507.2

PowerShell, Windows System, and Domain Auditing

THE MOST TRUSTED SOURCE FOR INFORMATION SECURITY TRAINING, CERTIFICATION, AND RESEARCH | sans.org

August 10, 2021

AUD507.2 | Auditing & Monitoring Networks, Perimeters, & Systems

SANS

PowerShell, Windows System, and Domain Auditing

Welcome to Section Two of the SANS AUD507 course! This course is written, maintained, and frequently taught by Clay Risenhoover. I am always looking for ways to improve this courseware. If you have questions or suggestions for how to improve the course, or if you need any additional materials referenced during the class, please let me know. If you find errors or inaccuracies in the course books, I encourage you to pass those on to me. You can email me at clay@risenhooverconsulting.com. Please put either "SANS" or "AUD507" in the subject line, to ensure I see the email.

This page intentionally left blank.

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. **Background and Plan**
   - *Course Coverage*
   - *Audit Plan*
2. PowerShell and WMI
3. Windows Auditing
4. Users, Groups, and Privilege Management
5. System and Resource Security
6. Windows Logging
7. Continuous Monitoring

This course provides an overview of techniques for performing a technical audit of Microsoft Windows systems. We explain the basic concepts behind Windows auditing and the various tools used to audit (and secure) a Windows system. Our goal is to place auditing concepts in the context of the Microsoft Windows operating system and to demonstrate, hands-on, many of the tools you can use to obtain information about a Windows system.

A larger and more important goal is to see how to design an audit process that informs the organization regarding risk, while simultaneously feeding back to the operations environment. What this means is that we should not only analyze systems and processes to determine risks and weaknesses, but we should also design and possibly even implement sustainable systems that can perform continuous monitoring within the Windows domain and alert administrators proactively to issues that can affect operational and security objectives.

Windows Operating Systems

- Course techniques are intended to address:
  - Windows 10+
  - Server 2016+
- Windows 10:
  - Most secure version of Windows yet
  - Introduces some interesting update and telemetry issues

The material in this section is designed to cover modern versions of Windows but to also address some of the legacy issues you may face. For instance, even though Windows 2008 Server passed the end-of-life marker quite some time ago, it is still fairly common to find a lingering installation. Most often, this is not the result of administrator or organizational laziness, but instead is necessitated by a piece of business-critical software that simply will not function on newer versions of Windows. This is a problem that is not unique to Windows environments.

Our primary focus is on what sorts of security features should be examined and what operational process questions should be asked to ensure effective controls for the Windows domain. To this end, a large portion of the material focuses on the security of endpoints (client workstations and laptops) and how the security of these endpoints is governed by the domain servers. We also spend time discussing server security and domain configuration practices that have a strong impact on the overall security of the domain.

Windows versions that are specifically covered include Windows 10, Server 2016, and Server 2019. For those who are still using older Windows versions today, you should definitely be giving feedback to your organization about the risks!

Windows 10 introduced some interesting issues for security. If you do not have strong patch management controls, you may find Windows 10 systems randomly installing updates and rebooting without warning. Even with updates mostly disabled, you can still find Windows 10 systems updating!

Baselining a System

- Enterprise defines standards:
  - Typically an administrator task
  - A lot of pointers on this as we go
- Image a system and snapshot it:
  - This is your baseline
  - We'll examine exactly what to baseline
- We monitor standard for changes:
  - Verify change control of image master
  - Verify continuous monitoring of variations from baseline by operations

Each enterprise must define or adopt a standard for the configuration of systems used within it. This can be done by adopting an existing configuration standard, such as the Level 1 Benchmark from the Center for Internet Security, or the Secure Host Baseline (SHB) used within the US Department of Defense. Even if existing configuration standards such as these are not a perfect fit for a specific organization, a great deal of time can be saved by adopting something like these, and then documenting how your organization chooses to vary from that standard.

After the standard is established, administrators have the responsibility to create a master image from which all, or nearly all, other systems are built. As a brief aside into process, if your organization is *not* building systems from an image, it's safe to say that you're doing something wrong! It becomes impossible to achieve any degree of consistency when the organization contains more than just a handful of systems unless these systems are built from a standard image.

With the image established, the auditor can step in. We have several possible tasks. One is to verify that the criteria used to build the master image is appropriate when compared to the organizational objectives. Another more common activity is to document the image in terms of a baseline, so that variations from this baseline can be detected.

Yet another and likely more important role that the auditor should play, but rarely does, is to verify that the operations team has an effective system to perform continuous monitoring of live systems in comparison to the established standards. We will focus on this throughout the section and see how this type of monitoring can be established.

Auditing a Windows Domain

- Even in a domain there are unique systems:
  - AD servers
  - Exchange servers
  - Certificate servers
  - Servers in general
  - Configuration masters
- Security of endpoints affects security of entire domain:
  - Integrated security allows for scalability

When auditing a Windows domain, we might mistakenly conclude that there is no need to look at individual systems because they are all tied together through the Active Directory (AD). This is a false assumption. Although the overall security of the domain is maintained centrally, and the configuration standards can be centrally controlled, the security of each individual system affects the security of the domain at large. Furthermore, even within a domain, there are systems that truly can be viewed and audited as standalone systems.

An example of this would be the Active Directory servers themselves. These servers contain all the configuration information for the entire domain, yet in many ways they are configured uniquely when compared to all other systems within the domain. The same can be said of Exchange servers, file servers, certificate servers, and any other server-class system filling a particular role within the enterprise domain.

Although systems of that type must be audited for the capability to fulfill organizational objectives specific to each role, the integrated nature of the security within the domain also allows for a great deal of automation and facilitates scalability. This is true for two main reasons.

The first reason is that some of the most critical settings that affect the security of the domain and individual systems are integrated into the domain. The second is that the centralized authentication store provided by Active Directory, coupled with the instrumentation inherent in the Windows operating system, allows virtually every important setting to be examined remotely and through automated means.

Operational Issues

- Separation of duties
- Principle of least privilege
- Procedural issues:
  - New account setup
  - Password change
  - Backup policy
  - Configuration management

When discussing the security of information systems, it is easy to get caught up in the technical issues involved. Remember that many important security issues are non-technical and rely on the behavior of individuals and the procedures they follow. These operational issues are not our primary focus, but we touch on them throughout this section.

Separation of duties is a process designed to ensure the security and integrity of a system, by preventing a single individual from possessing excessive power over a system or network. For example, the administrator who is responsible for maintaining a system should *not* be the person who reviews that system's audit logs. The idea is that a set of checks and balances are put in place so that any misconduct or error by one individual could be detected through the oversight of another individual.

The *principle of least privilege* is the philosophy that you should grant people only the minimum amount of permissions necessary for them to carry out their work. If people require only read access to a file, do not give them the ability to modify that file. If people need to update only a small number of fields in a database, do not give them access to the entire database. Limiting the scope of what people (or applications) can do limits the damage that can be done—accidentally or maliciously. If an application gets hacked, the attacker "inherits" whatever privileges are associated with that application.

Note that page D-2 of NIST Special Publication 800-53 (https://u.aud507.com/4-2), Appendix D, provides a high-level audit checklist that addresses both technical and nontechnical issues. It is one option for providing a framework to conduct audits of Windows systems and networks that address both technical and non-technical issues.

High-Level OS Checklist

- Basic system information
- Risk profile:
  - Running services:
    - Network
    - Local
  - Users, groups, and passwords
  - Protecting data
  - OS and application security
- Monitoring controls:
  - Auditing and logging

Now that we've touched on a few details, the majority of this course is dedicated to stepping through a sample audit of a Windows system and the different tools and techniques you can use to find out information about the system, and the status of its security. To do so, we use a high-level OS checklist as a framework. Please understand, you can apply this checklist to *any* operating system, and we also apply it during the Unix material. We detail how to go about applying this high-level checklist to Windows specifically.

Please take note that this high-level checklist applies to any operating system. Just as we have done throughout this course, the checklist uses a risk-based approach to work from the outside in, seeking to identify issues based on the level of risk that they would represent. This means that, in a real sense, this section's material is an example of the type and level of research that would be required on any operating system to use this high-level checklist.

Though we are primarily concerned with the technical aspects of auditing a Windows host, security does not rely solely on technology. It also depends on people and processes. We point out some key non-technical issues you should be aware of when conducting your audit, but this should not be considered an exhaustive list of non-technical issues.

Finally, we discuss not only auditing (in the sense of examining the system at a point in time to determine its security state) but also ongoing monitoring of Windows systems to detect unauthorized change or suspicious activity. And, of course, we discuss the all-important audit trail: The evidence that shows us who did what and when. We examine how to use Windows' built-in auditing mechanisms to piece together user and system activity.

Before we start digging into the checklist, we're going to take some time to go through two other important sections. We want to introduce you to a powerful tool for extracting data from a Windows system, and cover some basics of Windows scripting to allow you to begin to automate your tests.

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. Background and Plan
2. **PowerShell and WMI**
   - *PowerShell Fundamentals and Scripting*
   - Exercises 4.1: Scripting with PowerShell
   - *WMIC*
   - Exercise 2.2: Exploring WMI with PowerShell and WMIC
3. Windows Auditing
4. Users, Groups, and Privilege Management
5. System and Resource Security
6. Windows Logging
7. Continuous Monitoring

Before we go further in our discussion, it's important that we grapple with two topics. The first is the ability to remotely query important configuration information from member systems within the domain. The second is how we can mechanize our testing and scale it out to the entire domain. To accomplish these, we are going to examine PowerShell and Windows Management Instrumentation, and see how we can use these two powerful technologies together to automate our audit process.

We begin with a survey of PowerShell and how best to use it as an auditor.

Introduction to PowerShell

- Cross-platform scripting language
- Fully object-oriented
- Windows-native and cross-platform versions
- .Net (Core) integration

Microsoft describes PowerShell as "a cross-platform" (Windows, Linux, and macOS) automation tool and configuration framework optimized for dealing with structured data (e.g., JSON, CSV, XML, etc.), REST APIs, and object models. PowerShell includes a command-line shell, object-oriented scripting language, and a set of tools for executing scripts/cmdlets and managing modules." (https://devblogs.microsoft.com/powershell/announcing-powershell-7-0/)

This makes it ideal for daily use by audit, information security, and compliance professionals. The object-oriented nature of the shell GREATLY simplifies the handling of data in the pipeline, and the cross-platform compatibility in PowerShell core allows us to run the same code on multiple operating systems. Integration with the Windows .NET framework in PowerShell 5.1 and the .NET Core framework in the cross-platform versions allows users to extend the functionality of the language by including fully-functional .Net objects in their code.

PowerShell Editions

- Windows PowerShell
  - Powershell.exe
  - Windows native
  - Full .NET framework exposed
  - Stuck in version 5.1
  - No new features - security/stability updates only
- PowerShell Core
  - Pwsh.exe on Windows
  - Cross-platform
  - .NET Core framework
  - Version 7+ LTS
  - New features added regularly

There are currently two major branches of PowerShell:

Windows PowerShell is Windows-native and uses the full Windows .NET framework. However, version 5.1 is no longer under active development, and Microsoft is focusing their efforts on PowerShell Core, the cross-platform version.

PowerShell Core runs on Windows, MacOS and many Linux distributions, on both Intel and ARM processors. It uses the cross-platform .NET Core framework, which is a little different than the Windows version. It's under active development, which means that new features will be added only to PowerShell Core.

As of this writing, version 7.0 is the long-term support (LTS) version of Core, and version 7.1 is the "current" version.

PowerShell - Object-Oriented Shell

- Command return objects - not just text!
- Objects have properties (data) and methods (functions)

```
 Get-LocalUser | Get-Member | Select-Object Name, MemberType

Name                   MemberType
----                   ----------
Clone                      Method
ToString                   Method
AccountExpires           Property
Description              Property
Enabled                  Property
FullName                 Property
```


It's important to understand that PowerShell's command environment is **object-oriented.** This means that when you pipe the output of one command to the input of another, you are piping **objects** and not simply text. This allows for much more flexible selecting, sorting, grouping, and measuring of the output of a command.

In the example on the slide, the Get-LocalUser cmdlet is returning an array of objects which represent the local users on the host system. These objects have **methods,** or functions, that they can perform, and **properties,** or data, which they store about the user. Other commands which consume the output of the Get-LocalUser cmdlet can make use of these properties and methods as they process the results.

As you develop the skills and tools you need to automate compliance tasks, this object-orientation is a power tool in your toolbox.

The PowerShell Pipeline

- The vertical pipe character "|" is the "pipeline" operator. It is used to pipe the output from one command (an object) into the next command
- Pipelines allow for efficient processing of data, and for PowerShell commands to do a single thing well
- The pipeline operator allows for inserting a line break into a long PowerShell command
  - The grave accent character (`) also allows you to break a line

You may have used the pipeline operator in other shell environments to take the output of a command and pass it as input to the next command in the pipeline. PowerShell allows this, but it is different from many other shells in that the input and output are treated as objects (more about those later) rather than simply text.

We will make extensive use of pipelines in our data gathering work in this course. The pipeline allows us to string together many commands with limited functionality to get at the results we need.

Another handy use for the pipeline operator is to tell PowerShell that we are continuing a command on the next line. In fact, the only two ways to insert a line break into a command line are to put it after a pipeline operator, or to use the grave accent character to signal to PowerShell that the command continues on the next line.

Case Sensitivity

- PowerShell is not case-sensitive
- **`Get-aduser`** is equivalent to **`gEt-ADuSeR`**
- Good idea to use PascalCase for readability

```
PS C:\> GeT-aLIas pWd
CommandType     Name
-----------     ----
Alias           pwd -> Get-Location

PS C:\> Get-Location
Path
----
C:\
```


One thing to note as you begin to discover what PowerShell can do: unlike Unix shells which are case-sensitive, PowerShell is mostly case-insensitive. In fact, the few PowerShell tools which allow for case sensitivity require that it be explicitly enabled using run-time parameters or an alternative operator name. For example, the "match" operator has a "cmatch" version for case sensitivity.

Many PowerShell developers choose to use **camelCase** or **PascalCase** for their variable and function names. Since the PowerShell command shell has a tab-completion feature (begin typing a command name and hit TAB, and the shell will complete the name for you, choosing from a list of all possible matches), I often use tab-completion to let PowerShell do the capitalization for me!

Remember what case insensitivity means: if you create a variable called **$myvariable,** you can successfully access that variable by calling it **$MyVariable.** Likewise, you can call the **Set-Location** cmdlet **SET-LOCATION,** and it will work correctly.

Command Types: Cmdlets and Functions

- Cmdlets: Pre-compiled fully-functional commands
- Functions: Usually written in PowerShell and not pre-compiled

```
PS > Get-Command | Sort-Object Name | Select-Object CommandType, Name

CommandType Name
----------- ----
   Function A:
     Cmdlet Add-ADCentralAccessPolicyMember
     Cmdlet Add-ADComputerServiceAccount
     Cmdlet Add-ADDomainControllerPasswordReplicationPolicy
     Cmdlet Add-ADFineGrainedPasswordPolicySubject
     Cmdlet Add-ADGroupMember
      Alias Add-AppPackage
      Alias Add-AppPackageVolume
```


The commands we run in PowerShell come in many different shapes. The most commonly used is called a *cmdlet*. PowerShell cmdlets are pre-compiled commands, which are either included with the baseline install of PowerShell or loaded as part of an added-in module. Cmdlets stand alone; they need no other libraries or modules loaded in order to run. Often cmdlets are written in a high-level programming language like C#, before being compiled to run under PowerShell. The source code for cmdlets is not included with them.

Functions are often written in PowerShell and included as source code as part of a module. Beyond that, however, they tend to function just like cmdlets: The user can run the function directly at the shell, or as part of a script simply by calling it by name.

For our purposes of doing audit measurements, we make no distinction between using functions and cmdlets in our scripts or shell commands.

In the screenshot on the slide, we are using the Get-Command cmdlet to list the various commands installed and available on this system at this time.

Command Types: Aliases

- Short names for other cmdlets or functions
- Recommendation: Use only in interactive console; use full names in scripts for readability

```
PS C:\> Get-Alias

CommandType     Name                                               Definition
-----------     ----                                               ----------
Alias           %                                                  ForEach-Object
Alias           ?                                                  Where-Object
Alias           ac                                                 Add-Content
Alias           asnp                                               Add-PSSnapIn
Alias           cat                                                Get-Content
Alias           cd                                                 Set-Location
Alias           chdir                                              Set-Location
```


Aliases are used to make it faster to call certain commonly used commands, or to make the transition from other shells to PowerShell a bit smoother. Good examples would be the “ls” and “dir” aliases. The PowerShell way to list the files in a directory is to use the Get-ChildItem cmdlet. Since most new users are not familiar with this command, PowerShell provides “ls” and “dir” as aliases for Get-ChildItem. If a user types “dir” at the prompt, PowerShell treats it as if they had typed Get-ChildItem.

You can see what aliases exist on your system using the Get-Alias command. It will return a list of all aliases and the real commands they represent.

A good practice for PowerShell scripting is to use full command names rather than aliases in your scripts. This prevents future administrators and auditors who read your script from having to look up aliases to see what your code really does. When running commands directly in the shell, use all the aliases you want to speed up your work.

Command Name Format

- Cmdlets and functions are commonly named in a "Verb-Noun" format
- Noun is always singular
- Examples:
  - **Get-ADUser**
  - **Set-FileShare**
  - **Write-Host**
  - **Remove-Job**
  - **New-Object**

```
PS C:\> Get-Verb

Verb        Group
----        -----
Add         Common
Clear       Common
Close       Common
Copy        Common
Enter       Common
Exit        Common
...
PS C:\> Get-Verb | Measure-Object
Count    : 98
```


You may have noticed by now that PowerShell commands all have similar names, in that they all seem to use a "Verb-SingularNoun" format. The noun will always be singular, and there is a limited set of verb names which are used. You can get a list of valid verbs for your version of PowerShell using the Get-Verb command.

Knowing this convention can make it easier to "guess" what commands will exist for your needs. For instance, if you need to retrieve a list of Active Directory users, the command will be Get-ADUser; it will never be called Get-ADUsers, because the noun is always singular.

Command Parameters

- Used to modify command behavior
- Multiple parameters for most commands
- Syntax: **`some-command -ParameterName someValue`**
- For "positional" parameters, the parameter name can be omitted

```
PS C:\> Get-Help Get-Command

Get-Command [[-Name] <string[]>] [[-ArgumentList] <Object[]>]...

PS C:\> Get-Command -Name Get-ADUser

PS C:\> Get-Command Get-ADUser
```


Parameters are used to modify the way a command behaves. It's not unusual for a command to have several parameters. You get a list of parameters for a command by using the Get-Help cmdlet. If command's syntax description puts a parameter name inside square brackets, like the "Name" and "ArgumentList" parameters in the sample, above, it means that the parameter name can be omitted on the command line. This is known as a parameter being "positional."

The bottom two commands on the slide are equivalent, since "Name" is a positional parameter.

Special Mention – Quotation Marks

- Convention is to use single quotes (')
- Double-quotes (") are used for:
  - Including a variable's contents in a string
    ```
    $cmd="Get-ChildItem $path"
    ```
  - Including single quotes inside a string
    ```
    $name="O'Brian"
    ```
  - Using escaped characters
    ```
    Write-Host "Col1`tCol2"
    ```
    - ` is the "grave accent," and is used as the escape character (similar to "\t" in other languages)

As with any shell or programming environment, there are some special syntax and semantic rules you need to know. One of PowerShell's idiosyncrasies is its use of quotation marks. Single and double quotes are largely interchangeable, but there are some conventions and special rules you should know:

- You should use single quotes by default; this is the convention for most situations.
- Use double quotes when including a variable's value inside the string. PowerShell will replace the variable name with its value inside double quotes, but not within single quotes.
- When you need to include a single quote (or apostrophe) in a string, wrap the string with double quotes, or "escape" the single quote by preceding it with a grave accent or backtick character.
- Grave accent marks are used to escape other characters. In the example above, "`t" represents a TAB character. "`n" would represent, the newline character.

Getting Help

- Use the **`Get-Help`** command
- Shows the built-in help text for the command
- Most-current help file will be online:
  `Get-Help Get-ADUser –Online`

```
PS C:\> Get-Help Get-ADUser

NAME
    Get-ADUser

SYNTAX
    Get-ADUser -Filter <string> [-AuthType {Negotiate | Basic}] [-Credential <ps
    [-SearchBase <string>] [-SearchScope {Base | OneLevel | Subtree}] [-Server <
```


PowerShell gives you lots of ways to figure out how to solve problems. One of the most useful is the Get-Help command.

This command will show you the help text for a command. The Get-Help output will show you all of the command-line parameters for the command and the data types they require. There is even the -examples flag to Get-Help which will show examples of using the command in question.

The -online flag will open your default web browser to the current online help page for the command.

Updating Help Files

- **`Update-Help`** cmdlet downloads current copies of all help files
  - May take a while to run
- **`Save-Help`** cmdlet saves help files for offline updating
  - Good for non-internet connected hosts

```
PS C:\> Update-Help

 Updating Help for module AppBackgroundTask
    Connecting to Help Content...
    [ooooooooooooooooooooooooooo
```


To best use the help files, it is a good idea to keep them up-to-date on your computer. The Update-Help command is used to download current copies of the help files to a system for later use. Because it is downloading the files from the Microsoft website, it may take a few minutes for this command to finish running the first time you use it.

The Save-Help command can save a local copy of the help files, which can then be shared to machines with no ability to download files from the internet. If you have server subnets with no internet access, this allows your administrators to use PowerShell on those computers with the current help files installed.

PowerShell for Audit Evidence Acquisition

- Selecting and sorting data
- Managing output
- Data formats
- PowerShell Scripting
- Parameters/Functions
- Looping
- Conditionals

Microsoft advertises that PowerShell is designed for handling structured data. This is made easier by the many utility commands which allow us to "slice and dice" data in the pipeline any way we want. While I sometimes miss the power of a Unix tool like AWK, I more often find that the PowerShell techniques of selecting, grouping, and sorting data using single-purpose cmdlets and functions is very powerful and flexible.

In this section of the course, we discuss the many tools and techniques available to help us quickly gather and process the data we need.

Selecting and Sorting with PowerShell

- Get-Member
- Select-Object
- Where-Object
- Sort-Object
- Get-Unique
- Group-Object
- PowerShell comparison operators
- PowerShell formatting commands

Microsoft advertises that PowerShell is designed for handling structured data. This is made easier by the many utility commands which allow us to "slice and dice" data in the pipeline any way we want. While I sometimes miss the power of a Unix tool like AWK, I more often find that the PowerShell techniques of selecting, grouping, and sorting data using single-purpose cmdlets and functions is very powerful and flexible.

In this section of the course, we discuss the many tools and techniques available to help us quickly gather and process the data we need.

The commands listed on this slide are all useful for extracting just the right amount of data from our data-gathering efforts. We'll cover each of them in detail over the next several slides.

Get-Member

- Get a list of events, properties and/or methods for an object
- Choose which properties to select when gathering data
- Can specify that Get-Member return only properties (or events/aliases) with Type parameter

```
PS C:\> Get-Service | Get-Member -Type Properties
Name                  MemberType    Definition
----                  ----------    ----------
Name                  AliasProperty Name = ServiceName
CanPauseAndContinue   Property      bool CanPauseAndContinue {get;}
CanShutdown           Property      bool CanShutdown {get;}
```


One of the first steps I take when preparing to use an object for data gathering is to pipe a sample object through the Get-Member cmdlet. Get-Member is used to retrieve a list of the properties, methods, and events which are included with an object.

If you know that you are only interested in seeing the properties, methods, or events of an object, you could limit the results using the "Type" parameter. In the example on the slide, we have asked Get-Member to show only the properties of the object returned by Get-Service.

Select-Object

- Aliased to "select"
- Analogous to SELECT clause in SQL
- Specifies which **properties** to return for an object
- Limit which **objects** are returned using Skip, Index, First, Last parameters

```
PS C:\> Get-Service | Select-Object -Property Name,Status,StartType
-First 1 -Last 2
Name                  Status StartType
----                  ------ ---------
AarSvc_66cd7         Stopped    Manual
XboxNetApiSvc        Stopped    Manual
XenServerHealthCheck Running Automatic
```


The Select-Object command is used to filter the results returned from the previous command(s). Select-Object lets the user specify which properties should be passed on from the previous commands. This will often be the first part of the filtering you'll do to get only the results you need for your audit test.

When used with a list of properties, Select-Object will include only those properties in the output objects. When used with parameters like –Skip and –First, it will limit how many objects are output.

```
PS C:\> Get-WmiObject win32_NetworkAdapterConfiguration | Select-Object
Description, MACAddress, DHCPEnabled

Description                                MACAddress        DHCPEnabled
-----------                                ----------        -----------
Microsoft Kernel Debug Network Adapter                             True
Intel(R) 82574L Gigabit Network Connection 00:0C:29:9A:F4:F4      False
Bluetooth Device (Personal Area Network)   44:85:00:69:4A:87        True
WAN Miniport (SSTP)                                               False
WAN Miniport (IKEv2)                                              False
WAN Miniport (L2TP)                                               False
WAN Miniport (PPTP)                                               False
WAN Miniport (PPPOE)                                              False
WAN Miniport (IP)                          50:AE:20:52:41:53      False
WAN Miniport (IPv6)                        72:02:20:52:41:53      False
WAN Miniport (Network Monitor)             AC:40:20:52:41:53      False
```

Where-Object

- Aliased to "where" and "?"
- Analogous to WHERE clause in SQL
- Returns objects which meet criteria
- Two formats:
  - Script block (old): **`Where-Object { $_.Status -eq 'Running' }`**
  - Comparison (new): **`Where-Object Status -eq 'Running'`**

```
PS C:\> Get-Service |Where-Object Status -eq Running |select -first 2
Status   Name               DisplayName
------   ----               -----------
Running  Appinfo            Application Information
Running  AudioEndpointBu... Windows Audio Endpoint Builder
```


The Where-Object command is like a SQL WHERE clause. It allows you to specify the conditions under which an object should be returned from the command.

There are two formats for the Where-Object filter:

With the comparison statement format, objects are selected based on comparisons of the value of specified properties.

In the script-block format, the comparison is made against a variable called "$_" which contains the current object to be evaluated.

These two methods are functionally equivalent, but you may find situations where you prefer using one over the other. Most users seem to find the first format to be easier to read.

Sort-Object

- Aliased to "sort"
- Analogous to SQL ORDER BY clause
- Changes the order in which objects are returned
- Reverse the sort with Descending parameter
- Show only one of each sorted value with Unique parameter

```
PS C:\> Get-Service | Select Name,StartType | Sort StartType -Unique
Name                StartType
----                ---------
VGAuthService       Automatic
spectrum               Manual
NetTcpPortSharing   Disabled
```


You can think of the Sort-Object cmdlet as being similar to the ORDER BY clause in SQL. It takes in a collection of objects and outputs them sorted according to the criteria you specify. It is normally aware of object types and will correctly sort numbers or strings. Occasionally, it may be necessary to "cast" an object to the correct data type to get it to sort properly.

In this example, we have integers stored as strings. This often happens when using input formats like CSV. When we try to sort them, Sort-Object treats the values as strings and sorts them alphabetically, which is not likely what we want:

```
PS C:> "1","12","5" |Sort-Object
1
12
5
```


To correct this, we can explicitly cast the strings to integers to make sure they are handled correctly. In this example we will sort the integers in descending order:

```
PS C:\> "1","12","5" |Sort-Object { [int]$_ } -Descending
12
5
1
```

Get-Unique

- Aliased to "gu"
- Returns unique items from a sorted list
- Case sensitive in all versions of PowerShell

```
PS C:\> 'A','b','B','A' |Sort-Object -CaseSensitive |Get-Unique
A
b
B
```


Get-Unique is similar to the “uniq” command from Unix. It takes in a **sorted** collection of objects and outputs a collection of unique objects (no duplicates). This cmdlet is one of the few that is case sensitive in all versions of PowerShell.

Group-Object

- Aliased to "group"
- Analogous to the SQL GROUP BY clause
- Returns a new object with Count, Name, and Group properties
- Group property is a collection of matching input objects

```
PS C:\> Get-Service |Group-Object -Property StartType

Count Name        Group
----- ----        -----
  190 Manual      {AarSvc_66cd7, AJRouter, ALG, AppIDSvc...}
    7 Disabled    {AppVClient, NetTcpPortSharing, RemoteAccess...}
   61 Automatic   {AudioEndpointBuilder, Audiosrv, BFE, BITS...}
```


Group-Object works similarly to the GROUP BY clause in SQL. It groups its input object collection into a new collection of objects with Count, Name and Group properties. Because it is “destructive” to the input object, I often include this cmdlet toward the end of my pipeline.

Since a lot of audit data gathering involves counting things which have particular attributes, this cmdlet is an important one to learn.

PowerShell Comparison Operators

- Equality operators: -eq -ne -gt -ge -lt -le
- -like/notlike
- -match/notmatch
- -in/notin
- -contains/notcontains
- -is/isnot

Cmdlets like Where-Object and PowerShell constructs like the "if" statement, discussed elsewhere, make use of comparison operators to know what they need to do.

In addition to the mathematical-styled equality operations, there are a number of other operators which can work on arrays and objects.

On the next several slides, we will discuss several import comparison operators.

Comparison Operators: Equality

-eq equals
-ne not equals
-gt greater than
-ge greater than or equal
-lt less than
-le less than or equal

```
PS C:\> 1 -lt 3
True
PS C:\> 3 -le 2
False
```


Most PowerShell operators and syntax match the conventions used in modern Microsoft languages like C#. Some, like the equality comparison operators on this slide, are modeled after the operators in the PERL scripting language. These operators are less intuitive than the C# operators, but once you learn them, they're fairly easy to remember.

The operators on this slide are used for mathematical comparisons.

```
PS C:\> [Math]::pi
3.14159265358979
PS C:\> [Math]::pi -gt 3
True
PS C:\> [Math]::pi -gt 4
False
PS C:\> [Math]::pi -ge 3
True
PS C:\> [Math]::pi -lt 3
False
```

Comparison Operators: Like/Notlike

-like	Returns true when test string exists in another string
-notlike	Returns true when test string does not exist in another string

- Both allow the use of the '*' wildcard

```
PS C:\> 'AUD507' -like '*50*'
True
PS C:\> 'AUD507' -notlike '*40*'
True
```


The -like and -notlike operators allow for wildcard-based comparisons to see if a string contains (or does not contain) a particular substring. These operators are commonly used in filters for things like usernames, computer names, or IP addresses.

```
PS C:\> Get-NetIPAddress | Where-Object IPAddress -like '127*'


IPAddress         : 127.0.0.1
InterfaceIndex    : 1
InterfaceAlias    : Loopback Pseudo-Interface 1
AddressFamily     : IPv4
Type              : Unicast
PrefixLength      : 8
PrefixOrigin      : WellKnown
SuffixOrigin      : WellKnown
AddressState      : Preferred
ValidLifetime     : Infinite ([TimeSpan]::MaxValue)
PreferredLifetime : Infinite ([TimeSpan]::MaxValue)
SkipAsSource      : False
PolicyStore       : ActiveStore
```

Pattern Matching: Match/Notmatch

-match	For strings, returns true if given regex is included in the input. For collections, returns all objects which match. Case insensitive by default.
-notmatch	Returns false if given regex is included in the input.

```
PS C:\> "SEC557" -match "SEC[0-9]*"
True
PS C:\> "SEC557" -notmatch "SEC[0-9]{4}"
True
```


The match operators search for a regular expression in either a single string or a collection of objects. When used on a single string, they return a $true or $false value indicating whether there was a match.

When used on a collection of objects, they return all objects which match (or don't match, in the cast of "notmatch") the regex.

The case sensitive versions are "cmatch" and "cnotmatch." The explicitly case insensitive versions are called "imatch" and "inotmatch."

Comparison Operators: In/NotIn

-in	Returns true when test value contained in a collection
-notin	Returns true when test value not contained in a collection

```
PS C:\> 3 -in 1,2,3,4
True
PS C:\> 5 -notin 1,2,3,4
True
```


The PowerShell operators on this slide are used for string and collection comparisons. They allow the user to determine if a given string contains a particular substring, or whether a collection of objects has a particular object in it.

```
PS C:\> $ips = (Get-NetIPAddress).IPAddress
PS C:\> "127.0.0.1" -in $ips
True
PS C:\> "127.0.0.2" -in $ips
False
PS C:\> "127.0.0.2" -notin $ips
True
```

Comparison Operators: Contains/NotContains

-contains	Returns true when reference value contained in a collection
-notcontains	Returns true when reference value not contained in a collection

```
PS C:\> 1,2,3,4 -contains 3
True
PS C:\> 1,2,3,4 -notcontains 5
True
```


The PowerShell operators on this slide are used for string and collection comparisons. They allow the user to determine if a given string contains a particular substring, or whether a collection of objects has a particular object in it.

```
PS C:\> $ips = (Get-NetIPAddress).IPAddress
PS C:\> $ips -contains "127.0.0.1"
True
PS C:\> $ips -contains "127.0.0.2"
False
PS C:\> $ips -notcontains "127.0.0.2"
True
```

Managing Output in PowerShell

- Format-list - Formats output into one-line-per property
- Format-table - Formats output into one-property-per-column tabular format
- Measure-object - Get numerical data about the input object(s)

```
PS C:\> 1..10 | Measure-Object -Sum -Average -Maximum -Minimum -StandardDeviation

Count             : 10
Average           : 5.5
Sum               : 55
Maximum           : 10
Minimum           : 1
StandardDeviation : 3.02765035409749
```


The output from PowerShell commands can sometimes be pretty lengthy and difficult to read. Fortunately, there are utilities we can use to reformat the output to look the way we want. Two of the most commonly used are Format-Table and Format-List.

Format-Table presents data in columns going across the screen. It makes for very compact output, and if there are not many columns, it is very readable.

Format-List displays data with one property per row on the screen. This makes for much longer output, but it can be easier to read the value of each property when it is on its own line.

Most commands will default to using one of these formats based on the number and type of properties returned. You can override the default format by piping the output of the command through the tool you wish to use.

The Measure-Object cmdlet takes mathematical measurements of the input object collection, including the count, average, sum, minimum, maximum, and standard deviation for the collection.

Common Data Formats

- JSON - JavaScript object notation
- XML - extensible markup language
- CSV - comma-separated values

Microsoft markets PowerShell as being designed to handle structured data, and this is really one of PowerShell's big advantages over other languages. PowerShell has native tools for working with CSV, JSON and XML, and it can use .NET objects to do more with XML and HTML.

JSON in PowerShell

- Convert between PowerShell objects and JSON with
  - ConvertFrom-Json
  - ConvertTo-Json
- Test JSON validity with Test-Json (PowerShell Core)

The ConvertTo-JSON and ConverFrom-JSON cmdlets will be your go-to tools for processing JSON data. Given how that most web APIs and many other tools use JSON formatted data, you will make extensive use of these cmdlets during this course and your compliance automation career.

The Test-JSON cmdlet is used to test that JSON is valid. Spoiler alert: a lot of the JSON you will encounter in the wild will not pass this validation test, and you'll STILL be able to work with it pretty efficiently in PowerShell.

XML in PowerShell

- ConvertTo-Xml
- Select-Xml - uses Xpath queries
- Work with Common Language Infrastructure (CLI) XML using
  - Export-CliXml
  - Import-CliXml
- Export-CliXml can be used to saved encrypted credentials on Windows using the Data Protection API

XML is less well-supported than JSON. Converting data to XML with the ConvertTo-XML cmdlet is pretty straightforward. Reading XML data into an object is often trickier and may involve using a .NET XmlDocument object to handle the conversion.

PowerShell supports the common language infrastructure XML format with the *CliXML cmdlets. These are pretty handy for exporting an object to XML and then re-importing it later. Export-CliXML can even be used to store encrypted credentials (only in Windows PowerShell) in a file.

CSV files in PowerShell

- CSV is a common output format for LOTS of tools
- Many security APIs give results as CSV
- PowerShell has native cmdlets for CSV handling:
  - ConvertFrom-Csv: CSV pipeline input to an object
  - ConvertTo-Csv: input object to CSV
  - Import-Csv: reads a file into an object
  - Export-Csv: writes an object to a file

Many of the tools you will encounter doing compliance work will use CSV as an import or export format. The ConvertFrom-CSV and ConvertTo-CSV cmdlets process CSV data on the pipeline.

The Import-CSV and Export-CSV cmdlets work directly with files. Import-CSV uses different processing code than ConvertFrom-CSV and may yield slightly different results. This is good to know, because you can sometimes parse badly-formatted data with one of these tools, even if the other doesn't work.

PowerShell Scripting

- A script is a collection of commands saved in a file to execute at will
- Makes our code reusable and easy to distribute
- Makes automating and scaling possible
- Some PowerShell script features:
  - Parameters
  - Functions
  - For/foreach loops

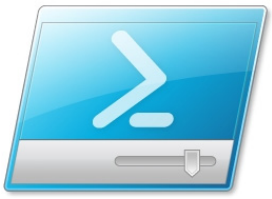
PowerShell is not just a shell. Like many command shells, it is also a scripting language. At its simplest, a script is merely a list of commands for the shell to run. At a deeper level, scripting is what allows us to eliminate repetitive work, and ensure that we receive consistent results, no matter who is doing the data gathering.

Automation and scaling to the enterprise level are not possible without scripting.

After a description of development environments, we'll look at two features that make scripting powerful. Parameters allow us to control some of the script's behavior at run-time. Functions allow us to reduce repetition in our code and make it more efficient. While PowerShell offers many types of loops, we will cover the For and Foreach loops as examples.

Development Environment Options

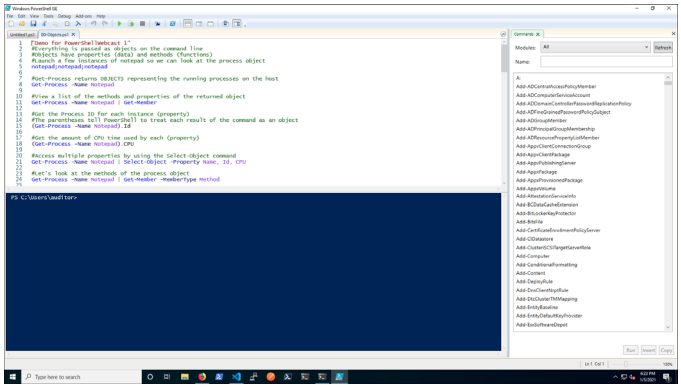
- PowerShell Integrated Scripting Environment (ISE)
- Visual Studio Code (VS Code)
  - SSH/SCP plugins for Linux file editing



You have several options available for developing your PowerShell scripts. While even a simple text editor like Notepad can be used for script creation and editing, there are much more powerful alternatives out there. Many of the integrated scripting and development environments allow for debugging and interaction with remote systems to happen in the editor environment, which can save you a lot of time.

Development Environments - ISE

- Included with PowerShell 5.1
  - No longer being actively developed
- Nice tab-completion
- Built-in help tools

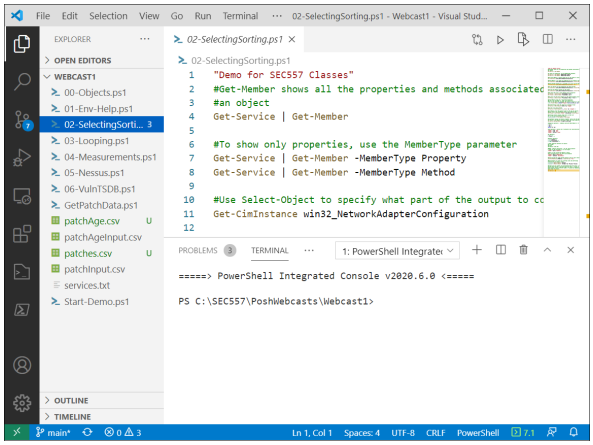


The Integrated Scripting Environment (ISE) is included with Windows PowerShell, but not with PowerShell Core. It allows for fairly rapid code development, with tab-completion for command and parameter names, built in debugging, and an integrated command shell.

The ISE will die with PowerShell 5.1, however. It is no longer being actively developed by Microsoft.

Development Environments - VS Code

- Cross-platform GUI for editing code
  - Windows
  - Mac
  - Linux (Snap, .deb, .rpm, .tar.gz versions)
- FREE
- IntelliSense
- Large range of plugins to extend functionality
- Git integration



Visual Studio Code (VS Code) is an open source, free development environment which supports PowerShell very nicely. It is cross-platform, so you can use the same editor on your MacOS and Linux machines as your Windows systems.

It has IntelliSense and tab-completion (if anything, it is sometimes TOO helpful with its auto-suggestions), and the ability to integrate directly with source code management systems like GitHub.

The real value of VS Code lies in the huge array of plugins available to extend the functionality. In this course, for instance, we use a plugin to manage remote server filesystems over SSH.

Parameters

- Parameters allow for the user to change the behavior of the script/function
- Parameter definition can optionally include a variable type and a default value
- Parameters must be defined in the first executable line of the script

```
param (
  [int] $MaxHosts=10,
  [string] $HostName,
  $ServiceCount
)
```


The first executable (non-comment) lines in a script can contain parameters to alter the behavior or settings of the script. Function parameters work exactly like the parameters for PowerShell cmdlets. You can specify a parameter value when calling the script by using a hyphen in front of the parameter name.

You can explicitly specify the data type of a parameter in the definition, and you can specify a default value if you like. The default value is used if the user does not supply a value for the parameter. If there is no default value, parameters which are not provided are set to $null.

Functions

- List of reusable statements with a name attached
- Can use parameters to refine behavior
- "Return" keyword will exit function and return the specified value
- Value of last command run is returned if there is no "return"
- Must be declared ABOVE any calls to the function (top-down parsing)

```
Function Get-PID {
  param( $ProcessName = "PowerShell")
  (Get-Process -Name $ProcessName).Id}
PS > Get-PID -ProcessName Notepad
12780
```


A function is a bit of reusable code to perform a specific task. Actually, many of the command line tools you already use in PowerShell are functions. Like scripts, functions can use parameters.

An interesting quirk of PowerShell is that functions must occur in the source code ABOVE any calls that are made to the function. This is because PowerShell is an interpreted language, which reads code from top to bottom during execution.

Functions can return values to the caller. This can be done explicitly using a return command, or, if no return command is used, the return value of the last command becomes the return value of the function.

For Loop

- Similar to the FOR loop in most languages
- Uses a starting state, condition for repeating, and command to execute on each repeat
  - Start with $x = 20
  - Continue while $x is less than 30
  - Increment (add 1 to) $x on every iteration

```
for($x=20;$x -lt 30;$x++) {
  Invoke-Expression "ping -n 1 10.50.7.$x"
}
```


The For loop is very similar to the for loop in languages like C#. There are three parts to the for-loop declaration:

- The starting state - This is often just a value assigned to a variable. In the example above, this is **$x=20**.
- The repeat condition - The loop will continue to run as long as the condition evaluates to $true. In the example, this is **$x –lt 30**.
- The repeat command - This will be executed on every iteration of the loop. In the example, it is **$x++.** That is a shorthand way of saying "increment (add one to) $x.

ForEach/ForEach-Object Loops

- Loop through all objects in a collection (ForEach-Object) or on the input pipeline (ForEach)
- When used in a pipeline, ForEach-Object is aliased to "Foreach" (confusing, we know…)

```
Foreach( $svc in (Get-Service) ) {
  Write-Host ($svc.Name).ToUpper()
}

#Next two loops are equivalent:
Get-Service | Foreach { $_.Name.ToUpper() }

Get-Service | Foreach-Object { $_.Name.ToUpper() }
```


Foreach loops are object-based loops which iterate over every object in a collection. This can be very helpful for doing the same operation on multiple objects. And remember that EVERYTHING is an object…

If you wanted to run the same code against every computer in a domain, you could easily accomplish this with a foreach loop.

Unfortunately, the PowerShell developers were a little inconsistent in their naming:

- Foreach is the loop type. It processes a collection of objects using a variable name assigned for the task.
- Foreach-Object takes in a collection on the pipeline and processes each object as the special variable $_.
- When Foreach is used with pipeline input, it is really an alias for Foreach-Object. This is why sometimes we hate aliases!

Conditionals: If/Else Statements

- If block runs code only if a condition is true
- Optional "else" block will be run if condition is not true
- Condition can be anything that evaluates to $true or $false (built-in PowerShell variables for true and false values)

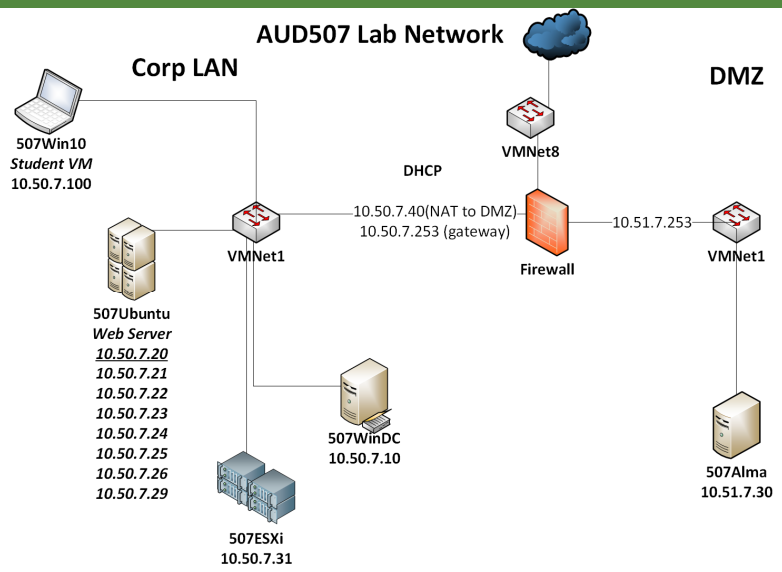
```
if( [Math]::PI -is [int]){
  Write-Host "Pi is an integer"
} else {
  Write-Host "Pi is not an integer"
}
```


Branching is the ability of a script to "make decisions" about which code to run based on conditional statements. (A conditional statement is anything that can evaluate to $true or $false).

If blocks make part of the code optional. Whatever is in the script block, between the curly braces, will execute only if the condition is $true.

You can optionally add an else block, which will run if the condition evaluates to $false.

Exercise 2.1: Scripting with PowerShell



Follow the directions of your instructor now to begin working on the PowerShell scripting exercise. As you proceed through the material in this book, see if you can leverage any of the techniques from this section to create automated monitoring scripts!

Scripting?

- We're not trying to make you scripters:
  - Consider the svcAccounts.ps1 script on the Windows 10 VM in the C:\scripts folder:
    - Can you see how this enables us to scale?
    - Who should be running these scripts?
  - We want you to take an existing script that's close and modify it.

Now that you've worked through that scripting lab, let's make something clear. We're not trying to turn you into professional script writers! This is clearly an administrator task, primarily. However, can you see the power of scripting? In fact, as an auditor, it is wise to understand the basics of scripting. If you work out some complex command line that allows you to retrieve an important piece of data, how many times do you want to have to work that out? How many times do you want to have to type it? Wouldn't it make much more sense to put it into a simple script?

Even more than this, the point of scripting is to introduce to you the feasibility of doing this type of testing at scale. This moves us into the realm of continuous monitoring. In our industry right now, everyone is spending money on continuous monitoring. Actually, at its core, how is continuous monitoring different from scalable, scriptable audit tests that run periodically, producing a continuous measure of compliance and/or security? With this in mind, it's clear that administrators should be interested in partnering with you to create and expand such useful scripts. Although you will be interested in the periodic results, an administrator should be interested in the continuous results!

We have an example of taking the simple principle illustrated in our lab (that is, a baseline audit test of services) and scaling it out using WMI to run many queries against every system in our domain. You can find this PowerShell script saved as "C:\Scripts\svcAccounts.ps1" on your Windows 10 VM. As we go through the material and you see an important test for you and your organization, ask yourself, "Could I add that test to the existing script?"

What if PowerShell Is Not Available?

- Version incompatibilities
- Restricted operation
- "Belt and suspenders" approach. We'll teach you multiple ways to get the audit evidence you need:
  - PowerShell
  - WMIC
  - Command-line tools
  - GUI tools

Throughout this course section, we will take a "PowerShell first" approach. We'll also teach you the other tools you can use to get the job done.

Frequently on audits, I find that PowerShell is not available due to security restrictions, or that older hosts either don't have PowerShell or have the wrong version. Sometimes, there's a more elegant solution to a problem than PowerShell presents. And occasionally, there simply is no PowerShell function to accomplish the task.

We'll teach you a good mix of command-line tools, GUI utilities, and PowerShell functionality to allow you to get the job done, no matter what.

WMI

- Windows Management Instrumentation
  - Introduced when Microsoft first built the first NT-class systems
  - Introduced "instrumentation" into the operating system
- Related to other standards
  - WBEM
    - Web-Based Enterprise Management
  - CIMv2
    - Common Information Model
    - Maintained by DMTF
    - Standards-based model for specifying information about systems

Windows Management Instrumentation (WMI) itself has been around for a very long time. Its origins go back to when Bill Gates hired a team away from Digital Equipment to build the "Next Technology" (or "New Technology") operating system for Windows. Mr. Gates recognized that, at that time, he had a wonderful operating system for individual users, but that he did not have the skill and expertise within Microsoft to compete in the emerging networked operating system world. The team that he hired had years of experience working with and developing mainframe and mini-computer operating systems, both of which are well known for robustness and reliability. Much of this reliability comes from the instrumentation built into the operating system, allowing for careful monitoring.

While WMI was not necessarily a design requirement, it was a very natural element for these developers to include in imitation of the larger-scale systems with which they were most experienced. When WMI was first created, there were no standards for enterprise information gathering or management of operating systems. Even when these standards first began to emerge, Microsoft had no real impetus to comply with any of them.

Since that time, WMI has come to support WBEM (Web-Based Enterprise Management) and the related CIM (Common Information Model) standards. In fact, when using WMI, you have the ability to interact with a tree-like structure within the internals of the operating system, querying out and even configuring data. One of the trees available is a CIMv2 branch, giving you direct access to everything that the Distributed Management Task Force (DTMF) has specified within the CIMv2 schema.

What Can You See?

- Almost anything:
  - Essentially exposes almost any setting for the purpose of troubleshooting and scripting
  - CIMv2: Computer Information Model
  - Sampling:
    - NIC configuration
    - Desktop settings
    - Users and groups
    - Password lockout status
    - System configuration information
    - Event logs

What can you see with WMI? The answer is, almost anything! Actually, not only can you see almost anything, but you can change a great deal of what you see as well! For instance, you can remotely start up programs, terminate processes, add users, force password resets, etc. Almost anything that can be done in Windows can be done using WMI.

In fact, there are many, many things that you can do via WMI that are not immediately obvious from the list of available features. This is because WMI is allowing you to access a CIM-compliant data store (CIM stands for Common Information Model). This is an open standard that defines attributes and relations between attributes within a computer system that can be queried for useful data about the system itself.

WMI with PowerShell

- Get-WMIObject
  - Prior to PowerShell 3.0, this was the only way to query WMI/CIM
  - Now deprecated, but sometimes works better
  - Uses DCOM (Distributed Component Object Model)
  - Dies with Windows PowerShell 5.1
- Get-CimInstance
  - Newer, preferred way to query
  - Uses Windows remote management (WinRM)
    - WS-Man protocol based on SOAP
  - More compliant with CIM open standards
  - Only option in PowerShell Core

Our preference today for querying WMI information will be to use PowerShell. As PowerShell has evolved, two different cmdlets have been created to perform WMI queries. The first, Get-WMIObject uses the same network protocols as the WMIC tool. In other words, if WMIC works against a remote system, Get-WMIObject is likely to work as well.

The newer, preferred way to query WMI is to use Get-CimInstance. It uses the Windows remote management protocols to query the Computer Information Model (CIM) schema on a system. Get-CimInstance is more compliant with the CIM protocols, but depending on your organization's security policies, the remote management tools may not be available to you.

It's good to know how to use both cmdlets and the WMIC command to give you the best chance of success during a particular test.

Windows Management Instrumentation Console (WMIC)

- Wmic.exe
- Runs *from* all modern Windows versions:
  - Open a command prompt.
  - Type **WMIC** and press Enter
- Runs *against* all modern Windows versions

WMIC stands for Windows Management Instrumentation Console. We will briefly explore how WMIC can be used to explore WMI, but we will focus most of our attention on accessing WMI data using PowerShell.

To get WMIC running, you simply need to open a command prompt and type **WMIC** followed by the Enter key. You can use the tool in two different modes: Interactive or non-interactive. In our class, we focus on using the non-interactive mode because we are most interested in seeing how this tool can be used in an automated audit process.

Using the non-interactive mode allows us to easily place our results into text-based output files. It also allows us to more easily insert WMIC commands into a script rather than requiring user interaction. During the various lab exercises in this section, we will be asking you to discover how WMIC can be used to extract some of the useful information that we are looking for.

WMIC Reporting (1)

- Essentially text based:
  - WMIC <object> list

```
PS > wmic startup list
Caption                     Command                                                                    Description
 Location                                                                                  SettingID  User
OneDrive                    "C:\Users\auditor\AppData\Local\Microsoft\OneDrive\OneDrive.exe" /background  OneDrive
 HKU\S-1-5-21-1020904641-2997690839-244422220-1000\SOFTWARE\Microsoft\Windows\CurrentVersion\Run             WIN10\auditor
ZoomIt                      "C:\ProgramData\chocolatey\lib\zoomit\tools\ZoomIt.exe"                     ZoomIt
 HKU\S-1-5-21-1020904641-2997690839-244422220-1000\SOFTWARE\Microsoft\Windows\CurrentVersion\Run             WIN10\auditor
SecurityHealth              %windir%\system32\SecurityHealthSystray.exe                                SecurityHealth
 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run                                                         Public
VMware VM3DService Process  "C:\Windows\system32\vm3dservice.exe" -u                                    VMware VM3DService Process
 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run                                                         Public
VMware User Process         "C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr                VMware User Process
 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run                                                         Public
```


Using WMIC is actually pretty easy once you understand what it wants. Typically, in non-interactive mode, you simply run "wmic <object> <verb>", where the object is the facility in the WMI namespace that you wish to access, and the verb is what you want to do to it.

As an example, if you wanted to get a report of all the items that will be run at startup, you could run "wmic startup list", where "startup" is an object in the WMI namespace and "list" is a verb that will list out all the nodes or properties in the object. In the example in the slide, you can see where these different items are being initiated as well. For instance, the "MSMSGS.EXE" service is being started from the HKEY->Current User->Software->Microsoft->Windows->CurrentVersion->Run registry key.

WMIC Reporting (2)

- Not always the "prettiest" output:
  - Often > 80 columns
  - Often scrolls off the screen
- There are other options:
  - Most options have a "Brief" selection.
  - All options visible through:
    - WMIC <object> list /?

One of the difficulties that you face with WMIC is that the output can be difficult to wrangle at first. Quite frequently, the output is far more than 80 columns in width, which leads to some difficult-to-read results. In addition, many of the items that you might request simply scroll right off the screen.

To help with this problem, you can use several options. However, WMIC offers help. If you want to get WMIC to do something, but you're not sure what options are available, simply type as much as you know and then add a **/?**. For example, you could type **wmic startup list /?**. This then shows you all the options available for the "list" verb in this context.

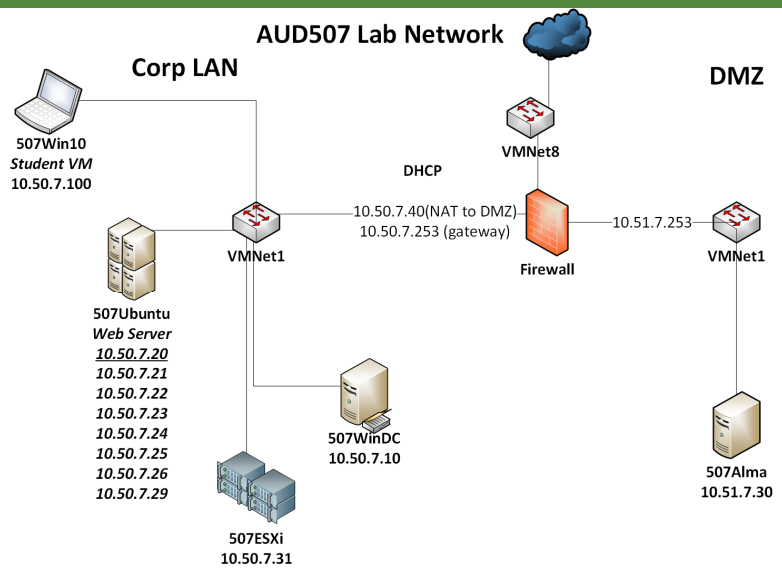
More Reporting

- Lots of options:
  - CSV
  - XML
  - Table
  - HTML Table
- Can also run through pseudo-script XSL definitions and translations

In addition to controlling the amount of output, you can also control the format of the output. The tool can easily create comma-separated values (CSV). These are wonderful for importing into Excel, XML (Extensible Markup Language, which is particularly easy to move into a database), a text-formatted table, or even an HTML-formatted table.

You're not limited to these formats, though. You can also specify an XSL (Extensible Stylesheet Language) definition file to translate and describe the format you want to produce, allowing you the flexibility to create a report in any format you like. We're going to stick with the HTML and text-based output for this now, but what you use is entirely up to you.

Exercise 2.2: Exploring WMI with PowerShell and WMIC



Follow the directions of your instructor now to begin working on the WMI exercise. As you proceed through the material, see if you can leverage any of the techniques from this section to create automated monitoring scripts!

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. Background and Plan
2. PowerShell and WMI
3. **Windows Auditing**
   - *System Information and Patching*
   - *Network and Local Services*
   - Exercise 2.3: Discovering Operating System and Patch Levels
4. Users, Groups, and Privilege Management
5. System and Resource Security
6. Windows Logging
7. Continuous Monitoring

This page intentionally left blank.

The Scenario

- You have been asked to evaluate the security of a Windows domain:
  - You know nothing at all about the domain and systems in question.
  - Where do you start?
- Start with standalone auditing:
  - Active Directory servers
  - Configuration masters

For most of the remainder of this course, we discuss (and walk through) the process of auditing a sample Windows system and then discuss how to apply this to a domain.

For purposes of this course (and to give you as much experience as possible examining a Windows host), we assume that the scope of our audit is broad. The system to be audited is completely unknown to us. Perhaps it is a system that was "inherited" from a previous administrator, and no one knows anything about the system's configuration, except for the admin who just left. Or perhaps the system belongs to a company that is just waking up to the importance of security; no one thought about it before, but suddenly (perhaps as a result of bad press, scrutiny by the shareholders, or an actual compromise on the network), management is interested in finding out whether the system is secure. We have been asked to find out as much information about the system and its current state as possible.

This gives us a great deal of room to play with in trying out different tools and techniques, but also leaves us with a bit of a conundrum: With such a broad playing field, where do we start?

The way to start an audit of this kind is to begin with a narrow scope. We'd like to get the biggest payoff for the smallest investment. From a risk perspective, this probably means that we should start with the Active Directory servers because the security of these systems impacts the security of the domain in a real way. Another possible starting point would be with the configuration masters used to deploy workstations.

Objective: System Identification

- Objective: Obtain basic information about the host you are auditing:
  - OS type: Windows 10, Server 2016, 2019…
    - Pro, Enterprise, Datacenter, 32 or 64 bit, and so on
  - OS version: Build number, Service Pack level, and such
  - System info: Uptime, registered user/company, and more
  - Basic hardware: CPU, memory, and disk
  - Partitions should be NTFS
- Purpose: Identify key aspects of the audited host

To start from the ground level, we first have to gain some basic information about the host in question. What version of Windows is it running? Windows 10? What kind of Windows 10 is it, though? Windows 10 Home? Professional? Enterprise? Is it a 64-bit install? This may sound like an obvious question, but sometimes the management requesting the audit (or even the system administrator responsible for the host) won't know this.

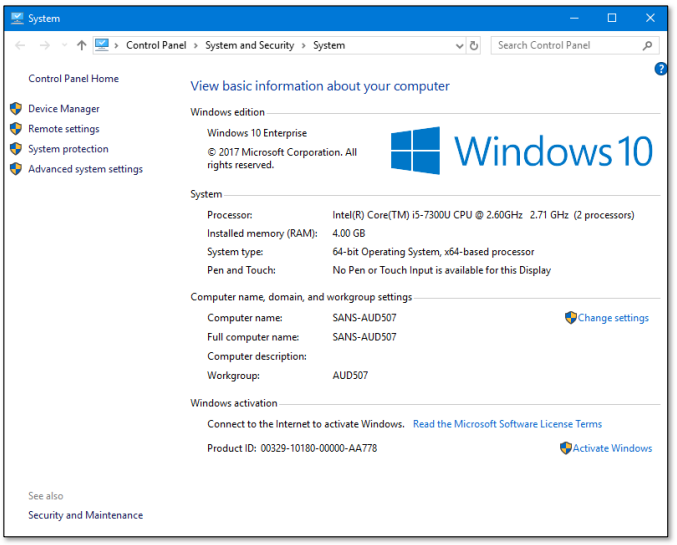
Also, knowing what flavor of Windows you are dealing with will affect which tools you use to conduct your audit. The native tools available in the OS (or their capabilities) may vary depending on the version of Windows you run.

By obtaining basic OS information, you build a minimal profile of the system you are auditing. This would include information such as the OS type and version, build (kernel) number, latest Service Pack installed, and so on. Some utilities may provide additional information, such as system uptime, the registered user and organization, and the location/name of the directory where the operating system is installed (sometimes referred to as the **systemroot**; also represented by the OS variable %systemroot% or %windir%).

You may also want to obtain basic information about the system architecture: CPU type and speed, memory, hard disks, and so on. You certainly want to determine what filesystem is in use: FAT, FAT32, or NTFS. For security and auditing purposes, you should use the NTFS filesystem because NTFS supports only the use of file- and directory-level permissions, auditing, and encryption.

Activities: Basic System Information

- Command-line Windows tools:
  - `systeminfo`
- GUI Windows tools:
  - System Information (`msinfo32`)



We can use a number of utilities to find out basic information, such as what version of the OS is running. Which one you use depends on your personal preference, and possibly on the level of access you have to the system (local versus remote; user-level access versus administrator-level access). Keep in mind that the majority of the tools we use are designed for administration, not auditing. Most of them require that you have administrator-level access to the system you examine. We don't recommend that auditors obtain administrator credentials; instead, we strongly recommend that an auditor team up with an administrator to perform all audit testing, especially in a domain or on a server.

The two most basic utilities are included in the Windows OS. These are the **ver** and **winver** commands. Both can be run from the command line, although **winver** displays its results as a GUI window. Both must run on the local machine.

The information provided by those utilities is somewhat limited. If you need additional information, you may want to use alternative tools; for example, systeminfo.exe is included with Windows XP and above. It is a command-line tool that provides extensive information about the OS, version, host, registered user, system uptime, memory, and even hotfix information.

Windows also includes the System Information utility (msinfo32.exe). This is a heavy-duty GUI tool that shows you everything from the OS version to environment variables to the filenames and versions of every Internet Explorer file on your system. In most cases, the System Summary page is sufficient for what you need!

For our purposes, we prefer tools that can be run from the command line and produce console- or file-based outputs.

Systeminfo.exe (1)

```
PS > systeminfo.exe

Host Name:                 WIN10
OS Name:                   Microsoft Windows 10 Enterprise
OS Version:                10.0.19041 N/A Build 19041
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:
Registered Organization:   AUD5x7
Product ID:                00329-10181-97955-AA622
Original Install Date:     12/28/2020, 2:05:08 PM
System Boot Time:          6/10/2021, 12:09:36 PM
System Manufacturer:       VMware, Inc.
System Model:              VMware Virtual Platform
System Type:               x64-based PC
Processor(s):              2 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 165 Stepping 2 GenuineIntel ~2400 Mhz
                           [02]: Intel64 Family 6 Model 165 Stepping 2 GenuineIntel ~2400 Mhz
BIOS Version:              Phoenix Technologies LTD 6.00, 7/22/2020
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC-06:00) Central Time (US & Canada)
Total Physical Memory:     4,095 MB
Available Physical Memory: 2,238 MB
Virtual Memory: Max Size:  6,399 MB
Virtual Memory: Available: 4,768 MB
Virtual Memory: In Use:    1,631 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\WIN10
Hotfix(s):                 6 Hotfix(s) Installed.
```


Previously, we mentioned that while WMIC is quite handy, there are a number of third-party tools that are sometimes easier to use. In some cases, we don't even have to venture afield to third-party tools! Something like systeminfo.exe is a handy tool for getting a lot of detailed information from the Windows system, as shown in this slide.

The only downside to this particular tool is that if the specific piece of information you are interested in gathering is not included in the output, there is no way to add it. For this reason, although this tool is good for a quick cursory look, we're going to recommend that you consider gathering this type of information with WMIC. Remember, if you want to gather a specific piece of information from a system for a baseline, it has little to do with your ability to process that data; it is governed by whether the collection of the data creates a negative impact on the ability to meet business objectives for which there is no commensurate need, or the ability to store that data.

Systeminfo.exe (2)

```
PS > systeminfo.exe /?

SYSTEMINFO [/S system [/U username [/P [password]]]] [/FO format] [/NH]

Description:
    This tool displays operating system configuration information for
    a local or remote machine, including service pack levels.

Parameter List:
    /S      system          Specifies the remote system to connect to.

    /U      [domain\]user   Specifies the user context under which
                            the command should execute.

    /P      [password]      Specifies the password for the given
                            user context. Prompts for input if omitted.

    /FO     format          Specifies the format in which the output
                            is to be displayed.
                            Valid values: "TABLE", "LIST", "CSV".

    /NH                     Specifies that the "Column Header" should
                            not be displayed in the output.
                            Valid only for "TABLE" and "CSV" formats.

    /?                      Displays this help message.

Examples:
    SYSTEMINFO
    SYSTEMINFO /?
    SYSTEMINFO /S system
    SYSTEMINFO /S system /U user
    SYSTEMINFO /S system /U domain\user /P password /FO TABLE
    SYSTEMINFO /S system /FO LIST
    SYSTEMINFO /S system /FO CSV /NH
```


Considering the command-line options available for the systeminfo.exe tool, you notice that we do have the ability to use this tool against remote systems. Not only that, we also have the ability to specify credentials if necessary. This can be quite useful when you deal with multiple standalone systems.

In addition, the output from this tool has some handy features for scripting and auditing. First, the output can be generated using a CSV format. When you choose this method, it displays the data using columns rather than rows. This means that each item displayed has a column header, and the data appears in the row beneath it.

This is useful, but what if you attempt to aggregate data from a large number of systems sequentially? The /NH option enables you to suppress the header output when using the CSV output mode, allowing you to append data from each system sequentially without having duplicate lines containing header information.

Questions to Ask

- OS in use: Current?
- Service pack or build number: Current?
- Fixed disk format: NTFS in use?
- Patch status: When was host last patched?
- Applications installed: Any unauthorized?

Even basic system information can tell you a great deal that is interesting from an audit standpoint. If the host runs Windows 7 in an environment where security or business concerns dictate that all hosts should use Windows 10 or later, you've already identified an audit concern. The same logic is true for Windows 10 build numbers. Simply knowing that we are running Windows 10 is not sufficient information to know if the OS is still supported by Microsoft.

What about Service Pack level? Service Packs represent major maintenance updates (security and otherwise) for Windows. If a host runs Service Pack 2 and the latest release is Service Pack 4, the host is missing many key security patches—another concern. In this same area, we need to concern ourselves with which "hotfixes" have or have not been installed between Service Packs.

If you can obtain information about the disk layout and format, determine what filesystem is in use: is it FAT32 or NTFS? Only NTFS supports the use of key security controls such as permissions, auditing, and (with NTFS v5 and higher) encryption. Hosts using FAT32 are missing these critical control features.

Some "basic information" tools may also generate a list of all installed applications. Although this list is not guaranteed to represent every application or utility installed on the host, it can provide a good general view of what's been loaded. A basic application list can help identify high-level security issues.

PowerShell OS Version

- There are no native cmdlets for querying the OS version
- Use CIM/WMI instead
  - **`Get-WMIObject Win32_OperatingSystem`**

```
PS C:\> Get-WmiObject Win32_OperatingSystem | ft

SystemDirectory     Organization BuildNumber RegisteredUser SerialNumber            Version
---------------     ------------ ----------- -------------- ------------            -------
C:\WINDOWS\system32               16299       Windows User   00329-10180-00000-AA778 10.0.16299
```


PowerShell does not currently offer native cmdlets for querying the operating system information of a host. The only alternative for now is to use the WMI/CIM cmdlets to query the Win32_OperatingSystem class. The information received will match the results from wmic os list queries.

WMIC OS

- wmic os
- wmic os list brief

```
PS > wmic os list brief
BuildNumber  Organization  RegisteredUser  SerialNumber             SystemDirectory      Version
19041        AUD5x7                        00329-10181-97955-AA622  C:\Windows\system32  10.0.19041
```


Of course, WMIC offers the ability to retrieve this information as well. The OS alias enables you to retrieve the OS version information easily without extra software.

If your computer is already up and running, why not open up a command prompt and try this command? Remember, during the exercises, we ask you to discover ways to extract precisely this sort of information.

PowerShell: Drives

- Get-PhysicalDisk
- Get-Partition
- Get-Volume (filesystem)

```
PS C:\> Get-PhysicalDisk

FriendlyName             SerialNumber MediaType CanPool OperationalStatus HealthStatus Usage         Size
------------             ------------ --------- ------- ----------------- ------------ -----         ----
VMware, VMware Virtual S              SSD       False   OK                Healthy      Auto-Select 100 GB
```

```
PS C:\> Get-Volume

DriveLetter FriendlyName FileSystemType DriveType HealthStatus OperationalStatus SizeRemaining     Size
----------- ------------ -------------- --------- ------------ ----------------- -------------     ----
D                        Unknown        CD-ROM    Healthy      Unknown                     0 B      0 B
C                        NTFS           Fixed     Healthy      OK                  80.13 GB 99.54 GB
                         NTFS           Fixed     Healthy      OK                  82.31 MB   467 MB
```


PowerShell has a few native cmdlets for gathering information about disks, partitions, and volumes on a system.

Get-PhysicalDisk returns information about disk drives installed in the system.

Get-Partition returns information about the partition table and partitions on the physical disks. This command is good to use to find out what filesystem format type (NTFS vs. FAT32) is in use on a partition.

Get-Volume gives information about the logical storage volumes Windows is using on the disks.

WMIC: Drives

- WMIC diskdrive
- WMIC logicaldisk

```
PS > wmic.exe diskdrive list brief
Caption                                 DeviceID            Model                                   Partitions  Size
VMware, VMware Virtual S SCSI Disk Device  \\.\PHYSICALDRIVE0  VMware, VMware Virtual S SCSI Disk Device  1           64420392960

PS > WMIC.exe logicaldisk list brief
DeviceID  DriveType  FreeSpace    ProviderName  Size         VolumeName
C:        3          29002514432                64422408192  Windows 10
D:        5
Z:        4          0                          0            Shared Folders
```


WMIC can also be used to retrieve information about both the physical and logical volumes on the system. It is important to look for both as an auditor because a simple way to store unauthorized data on a system without it being easily detectable is to unmount the volume. If the volume isn't mounted, or perhaps if it isn't supported by the OS directly, it will be virtually invisible. In this way, we can quickly see whether any "extra" volumes are available.

It's also important that we verify that NTFS is used for all the fixed media. The security of our Windows systems today is largely tied to NTFS permissions. If we were to format our fixed media using FAT32, we would essentially gut the security system of the operating system! Here's an example of the output:

C:\Documents and Settings\Administrator>wmic diskdrive list brief
Caption DeviceID Model Partitions Size
Virtual HDD [0] \\.\PHYSICALDRIVE0 Virtual HDD [0] 1 33550917120

C:\Documents and Settings\Administrator>wmic logicaldisk list
Access Availability BlockSize Caption Compressed ConfigManagerErrorCode ConfigManagerUserConfig Description DeviceID DriveType ErrorCleared ErrorDescription ErrorMethodology FileSystem FreeSpace InstallDate LastErrorCode MaximumComponentLength MediaType Name NumberOfBlocks PNPDeviceID PowerManagementCapabilities PowerManagementSupported ProviderName Purpose QuotasDisabled QuotasIncomplete QuotasRebuilding Size Status StatusInfo SupportsDiskQuotas SupportsFileBasedCompression VolumeName VolumeSerialNumber
C: FALSE **Local Fixed Disk** **C:** 3 **NTFS** 20739600384

Audit Objective: System Patches/Updates

- Objective: Ensure system is up to date with critical security patches
- Purpose: System updates and security patches protect the host from compromise:
  - Most compromises occur through known vulnerabilities that were simply never fixed
  - Patching is one of the easiest ways to address this problem
  - Note that unsupported software may be "unpatchable" and vulnerable by default

A key goal of any audit (but particularly a Microsoft Windows audit) is to ensure that the system is up to date with critical security patches. Installing a system securely is not enough. Systems must be monitored and maintained over time, and one of the most critical maintenance tasks that an administrator must perform is updating or patching the system.

Vendors release patches to fix bugs in the software, including security vulnerabilities. Although it is a good idea to keep your system patched and updated with all fixes, every system must be patched against all security vulnerabilities. Many of the most well-known and most damaging attacks are carried out by exploits that take advantage of *known* vulnerabilities for which patches, or workarounds, are readily available. Attackers rely on the "easy prey" of administrators who fail to patch their systems, leaving seriously vulnerable systems available on the network.

Although it is true that the window of time between the announcement of a vulnerability/release of a patch and when an exploit is first seen in the wild continues to shrink, the patches are there and getting them on the hosts remains one of the best ways to protect your hosts and networks. Patch status should, therefore, be a key item of interest in any system audit.

Microsoft's support policy may change periodically, so you should check Microsoft's website for the most current information about end-of-life products. When a product has reached end-of-life, there will be no future security patches, even when vulnerabilities are discovered.

Audit Tools: Patch Status

- Patch management tools:
  - Microsoft WSUS
  - Microsoft SCCM
  - Patchlink (now Ivanti)
  - Shavlik (now Ivanti)
  - IBM BigFix

A number of tools are available that can check (and, in some cases, maintain) patch level.

Many third-party tools (vulnerability scanners, audit tools, and management software) also include checks for system patch status. The **Microsoft Baseline Security Analyzer (MBSA)** was a freely available graphical utility from Microsoft that could perform several basic security checks on Windows systems and selected Windows applications, such as IIS, SQL Server, and Internet Explorer. Unfortunately, Microsoft is no longer updating MBSA and there is no ready replacement for it.

There are also tools available from both Microsoft and third-party vendors that perform patch checking and patch management across multiple systems. Microsoft has two offerings that are of particular interest. The first is Windows Server Update Service (WSUS), which is a role that you simply need to enable on Windows Server. Although it is officially limited to managing patching of Microsoft products only, it is a great deal better than nothing! Reportedly, if you are willing to invest significant time and effort, you can actually script it to apply third-party patches as well, although we rarely see this done in practice.

Microsoft's second offering is System Center Configuration Manager. It can be leveraged to enforce patching for Microsoft and third-party software within managed systems in a domain.

If the organization has a central patch-management solution, we'd like to leverage its reporting capabilities during any audit of the security of a domain. We'll come back to this idea in a moment.

Types of Windows Patches

- Service Packs:
  - Major updates that roll up previous security and non-security patches; may include new features
  - Replaced with builds in Windows 10
- Hotfixes or Critical Updates:
  - Fix for single critical issue affecting security or system stability
- QFE fixes:
  - Interim fix for a single, specific issue; usually available only from Microsoft Support

Before we discuss various tools that can help you verify patch status, we need to talk a bit about Microsoft's patch process. Different vendors have different policies for releasing patches and different methods for installing patches. Microsoft issues three different types of patches:

- **Service Packs:** Service Packs are released for Windows operating systems and major applications, such as Microsoft Office or Exchange Server. These are major updates that act as a "rollup" of all (actually, most) patches issued to date. They may also include enhancements or add-on components to the original software. Service Packs are released publicly and may contain hundreds of fixes, from security patches to minor bug fixes. They are usually released every 6–12 months. Service Packs go through thorough testing prior to release and come with extensive documentation regarding the fixes included in the Service Pack, any known issues, and so on. Microsoft is using semi-annual builds of Windows 10 to replace the old service pack function.

- **Hotfixes (sometimes called critical updates):** Hotfixes are patches intended to address a single issue or a few related issues. Hotfixes are released publicly and address critical security vulnerabilities or severe system problems. They go through some degree of testing but are not as thoroughly tested as Service Packs.

- **QFE fixes:** Quick Fix Engineering (QFE) fixes are intended to address a single specialized issue—either a widespread but minor problem, or a problem that affects only certain systems (based on interaction with another application or device driver, for example). QFE fixes do not undergo extensive testing; they are literally a "quick fix" designed to temporarily address a specific problem for those who may be experiencing that problem. As such, they are generally not publicly released. Instead, Microsoft publishes a Knowledge Base article describing the problem and the fix.

Unless there is a business reason to *not* patch a system, ideally hosts should be kept up to date with all security patches within 7–14 days of their release.

How Do Systems Get Patched?

- Install hotfix:
  - Automatic Updates, Windows Update website, patch management software, manual install, and so forth
- During install:
  - Updated binaries copied to disk
  - Registry entry written to show hotfix is installed
  - Uninstall directory may be written to %windir%
- Usually requires reboot of system

Installing a Service Pack or hotfix on a Windows system is straightforward. You run the hotfix executable or select the patches to install from an update site such as Windows Update or use some type of patch management tool to distribute the patch to your hosts. Whatever method is used to install the patch, the updated binaries are copied to your disk, and Windows generally writes an entry in the registry to show that the hotfix has been installed. Windows may also create an uninstall directory under the %windir% directory. (**Note:** %windir% is a variable used to represent the directory where the Windows operating system is installed. In most cases, this will be C:\winnt or C:\windows. You may sometimes see the variable %systemroot% used to represent the same thing.)

The uninstall directory will be compressed (to save space) and will be named with a dollar sign ($) and the number of the Knowledge Base (KB) article relating to the problem/fix (for example, C:\winnt\$NtUninstallQ329048$). The uninstall directory contains the original binaries (the ones replaced by the hotfix) and instructions for uninstalling the patch.

Finally, in most cases, Windows requires a reboot after you install the patch. A reboot may be required for any number of reasons:

- The patch needs to replace binaries that are locked and in use by the operating system. A reboot is needed to "release" the files so that the new versions can be copied into place.
- The patch needs to modify a portion of the registry that is locked and in use by the operating system. Again, a reboot is needed to allow the changes to take place.
- The patch has modified files or registry entries that will not take effect until the system reinitializes during a reboot.

The need to reboot the system causes some difficulties with patch management. For example, if you have to reboot the system every time you install a patch, this means that the system cannot be patched without at least minimal downtime, even if it is only five minutes. This may make it difficult to patch mission-critical servers.

Difficulties with Patch Validation

- Different utilities give you different results
- Depends on how the patch level is checked:
  - Check for registry entry?
  - Check file version/checksum?
  - Both?
- Can always manually check:
  - Based on information in KB article

If installing patches is relatively simple, why is it so hard to determine the patch level of a system? Unfortunately, it is not always easy to determine whether your system is "secure" and whether your patch level is up to date. There are any number of utilities that can be used to check your patch level. Many of them give you different results.

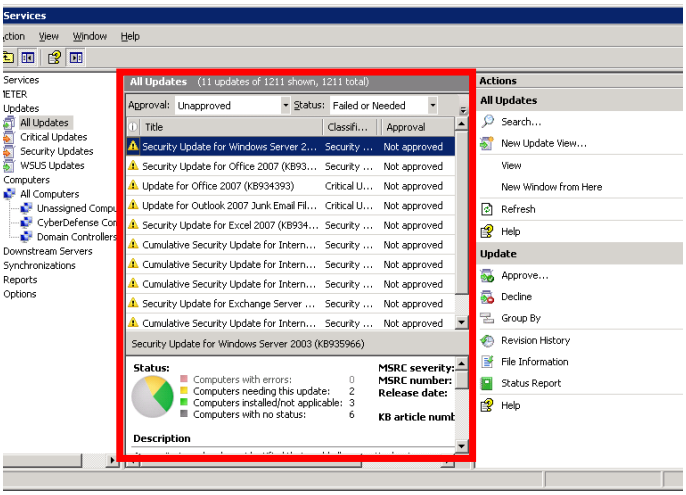
Part of this problem is due to conflicting or confusing information provided by Microsoft. When a security-related problem is discovered, Microsoft issues a security bulletin with a numbering scheme similar to MS18-038 (where "19" is the last two digits of the current year, and "038" means the 38th security bulletin issued this year). If you monitor the Microsoft Security Bulletin mailing list, you'll be used to seeing and working with those numbers. However, if you use Windows Update or almost any other method to patch your system, you won't see those numbers; instead, Microsoft uses a six-digit Knowledge Base (KB) article number such as 810030. Figuring out how the KB number relates to the Security Bulletin number is not always easy, so it is not always obvious which problems you've patched.

Another issue is that after a patch is released, any vendor that provides patch-management or patch-checking software (including Microsoft) must update their patch database to include information about the latest patch. Depending on how quickly your vendor responds, it may take anywhere from a few days to a week or more for a new patch to make it into your vendor's database.

Most patch-related KB articles include a list of the specific filenames, dates, and versions that are installed by the patch. In a worst-case scenario, you could use the information referenced in the KB article to manually check the files, although this is tedious and time-consuming and not ideal for conducting an audit. However, it is a way to be quite sure about what's installed.

What's Available?

- Management solution takes care of figuring out what's available:
  - Allows for testing
  - Phased rollout
  - Easily find out the oldest patches that have not been applied



Managing audit data for patch management manually or through scripted tools is possible, but it is easier to use something such as WSUS. WSUS, which has undergone a number of name changes, is simply an added server role in Server 2008 R2 and higher. It not only provides a handy way to roll patches out in a controlled manner after testing but also provides an interface for keeping track of which patches have been applied where. Although we're not saying that you should scrap your current patch management to use WSUS, if you do not have a patch management product currently, or if what you have is not working for you, you should definitely have a look at it. Although I would push an organization to something more "Enterprise Grade" such as Microsoft's SCCM (the replacement for SMS), Patchlink, or BigFix, if you have nothing now for patch management, then you should consider WSUS at a minimum.

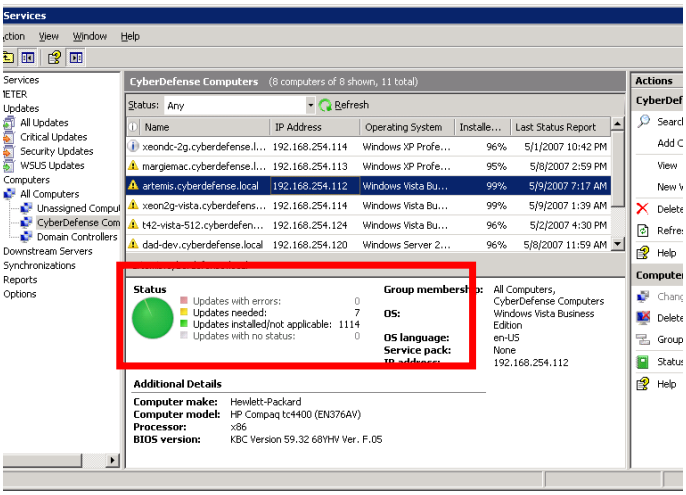
In the screenshot, you can see the MMC console for managing the patches that should be applied within the domain. This screenshot was taken on a Tuesday morning after the WSUS server had synchronized with Microsoft to obtain the most current patches.

When patches are approved, they can be approved for installation into specific computer groups. This means that the administrators could do their testing by creating groups that contain test systems or early adopters. After fully testing, the update could be approved again, this time for larger distribution. If there are systems that cannot work with the patch, they could be placed into a group that indicates this and to which the patch is not released.

Before we continue, an important fact to note is that WSUS is designed to support patching Microsoft products only. We don't mean that it can't patch Linux systems. Obviously, this is true. What we mean is that it can patch Microsoft Office, but it absolutely cannot patch Adobe Acrobat!

What's Needed?

- Management software doesn't have to wonder what's installed:
  - It was there when the installation happened
  - Knows if it failed
  - Knows what's missing



The other particularly useful thing about WSUS is the capability to quickly obtain a report showing how a specific computer is patched, how a group of computers is patched, or how all the computers in the domain are patched. The report indicates high-level statistics regarding the types of patches missing or applied, but you can also drill into the report to get details of exactly which patches have been applied, have not been applied, or have failed to apply.

This can be useful, especially in the context of our discussion regarding continuous vulnerability/risk remediation. If the patch-management process requires that all relevant patches are applied within 30 days of release, we can easily create a custom report that shows us only systems with missing patches that were released more than 30 days ago.

A key point with regard to Windows patch management is that centralized patch management is effectively a requirement. Without it, it can be very difficult to determine which patches have been applied or are missing at scale. When centralized patch management is used, it *knows* which patches are applied because *it was there when they were applied.* It also knows when patches fail, obviously.

There is still a need to validate that the solution in use is properly deployed and accurately reporting. This can be accomplished through sampling and spot checks.

Inquire about Process

- It's not simply a matter of ensuring things are patched:
  - When was it released? How was it tested? What was the back-out strategy?
- Example Process: 96 hours to patch:
  - 24 hours, applied to "power users"
  - 48 hours, applied to noncritical servers
  - 72 hours, applied to all users
  - 96 hours, applied to all servers

As I'm sure you realize, it's not simply a matter of determining whether the systems are patched. Of course, if systems are not patched, that's a serious issue, and we must determine what is breaking down in the process that's allowing a system to go unpatched without an accepted risk exception or other compensating control.

What if everything is patched? We'd recommend that you increase the difficulty of the audit slightly and ask deeper questions about the processes. Start to determine the mean time to patch within the organization. How long does it take, on average, for a patch to be fully applied to relevant systems after its release?

You also want to inquire about the back-out strategy and testing process. This is where most organizations that are patching fall down. One way that a client of mine has addressed this is by creating a testing group within the enterprise.

Within each department, whether "official" or not, there is an IS contact. This is the person who people typically go to before they call the help desk. These individuals represent a cross-section of the enterprise from a functional level. They have been invited to participate as testers and, for this duty, they are given elevated rights on the systems that they work with. Whenever a new patch is released, the administrators have 24 hours to test it out and develop a back-out process. Clearly, this won't be thorough, but even with 30 days it is unlikely to be completely thorough. At this point, it is released to the testing group. At 24-hour increments, the patch is released to a larger and larger set of systems until the entire enterprise is patched within 96 hours, unless something exceptional occurs, triggering a second set of criteria.

Installed Patches

- PowerShell Get-Hotfix
- WMIC QFE list

```
PS C:\> Get-HotFix

Source        Description      HotFixID      InstalledBy          InstalledOn
------        -----------      --------      -----------          -----------
SANS-AUD507   Update           KB2693643     NT AUTHORITY\SYSTEM  11/17/2018 12:00:00 AM
SANS-AUD507   Security Update  KB4053577     NT AUTHORITY\SYSTEM  12/14/2017 12:00:00 AM
SANS-AUD507   Update           KB4073120     SANS-AUD507\student  12/1/2018 12:00:00 AM
SANS-AUD507   Security Update  KB4074595     NT AUTHORITY\SYSTEM  2/14/2018 12:00:00 AM
SANS-AUD507   Update           KB4078408     NT AUTHORITY\SYSTEM  2/14/2018 12:00:00 AM
SANS-AUD507   Security Update  KB4074588     NT AUTHORITY\SYSTEM  2/14/2018 12:00:00 AM
```


PowerShell's Get-Hotfix command will return a list of installed patches on a system. Because the object returned includes an "installedOn" property, you can query to find only the patches installed within the current patch window. If your workstation patch window is 30 days, you could query all patches installed in the last 30 days with this command:

```
Get-HotFix | Where-Object { $_.InstalledOn -gt (Get-Date).AddDays(-30) }
```

The unfortunately named WMIC alias QFE also returns a list of installed patches on a system. Start with the command:

```
wmic qfe list brief
```

Installed Applications

- To find installed applications, use:
  - Get-WMIObject Win32_Product
  - wmic product list brief
- For a more definitive answer, search the HKLM and HKCU hives of the registry for "uninstallPath" keys
  - Get-ChildItem -Path HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall

```
PS C:\> Get-WmiObject win32_product | Sort-Object Name | ft -Property Name, Version, InstallDate

Name                                                   Version        InstallDate
----                                                   -------        -----------
Citrix XenCenter                                       7.4.0          20190216
Google Chrome                                          72.0.3626.109  20181130
```


Finding all of the software installed on a system can be a bit of a hit-or-miss proposition. There's not a single source of this data on a Windows system. The tools mentioned on this slide use the WMI product class, which is a list of everything installed or managed by the Microsoft Installer service. This list would be very similar to what would have shown up in the old "Add or Remove Programs" control panel in older versions of Windows.

Since not every application installs using the MSI service, another method would be to use PowerShell to iterate through the HKLM and HKCU registry hives looking for settings from installed software.

Audit Interview Questions

- Change control policy?
  - How much testing is preferred/required before deploying patches?
- Scheduled maintenance?
  - Does the site have a regular maintenance schedule? How often does it occur?
- Compliance policy?
  - Is your organization required to install certain patches, or maintain a particular patch level? Are you compliant with this policy?
- Exception policy?
  - When is it okay not to patch?

Finally, we have a brief mention of some of the non-technical issues related to patching and patch management. You may want to address these during your audit as well.

Issues include the following:

- **Change control policy:** Although it is important to keep hosts patched for known security vulnerabilities, it is also important to ensure that a new patch will not introduce other, unexpected vulnerabilities and that it will not break any critical applications. Are there policies or procedures relating to testing patches prior to deployment? How does the organization balance the need to patch with the need to test?

- **Scheduled maintenance:** Patches are issued on a regular basis, particularly for Microsoft products. Are there policies or procedures related to scheduled maintenance for systems on the network? This includes "low-impact" systems like user workstations as well as "mission-critical" systems such as key business servers, which often go unpatched because no one is willing to take them down for maintenance.

- **Compliance policy:** Is the organization required by policy, regulation, or law to maintain a certain patch level? This may be an explicit policy requiring the installation of specific patches, such as the US Department of Defense Information Assurance Vulnerability Alert (IAVA) policies, or a more broad "due diligence / best practices" clause regarding the security or privacy of information, systems, or networks (for example, in legal or regulatory statutes such as HIPAA, FISMA, and others).

- **Exception policy:** Are there policies or procedures relating to exceptions to an existing patch policy? Are there situations in which it is okay not to patch? Who makes these decisions and assumes the risk in such cases?

Network and Local Services

- Ensure only necessary features of OS are installed/running
- Audit tools:
  - Services: sc, tasklist, psservice, Get-Service, Get-Process
  - Ports: nmap, fport, netstat, Get-NetTCPConnection

Another key objective in your audit will be to determine which services and components are installed on a given Windows host, and which ones are essential to the system's operation. Any that are not essential should be uninstalled or disabled. Think of this as the principle of least privilege for your computer; if the system does not serve up webpages, there is no need for it to be running a web server.

When examining components and services, we want to examine the system from a few different angles. First, we'll want to identify which services are "listening," or willing to accept connections from other hosts on the network. Because these services are externally accessible (available on the network, though access to those ports may be filtered by firewalls or routers), they present a higher risk. However, we also want to look at the overall list of services installed on the system. Microsoft (like many vendors) ships its OS with a number of services, many of which are unnecessary or even risky. Identifying these services allows you to remove, disable, or otherwise take steps to mitigate risk to the system.

Unneeded Services

- Many services installed by default are not required for operation:
  - Example: IIS, SMTP, Messenger, and such
- Services may contain vulnerabilities
- Unused services unlikely to be patched
- Rogue services may indicate malware infection:
  - Know your systems

The default installation of most operating systems and applications includes numerous components (and associated services) that are not needed except in certain operating environments. The reason for this is that the majority of vendors are not concerned with their operating system (or application) being secure out-of-the-box. They are more concerned with ease of use: Convenience for the administrator or end user. Unfortunately, this means that services, components, or other "features" that contain vulnerabilities may be installed by default—often without the administrator's knowledge. Some components may contain inherent vulnerabilities (that is, the Simple Network Management Protocol [SNMP]). Others may include features that allow them to be made secure, but these features are not enabled by default. Finally, some components may introduce vulnerabilities in the form of bugs that are discovered later (for example, Internet Information Server). If the administrator is not aware that the components are installed, he is unlikely to patch any vulnerabilities that are discovered in those components.

A related scenario is the administrator who carefully reviews the installation option but chooses to install numerous services or components that she is not using now but "may" use sometime in the future. The idea is to have the service available "just in case" (and to avoid having to go back and install it later). Again, these components may contain vulnerabilities, and if the components are not used, they are also not likely to be patched.

Every running service is a potential hole into your system for an attacker to use. If a service is disabled or not installed, that is one less door into your host. Disabling unused services may also improve your system's performance, as there will be fewer background processes running on the host. As an auditor, one of the key issues you should look for is whether the audited host is running only necessary services and components.

One of the key ideas taught in nearly every security course that SANS offers in addition to being a critical knowledge point in the 20 Critical Controls is that we must *know our systems.* Baselines provide a key piece of data that helps us know our systems, and knowing which services are running is absolutely part of that!

How Do I Check Services?

- Listening services (open ports):
  - "Outside": From an external host (port scanner)
  - "Inside": From on the host itself (netstat)
- Good to do both to correlate results:
  - Outside scan may be more trustworthy
  - Local services may not be "visible" to outsiders
- Full list of services:
  - Services MMC
  - psservice.exe, sc.exe, tlist/tasklist
  - wmic (of course!)

When auditing which applications and services are running on a host, it's important to do so from two perspectives. First, you want to determine which services are "listening" on the host, ready and waiting for outside users to connect. These are the services that provide a potential means for someone to remotely connect to your host. Because they are potentially accessible to outsiders, they present a higher level of risk for external attack (or infection by a network worm).

Recall that in TCP/IP networking, connections are defined by both IP addresses (source and destination) and ports: A 16-bit number between 0 and 65535. Services willing to accept connections from the network "listen" on a particular port. Determining which ports are "listening" can give you a good idea of the services running on that host. Port scanners are used for this purpose.

After you have enumerated the listening ports, you need to determine the services associated with them. By convention, commonly used services (web, mail, telnet, and such) always use the same "well-known" port. Note that this information is not foolproof; just because port 80 is listening on a system does *not* mean that a web server is running. Though commonly used services "should" run their associated well-known ports, they are not required to. We can map the port to the executable or the process using the netstat tool.

In addition to a list of ports and their associated services, you also want to know the full list of services installed on a host. Regardless of whether they listen for external connections, many Windows services are unnecessary and can be disabled. And some viruses, worms, or other malicious code attempt to install themselves as a service to ensure that they re-activate even after a system reboot. So, your first priority is identifying all services and determining which (if any) are unknown or unnecessary. Various Microsoft and third-party tools, such as Sysinternals' psservice.exe and Windows' sc.exe, can provide us with this information.

Port List Comparison

Inside

```
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:88             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:389            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:464            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:593            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:636            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3268           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3269           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5722           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49152          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49153          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49154          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49155          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49157          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49158          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49161          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49171          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49194          0.0.0.0:0              LISTENING
  TCP    10.0.1.20:53           0.0.0.0:0              LISTENING
  TCP    10.0.1.20:139          0.0.0.0:0              LISTENING
  TCP    10.0.1.20:3389         10.0.1.54:58208        ESTABLISHED
  TCP    10.0.1.20:5722         10.0.1.6:61333         ESTABLISHED
  TCP    10.0.1.20:64538        10.0.1.20:135          TIME_WAIT
  TCP    10.0.1.20:64540        10.0.1.20:135          TIME_WAIT
  TCP    10.0.1.20:64543        10.0.1.20:135          TIME_WAIT
  TCP    127.0.0.1:53           0.0.0.0:0              LISTENING
```

Outside

```
Starting Nmap 5.21 ( http://nmap.org ) at 2010-12-06 10:53
Nmap scan report for enclave-dc-1 (10.0.1.20)
Host is up (0.00024s latency).
Not shown: 65513 closed ports
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
3389/tcp  open  ms-term-serv
5722/tcp  open  unknown
5900/tcp  open  unknown
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49157/tcp open  unknown
49158/tcp open  unknown
49161/tcp open  unknown
49171/tcp open  unknown
49194/tcp open  unknown
```


Just as an example of the concept of comparing an inside versus an outside picture of the network services, consider this screenshot. On the left side, you can see the results of running netstat –an -p tcp. This displays all the listening TCP ports from the localhost. On the right side, you can see the results of running nmap –n –p 1-65535 against the same server remotely.

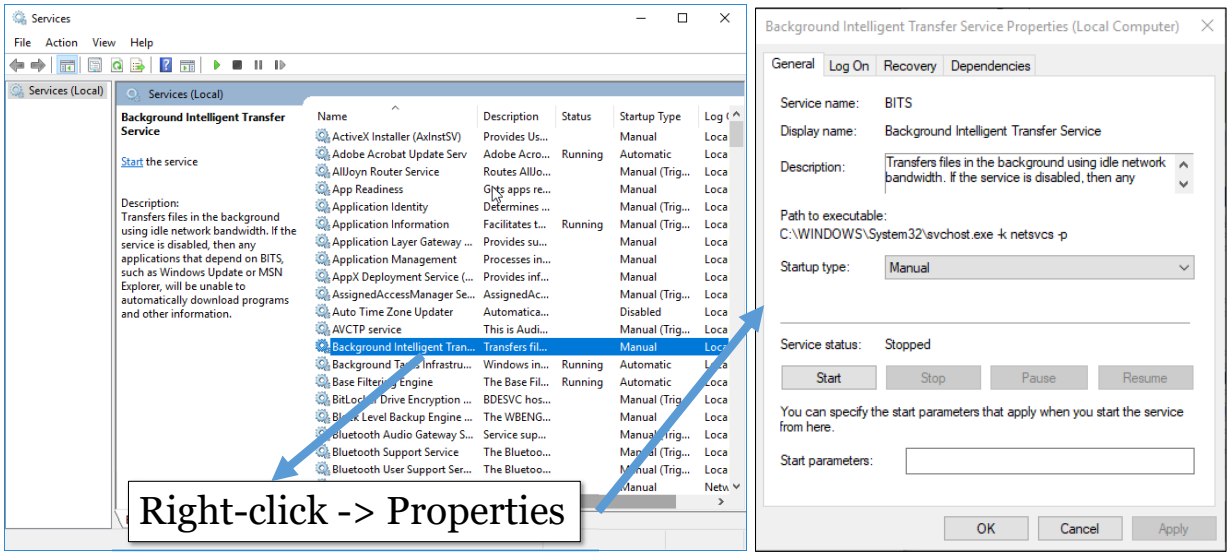
Comparing the results does take just a bit of effort because the ports display in different orders (what a perfect task for a script!), yet we can quickly realize that a port is listening from the outside that is not visible from the inside. For the purposes of the class, we have highlighted the abnormal port in red: Port 5900/tcp. A bit of research reveals that this is typically the port used for VNC, a piece of remote-control software. The biggest question, however, is why doesn't this port show up from the inside? This appears to indicate that the machine considered has been compromised or infected in some way.

Lists of All Services

- Any risky services installed?
  - Hard to determine - Consider using the baseline image to determine what's normal, and then spend research time on services beyond that list
- Regardless of port usage, some services may be vulnerable/risky
- Malicious code frequently installs itself as a service:
  - Useful for identifying infection/Trojan
  - Can only do this if you have a baseline of what belongs on the system

Port scanners and tools such as FPort and Openports focus on ports that "listen" for connections on the network stack. Other services may provide only some function to the local operating system, but can still be risky or unnecessary, or both. In addition, malicious code (viruses, Trojans, bots, and more) sometimes installs itself as a service, attempting to masquerade as a legitimate Windows process. In these cases, you want an audit tool that can provide you with a list of all services installed on a given host.

Tool: Services MMC



The Services MMC is the most direct method to view the list of installed services and their status. However, because it is a graphical utility, it can be cumbersome to work with from an audit standpoint. (That is, it's not easy to record the status of the services from the GUI, short of performing a screen capture.) However, this utility can "drill down" into a service's configuration, if necessary, to see a service's dependencies or to view/change the account used to run the service.

Tools: psservice / sc

- psservice from Microsoft
  - Part of PSTools
  - Previously from Sysinternals/Winternals
- sc.exe (Service Control)
- Command-line list of all installed services and their states
- Local or remote host

```
SERVICE_NAME: Eventlog
DISPLAY_NAME: Event Log
Enables event log messages issued by
   Windows-based programs and components
   to be viewed in Event Viewer. This
   service cannot be stopped.
        GROUP              : Event log
        TYPE               : 20
   WIN32_SHARE_PROCESS
        STATE              : 4  RUNNING
   (NOT_STOPPABLE,NOT_PAUSABLE,ACCEPTS_SH
   UTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```


While the Services MMC provides a great deal of detail in a convenient format, it is not very practical from an audit standpoint. A more useful tool would run from the command line (so it could be scripted, if necessary), would provide detailed information about the services installed, and could be run against either a local or remote host.

Fortunately, there are tools that meet these criteria: Microsoft's sc.exe (Service Controller) and Sysinternals' psservice.exe (from the PSTools package).

The Service Control utility (sc.exe) has been included since Windows XP and is a highly flexible utility that can be used to start, stop, pause, query, create, and delete services. It provides a great deal of service data, and the fact that it is included with all modern versions of Windows makes it a useful audit tool. See the command-line help for the full range of options available with sc.exe; for auditing purposes, the “query” and “queryex” (query extended information) command-line switches are the most useful.

A comparable utility is the psservice.exe tool that comes with Sysinternals' PSTools suite. Psservice can perform tasks similar to sc.exe, with the exception of creating and deleting services and some queries. However, psservice has the advantage that it can be run against both local and remote hosts.

While both tools provide nearly identical output, the details vary slightly. For example, psservice includes the detailed description of each service by default, but sc does not. Alternately, sc with the queryex option includes the process ID and additional status flags for the service that are not displayed by psservice.

PowerShell: Services and Processes

- Get-Service returns information about installed services and their current status
- Get-Process returns information about running processes on the system

```
PS C:\> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)      Id  SI ProcessName
-------  ------    -----      -----     ------      --  -- -----------
    361      21    10844      26888       0.13    2848   1 ApplicationFrameHost
     42       4     2312       2988       0.02    6568   1 cmd
    219      14     7900      23316       1.64    1048   1 conhost
    188      11     8184      14760       0.22    4248   1 conhost
    215      13     6224      17924       0.05    6836   1 conhost
    394      13     1576       4524       0.81     396   0 csrss
    367      17     1744       5028       4.50     500   1 csrss
```


Services and processes are easy to query and administer with PowerShell. Get-Service can be used to query information about all services on a machine, whether they are enabled and running or not.

Get-Process returns information about running processes on a system.

The information returned by both of the commands could be very useful in baselining a system. Remember the script you analyzed in the first lab? It was using the output of Get-Service as a baseline test.

Tools: wmic (1)

- wmic service list brief

```
C:\Documents and Settings\Administrator>wmic service list brief
```

ExitCode	Name	ProcessId	StartMode	State	Status
1077	Alerter	0	Disabled	Stopped	OK
0	ALG	1060	Manual	Running	OK
1077	AppMgmt	0	Manual	Stopped	OK
0	AudioSrv	1972	Auto	Running	OK
1077	BITS	0	Manual	Stopped	OK
0	Browser	1972	Auto	Running	OK
1077	CiSvc	0	Manual	Stopped	OK
1077	ClipSrv	0	Disabled	Stopped	OK
0	cohrence	1936	Auto	Running	OK

Of course, you can also gather information on the status of services using the WMIC interface as well. The WMIC interface can be used to start, stop, change the startup mode, etc. For our purposes, the ability to easily and quickly list the installed services with the startup status creates a very useful piece of our baseline. Later, we'll take a look at how we can pull this information out of large numbers of machines easily within a domain, in order to verify that the systems are configured to match the configuration master from which they were made.

Tools: tasklist

- Command-line list of running processes:
  - Allows remote queries
  - Tasklist /S system

```
PS C:\> tasklist

Image Name                     PID Session Name        Session#    Mem Usage
========================= ======== ================ =========== ============
System Idle Process              0 Services                   0          8 K
System                           4 Services                   0         92 K
smss.exe                       292 Services                   0        864 K
csrss.exe                      396 Services                   0      4,516 K
wininit.exe                    484 Services                   0      5,924 K
csrss.exe                      500 Console                    1      5,028 K
winlogon.exe                   580 Console                    1      8,532 K
services.exe                   616 Services                   0      8,216 K
```


The tasklist utility is included in base Windows installations. Tasklist is primarily intended as a sort of command-line version of Task Manager and can be used to provide a list of running processes on a given host. Without the "/S" option it will list processes on the localhost. The "/S" option allows you to retrieve a process list from a remote host as well.

One of the most useful features of tasklist is the ability to list the specific Windows services that are running within a given process. Some processes on Windows—most notably svchost.exe, but also lsass.exe and others—are generic processes that are used to run multiple services. It can be useful to know what services are being controlled by a given process.

In addition, malicious code such as viruses, worms, and other tools frequently attempt to hide their presence on a Windows system by using an innocuous process name—such as svchost.exe. If tasklist shows a service process (such as svchost.exe) that does not include a list of services associated with it, you probably want to investigate that host a bit further!

Another easy way to figure out what a particular svchost invocation is doing is to bring up task manager, find an svchost that you'd like to examine, right-click on it, and select "Goto Service(s)." It will now highlight all of the services that the svchost selected has spawned.

Tools: wmic (2)

- wmic process list brief:
  - How can this be used for detecting compromise?
  - Could you determine which processes *must* be running?
  - Would it be valuable to alert if one or more of them disappears?

```
PS C:\> wmic process list brief
HandleCount  Name                  Priority  ProcessId  ThreadCount  WorkingSetSize
0            System Idle Process   0         0          2            8192
1958         System                8         4          101          94208
52           smss.exe              11        292        2            884736
387          csrss.exe             13        396        10           4624384
146          wininit.exe           13        484        1            6066176
371          csrss.exe             13        500        13           5148672
```


WMIC can also be used to extract information on running processes. Like tasklist, this can be used remotely against any host where we have appropriate credentials within the domain.

Please remember our purpose in offering you multiple tools to solve a given problem; certain tools will lend themselves to specific purposes and installations. We are trying to give you multiple options so that when you need to retrieve a piece of information or need to write a script to automate monitoring or auditing, you have the largest number of tools in your toolbox. This way, when you need a wrench, not only will you have a number of wrenches, but you'll probably have at least one wrench that's a perfect fit!

You may wonder how something like tasklist or process list could ever be used in a security audit or with a baseline. To explain, let's first talk about how to create the baseline.

Rather than being interested in all of the processes that are running at any given time, we are much more interested in the *required* processes for a system. To determine this, image a system and then baseline which processes are running. These are the processes that you will expect to find on *every* system, though the PIDs will, of course, change.

Armed with this list, an administrator would periodically—preferably daily or perhaps more often—sweep the entire domain, checking to see if any of these processes are *missing* from domain systems. We've found in many, many cases that malware will either kill or otherwise hide some of these default processes!

While the administrator will not know what the malware is, there is a clear sign that a very significant change in behavior has occurred. If rebooting the system does not clear it up, the best course of action is to treat it as compromised!!

Advanced Audit / Continuous Monitoring Question

- What if the administrators created a baseline of which services *must* be running on a newly imaged system?
  - Antivirus, endpoint security, LSASS, SMSS, CSRSS, svchost, etc.
- How difficult would it be to create a script that:
  - Queries the currently running processes from a remote system?
  - Compares the list of processes to a list of *required* processes?
  - Generates a notification if anything is *missing*?

Let's pause our audit research for a moment and stand back. There's a really cool continuous monitoring system that can be designed with what we know so far, and it probably adds a detection capability that your organization does not have currently.

Imagine that the administrator takes some time to create a baseline of the processes that are running on a newly imaged system. More specifically, he identifies precisely which processes *must* be running, and possibly the number of instances of each. Armed with this information, he now sets out to write a script.

The script remotely queries a workstation using WMI, obtaining a list of all running processes. Once the list has been obtained, the script verifies that each required process is actually running, possibly verifying the number of copies of each of the processes as well. What if a process is missing?

Think about how a process could go missing. You would have to be an administrator on the local system to stop one of these processes. Malware frequently makes changes here, either adding or removing some of these processes in an effort for self-preservation. If the script notes anything is missing, it can generate a notification.

This would give us the ability to detect any significant event that causes critical required processes to terminate on a workstation. Could this be zero-day malware? Certainly. Could it be a user who is engaged in unauthorized activity? Absolutely. Regardless of what the cause turns out to be, it gives us a method for detection.

What's really awesome about this approach is that this script could be automated to run periodically against every workstation in our domain! It will have next to no impact on the systems or network and gives us an instant detective capability that goes beyond standard malware signature detection. What does this cost? Just a small amount of time.

Audit Interview Questions

- Are there installation and configuration processes or standards for Windows systems?
- Is there any policy describing allowed or disallowed services?
- Are system administrators familiar with the standard services and ports that should be present on their systems?
- Are periodic checks performed to detect new or changed ports or services?

Again, although the focus of this course is on the technical aspects of auditing Windows, we include a few periodic reminders of some of the policy issues you should consider.

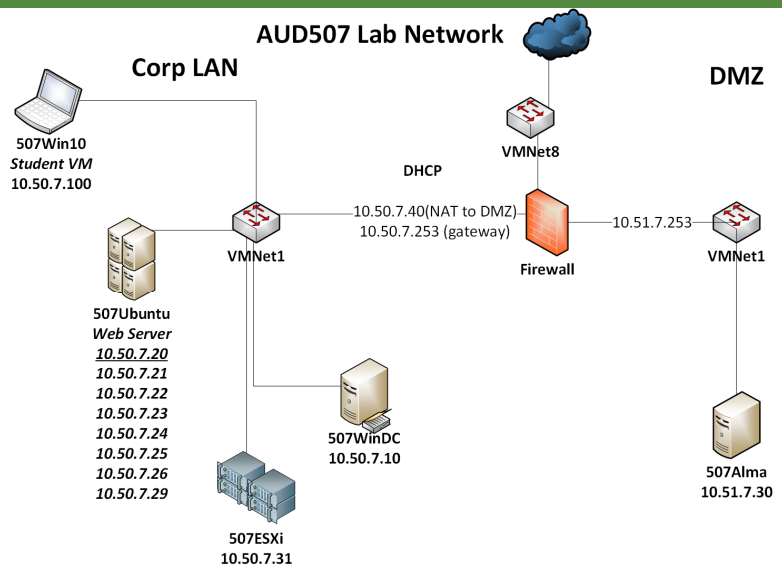
Are there installation and configuration processes or standards for Windows systems? Consistency in installing and configuring Windows ensures that after you determine a "secure" configuration for your environment, that configuration is used throughout your organization. Repeatable (or preferably, automated) procedures can further this goal.

Is there any policy describing allowed or disallowed services? Policy prohibiting risky services (for example, Telnet or SNMP) provides written guidance for what is acceptable within an organization.

Are system administrators familiar with the standard services and ports that should be present on their systems? As the people who (in theory) work most closely with systems on the network, system administrators are in many ways the best people to be on the lookout for something out of the ordinary that may indicate a problem (such as a new service or listening port that indicates a Trojan has been installed on the host). Yet it is surprising how often sysadmins don't know what "should" be running on a host or cannot determine whether a given process is "legitimate." Although system administrators can't know every service on every port on every host, they should at least have some idea of what is "normally" there.

Are periodic checks performed to detect new or changed ports or services? A new port or service could indicate that a new application has been installed, or it could indicate the presence of malware on the host. Part of good security is "auditing" (in the sense of monitoring over time) a host or network to detect changes to a previously "known good" baseline. Is this type of monitoring performed? This could include periodically port-scanning a host for new ports, or "diff"-ing the output of a particular utility run at two different points in time.

Exercise 2.3: Discovering Operating System and Patch Levels



Please open your workbook to Exercise 2.3: Discovering Operating System and Patch Levels and follow the instructions to complete the exercise.

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. Background and Plan
2. PowerShell and WMI
3. Windows Auditing
4. **Users, Groups, and Privilege Management**
   - *User Management*
   - *Querying Active Directory*
   - *Group Membership*
   - *Password Assessments*
   - Exercise 2.4: Querying Active Directory
5. System and Resource Security
6. Windows Logging
7. Continuous Monitoring

This page intentionally left blank.

Users Management Considerations

- Any operating system:
  - Only valid users on the system
  - Groups have appropriate membership
  - No blank passwords
  - Reasonable password policy
  - "Strong" passwords in use
- OS-specific:
  - Local accounts versus domain accounts

The next step in our general operating system audit process is to evaluate the users and groups present on a system. There are a number of security issues related to users and groups, including the following:

- **Only valid users on the system:** User accounts should be enumerated and reviewed to ensure that all accounts present belong to valid system users. This includes "special-purpose" accounts that may be created as "template" accounts, or for use by a service or application.

- **Groups have appropriate memberships:** Windows uses groups to collect users with similar security requirements, assigning permissions to system resources to the group as a whole instead of to individual users. This simplifies permissions management. However, it is not uncommon for group memberships to become skewed over time. Administrators are often called on to deal with problems on-the-fly, such as "So-and-so can't access this file, can you add him to such-and-such group?"

- **No blank passwords:** You would think that after all the information and education about the importance of setting passwords, changing passwords regularly, using strong passwords, and so on, we would have eliminated passwords as a security vulnerability, yet this remains an important test.

- **Reasonable password policy:** It's not enough to simply require passwords. You must have controls in place on the system to enforce a reasonable password policy, covering such items as how often passwords must be changed, whether the system "remembers" passwords to prevent their reuse, and how many bad logon attempts can occur before an account is locked out.

- **"Strong" passwords in use:** Passwords are still a common avenue of attack. If an attacker can guess a simple password or obtain a copy of the password file, or if he can "crack" the encryption that protects the passwords, he can log on to your system masquerading as a legitimate user. Passwords should be strong enough to resist guesses, dictionary attacks, and brute force attacks.

The last issue on the slide, **Local accounts versus domain accounts**, is a Windows-specific issue.

Other Account Issues

- Use of expiration dates
- Limit logon hours
- Special accounts:
  - Administrator/Guest
  - Built-in accounts:
    - IUSR/IWAM, TSInternetUser
    - HelpAssistant, SUPPORT

Some additional precautions can be taken on a Windows system to limit the use of user accounts: Remember the **principle of least privilege!** Accounts can (and should) be created with **expiration dates**, after which the accounts are no longer valid. This is especially useful with temporary or contract employees but can be useful for "regular" employees as well.

In addition, you can **limit the logon** hours during which the account can be used. This may not be practical for some of your "power users" who regularly VPN into the office at 2 AM to work but can be useful for employees who normally work a fixed set of hours. If the account is used only between 8 AM and 6 PM, configure the account so that it is only accessible during that time.

Finally, you should be aware of **"special" accounts** that may exist on a Windows system. The **Administrator** and **Guest** accounts are created by default on all Windows systems. Neither can be deleted via the Windows operating system. In addition, the Administrator account can optionally be disabled on modern versions of Windows.

When you install Internet Information Server (IIS), Windows creates two additional accounts: **IUSR_<machinename>** and **IWAM_<machinename>**. These accounts are used to provide "anonymous" access for people who visit the website. IUSR and IWAM are both considered part of the Guests group by default and have all the privileges and permissions associated with that group.

Some versions of Windows create two additional accounts during installation: **HelpAssistant** and a **SUPPORT** account. HelpAssistant can be used with Windows' Remote Assistance feature to allow a remote user to view or take control of a local user's desktop. The SUPPORT account can be used by the Microsoft Help and Support Center to provide remote assistance to a user as well. If you are not using these features, these accounts should be disabled.

Local versus Domain Accounts

- Nearly every system has local accounts:
  - Administrator (never locks out)
  - HelpAssistant
  - Etc.
- Desktops:
  - Disable them: Domain accounts rule
- Laptops:
  - Have a rational policy
  - Look into Microsoft LAPS

With the exception of your domain controllers, every Windows computer in your domain has local accounts on it, in addition to being accessible through domain accounts. Are these accounts disabled? What about the local administrator account? What about on laptops that are sent with employees to training?

On desktops and servers, it is generally best practice to disable all the local accounts unless there is some specific need for them. When the computer is a domain member, the local accounts become, essentially, unnecessary.

Laptops used by road warriors might be an exception, however. When our employee is in the middle of Africa and needs to do something that requires administrative rights, it may not be acceptable to say, "Sorry, you can't do that." Instead, these administrative accounts should have strong, unique passwords that can be given to the remote user if the situation warrants it. Imagine, for example, that the company has paid $6,000 for a training class and the employee does not have the rights that were published in the laptop requirements.

If this local administrator password is ever issued to an end user, however, it must be changed by the help desk when the user returns to the office. How do we get the user to bring his computer to the help desk? As it turns out, we don't have to. A domain administrator can remotely reset the local administrator password if it becomes necessary to do so.

There's even a tool available from Microsoft to help with this. Look at Local Administrator Password Solution (LAPS, https://u.aud507.com/4-9). This generates random local administrator passwords for your system, storing them securely in Active Directory, and provides a mechanism for domain administrators to make these passwords available to your help desk when needed!

Audit Objective: Valid Users

- Objective: Authorized users:
  - Ensure only valid, active user accounts are present
- Audit activities:
  - PowerShell ActiveDirectory module
  - DSQuery/DSGet
  - PowerShell Get-Local* cmdlets
  - Netwrix, ManageEngine and other commercial tools

The first step in our audit of users and groups is to ensure that only valid user accounts are present on the system and that any accounts present are actively used. (If an account is unused, it is probably not needed.)

We also want to verify some of the parameters associated with the user account (whether the account expires, whether the account has restricted logon hours, and so on); we'll discuss some of these in detail in the next few slides.

Various command-line and graphical tools enable you to extract or dump user information from both local and remote hosts.

Local Users and Groups

- Local accounts and groups affect the security of endpoints
- Unused/unneeded local accounts should be disabled
- Cmdlets for auditing:
  - Get-LocalGroup
  - Get-LocalGroupMember
  - Get-LocalUser

```
PS > Get-LocalUser | Select Name, Enabled | Where Enabled -eq $true

Name     Enabled
----     -------
auditor     True
sshd        True
```


Checking local accounts and groups is easy with the cmdlets listed on the slide.

- Get-LocalGroup will return a list of all groups on the system.
- Get-LocalUser returns the list of users.
- Get-LocalGroupMember returns a list of users or groups who belong to a group. Note that this command does not recurse. It will take a custom script to recursively list all members of any nested groups.

ActiveDirectory Module for PowerShell

- Requires the remote server administration tools (RSAT)
- Available as a feature in newer Windows 10 releases

```
PS C:\> Import-Module ActiveDirectory

PS C:\> Get-Command -Module ActiveDirectory | Measure-Object
Count    : 147
```


The ActiveDirectory module for PowerShell installs as part of the remote server administration toolkit (RSAT). RSAT is now installable as a feature in newer releases of Windows 10.

The AD module includes commands to query and administer all types of objects in Active Directory. Microsoft has begun a strong movement toward using PowerShell as the primary means of managing Active Directory, so it is worth the time to learn to use PowerShell to query our AD domains.

Some of the Get-AD* Commands

```
Get-Command Get-AD* | Select Name
```

```
Get-ADComputer              Get-ADServiceAccount
Get-ADDomain                Get-ADTrust
Get-ADDomainController      Get-ADUser
Get-ADForest                Get-ADUserResultantPasswordPolicy
Get-ADGroup
Get-ADGroupMember
Get-ADObject
Get-ADOrganizationalUnit
```


As auditors, we're concerned with the Get-AD* Active Directory commands. These will return information about any attribute for any object in the directory if you know how to use them.

You can see from the list on the slide that we can query about more than just AD objects, though. Server roles, domain trust relationships, and password policies are just a few of the things we can query.

PowerShell: Active Directory Forest Information

- Get-ADForest
  - Information about the current or specified forest

```
[507dc]: PS C:\> Get-ADForest -Identity aud507.local
ApplicationPartitions : {DC=DomainDnsZones,DC=AUD507,DC=local,
DC=ForestDnsZones,DC=AUD507,DC=local}
DomainNamingMaster    : Day4-DC.AUD507.local
Domains               : {AUD507.local}
ForestMode            : Windows2016Forest
GlobalCatalogs        : {Day4-DC.AUD507.local}
Name                  : AUD507.local
PartitionsContainer   : CN=Partitions,CN=Configuration,DC=AUD507,DC=local
RootDomain            : AUD507.local
SchemaMaster          : Day4-DC.AUD507.local
Sites                 : {Default-First-Site-Name}
```


One of the first steps in auditing a domain is to gather information about the forest in which the domain resides.

The Get-ADForest command will retrieve information about the current forest.

PowerShell: Active Directory Domain Information

- Get-ADDomain
  - Information about the current or specified domain

```
[507dc]: PS C:\> Get-ADDomain -Identity aud507.local |Select-Object
Name, DNSRoot, DomainMode, Forest, NetBIOSName, RIDMaster


Name         : AUD507
DNSRoot      : AUD507.local
DomainMode   : Windows2016Domain
Forest       : AUD507.local
NetBIOSName  : AUD507
RIDMaster    : Day4-DC.AUD507.local
```


The Get-ADDomain command retrieves lots of information about the domain being queried. Truncated sample output is below:

```
[507dc]: PS C:\> Get-ADDomain -Identity aud507.local

ComputersContainer                  : CN=Computers,DC=AUD507,DC=local
DeletedObjectsContainer             : CN=Deleted Objects,DC=AUD507,DC=local
DistinguishedName                   : DC=AUD507,DC=local
DNSRoot                             : AUD507.local
DomainControllersContainer          : OU=Domain Controllers,DC=AUD507,DC=local
DomainMode                          : Windows2016Domain
DomainSID                           : S-1-5-21-2061662408-1420527023-517083644
ForeignSecurityPrincipalsContainer : CN=ForeignSecurityPrincipals,DC=AUD507,DC=local
Forest                              : AUD507.local
InfrastructureMaster                : Day4-DC.AUD507.local
LinkedGroupPolicyObjects            : {CN={31B2F340-016D-11D2-945F-
00C04FB984F9},CN=Policies,CN=System,DC=AUD507,DC=local}
LostAndFoundContainer               : CN=LostAndFound,DC=AUD507,DC=local
ManagedBy                           :
Name                                : AUD507
NetBIOSName                         : AUD507
ObjectClass                         : domainDNS
```

PowerShell: Active Directory User and Group Membership

- Get-ADUser
  - Information about users (specify one or many with -identity or -filter)
- Get-ADPrincipalGroupMembership
  - List of groups a user belongs to
  - LACKS A RECURSIVE OPTION - Use DSGET for this!

```
[507dc]: PS C:\> Get-ADPrincipalGroupMembership -Identity MTams

distinguishedName : CN=Domain Users,CN=Users,DC=AUD507,DC=local
GroupCategory     : Security
GroupScope        : Global
name              : Domain Users
SamAccountName    : Domain Users
```


The Get-ADUser command gives information about one or many users specified with the -identity or -filter options.

Get-ADPrincipalGroupMembership returns a list of all the groups the specified user is a member of. This command lacks the -Recursive option its Get-ADGroupMembership counterpart has. There is an older command line tool called DSGet (mentioned later) which allows us to better obtain group membership information for a given user.

Auditing Aside: Know Your Tool Limitations!

- Get-ADPrincipalGroupMembership is unable to recurse
- Won't return nested group membership
- DSGet works great for this – know your tools!

```
AD > Get-ADPrincipalGroupMembership -Identity DGunny | Select Name | fl
Name : Domain Users
Name : ServerAdmins

AD > dsget user 'CN=Doyle Gunny,OU=Server Admins,OU=Information
Technology,DC=AUD507,DC=local' -memberof -expand
"CN=ServerAdmins,CN=Users,DC=AUD507,DC=local"
"CN=Domain Admins,CN=Users,DC=AUD507,DC=local"
"CN=Schema Admins,CN=Users,DC=AUD507,DC=local"
"CN=Administrators,CN=Builtin,DC=AUD507,DC=local" ...
```


It's always important for an auditor to know the strengths and weaknesses of the tools they use. While we prefer PowerShell for most of our Active Directory query work, sometimes the tool limitations dictate that we should use something else. Here is a perfect example. Get-ADPrincipalGroupMembership cannot recurse any further than the first-level groups to which a user belongs. If a user is in "nested" groups (one group is a member of another), that information will not be reflected in the results.

The old-in-the-tooth DSGet.exe tools handles this with ease. Know your tools and their limitations and capabilities, and your audits will be much easier to conduct!

DSQuery/DSGet

- General-purpose Directory Services Query tools:
  - Any object in the AD
  - View properties
  - Can be used with other tools for scripting updates
  - Great source of audit data

The dsquery tool is an incredibly useful utility. It provides you with a command-line interface to the Active Directory services in your domain. This means that you can use this tool to find any object in the AD, and using it with a few other tools, you can extract any information about the objects easily.

Although you will probably discover that not all the information is in a format that is readily usable, with a little bit of effort and perhaps some scripting, you can usually get the information into an extremely useable form. You'll look at this more closely as we get into the exercises. One of the things that you start doing is learning some basic scripting so that you can create a sort of automated baseline tool. With a little bit of effort, you should be able to use the script you create as soon as you get back to your office!

PowerShell: Active Directory Groups and Group Members

- Get-ADGroup
  - Information about groups (can also use -identity and -filter)
- Get-ADGroupMember
  - Lists members of a group (able to recurse)

```
[507dc]: PS C:\> Get-ADGroupMember -Identity "Domain Admins" | Measure-
Object | Select-Object Count | Format-List
Count : 5

[507dc]: PS C:\> Get-ADGroupMember -Identity "Domain Admins" -Recursive
| Measure-Object | Select-Object Count | Format-List
Count : 70
```


The Get-ADGroup command gives information about one or many groups specified with the -identity or -filter options.

Get-ADGroupMember lists the members of the specified group(s) and has a recursive option to show users who are members of nested groups (user is a member of a group which is a member of the group in question).

Note in the screenshot that while the Domain Admins group appears to have only five members in the first query, when we recurse, it has many more members than that!

"Orphaned" User Accounts (1)

- Unused accounts left on system:
  - User leaves organization.
  - User has an account he never uses.
- Look for:
  - Accounts unused > 30 days
  - Accounts never logged in to
- Check with user/management to see if account is needed
- Disable or delete unneeded accounts

When auditing user accounts that exist on a Windows system, one of the main issues to look for is "orphaned" or unused accounts. It is not uncommon in the daily rush and pressures of a production environment for a "communications gap" to exist between system administrators and the Human Resources department. Users may be promoted, transferred to other departments, or leave the company… and someone "forgets" to tell the system administrators to disable or delete their accounts. In addition, users may be granted accounts to access systems that they use infrequently, if at all. These "orphaned" user accounts should be located and disabled or deleted if they are not in use. In general, accounts that have not been used for 30 days or more (less in high-security environments) should be investigated to determine whether they are necessary or can be removed from the system.

Orphaned Accounts with PowerShell

- Filter on LastLogonDate and/or PasswordLastSet properties
- Dump results to CSV for reporting to management

```
PS > $cutoffDate = (Get-Date).AddDays(-90)
PS > Get-ADUser -Filter 'PasswordLastSet -lt $cutoffDate' | Measure-Object |
Select-Object Count | Format-List

Count : 18

PS > Get-ADUser -Filter 'LastLogonDate -lt $cutoffDate' | Measure-Object |
Select-Object Count | Format-List

Count : 3
```


"Orphaned" accounts are easy to find with PowerShell's Active Directory module. The "PasswordLastSet" and "LastLogonDate" properties can be queried directly using the Filter parameter.

In the screenshots above, we are counting the number of users who have not reset their password or logged in within the last 90 days. You could easily export the list of users to a CSV file for management to investigate.

Orphaned Accounts with DSQuery

- Inactive users:

```
dsquery user –inactive <numweeks> -limit 0
```

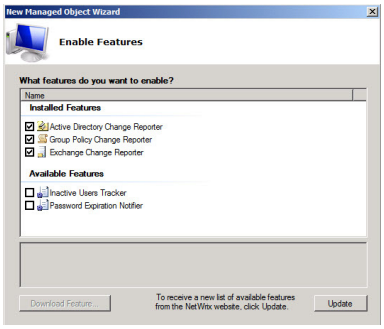
- # days since last password change:

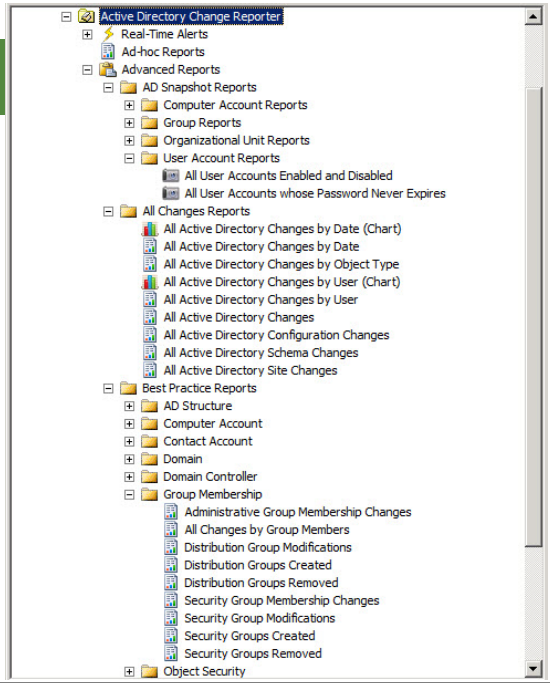
```
dsquery user –stalepwd <numdays> -limit 0
```


Using DSQuery, you can retrieve information about orphaned accounts quite easily, as shown in the slide. The "inactive users" query enables you to return results based on the last login date. The second query is almost as good. This query enables you to list all the users whose passwords have not been changed in a certain number of days. If you line this up with your password change policy, orphaned accounts quickly appear.

Tool: Netwrix/ManageEngine

- Commercial tools:
  - Analyze AD
  - Extract data
  - Audit the AD for all changes





If you detest running LDAP commands manually (or scripting them) and you have a few dollars to spend, you might want to look at Netwrix Auditor for Active Directory. It's an easy-to-use GUI-based tool that allows for exploration, analysis, and, most important perhaps, change detection. You can also configure alerting and other reporting, though that may be of less immediate interest for us.

In terms of cost, the good news is that it is a perpetual license. At this point, I do not feel confident telling you that all future upgrades will be included in that, but at least you don't *have* to pay a yearly licensing fee. There is, however, a yearly fee for software support, which is where your upgrades will come in! Still, the price isn't outrageous.

At least at the time of this writing, you would be looking at approximately $10–$16 per user, depending on the total volume of users. This can add up quickly, but there are definitely price breaks for larger numbers of users.

Service Accounts

- Services must run in the context of a user account:
  - If service is hacked, attacker has privileges of service.
  - Many services run as SYSTEM/LocalSystem.
  - Many applications run with Administrator or Domain Administrator access by default.
  - Should run with minimum required privileges.
- Check account used by various services:
  - How frequently should these passwords be changed?
    - Check your policy…
    - But really, not often! No one should know them, period!

Keep in mind that services also run in the context of a given user account on a Windows system. The service runs with the full privileges of the account that it uses, so if an attacker can compromise a given service, he can obtain the same level of privilege as that service account.

You should check the accounts that are used by the services running on your system. In accordance with the principle of least privilege, services should run only with the minimum permissions necessary to operate. Unfortunately, many Windows services run as **LocalSystem** (sometimes referred to as **SYSTEM**): A full-privileged account roughly equivalent to Administrator.

Back in Windows XP, Microsoft introduced two additional standard accounts used by services: **Local Service** and **Network Service**. Local Service runs on the local system with the privileges of the Users group; it accesses network resources as an Anonymous user. Network Service also runs on the local computer with the privileges of the Users group but accesses the network using the credentials of the computer on which it runs. These accounts provide more limited options for running services without requiring them to have full LocalSystem- or Administrator-level access.

Services can also be configured to run using specific accounts created for their use. You can create standard user-level accounts (or administrator-level accounts, if required) and assign them to a service. Third-party applications installed on Windows systems may also create services as well as accounts that are assigned to those services. Unfortunately, many applications create and use Administrator-level or Domain Administrator–level accounts by default. Although some applications may legitimately require that level of access, in many cases the application can be run with lower privileges. If you can run a service with a lower-privileged account, do so. If you can't, accounts associated with those services should be closely monitored, given long and complex passwords, and made as secure as possible to prevent misuse.

Audit Objective: Appropriate Groups

- Objective: Group memberships are appropriate and enforce least privilege.
- Audit activities:
  - dsquery
  - Netwrix and other commercial tools

In addition to auditing user accounts, you must also pay attention to the group membership of various accounts. As Windows uses groups to assign privileges and grant access to system resources, inappropriate group memberships (such as "regular users" being part of the Administrators group) can grant users access in excess of their needs. Various tools can be used to obtain a list of groups and group memberships. In general, you need to review those lists with appropriate personnel at your audit site (management, project managers, and system administrators) to have them verify that group memberships are appropriate.

Group Memberships

- Groups used to grant permissions and privileges.
- Membership restricts (or grants) access to resources.
- Sensitive groups should be monitored.
- "Administrator" groups should have few people.
- How many in Schema Admins?

You also want to determine which groups are present on the system. Every user on a Windows system must be a member of at least one group. By default, Windows does not grant any rights or permissions to individual users—only to groups. A user gains the ability to do things on the system based on the rights and permissions assigned to the groups of which he is a member.

Windows includes a set of default, built-in groups that define common system roles such as Administrators, Users, Server Operators, Power Users, and so on. Administrators can also create their own groups to grant access to resources. It is important to keep track of group memberships so that you can determine who has access to what. For sensitive groups, membership should be monitored closely. Group names should be indicative of what the groups are for!

Verify that there are no empty groups in your domain, with the exception of your Schema Administrators group. For example, Windows Servers include a number of limited-purpose administrative groups such as Server Operators and Account Operators. These are designed to allow you to delegate limited administrative authority to certain users, without giving them the full power of an Administrator-level account. If these groups are unused, it might be possible for an attacker to remain "under the radar" using one of these groups (versus creating an account in the Administrators group, which is more likely to get noticed).

The administrative groups (Domain Admins, Enterprise Admins) are the most sensitive groups in a domain environment. Membership in these groups should be strictly monitored and limited to a small number of individuals. If you have 10 or more administrators for your domain controllers, you're likely not applying the concepts of least privilege and separation of duties appropriately!

The Schema Administrators group is a particularly sensitive group. Although we were promised that Server 2008 would include a schema-level restore capability, such a restore feature still doesn't exist today with Server 2019 released! For this reason, we strongly suggest that there be absolutely no members in the Schema Admins group; if a schema change needs to be performed, a user or group would be moved there only temporarily.

Windows Passwords

- Access to system should be restricted/controlled
- Windows requires username/password:
  - But password can be blank/weak
- Key issues for audit are:
  - Existence of passwords
  - Regular password changes required
  - Force use of "strong" passwords
  - Good encryption used to protect passwords

Another key issue for OS security is how access to the system is controlled. A secure operating system must somehow restrict or control access to both the system and to individual resources on the system.

Windows requires a valid username and password for any user attempting to access the system. However, if an appropriate password policy is not set, "valid" passwords may include a blank password or an easily guessed password (such as a password that is the same as the username).

Please note that although Windows XP and later still allow the use of blank passwords (unless restricted by policy), by default, accounts with blank passwords can only be used to log on to a system at the console; they cannot be used over the network.

Key issues for the auditor to be aware of regarding user accounts and authentication include:

- Passwords are required; no blank passwords are allowed.
- An appropriate password policy should be in place, specifying things such as minimum password length, account lockout parameters, password history retention, and so on.
- Passwords should be complex and chosen so that they are difficult to guess/crack. An attacker who can guess (or crack) a password can log on to your system and masquerade as a legitimate user.
- Passwords should *never* be stored in cleartext. Appropriate encryption should be used to protect passwords, both at rest (in storage, such as within Active Directory or the SAM portion of the Windows registry) and in transit (such as when sent over the network during a remote logon).

Strong Passwords

- Must not be able to guess or "crack" password:
  - Windows makes cracking passwords easy
  - Unless LM hashes are disabled, LM hashes still stored by default
  - NTLM hashes really are not much better today
- "Password1" still passes "complexity requirements"
- Use a password-cracking tool to audit strength of passwords
- Consider two-factor authentication

Even today, most Windows systems still use password encryption and authentication protocols that were first used with Windows 95. The old LAN Manager (LM for short) authentication protocol used DES-based methods both to store encrypted passwords and to transmit them across the network during authentication. Several weaknesses in Microsoft's implementation made these passwords easy to crack:

- Case sensitivity is ignored
- The password is truncated to 14 characters
- The password is encrypted as two seven-character pieces
- No random information (salt) was included in the encryption process

Microsoft introduced better encryption and authentication protocols with later versions of Windows. However, the default behavior of the OS (even today) is to use the weak LM encryption and the slightly better NTLMv1 encryption in parallel, and to not use the stronger NTLMv2 encryption at all unless it is explicitly enabled:

Older Windows hosts (NT through XP) in a standalone (non-domain) environment used LM/NTLMv1 in parallel unless configured otherwise. Windows Server 2003 and newer use NTLMv1 only, which is better but still has its own weaknesses.

Windows hosts in any environment (standalone or domain) use LM/NTLMv1 in parallel to communicate with "downlevel" (NT, ME, 98) hosts, unless configured otherwise. Beginning with Windows 2000 domains, all hosts now use Kerberos for authentication.

As you can see, in most cases, this leaves us with the old LM protocol as our weakest link, and passwords encrypted using LM are extremely easy to break.

Passwords at Rest

- Password hashes stored in SAM database or Active Directory
- LM hash (DES-based) and NTLM hash (MD4) can be stored in parallel
- Use NoLMHash to disable LM storage:
  - Group Policy setting

Passwords must be protected in two locations: When they are stored on the hard disk, and when they are transmitted across the network during the authentication process.

Windows uses a one-way hash algorithm to store the passwords securely. On most Windows systems, the passwords are stored in the local Security Accounts Manager (SAM) database, which is part of the system registry. On a Windows domain controller, the encrypted passwords are stored in Active Directory. Newer features in 2008 and higher even allow us to limit how much data is replicated to various servers so that certain user password hashes are only present on specific servers.

To support backward compatibility with older Windows systems, all store both the LAN Manager hash and the NTLM hash in parallel. The LM hash represents the "weak link." After that has been found, it is only a matter of time before the NTLM (case-sensitive) password can be obtained as well.

Windows gives you a few options to help protect passwords in storage. One method that can be used to protect the passwords is to disable storage of the weaker LM hash altogether. This option is supported by Windows 2000 and later.

Passwords in Transit

- Windows authentication uses challenge/response or Kerberos
- Challenge/response by default uses LAN Manager and NTLMv1 in parallel:
  - LM sends the hash:
    - It's not your password that authenticates you… It's your hash
  - NTLMv1 is horribly vulnerable to replays
  - NTLMv2 has issues as well – still vulnerable to a man in the middle
- Microsoft recommends Kerberos:
  - Can be difficult to require this given other NTLM-based dependencies

To protect passwords as they are transmitted across the network, you must enable the use of either Kerberos or the NTLMv2 authentication algorithm. Keep in mind that Windows 2000 or later systems that are members of a Windows 2000 or higher domain use Kerberos for authentication. However, for standalone systems or for communication with other clients and servers (older Windows systems, SAMBA systems, and OS X systems), Windows uses LM and NTLMv1 in parallel. This is extremely bad. Worse, in the most modern (Server 2016 and Windows 10) operating systems, you must *downgrade* security to successfully communicate with these non-Microsoft systems! (Please note that it is possible to configure this to use more secure protocols, but it is *much simpler* for the administrator to just downgrade security.)

LM, or LAN Manager, simply sends the actual LM hash across the network directly. This is extremely serious because it's not actually your password that authenticates you. After all, the system doesn't actually store your password; the system stores the *hash* of your password. If a malicious user can see the hash, he can simply use that hash directly!

NTLMv1 has similar problems. Although this is a challenge-response system, it is vulnerable to a replay attack because the system acting as a server does not "remember" what it sends as a challenge. This allows an attacker to replay any previous authentication response, providing a challenge to the server. Windows systems can be configured to require the use of NTLMv2 through the local security policy editor or via Group Policy.

NTLMv2 is still vulnerable to man-in-the-middle attacks. While these are mitigated through the use of SMB Signing, the better option is to require Kerberos authentication. Kerberos naturally includes signing and the identity of the requesting workstation, not just the user, for each ticket request. Even though this is definitely the *right* way to do things today, you may find that it is quite difficult to do so. Legacy systems and connections to other operating systems that are authenticating into your Active Directory may be completely incapable of speaking Microsoft's version of Kerberos, leaving NTLMv2 as your best option.

Password Testing

- Three main ways:
  - Dictionary
  - Hybrid
  - Brute force
- We don't care about brute forcing:
  - If someone has our hashes, we've already lost
  - Expect that they will be broken in minutes
- We care if passwords can be easily guessed

Automated password attacks are carried out in one of three ways:

- **Dictionary attack:** The cracking tool is fed a list of common words and password patterns (password, printer, 1234abcd, and such). Each word or pattern is encrypted with the same algorithm used by Windows. If the encrypted word matches an encrypted password from the password file, the passwords must be the same.

- **Hybrid attack:** Even if users do not pick plain dictionary words, they often use simple tricks such as appending numbers or symbols to the beginning or end of words or substituting numbers for letters. Some cracking tools contain logic to attempt these common permutations if a dictionary attack fails.

- **Brute force attack:** A brute force attack tries every possible combination of numbers, letters, and symbols until the correct password is found. This means that even long, complex passwords such as ^6AdE^!3hIoqKSzZ would be found eventually… given enough time. A "strong" password should withstand a brute force attack until the maximum password age is reached and the password must be changed.

Passwords are consistently one of the prime weaknesses in any system. This is partly due to increased computing power, which allows password attacks to work more and more quickly. A more significant factor is user behavior: Choosing poor passwords, writing passwords down, sharing passwords with others, and so on.

In 2003, cryptographic researchers outlined a method to further decrease the time needed to crack Windows passwords. Sometimes referred to as *rainbow chains* or *rainbow tables*, the process involves precomputing a lengthy list of possible LM password hashes. Generating the hash value is the labor-intensive part of password cracking; if hashes are generated in advance, all that remains to be done is to compare password hashes obtained from a system with the precomputed password hashes. One website even hosts a database of precomputed hashes and allows you to upload hashes for cracking.

Tool: Password Assessment

- Password assessment tools ("crackers"):
  - L0phtCrack 7 ($595 < 500 accounts, $1600 < 5000, etc.)
  - Rainbowcrack (Fast... sort of):
    - Not meaningful for us
  - Cain and Abel (Awesome! May be wrongly flagged as malware)
  - John the Ripper (Our preferred tool)
  - Microsoft Baseline Security Analyzer:
    - Checks for a few basic, "really dumb" passwords
- **Only run these tools with permission**

Auditors are strongly encouraged to audit user passwords on a regular basis. As this is one of the prime areas of weakness (blank passwords, non-expiring passwords, and weak passwords), it should certainly be examined.

A number of tools are available that can assist with password assessment. One caveat: Some audit processes (password assessment, vulnerability scanning, and so on) are no different than the tools and techniques used by attackers to perform reconnaissance and break into systems. The only difference between you using these tools and a "bad guy" using these tools is that the bad guy doesn't have permission. Make sure you do; if you plan to run these tools, explain the reason for their use in advance and be sure to get permission, in writing, to use them.

Also, be aware that some organizations may want to know that a particular account has a "weak" password (for example, it is crackable by some assessment tool), but may not want to know (or wish for you to know) what the password actually is. Some tools display the cracked password; some simply let you know a password was "cracked." Be aware that this is sensitive information, so plan accordingly for your audit reporting.

L0phtCrack 7 is probably the best-known Windows password cracker. L0phtCrack has served as more or less the standard in Windows password cracking for years. It has almost gone out of existence several times but is again available from L0phtcrack.com.

Other outstanding products to consider are RainbowCrack, which is extremely fast, but only after you have spent several days precomputing hash databases. RainbowCrack is also not "point-and-click" friendly, so if you try it, be prepared for some work. Another super tool to examine is Cain and Abel, which is a good all-around password tool with a nice interface and lots of other cool features.

For performing password audits on Windows and any other systems, the author recommends John the Ripper. It is versatile, can be clustered, supports GPUs, and runs on any operating system.

Getting the Hashes

- Hashes stored in the registry of AD servers:
  - NTDS.dit
  - Not normally accessible:
    - Use NTDSUtil to perform install-from-media (IFM) backup
      - Use NTDSAudit tool to extract hashes and SysKey
    - Leverage the volume shadow service to copy the Domain Controller's registry (NTDS.dit) and extract the hashes from that

The best approach today to extracting the protected password hashes is to use the Windows backup features. Microsoft doesn't actually support any mechanism for getting at the hashes. The hashes are stored in the registry of the AD servers in the NTDS.dit file. This file is locked whenever the system is running and, further, it's encrypted. To access it, we can either boot the system into an alternative operating system and copy the relevant files or use the Volume Shadow service, which is designed to allow you to get access to locked files to back them up!

Here's the process that we've found to be most reliable: Have the admin backup NTDS by using the NTDSUtil program to perform an install-from-media (IFM) backup. Then use the NTDSAudit tool from Dionach to extract the hashes using the backed-up file and the SysKey retrieved from the domain controller's registry. Full details can be found at:

https://u.aud507.com/4-13

Audit Interview Questions (1)

- User account management:
  - Who requests accounts? Who can create accounts? How are they notified of departing employees?
- Group management:
  - Who determines group members? Who reviews membership?
- Passwords:
  - Is there policy addressing passwords or authentication? Are passwords regularly assessed? Who is allowed to do this?

There are some non-technical policy issues that should be considered when reviewing user accounts and groups within a domain.

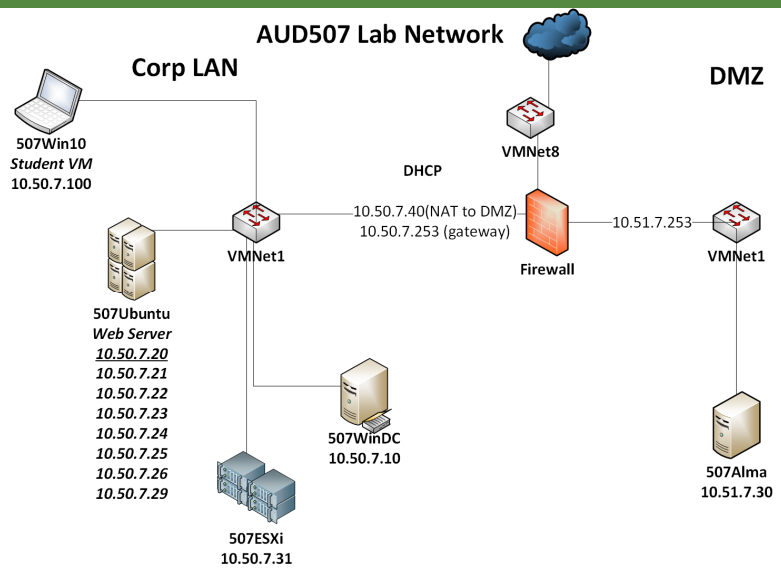
For user accounts, who has the authority to create accounts? What type of authorization is required to request an account (for example, supervisor's signature, explanation of business need to access resource, and more)? How are passwords for new accounts distributed (verbally, in-person pickup with check of photo ID)? Who is allowed to modify accounts, including unlocking accounts or resetting passwords? How are these procedures handled?

Are there procedures in place to identify inactive accounts and disable or delete them? Is the IT department notified of departing employees? Does your organization allow departing employees to retain account access for a limited period of time (such as to retrieve email and so on)?

In terms of groups, who can create and manage groups? Who can request the creation of a group? Who determines group memberships? Who determines the resources that a group has access to, and the level of access provided? Does anyone review group memberships on a periodic basis to ensure membership is appropriate?

Regarding passwords, does the organization have a written policy regarding passwords and authentication requirements, including password strength and periodic password changes? Are passwords assessed or audited (for example, cracked)? Who has the authority to perform such assessments?

Exercise 2.4: Querying Active Directory



Open your Lab Workbook to Exercise 2.4: Querying Active Directory, and let's practice getting information from Active Directory using PowerShell and DSQuery.

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. Background and Plan
2. PowerShell and WMI
3. Windows Auditing
4. Users, Groups, and Privilege Management
5. **System and Resource Security**
   - *Protecting Data: Rights*
   - *Protecting Data: Permissions and Shares*
   - *Group Policy*
6. Windows Logging
7. Continuous Monitoring

Now that we've covered most of the external threat vectors (patches, network services, local services, and users), let's turn our attention to more of the internals of the Windows domain.

Audit Objective: Restrict Access to Resources

- Objective: Enforce least privilege/need to know:
  - Control who can access objects and what actions users can perform
  - In short, permissions
- Audit activities:
  - Command-line tools
  - Sysinternals' AccessEnum
  - Somarsoft DumpSec
  - Security Configuration and Analysis

A key security feature of any operating system is its capability to control access not only to the overall system, but also to individual resources on the system, such as directories, files, printers, and so on. This is commonly referred to as **authorization**: After a user has successfully logged on to the system (**authentication**), what resources (files, applications, and directories) are they authorized to access?

Not all users should have access to all parts of the system. Operating system files (binaries, executables, and configuration files) should be restricted to privileged users only, to prevent malicious users from replacing system files with trojaned copies or to prevent ignorant users from accidentally deleting key files. A virus hoax in the spring of 2002 warned users to delete the jdbgmgr.exe file from their Windows systems (the file with the "teddy bear" icon) because it was a virus. The file is actually a normal part of the operating system: The Visual J++ debug manager. Nevertheless, many users deleted this "virus" from their system. Fortunately, the file is not a critical component of the OS.

In addition, user-created documents may contain sensitive information and should not be available to just anyone. Printers may also require limited access/use, such as printers used to print sensitive documents, or they may use custom forms (such as invoices, purchase orders, or even checks) that are costly if wasted by misdirected print jobs.

Windows should always use the NTFS filesystem (versus the older FAT/FAT32 filesystem), which allows granular control over the permissions assigned to a given object. NTFS uses Discretionary Access Control Lists (DACLs, pronounced "dackles") to determine who can access an object and what exactly they can do to it. Permissions can be assigned to users and/or groups. Note that NTFS also supports a System Access Control List (SACL, pronounced "sackle"), which is used to define audit policies for NTFS objects.

As an auditor, you may need to be aware of the permissions assigned to particular files or directories (or the entire filesystem), and to determine whether those permissions are appropriate.

Data at Rest

- Local access controls often ignored:
  - Bad guys are "outside" on the internet
  - Successful "outsiders" still have some level of access
- Insider attack potentially more damaging:
  - Already has access to system
  - Knows more about location of data and any existing security

Local access controls, such as permissions, are often ignored for a number of reasons. One is the perception that all the "bad guys" are "outside" on the internet. Therefore, if you have strong perimeter protection (firewalls, packet-filtering routers, and a perimeter intrusion detection system) or system-wide access controls (accounts, passwords, security tokens, and smart cards), there is no need to worry about protecting the data.

That attitude is a misperception. It is true that the greatest number of attacks (probes, scans, and so on) originate from the internet at large. However, although insider attacks are less frequent, they are often the most damaging. Insiders, by definition, already have some level of access to the system (a valid logon, for example) and may have knowledge of where particularly "interesting" data is stored, and what protections may (or may not) be in place. In such a case, providing strong authentication without providing any control over access to resources leaves your network wide open for misuse, abuse, theft, or malicious destruction.

Strong permissions or other access controls also help protect you from outside attack. If someone can break into your system from "the outside," he will have the same level of access to resources as the service or application he compromises. (Someone who hacks your web server, for example, will have the same level of privilege as the web server application.) Setting appropriate permissions can help to limit the damage from an external break-in, or at least buy you some time to detect the problem before it becomes significantly worse.

Protect Data at Rest

- Sensitive files (OS binaries, corporate data) must be protected
- Various methods:
  - Restricting system access
  - Limiting explicit rights/privileges
  - Enforcing resource permissions
  - Using encryption

Sensitive data, such as your operating system binaries and sensitive corporate information, must be protected from unauthorized access (confidentiality), modification (integrity), or destruction (availability). We have already discussed two methods of protecting your data: Requiring user authentication and using permissions to protect resources.

In addition, access to system resources can be further restricted by limiting the explicit rights or privileges granted to users and groups. In addition to access control lists (permissions), Windows supports the use of privileges to carry out specific tasks. Limiting privilege use restricts the types of actions users can take on the system.

Finally, encryption can also be used to protect data. Encryption uses one or more keys combined with an algorithm that is used to convert plaintext into unreadable ciphertext. The plaintext can be recovered only by someone with the appropriate key.

Encryption provides a few additional advantages over "just" authentication and permissions. Those protections may not always be available. (NTFS cannot be used on a CD/R, for example.) Or they may be subverted; for example, someone with physical access to a Windows system can boot the system to an alternative operating system, bypassing any restrictions (such as NTFS permissions) placed on the data and enforced by the OS.

In those cases, encryption provides added protection because even if permissions are bypassed so that an attacker could actually gain access to the file, the file remains unreadable unless the attacker also has the key to decrypt the information. Encryption is frequently used to protect passwords or password files as well as sensitive user data. Windows uses encryption to protect the Security Account Manager (SAM) database. Windows 2000 and later also provide built-in file encryption through the Encrypting File System (EFS).

Windows Rights and Permissions

- Rights (also called privileges) are specific tasks:
  - Log on locally, change system time, load device drivers, shut down system, and more
  - Generally, things you can "do"
- Permissions are access controls:
  - Read, add, modify, execute, delete, change permissions, and so on
  - Generally, actions you take with regard to NTFS objects

Let's take a moment to distinguish between *rights* (sometimes called *privileges*) and *permissions*.

Permissions (also called *access control lists*) are something you should be familiar with. Permissions define the standard types of access to objects, such as read, execute, write, delete, and so on. Most often, we associate permissions with objects such as files and directories. In Windows, nearly every object can have permissions, including printers, registry keys, services, and Active Directory objects and object properties. Each of these objects has a particular set of permissions associated with it. (Printers, for example, would include permissions such as the ability to print to the printer, to manage print jobs that you submit to the printer, to manage others' print jobs, to modify the printer properties, and so on.)

A concept that might be more unfamiliar is that of rights or privileges. Windows defines a set of user rights that can be assigned to users or groups. These rights are specific tasks that can be carried out on the system. Rights include common activities such as logging on locally, accessing the computer from the network, installing device drivers, changing the system time, and adding workstations to a Windows domain. Rights also include more esoteric activities, such as the ability to log on as a service, increase the scheduling priority of a task, and act as part of the operating system.

It is the combination of rights and permissions that define what a user or group can do on a system. As such, both must be taken into consideration for your audit to ensure that rights and permissions are assigned appropriately.

RBAC Should Be Used

- Access granted to users or groups:
  - Users "inherit" rights of groups of which they are members.
- In domains, groups can be nested:
  - Groups "inherit" rights of groups in which they are placed.
- **AGDLP** – Accounts → Global groups → Domain groups→ Local groups → Permissions (or rights)

Rights and permissions can be granted to users and/or groups. Best practice is to grant permissions to groups instead of individual users. This is, effectively, how we implement Role-Based Access Control in Windows because Windows does not have "Role" objects. It is much simpler to grant access to a single group rather than to grant access to 20 individual users. In addition, as your organization grows, groups make it easier to keep track of not only *who* has access to resources, but *why* they have access.

When granting rights or permissions, keep in mind that users "inherit" or obtain the rights and permissions of groups of which they are members. Also, in a Windows domain environment, it is possible to nest groups inside of other groups. Groups "inherit" or obtain the rights of the groups in which they are placed.

This is exactly how Windows grants power, so to speak. Even the Administrator account has no inherent power of its own. It obtains its capabilities by being made a member of the Administrators group. The Administrators group is then given extensive rights and permissions on the system.

In domain or multidomain (tree/forest) environments, the ability to use different types of groups as well as "nested" groups allows for great flexibility in assigning rights and permissions. But this flexibility can also lead to complex configurations that are difficult to sort out and manage.

Microsoft recommends the use of a particular method to assign rights and permissions in a domain environment: AGDLP. **AGDLP** is an acronym that stands for taking user Accounts, placing them in Global groups, which are then placed in Domain groups, which are then placed in Local groups, which are then assigned Permissions.

If you are in a single-domain environment, you will be dealing only with Global and Local groups. If you are not in a domain at all (that is, workgroup environment/standalone hosts), you will be dealing only with Local groups.

Just Enough Administration

- Server 2016 introduces JEA (Just Enough Administration)
  - Create tailored access for individuals
  - Permit remote elevated privileges on specific systems in the AD
  - Based on and relies upon PowerShell >= 5.0
    - Also requires Windows Management Framework
- This should dramatically reduce the number of individuals who must be granted broad-ranging rights in the domain

A new addition to rights management that becomes available to you when you migrate to Server 2016 is Just Enough Administration (JEA). This system further seeks to reduce the number of rights that need to be assigned to a user, even a user who needs to perform some administrative functions.

Rather than grant the rights to the user directly, JEA is used to create a profile that can be leveraged within PowerShell 5. This secure operating profile will allow the user to use PowerShell to query, configure, monitor, or otherwise interact with remote systems with whatever rights have been assigned, even though the user himself does not have these rights.

This *should* mean that even fewer users and groups will have special rights, though we expect that there will be a slow uptake since using this requires not only the administrator himself, but also the users to whom these profiles are assigned, to have a fair amount of competence with PowerShell.

JEA Configuration

- User assigned to role (usually AD group) in session configuration (.pssc) file

```
RoleDefinitions = @{ 'AUD507\JEAAuditors' =
@{ RoleCapabilities = 'JEAAuditRole' }; }
```

- Roles given access to PowerShell functionality in role capability (.psrc) file

```
ModulesToImport = 'ActiveDirectory'
VisibleCmdlets = 'Get-*'
```

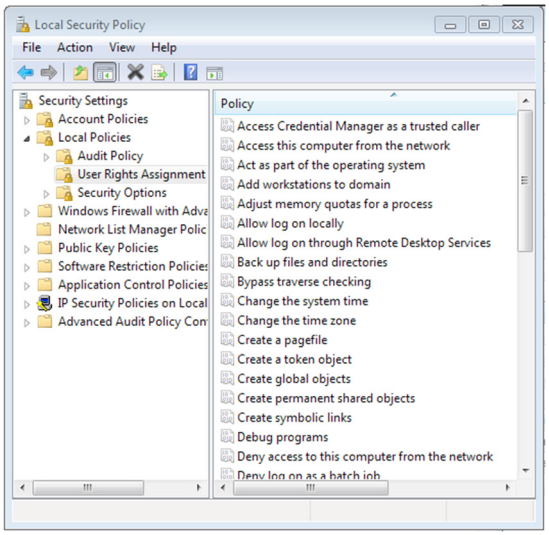

To configure JEA, the administrator will create roles on the target systems using a PowerShell session configuration (.pssc) file. Roles will be given names and an array (list) of capabilities.

Then, those roles are assigned the ability to perform certain actions using PowerShell remote sessions on the target systems. When the user enters the remote session, they will have access to only the modules, cmdlets and functions that have been assigned to them by the administrator.

On this slide, we've created a JEA role for auditors to be able to enter a session on a machine, access the ActiveDirectory module, and run any cmdlet whose name starts with "Get-".

Windows Rights

- Specific tasks or actions on the system
- Can be explicitly granted
- Rights not explicitly granted are implicitly denied
- Some can be explicitly denied



Windows rights are specific tasks that can be carried out on the system or acts that can be performed. They define various ways that users can interact with the system. Rights include things such as the ability to manage the audit log, the ability to modify the system time, and so on. By default, Windows grants a number of rights to groups on the system to allow abilities commensurate with the roles of those groups. For example, on a Windows Domain Controller, Administrators have the right to "log on locally" (at the system console), but Users do not.

Rights must be explicitly granted. If a group is not granted a right, the group is considered to *not* have that right; it is implicitly denied.

For an explanation of the rights in Windows, look at this MSDN article: https://u.aud507.com/4-18

Next Several Slides

- Majority of the rights in Windows systems today:
  - Slides contain the rights and who should have the rights
  - Recommendations based on:
    - Microsoft hardening guides
    - Practical experience in enterprises: Sometimes Microsoft's recommendations are too loose
  - Notes contain detailed description of each right
- Your instructor will *not* cover these slides with you
  - Feel free to ask questions about a right if you've always wondered what one does

Over the next several slides, we detail the majority of rights available in the Windows operating system. There are a few that we do not list, because their function is very clear and having those rights may not create immediate security consequences. Within the slides, you can find the name of the right along with recommendations for who should have that right. In some cases, you see that no one should have the right because it's not necessary or because it creates a security vulnerability.

The recommendations in the slides are based on the Microsoft best practices hardening guide for Server 2016, but this is just a starting point. When reviewing these guidelines, it was discovered that in several cases Microsoft's recommendations are a bit too loose! As a result, these recommendations are tweaked to be slightly more secure than what Microsoft suggests.

The notes for each slide contain a detailed explanation of each of the rights described in the slides. You may want to hang onto and even share these pages! Finding all this information in one place is actually tough!

It is unlikely that your instructor will explain every right, one by one. First, this is far more detail than you need as an auditor. Instead, you need to know where to find a description should you have a question about a right. Second, this would be even more detail than most administrators are familiar with!

That said, ask about any of the rights should you have questions. Your instructor will be more than happy to assist you, or you can write to me at any point after the course using the email address found on the title page for this book!

Rights Recommendations (1)

- Access Credential Manager as a Trusted Caller:
  - No one
- Access from Network:
  - Administrators, Authenticated Users
- Act as Part of the Operating System:
  - No one
- Backup Files and Directories:
  - Administrators, Backup Operators
- Change System Time:
  - Local Service/Administrators

Access credential manager as a trusted caller is a right that should never be assigned to users and is used only by the Winlogin service. Anyone having this right is able to access absolutely any credential stored in the credential manager, leading to the ability to impersonate any user or entity.

The second right gives users the ability to **access a computer from the network**. This right is necessary to access things such as file shares. For this reason, both administrators and authenticated users should have this right. The Authenticated Users group represents all users who are currently logged in to systems using their credentials.

Our third right is **act as part of the operating system** and should not be granted to anyone. This is a high-level right that essentially creates a trusted process that can take any action on the system. This right is included for developers who might need to write a service to run as a part of the operating system. If that were the case in our situation, we could create a service account and grant this right to just that service account. However, this is a rare right to actually need.

The **backup files and directories** right should be assigned only to administrators. An exception would be if you have a specific set of operators within your organization who are responsible for maintaining backups. If you do, then it would make sense to create a backup operators group and to give this right to that group as well (though there is already a Backup Operators group built in). Because this group already exists, it would be rare to create another or to assign this right to any other users.

The right to **change the system time** may not seem particularly important but can actually have some serious consequences. This right should be granted to the Local Service group and the Administrators group. If you allow users to change the times on their systems, they can end up breaking their ability to authenticate to the domain. This can lead to a difficult troubleshooting problem. To prevent this type of issue from arising, it is simplest to prevent people from changing the system time. Also, changing the system time might allow a user to effectively forge log entries.

Rights Recommendations (2)

- Add Workstations to the Domain
  - Administrators, Help desk users
- Create Token Object:
  - Administrators
- Create Global Objects:
  - Administrators, Local/Network Service
- Create Permanent Shared Objects:
  - No one
- Debug Programs:
  - No one

The **add workstations to the domain** right, as its name directly states, permits users having the right to join workstations to the domain. It is completely reasonable for your computer support staff to have this right. It is interesting to note that, by default, every user has the right to join up to 10 workstations to the domain even without this right.

The **create token object** right should be assigned only to the LocalSystem group. This right, if granted to a user or other group, allows individuals with the right to create an access token that can then be used to access any local resource on the system. Essentially, this means that this user or group would take over the system. This right must be closely controlled.

The **create global objects** right manages which users and groups have the ability to create sections or symbolic links in the object manager. Assigning this right to users can be a security risk. It should be assigned only to trusted users, though the default settings of administrators, local service, and network service are appropriate for most installations. There is a misunderstanding that users need this right to access terminal services. This right would be necessary to create global objects through terminal services, but the creation of session objects can still be accomplished without granting this right.

The **create permanent shared objects** right, if granted to users, would allow them to create shared objects on domain controllers. This right should not be assigned to anyone. Administrators, for instance, do not need this right to shared objects.

The **debug programs** right gives someone the ability to attach to a running process. When attached, the user can view the entire process space, including the memory. In addition, it is possible to inject arbitrary code into the running process and change the overall function of an existing program. No users require this right. It might be necessary for the developer on the developer machine.

Rights Recommendations (3)

- Deny Access from Network:
  - Guest
- Enable Trust for Delegation:
  - Administrators
- Force Remote Shutdown:
  - Administrators
- Impersonate Client:
  - Administrators, Local/Network Services

The **deny access from network** right allows us to explicitly list users or groups that should never be allowed to access the computer from the network. Globally, in our domain it's safe to say that guest users should be specified here. If we were looking at a Group Policy object that applies to domain servers that do not offer file sharing, it might make sense to include additional users in here. The principle of least privilege will dictate which groups show up here.

The **enable trust for delegation** right should have only the Administrators group listed. This right would give the user the ability to run a process on the remote machine under the credentials of another designated user or service.

Force remote shutdown is another right that is reserved only for administrators. I have seen some situations in which a separate service account is set up for the SMS service. If that's the case, then it is completely appropriate for that account to be listed here as well. As indicated, this right allows a user to force a remote shutdown of a computer without user interaction.

The **impersonate a client** right is assigned by default to administrators, a local service, and a network service. This right gives the user the ability to run commands or access files and services as a different user. The computer system does this all the time. For instance, using EFS, the user's private key is used to decrypt the information. If you access EFS data remotely, then that remote computer system is actually impersonating your user to access your private key on that remote system. No users require this right.

Rights Recommendations (4)

- Increase Priority:
  - Administrators
- Load/Unload Drivers:
  - Administrators
- Modify Firmware Environment:
  - Administrators
- Volume Maintenance:
  - Administrators

The ability to **increase the priority** of running tasks should be reserved only for administrators. If users have the ability to increase the priority of their tasks, especially on shared systems, it can lead to resource starvation issues. Typically, there is no good reason why a user would ever need the right to increase the priority of a task.

Loading and unloading drivers, possibly changing the drivers that are stored on the system currently, can lead to serious consequences. In many cases, malware will attempt to install itself by replacing an existing driver. It may also attempt to install itself by creating a new driver and installing it on the system. The loading and unloading of drivers should be assigned only to administrative users and can provide some protection against some forms of malware.

The **modify firmware environment** right should be limited to the administrators and the LocalSystem account. This right is used by the operating system to mark in the firmware environment the last known good configuration. On newer processors, additional information (for instance, the ability to control the default operating system on startup) may also be changed with this right.

The **volume maintenance** right should be granted only to administrators. This right allows users to remotely perform maintenance on volumes (for instance, initiating a defragment remotely). In addition, this right gives users the ability to explore discs and extend files into memory, potentially giving them the ability to read or even modify data that they would not normally have access to.

Rights Recommendations (5)

- Replace Process Level Token:
  - Local/Network Service
- Shut Down System:
  - Administrators, Backup Operators, Interactive
- Add Workstations:
  - Administrators, Delegated Right
- Local Logon:
  - Servers: Administrators
  - Workstations: No setting

The **replace process level token** right is similar to the impersonated user right. More specifically, the replace process level token right gives one the ability to initiate an API call as another user or service. This right should be assigned only to the Local Service and Network Service groups.

All users need the ability to **shut down systems**, but only interactive users should be assigned the right along with backup operators and administrators. If we were to include the more general authenticated users with this right, that would allow a user to potentially cause the shutdown of another user's workstation. Restricting this to interactive users requires that the user be physically logged in to the console to start the shutdown process.

The **add workstations** right is assigned to the administrators group by default. In addition, it is quite common to grant this right to individuals working on your help desk who are responsible for the installation of new workstations. General users should not have this right, however, to prevent them from installing unauthorized systems into your domain area. **You might be disturbed to learn that, unless your AD Schema has been modified, any authenticated user is allowed to join up to 10 different systems to the domain, regardless of this right.**

The **local logon** right should be controlled definitely for the name servers and domain workstations. Domain workstations generally permit all users to log on. For that reason, there's no need to list anything in this setting. On domain servers, however, this right should be granted only to individuals in the Administrators group. This prevents the user with access to the data center from walking up to a server and logging in interactively at the console.

Rights Recommendations (6)

- Allow TS Logon:
  - Servers: Administrators
  - Workstations: ?
- Deny Logon as Batch Job:
  - Guests, likely most of your User groups
- Deny Logon Locally:
  - Guests, your "Services" groups
- Deny TS Logon:
  - Guests, your "Services" groups, ?

The **allow terminal services logon** right should be assigned to the Administrators group. Depending on policy, additional users may be permitted to use this. This right allows a user to remotely connect to the terminal services using the remote desktop tool in the Windows operating system. Of course, if a terminal server is in use, it makes sense that on that server individuals would be given the right to log on. This is typically managed by using the group named remote desktop users.

In the slide, we have listed what Microsoft considers best practice for the **deny logon as batch job** right. In the opinion of many administrators, guests is a good start. However, apply the principle of least privilege here. Do you allow users to schedule tasks to run as batch jobs? If you do, then those users and groups would not need to be listed here. However, if you do not want this type of activity, perhaps additional groups should be listed here.

The **deny logon locally** right and **deny terminal services** logon right should need no explanation. The guests group is included in these rights by default. A suggestion for implementing principle of least privilege for administrators on their own workstations is to include the Domain Administrators group in the deny logon locally right only on the administrators' workstations. This will force them to use their "normal" account to do things like read email. When they have a need to do something requiring administrative rights, they can use the **run as** command to gain the necessary rights temporarily.

Rights Recommendations (7)

- Generate Security Audits:
  - Local/Network Services
- Log On as Batch Job:
  - Administrators
- Log On as Service:
  - Service Accounts, Local/Network Services
- Restore Files and Directories:
  - Administrators, Restore Operators

The **generate security audit** right gives services the capability to generate security audit events. This right should be assigned to the Local Service and Network Service groups. The next right, **log on as batch job**, is typically limited to the Administrators group. As we discussed when dealing with the denying log on as batch job right, it may be that specific users or perhaps service accounts have the ability to log on as a batch job. If that's the case, the recommendation is to create a specific group to which this right is assigned and then document which users and service accounts require this right and add them into that group.

Similarly, the **logon as service** right should be limited to Local Service, Network Service, and any service accounts that are necessary for your environment. For example, if you were to install a Lotus Domino server, the Domino server should run under its own service account. The service account would be added to, perhaps, a service accounts group, which would be added to this right.

As with the backup files and directories right, the **restore files and directories** right needs to be tightly controlled. Only administrators and people actually involved in the backup and restore process need to have this right. Having this right allows you to bypass file access controls and permissions to overwrite data on the system.

Rights Recommendations (8)

- Take Ownership of Files and Objects:
  - Administrators
- Access Credential Manager:
  - No one
- Adjust Memory Quotas For a Process:
  - Administrators, Services
- Whew!!!

The last three rights that we need to concern ourselves with are the **take ownership of files and objects** right, the **access credential manager** right and the **adjust memory quotas for a process** right.

The ability to **take ownership of files and objects** gives one the ability to bypass file access permissions by simply changing the ownership of the object. Only administrators should ever have this right.

The right to **access credential manager** would give one direct access to all the cached credentials and other credential information stored in memory in the LSASS process. Microsoft best practices do say that no one needs this right. (I have never seen the need to assign this right to any service or user.)

The **adjust memory quotas for a process** right allows the granted user account to define the maximum amount of memory that may be used by a process on the system. Users should never have this right on a server system because it would allow a user to effectively set this memory priority of his own process to be higher than most everything else. On a workstation, this right does not have as much potential for disaster, but you would have to wonder why a user would need this right.

As you can see, there are quite a number of rights available in Windows systems. Documentation of what these rights actually do is rather limited, so it's not uncommon to find administrators are not sure what these rights mean. It's my hope that, coupled with the best practices document from Microsoft, you will find this information useful in determining what the appropriate settings are for your domain.

Finding Rights

- Various tools can report on the system:
  - Even so, remember that the rights do not exist on the user objects
  - Rights are controlled via the registry only
  - This means that the rights can change from machine to machine
- Querying a machine tells you which rights are granted to which users based on the Group Policy *currently* applied

User rights can be viewed in a number of ways. Any of the Windows security and administration tools (Local Security Policy and Group Policy) can be used to review user rights assignments. However, as those are all graphical tools, they are not ideal for an automated audit.

It is important to remember, however, that the user rights cannot be observed by analyzing a user object. The user rights are controlled exclusively through the registry and are typically configured by Group Policy. This means that if you query a system to determine who has which rights, the rights that you see configured are specific to that system and user combination with those group policies applied. You will want to more thoroughly inventory right assignments when you review the group policies that are in use within the organization's domain. Because the rights should be managed via Group Policy, we can find that the rights assignments change from system to system depending upon which Group Policies have been applied.

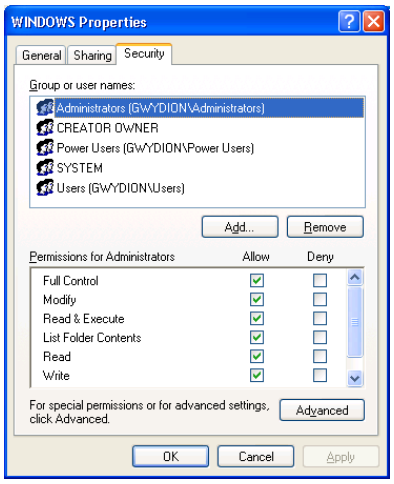
User Rights PowerShell Module

- User rights are handled by the local security administrator subsystem
- Best information for current user rights on a system is in memory
- There's no native PowerShell tool for querying rights
- PowerShell module available on TechNet for querying/setting current user rights

Because it is difficult to query rights on a system natively using PowerShell, I have found the script available at https://u.aud507.com/4-21 to be extremely helpful on my audits. This module implements commands which allow the auditor to query for all users with a specific right, and to query for all rights assigned to a particular user.

Windows NTFS Permissions

- Can be granted or explicitly denied:
  - Permissions not explicitly granted are implicitly denied
- "Deny" overrides "allow"
- NTFS permissions are cumulative:
  - Effective permissions are **sum total** of all permissions



Permissions (technically known as Discretionary Access Control Lists, or DACLs) are the common types of access that you are already familiar with. Typical high-level permissions include read, execute, write, modify, and full control (which includes the ability to modify permissions).

Windows also allows you to drill down into these high-level permissions and assign fine-grained access controls. For example, the "read" permission is actually composed of the individual low-level permissions of traverse folder (for directories)/execute file (for files); list contents of folder/read file data; read file attributes (such as Read-only, Hidden, and System); read extended file attributes (which may be added by other programs or utilities); and read the permissions on the file.

Permissions can be explicitly granted or explicitly denied. However, use of any explicit "deny" is not recommended, except in special circumstances, as it can have undesirable effects. Similar to user rights, if a permission is not explicitly granted, then it is considered to be implicitly denied. In other words, simply not granting the "Users" group any access to a file is equivalent to explicitly denying access to the Users group. Note that when both allow and deny permissions are used, deny permissions will always override allow permissions.

Finally, keep in mind that NTFS permissions are cumulative; a user's access to a given object is the sum total of all access they have been granted (individually or via group memberships). For example, if Susan is a member of both the Users and Managers groups, and Users has been granted Read access to a file and Managers has been granted Modify access, Susan will have Modify + Read access to the file. (Technically, her access is Modify because the Modify permission includes Read access.) The only exception, of course, would be that any explicit "deny" permissions would override any allowed permissions.

Permissions and Special Groups

- Windows includes several "special" groups:
  - Members cannot be viewed directly
  - Membership may be dynamic
  - Groups can be used when assigning rights or permissions:
    - May inadvertently grant excessive permissions via these groups
  - Everyone, Authenticated Users, Interactive, Network, Creator/Owner…

When you audit rights and permissions, keep in mind that Windows uses several "special" groups to grant access to resources. These "special" groups are not visible in the standard list of users and groups visible through Computer Management or Active Directory Users and Computers.

For example, two of the special groups used by Windows are the **interactive** and **network** groups. These groups are used to represent the set of users who are currently logged in locally (interactively) and via the network, respectively. So, the groups' memberships consist of whatever set of users is currently logged in to the system using that particular method.

The two special groups you will most often encounter when reviewing rights and permissions are **Everyone** and **Authenticated Users**. The Everyone group is a holdover from the early days of Windows NT. Under 2000 and below, Everyone includes, literally, ***everyone***. With 2003, a change was made in the security system so that Everyone no longer includes anonymous or unauthenticated users. In fact, today these two groups are essentially equivalent.

Keep in mind that there are cases in which even Authenticated Users may provide too much access because it can include accounts such as Guest, IUSR, and IWAM. In such situations, you can use the standard Users group to limit access to those who are explicitly members of the Users group (which would generally exclude Guests and more).

Finally, you may see permissions entries for another "special" group called **Creator/Owner**. Creator/Owner is most often used to grant access at the directory level and is intended to apply to the contents of that directory: That is, subdirectories and files. Creator/Owner is the individual who creates an object and is, therefore, the owner of that object. As the owner of an object, the individual has full control over the object, including the ability to modify permissions. It is frequently used on shared folders where multiple users need access. Creator/Owner may be given Full Control over the folder and a different group (for example, Sales) may be given Read access.

Windows Share Permissions

- "Shared" objects are made available to other users across network
- Must be explicitly shared:
  - Administrative shares created by default
  - Default permissions grant full access:
    - And that makes sense
- Shares have permissions, too
- Weak/nonexistent share permissions are a top security issue

Another critical element to check when reviewing permissions on a Windows system are the permissions assigned to shared objects, or "shares" for short. Windows allows objects (most commonly directories and printers) to be "shared"—made accessible to other users on the network.

Windows does not create publicly accessible shares by default; resources must be explicitly made available to others. However, Windows creates "administrative shares" automatically and shares the root of each drive, along with the Windows directory (\winnt\system32 or \windows\system32), and makes those shares available to Administrative users only. (C:\ is shared as C$, D:\ is shared as D$, and so on. The Windows system32 directory is shared as ADMIN$.) Administrative shares can be permanently disabled only by editing the registry.

Of greater concern are shares that administrators (or users) create. Shares can have high-level permissions assigned to them (read, modify, or full control), but the default permissions on modern Windows shares are Everyone = Read. This allows anyone, including null session users, to connect to the share. You may feel that this is bad, but it actually makes sense. If an administrator is creating a share, he obviously wants people to access it!

If there are inappropriate NTFS permissions set on the underlying filesystem (such as Everyone = Full Control), then anyone can access the data as well. (Note that when share permissions and NTFS permissions interact, it is the *most restrictive* set of permissions that apply. So, if the share permissions are set to Everyone = Full Control but the underlying NTFS permissions are set to Everyone = Read, then access would be limited to Read only.)

Not only can unprotected shares allow unauthorized access to data by users (or attackers), they are also frequently used by viruses and worms as one method of propagation and can allow malware to run rampant through a network. Viruses/worms such as Klez, Bugbear, and Opaserv all use unprotected shares to propagate (among other methods).

Get-SmbShare and Get-FileShare

- Get-SMBShare – retrieves a list of all server message block shares
  - Includes the inter-process communications share: IPC$
- Get-FileShare – returns list of all SMB shares for filesystems

```
PS C:\> Get-SmbShare

Name    ScopeName Path        Description
----    --------- ----        -----------
ADMIN$  *         C:\Windows  Remote Admin
C$      *         C:\         Default share
D$      *         D:\         Default share
IPC$    *                     Remote IPC
```


These two PowerShell commands are used to retrieve a list of Server Message Block (SMB) file shares on a system. Get-SMBShare returns all shares, even those used only for inter-process communications. Get-FileShare returns only those shares which are created for disk filesystems.

Each can be run against a remote server by invoking an existing CimInstance:

```
$cred=Get-Credential -UserName auditor -Message "Please enter a
password for auditor"
$cim = New-CimSession -ComputerName 507dc -Credential $cred
Get-SmbShare -CimSession $cim
Get-FileShare -CimSession $cim
```

Get-ACL

- Get-ACL retrieves information about access control lists on objects
- Examples:

```
Get-Acl -Path C:\Windows\system32\

Get-Acl HKCU:\Software\Microsoft\Windows
```


This page intentionally left blank.

Audit Objective: File Integrity

- Objective: Ensure key system or data files are not tampered with or modified
- Audit activities:
  - Tripwire
  - OSSEC

As a final note in discussing the security of local resources, we want to mention the issue of file integrity. File integrity attempts to ensure that critical files, whether OS binaries or sensitive data, have not been modified, replaced, or otherwise tampered with. The primary concerns with file integrity are that files have been modified intentionally or accidentally by legitimate users, or that files have been intentionally modified by an attacker. Users may accidentally overwrite data; administrators may accidentally overwrite a newer patch with an older one. Malicious users may attempt to damage or destroy (either obviously or subtly) critical data to take revenge on an employer. Attackers may attempt to replace legitimate system files with trojaned copies. How can you detect these types of changes?

As an auditor, the scope of your audit may or may not extend to verifying file integrity. In general, if the audit requires this level of detailed inspection, then there is already a suspicion that something is wrong—either a host has been compromised or a user is engaging in illicit activity such as fraud, and an investigation is underway. Under such circumstances, the investigation will probably be handled by incident response personnel instead of auditors, though this may not always be the case. Either way, auditors should be familiar with the issue of file integrity and have knowledge of various tools that can be used to ensure it. To do a good job here, though, you need a commercial solution.

Tool: OSSEC

- Tripwire is the most commonly installed solution
  - If licensing is an obstacle or will take time, consider OSSEC
- OSSEC:
  - Most operating systems
  - Centralized monitoring
  - Definitely requires effort
    - No real interface
    - Heavy reliance on SIEM

```
2018 May 12 10:12:31 Received From: windows-ossec->syscheck Rule: 122 fired (level 7) -> "Integrity checksum changed for 'c:/windows/system32/config/etc/hosts' Size changed from '46' to '123'

Old md5sum: 'a62de8f1b231482bf1e3a2341fedaa32'
New md5sum: '29381736482fed8abcde924b32fe8d12'
Old sha1sum: '1938de7f4cd425abbee1986b30d5c542b468cbbd'
New sha1sum: 'bc6efd7af2410c73a74251ad217fefeef133b8e9'
```


The best method for validating file integrity is through the use of a **cryptographic hash**. A hash is a one-way encryption algorithm that takes input of variable length (that is, any file) and generates a fixed-length output (normally 128 bits or 160 bits in length). The key aspect of a hash (for the purposes of file integrity) is that a hash value is, for all practical purposes, unique. No two files generate the same hash value, and if a file is modified, even slightly, the hash value of the modified file will be different from the hash value of the original.

If you generate cryptographic hashes of key files when they are in a "known good" state, and then check those hashes at a later time, if the hashes have changed, then you know the files have been modified. Although there are commercial products that can generate and monitor hash values, you can also do this on your own. All you need is a simple utility that can generate the hash values for you. (MD5 and SHA-1 are the most common algorithms used.)

There are many tools available to support this. The most commonly used in the enterprise is Tripwire. If your organization isn't using something on workstations within your domain to monitor for file and registry changes, and the cost of Tripwire is too high, you might consider recommending OSSEC (https://u.aud507.com/4-23). This tool does support central monitoring, reporting, and alerting, in addition to being able to perform file and registry monitoring. It can also perform pretty much any test on a local Windows computer that you would like to automate.

As good as this sounds, be aware that installing, managing, and working with OSSEC will definitely require effort. Not only is there currently no management interface (there was one, but development ceased some time ago), but all of the configuration and customization is the responsibility of the administrator through the use of text files. This makes the requirement of a useable central log management solution very important. In the slide you can see an example alert being generated by the “syscheck” agent, successfully identifying a modification to the “'hosts” file on a Windows system.

Audit Interview Questions (2)

- Is there policy regarding permissions, authorization, or access to data/resources?
- Are permissions reviewed or managed for OS and application binaries?
- Are permissions managed for data? Who grants access to data? Are permissions reviewed periodically?

Some non-technical issues to consider in your audit include the following:

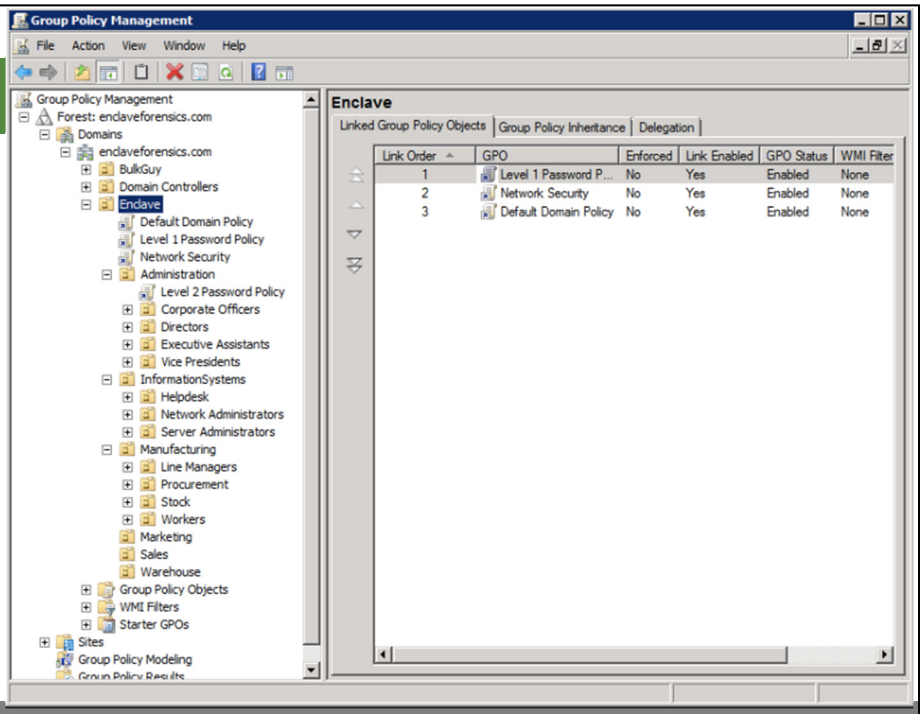
- Is there policy regarding permissions, authorization, or access to data/resources? This may include procedures that specify permissions that should be set on operating system files and directories, or policy that specifies who is responsible for approving access to data (project files, proposals, financial information, and more).

- Are permissions reviewed or managed for OS and application binaries? Operating system and program files should be reviewed to ensure that only authorized users have access to files. In most cases, only administrators should modify or delete program binaries. Users should require only read/execute access to program files, and then only to those applications that they actually need to run.

- Are permissions managed for data? Who grants access to data? Are permissions reviewed periodically? An organization's data is generally its "crown jewels" in some form, whether that is financial data, customer lists, contracts or proposals, source code, design, and engineering data… you name it. As potentially the most valuable resource in the organization, such information must be protected. An organization should have policies and procedures for granting, managing, and revoking access to sensitive data. Permissions should also be reviewed periodically to ensure they are still appropriate.

Group Policy

- The right way to manage Windows settings



The primary tool for managing security in a domain is Group Policy. The Group Policy Management Console (pictured in the slide) allows you to view or modify group policy objects (GPOs, which are collections of settings) and select which organizational units (OUs) they are applied to within the domain. One of the first things worth noting is that group policies are not applied to groups. In fact, they cannot be applied to groups. They can only be applied to organizational units.

This OU structure is the first thing we want to take a close look at. Once we are satisfied that the structure makes sense, we will consider the policies themselves.

Group Policy Questions

- What you're looking for:
  - Evidence of a consistent and considered design
  - Enterprise requirements expressed in default domain policy
  - Issue-specific policies broken out into specific policies and applied appropriately

So, what exactly should you be looking for in terms of Group Policy when you examine an Active Directory? There are three primary things you are interested in.

The first is that you are looking for evidence that the OU (Organizational Unit) structure within Group Policy is reflective of the organization in some logical way and that it appears to have some logical consistency. An administrator should explain the organizing principles behind the design of the structure, proving that it is a thoughtful design rather than an ad-hoc expression of the organization.

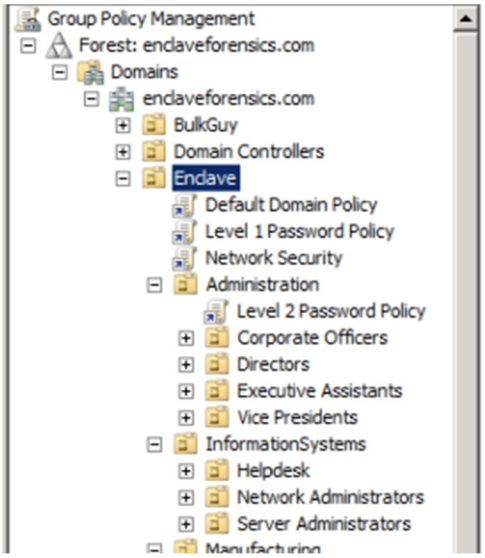
There are strong opinions as to whether this structure should be geographic, functional, or follow some other design structure. Frankly, we don't care what the structure is. Do not try to inflict your personal preferences on the business. Simply verify that the structure was actually *designed* rather than *evolved*.

Next, we should find that every single requirement that applies throughout the enterprise has been documented somewhere in the Group Policies. Typically, you find enterprise-wide requirements within the Default Domain Policy, the policy that is applied to every OU within the enterprise by default.

Finally, it is rare to find that there are no issue-specific policies that have been created. These can deal with by allowing an exception to an enterprise-wide policy, a requirement for a tighter policy, a Group Policy Object (GPO) that has been created to address some type of permissions or rights conflict, and so on. These policies should be broken out and address specific things. For example, if you are looking at a GPO titled "Network Security Settings," you would be extremely surprised to find something about password settings or screen saver settings within it!

Group Policy Structure

- Evidence of conscious design?
- Connection with organizational structure or function?
- Layered approach toward policy?



For example, consider the group policy OU structure in the slide. To view this structure, you must open the Group Policy Management Console.

Looking at the structure on the right side of the slide, consider these questions:

- Is there some evidence of a conscious design? Clearly, the answer is yes. Although we may personally prefer an approach that creates geographical containers at the top level, this organization has chosen to use a more functional approach. Still, we can see that this structure didn't just "happen"; it was designed.

- Is there some connection between the OU structure and the organization? Again, the answer is clearly yes. We can see high-level containers representing everything from the organization in general down to high-level roles or departments within divisions in the organization.

- Is there evidence of a layered approach toward security? Again, the answer is yes. Group Policy is applied in layers, like an onion. The policies furthest away or highest in the tree are applied first and then the rest of the policies are applied in turn, moving closer and closer to the OU in which an object resides. In this example, individuals within the Administration container would first have the Default Domain Policy, Level 1 Password Policy, and Network Security policy applied. Next, the Level 2 Password Policy would be applied.

- What happens when a closer policy differs from a policy previously applied? That depends. Typically, the closer policy overrides the policy that is further away. However, it is possible to mark a GPO as "No Override," which prevents any setting that it controls from being modified by a policy applied later in the chain.

Tool: Group Policy

- Useful to apply/enforce numerous settings:
  - **Computer** policies to HKEY_LOCAL_MACHINE at boot time
  - **User** policies to HKEY_CURRENT_USER at login time
- Group Policy can control wider range of settings
- **Group Policy has no built-in audit capability - it is an enforcement-only system**
- Use the Policy Analyzer from the Microsoft Security Compliance Toolkit to audit how group policies are applied

In a Windows domain, you can create custom templates, import them into Group Policy Objects, and automatically apply those GPOs to selected users and/or computers through Active Directory. Group Policy allows the functions of the Security Configuration and Analysis tool to be applied easily and automatically to multiple computers and users. A full discussion of Group Policy is beyond the scope of this course, but some of the advantages are described here.

Group Policy can be used to create policies/settings for both computers *and* users. Computer-based settings are applied to the HKEY_LOCAL_MACHINE registry hive when the system boots. User-based settings are applied to the HKEY_CURRENT_USER registry hive when the user logs in. Note that because user settings are triggered at login, users' security settings will "roam" with them. Regardless of which computer is used to access the network, the security configuration will be consistent.

One advantage of Group Policy is that security settings can be applied in "layers," with different settings applied at different levels of the domain structure (domain, OU, individual host, and so on). Group Policy has limited use from a strict audit standpoint because there is no built-in analysis function to compare Group Policy to a system's current state. Instead, Group Policy settings are simply reapplied every 90 minutes (plus or minus 30 minutes) for workstations and servers.

Keep in mind that when you audit security on a Windows system that is part of a Windows domain, that system may be subject to more than one security policy. In other words, it may be subject to various Group Policy Objects configured within the domain that are applied to the system when it boots. The gpresult.exe command-line tool or the Resultant Set of Policy Wizard/MMC snap-in can be used to help determine which policies are applied, in what order, and what the resultant (actual) settings are.

Microsoft has tools for managing and verifying group policies, gathered as the Microsoft Security Compliance Toolkit. One of the tools, Policy Analyzer, is built to verify that policy settings have been correctly applied to a system. The toolkit is available from: https://u.aud507.com/4-20

Audit Interview Questions (3)

- Has security policy within the organization been mapped to technical controls available within Security Templates/Group Policy?
- How much of the organization's security policy is (or could be) enforced or managed via these tools?

Group Policy and Security Templates are the two primary tools available from Microsoft to manage security (and other) configuration settings in a large environment. Group Policy, in particular, allows a central point of management for a range of computer security settings and user environment settings. In a Windows environment, these tools provide the primary technical controls that can be used to implement and enforce an organization's security policies on servers and workstations.

As part of your audit, it is worthwhile to consider whether the organization's existing security policies have been reviewed to see whether/where they map to controls that are available via Security Templates and Group Policy. In many cases, Security Templates and/or Group Policy can be designed that will help you to implement applicable portions of your written policy. This is, of course, highly desirable in terms of consistent management of hosts and consistent implementation of policy. It is also useful from an audit/oversight standpoint for organizations that need to demonstrate due diligence and/or privacy compliance with legal or regulatory issues such as HIPAA, FISMA, Sarbanes-Oxley, or others.

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. Background and Plan
2. PowerShell and WMI
3. Windows Auditing
4. Users, Groups, and Privilege Management
5. System and Resource Security
6. **Windows Logging**
   - *Log Configuration and Recommendations*
   - *Centralized Log Management*
7. Continuous Monitoring

In this section of the course, we discuss Windows' auditing capabilities and effective strategies for reporting on events. In addition, we spend some time examining some of the oddities in Windows event logging that you should be aware of when auditing Windows events.

Audit Objective: Audit Trail

- Objective: Ensure security logging is enabled and configured
- Audit Activity:
  - Event Viewer
  - Audit Policy

Our audit would not be complete if we did not ensure that, in addition to being properly secured, Windows hosts have an appropriate audit trail to trace activity on the system and establish accountability for actions taken on the host. We need to confirm that auditing is both enabled, and properly configured to record events of interest.

Tool: Event Viewer

- Primary logging tool in Windows
- Track a wide range of successful and unsuccessful user activity, as well as some system activity
- Examples:
  - Logons/logoffs
  - Use of rights and permissions
  - Access to objects
  - Track processes

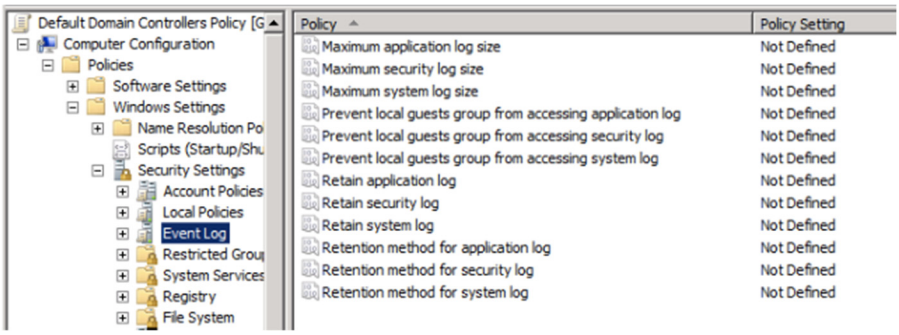
The primary tool for auditing and tracking user activity on a Windows system is the Windows Event Viewer. Windows can perform an extensive, highly detailed amount of logging and auditing. The trick is in knowing what to log, how much to log, and how to interpret and manage the information you collect.

Windows Event Viewer actually maintains three log files: The System log, which records system events; the Application log, which records events from various applications; and the Security log, which records security-related events. Windows servers also include separate, specific logs for monitoring DNS, Directory Services (Active Directory), and file replication activity, respectively, if these services are installed.

Other services and applications on Windows may have their own logs that are stored in various locations. For example, the Internet Information Services server (formerly known as Internet Information Server) maintains its own set of text logs to record web server connection attempts. The Windows backup utility (ntbackup.exe) keeps its own set of text logs as well. These logs may be useful for the correlation of events, but for security and auditing purposes, we will be primarily concerned with the Event Viewer Security log.

Configuring the Log Files

- Critical:
  - Size
  - Retention time/size and actions



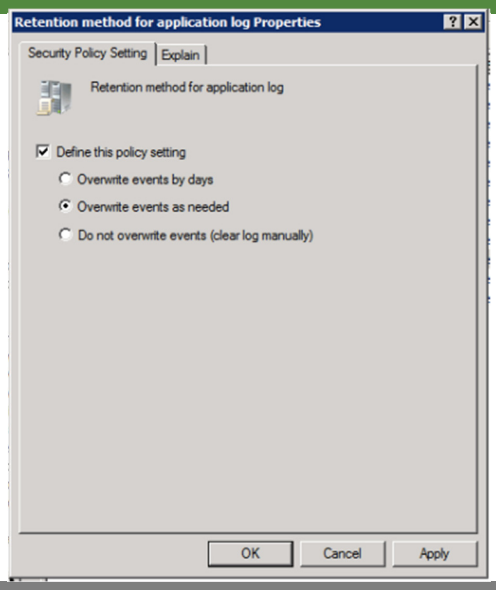
To be useful, the log files must be configured properly. The key elements to configure are the files' maximum size and the retention options.

Location: By default, the log files are located in the %systemroot%\system32\config directory. The files themselves have an *.evt extension (i.e., SecEvent.Evt). The location of the files can be changed by editing the registry value HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Eventlog\<log name>\File.

The amount of free disk space should be a consideration when deciding where to place the log files. Wherever the files are located, you should set appropriate NTFS permissions on the files themselves, as well as on the registry keys that control their configuration. Also, be sure that the size and wrapping options are set appropriately.

Retention Strategy

- Are there any good options here?
  - Overwrite as needed
  - Overwrite by days
  - Don't overwrite



Overwrite events as needed: When the maximum log file size is reached, events at the beginning of the log **will be overwritten** to make room for new events, regardless of the age of the events themselves.

Overwrite after X days (default is 7 days): When the maximum log file size is reached, logs older than the specified number of days **will be overwritten** to make room for new events. If there are NO events older than the specified age, system activity will continue, but **no new events will be logged.**

Do not overwrite events: Events will **not** be overwritten under any circumstances. The log must be manually cleared to make room for new events. If the log reaches its maximum size without being cleared, system activity will continue but **no new events will be logged.**

What's the right setting, then? The answer is that the administrator truly needs the ability to centralize these logs from important systems. To do so, the administrator needs to determine how quickly the logs are growing and based on that, how quickly the logs can be aggregated. With this information in hand, he should calculate how much space the logs need to have to be aggregated off the system before they begin to be overwritten.

I recommend, after this number is determined, that you at least double it! Things change over time and, of course, you must account for the aggregation server being offline for a period of time.

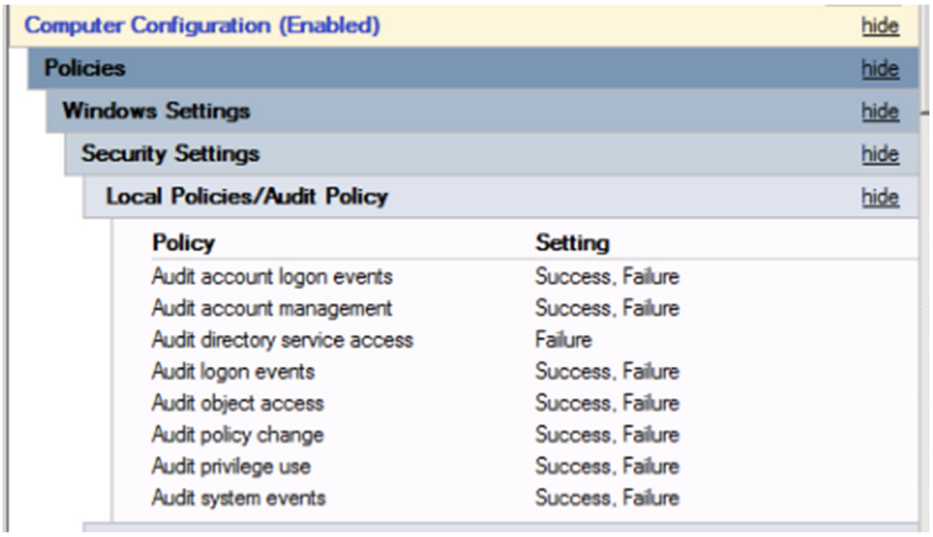
Tool: Audit Policy

- Historically, Windows does **little/no** security logging by default:
  - Modern versions turn on some auditing
- Ensure Audit Policy is appropriate:
  - What events are logged
  - Whether successful events, failed events, or both are logged

In addition to simply controlling access to objects on a system, you also want to have a record of that access: In short, a record of key events on the system. This is your primary audit trail, and you must use it to reconstruct system events, such as who performed certain actions, when those actions occurred, and what objects were acted upon. Auditing should, at a minimum, include a means to identify individual users and record significant events such as logons, logoffs, changes to security settings, and successful or failed access to critical files. Windows gives you the option to log successful events (for example, a user successfully logged on), failed events (for example, a user failed to log on for some reason), or both. You must decide what level of logging is both useful and appropriate for your environment.

Windows provides extensive and flexible auditing capabilities, primarily through the Event Viewer utility. With Windows, the problem is *not* an inability to log events; the problem is often that Windows can log *so* much information that it can be difficult to sort through and make sense of it. (Well… that and the fact that little to no security logging is done by default.)

What Should I Audit?



This screenshot is from a Default Domain Group Policy. These are possible suggested settings for your environment. Following are some basic definitions for the majority of these settings:

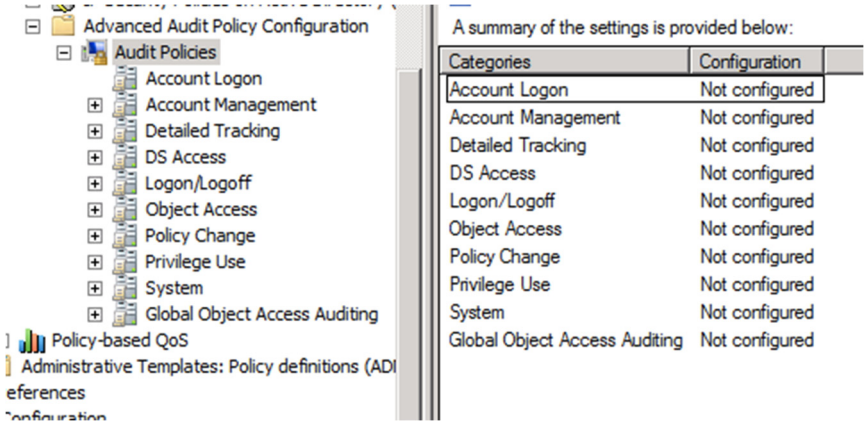
- **Audit account management:** Account and/or group creation or deletion, changes to group membership, and so on. (Recommended setting: Successful and failed events.)
- **Audit logon events:** Standard user and service logons and logoffs. (Recommended setting: Successful and failed events.)
- **Audit object access:** Access to all auditable objects, including files, directories, registry keys, and printers. Note that in addition to enabling auditing, you must also configure each object to be audited, the type of access to be audited, and the user(s) and/or group(s) whose access should be audited. (Recommended setting: Primarily failed events for selected objects, possibly both successful and failed events for critical files.)
- **Audit policy change:** Changes to the Audit policy, including enabling or disabling auditing, and assigning/revoking user privileges. (Recommended setting: Successful and failed events.)
- **Audit privilege use:** The use of user rights or privileges. Auditing this category can potentially generate a large number of events. (Recommended setting: None, or possibly failed events only.)

Auditing the use of user privileges can generate a large number of log entries, particularly if you audit "success" events. For this reason, Microsoft recommends that you not audit privilege use, or if you must audit it, that you audit only "failure" events.

For a complete list of Windows privileges, see https://u.aud507.com/4-18

2008 R2 and Above

- More granular settings are available
  - We recommend that you grab as much as you can, not seek to exclude



If you want, if you run Server 2008 R2 or above and if your desktops are all above Windows Vista, you may also set more granular audit policies. This way, you can choose to audit some aspects of directory service access, for example, without having to audit everything within the bigger bin.

This is a great feature, though I prefer to collect everything I possibly can. Unless something is actually crippling performance, it's hard to justify *not* logging it.

Finally, let's reiterate: If you're controlling pre-Vista systems (do you have any XP hosts?) then you *cannot* use these advanced settings. If you'd like to and you have a mixed environment, administrators could certainly create two sets of policies and logically segregate out the pre-Vista systems. Honestly, though, unless there is a driving need for a granular feature and I have a mixed environment, as an administrator, I would greatly prefer to deal with the big bins rather than manage groups of different kinds of systems with different GPOs being applied to them.

Logon Events

- Two types of events:
  - Audit Account Logon Events
  - Audit Logon Events
- What's the difference?!?
  - Account = authentication
  - Logon = session

Here's an important tip. When you look at the audit options for Windows logs, you see two types of logon events that you can audit: Account logon events and just plain-old logon events. So, what's the difference?

Account logon events are generated whenever someone authenticates by presenting credentials. This would include the initial logon to the desktop and is usually recorded at the domain controller.

Logon events, however, are generated every time there is a session. So, what's the difference? Imagine that you log on to your desktop in the morning. That would be an account logon event. After you log on, you browse out to a network share. When you access the network share, you generate a logon event! You can see that this can be confusing!

Auditing Logons/Logoffs

- In a domain, "most" logon activity will be recorded on the Domain Controllers
- Don’t overlook local logons to individual systems
- User sessions can be tracked by matching up Logon IDs

Logon activity and logoff activity are among the most critical events you will want to ensure are recorded on a Windows host. This is basic security information relating to who gained access to the system and when. This type of activity is logged if you enable auditing for "Logon events" (general authentication activity) and "Account Logon events" (domain/Kerberos-related activity).

In a Windows domain environment, most logon/logoff activity will be recorded on the Domain Controllers, so this information should be fairly consolidated for easy review. (This was not the case with NT, which logged even domain logons on the local workstations.) However, keep in mind that local logons to individual servers or workstations (for example, using local/nondomain accounts) will be recorded on those individual hosts. Don't neglect the logs from individual machines when considering all your audit data.

The information recorded when Windows logs a logon/logoff event can tell you useful information about how the logon occurred, such as whether it was a local (console) logon or a network logon. In addition, you can track the time and duration of an individual user's session by matching up the LogonID field from individual logon and logoff events.

Querying Logon Events with PowerShell

- Get-EventLog and Get-WinEvent used to retrieve events
- Most interesting information is in the Message or Properties field, which contains all of the event data
- Message field is built using ReplacementStrings array

```
Get-Eventlog security -InstanceId 4624 | select-object timegenerated,
@{Name="UserName";Expression={ $_.ReplacementStrings[5] }} | where { $_.username
-like 'student' }

TimeGenerated        UserName
-------------        --------
2/22/2019 3:26:06 PM student
2/22/2019 3:26:06 PM student
```


The most useful data in a Windows event log entry retrieved by Get-EventLog is often in the Message portion of the log, which is a long paragraph of text. This would be difficult to parse, except for the fact that the paragraph is actually built by plugging values into specific parts of a message template. Those values are usually what we're interested in, and they are saved in a PowerShell array called ReplacementStrings. The array will be different for each event type since the events all log different data.

For events retrieved by Get-WinEvent, the juicy information is in the Properties field of the record returned.

Audit Special Mentions

- Process Tracking:
  - All or nothing
  - This one can cripple performance
  - Lots of logs
- System Events:
  - Start up, shut down, loading auth packages
  - Some things are logged in odd places:
    - Starting/stopping event logger goes to "System," not "Security"
- You really need everything:
  - System, Security, Application and application-specific logs

Audit process tracking: Audits the creation and deletion of all processes on the system. Can potentially generate a vast number of events. (Recommended setting: None; though if you don't mind dealing with a lot of log data, this can be useful information.)

Auditing process tracking is an all-or-nothing affair: You either audit every process created (for success or failure or both) or no processes. You cannot pick and choose. Enabling auditing of process tracking can, therefore, generate a large amount of log data, particularly on busy systems.

Process tracking also records the Process ID of each new process. The Process ID can correlate a process's creation (Event ID 4688) with that process's termination (Event ID 4689) to determine how long a process ran.

Audit system events: Events such as system startup and shutdown, the loading of logon processes, authentication and notification packages, and clearing the audit/security log. (Recommended setting: Successful and failed events.)

If auditing is enabled for system events, Windows records events such as Windows starting, the loading of authentication packages (Event ID 4776), the registering of trusted logon processes (Event ID 4611), and the clearing of the Security log. These events are all recorded in the Security log.

Some additional security events of interest may be recorded in the System log. The System log automatically records when the Event Log service starts (Event ID 6005) or shuts down (Event ID 6006), which roughly correspond to Windows starting up or shutting down. An unexpected system restart records Event ID 6008. Event ID 6009 is logged whenever Windows starts and records the current build. Finally, Windows can be configured to write an event to the System log if a fatal exception/stop error (blue screen) occurs. This is Event ID 1001.

Auditing Policy Change

- Security log records:
  - Assignment or removal of user rights
  - Changes to domain trusts
  - Changes to audit policy
  - IPSec, Kerberos, Quality of Service, Encrypted Data policy changes

It is critical to monitor the audit policy for your Windows network. Policy changes should always be investigated to make sure they are legitimate. Attackers may try to modify your audit policy or disable auditing altogether to hide their activity on your network.

If you enable auditing of policy change events, Windows records information such as when the audit policy was modified (this includes whenever Group Policy is applied to the host, if Group Policy is used to configure audit settings); when domain trust information changes; and when specific policies (such as IPSec, Kerberos, or EFS policies) are modified.

Note that the assignment or removal of user rights and permissions (that is, change system time, add workstation to domain, and so on) is also audited under Policy Change (not under Account Management, as you might think).

For additional information on auditing policy changes, see https://u.aud507.com/4-24

Sample Policy Change Event

Event ID: 4715
Audit Policy Change:
New Policy:

Success	Failure		Changed By:
+	+	Logon/Logoff	**User Name: SQUREPANTS$**
+	+	Object Access	Domain Name: BIKINIBOTTOM
-	+	Privilege Use	Logon ID: (0x3E7)
-	-	Account Management	
+	+	Policy Change	
+	+	System	
-	-	Account Logon	

Event ID 4715 indicates that the audit policy has been changed for this system. The event shows the *current* (changed) audit policy but does not show you specifically what has changed. You would have to be familiar with your required audit policy to recognize what had changed or do additional research to find out what the previous policy was.

Note that under some circumstances, the User Name of the user who performed the action will be listed as the **computer account** (SQUARPANTS$ in this case) and not the name of a "normal" user. This indicates that the change was performed by the SYSTEM account, for example, by applying a security policy or Group Policy Object at boot time. You can also identify this based on the Logon ID. You will typically find a Logon ID of 0x3E7, 0x3E5, or something similar for SYSTEM account events.

A deficiency in the logging is that frequently there is not enough information in the events to figure out what is going on. In the event pictured here, we can see what the *new* policy is, but can you tell what has been changed?

While the settings are certainly suspicious (account logon events are not recorded and neither are account management events), there is no simple way to figure out what the settings were without searching for the previous 4715 event. What this means is that the most critical thing that you would want to know simply isn't recorded.

Audit Objective: Log Management

- Objective: Ensure logs are cleared, rotated, and/or consolidated
- Audit Activity:
  - Verify centralized log management
  - Verify some type of alerting/reporting

Understanding Windows auditing, configuring it properly, and tuning both your audit settings and your Event Log settings to support your audit policy is a challenge unto itself. But that's only half the battle! After you've done that, you need to consider how to *manage* your logs. You can log all the data and events in the world, but it is of no use to you if you can't take that data and somehow translate it into useful information.

Log Subscriptions

- New feature:
  - Manually add computers whose events should be forwarded
  - Ability to limit event collection based on criteria
- A great step forward:
  - It's a pain to subscribe to 10,000 systems
  - It also might not be a great idea…
  - …Windows event logs fall over when they get too large

Log Subscriptions is another new and fantastic feature that was introduced at the time that Vista was released. It enables you to configure a system to receive Windows events from other systems within the domain. You even have the ability to define alternative credentials that should be used to obtain the events from other systems.

The remote systems to which you are subscribing can either be configured manually, through Group Policy, or via the Event Viewer on the subscribing system. When creating a subscription to remote systems, you have the ability to define criteria that determines which events will be forwarded. You can choose to forward all events, which is a big step toward centralized aggregation and analysis.

It is, however, a bit of a pain to try to subscribe to large numbers of systems, which might be necessary when we consider where events are stored in a Windows domain. Also, even if we could easily subscribe to everything in the domain, it may not be the best strategy to try to use Event Viewer to manage aggregated logs from hundreds or thousands of systems.

A strategy that some administrators have used to store logs long term is to simply bump up the log size to several gigabytes. Although this doesn't sound like a bad idea and Windows cooperates happily, the system is headed for trouble. One day the system will begin randomly crashing. We're talking "blue screen of death" crashing. Troubleshoot all that you want; you're going to have a hard time finding the actual cause… your log file has grown to several hundred megabytes.

I don't believe a study has been conducted to figure out exactly what the magic number is but suffice it to say that when Windows log gets to a certain size, the server begins to fall over randomly. Not a great log management strategy!

Managing Log Files

- Logs must be centralized:
  - Otherwise, logs overwrite or halt
- Aggregation must be continuous:
  - Attackers attempt to clear logs; keep a secure copy elsewhere.
- These days you *must* have a SIEM:
  - ELK, Splunk, Arcsight, SCOM/ACS, and so on

One of our first priorities is managing the Event Logs. We discussed this topic in part earlier when we discussed Event Log configuration. You need to manage the logs to make sure there is always room in the logs for new events. This can be accomplished by setting the log file size large enough to hold data for X number of days; then configuring the files to overwrite after X number of days. If you back up your files every X days or less, you will not overwrite log events.

The problem with this method is that if an unusually high amount of logging occurs, events can still be overwritten or not logged. So, the best approach is to set your Event Logs to NOT overwrite at all, and make sure you back up and clear your logs on a regular basis. Unfortunately, the only method that Windows provides for clearing the logs is manually through the Event Viewer GUI, which is not a practical solution if you have more than a few machines to take care of!

Another problem we face is how to consolidate, sort, or otherwise manage the log file data. Event Viewer provides some filtering and sorting capabilities, but they are limited, and it is not possible to search for keywords or to sort on all fields. It would be nice if we could:

- Archive log data to a secure, central location
- Put data into a form that is more easily parsed or queried

Therefore, you *must* have some type of centralized system that is aggregating, correlating, and hopefully even alerting on logged events. We aren't particular about which you have, but you must have *something*.

Course Roadmap

- Enterprise Audit Fundamentals; Discovery and Scanning Tools
- **PowerShell, Windows System, and Domain Auditing**
- Advanced UNIX Auditing and Monitoring
- Auditing Private and Public Clouds, Containers, and Networks
- Auditing Web Applications
- Audit Wars!

Section Two

1. Background and Plan
2. PowerShell and WMI
3. Windows Auditing
4. Users, Groups, and Privilege Management
5. System and Resource Security
6. Windows Logging
7. **Continuous Monitoring**
   - *Scripting, Auditing, and Monitoring*
   - Exercise 2.5: Permissions, Rights, and Logging

Let's put everything we've talked about into some perspective.

Continuous Monitoring

- You will go through a lot of work to determine what constitutes a "secure" system
- After you have your system configured, take a **baseline** of this information
- Periodically re-baseline and look for changes
- Allows you to monitor for security problems over time

When you have your system configured just the way you want it, take a snapshot of the system to create a **baseline** of what your proper, secure configuration should look like. After you have this baseline, you can periodically re-baseline the system (take another snapshot) and look for changes. If you detect any changes, you can then investigate to see whether the change is legitimate or not. (Did someone install a new software package? Did someone misconfigure the box and turn on something they shouldn't have? Or have I been "0wn3d" and are "5cr1pt k1dd13z" now uploading their warez to my web server?)

Note that there may be some information on your system that you want to audit or check over time that is not easily validated using standard tools such as Security Configuration and Analysis or vulnerability scanners. For example, SCA can't double-check your list of open ports, or make sure that no mysterious new drivers, services, or users have shown up on your system. Not to fear; we can use our command-line utilities to automatically check this information as well.

Remember the Scripts?

- Think about the script that you wrote this morning.
  - Think about the scripts in the Scripts folder on your Windows 10 VM.
- Did you see things today that you know your team should monitor for?
- Can you see how to script these things?
  - Can you see how to monitor *continuously*?

Think back to the basic script that you wrote this morning in the first lab. Take into consideration the scripts included in the Scripts folder on the Windows 10 VM. Now reflect on everything that we've spoken about throughout this section. Can you see the power of scripting?

Hopefully, as we covered material, you heard and saw things that made you say, "Wow, I need to do that!" If you felt that way at any point during the material, think about this question: Could you, with a little bit of effort, turn that thing you saw into a couple of lines in a script? Could you create a test, whether it's a baseline test or some other test, to generate an alert if the setting changes or is incorrect?

We hope that it is clear that automation is the way to go. There's just no way for an administrator to keep track of everything manually. More than this, though, it is absolutely critical that we build scalable monitoring systems! Hopefully, you appreciate that you can create extremely powerful continuous monitoring systems that scale out to your enterprise domain environment for nothing more than an investment in time!

How to Apply This

- Auditors are not the people who go and automate "the things."
  - Can your research help you identify what might be automated?
  - Can your research reveal things that could be monitored?
- Is it unrealistic to believe that administrators are responsible for the ongoing security of their systems?
  - How valuable is automation for these tasks?
  - How does your organization measure up to current administrative trends?

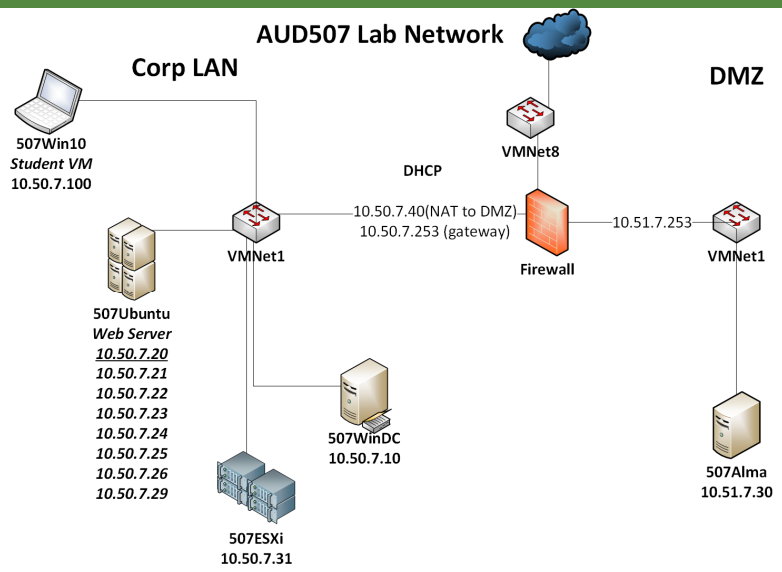
Since auditors really aren't the right people to be writing scripts and automating administration and monitoring tasks, what is the big takeaway?

We hope that, as you've covered the material, you have seen and heard things that were new, helpful, and possibly big potential improvements for your organization. When you think about those kinds of things, we hope that you can visualize how many, if not most or all, of those things can be automated with a script of some kind.

If *you* can imagine how those things can be automated, shouldn't an administrator *already be doing it?* We would argue, "Yes!" The industry has steadily been moving toward a higher and higher ratio of users and systems per administrator. Where is your organization on that continuum? The only way to effectively reduce IT staff while maintaining a high level of quality and service is to develop and document strong practices while simultaneously automating and monitoring as much as is possible.

If your administrators invest time into automation and automated monitoring, they will be able to move from the typical reactionary model of IT support to proactive support and strategic planning. As auditors, we are in a wonderful position to stand back, observe what's happening, and make recommendations that can help the organization to mature and improve!

Exercise 2.5: Permissions, Rights, and Logging



Feel free to pick up on whichever lab you are working on; you do not need to move forward if you are not ready. If you are ready, please move on to Exercise 2.5: Permissions, Rights, and Logging.

.

Daily Status Update Agenda

- Fieldwork completed today: Audited AD DC:
  - Well over 50 Domain admins
  - Number of Schema Admins?
  - Stale user accounts?
  - Windows DC well patched?
  - Windows 10 well patched?
- Other issues?
- Questions for auditee?

Think back on the fieldwork you completed today.

What sort of issues did you notice with regard to least privilege and configuration management?

Thank You!

This brings us to the end of Section 2. If you are taking the class at a conference, please take a moment to complete an evaluation form. You will be given a different evaluation after each section of the class.