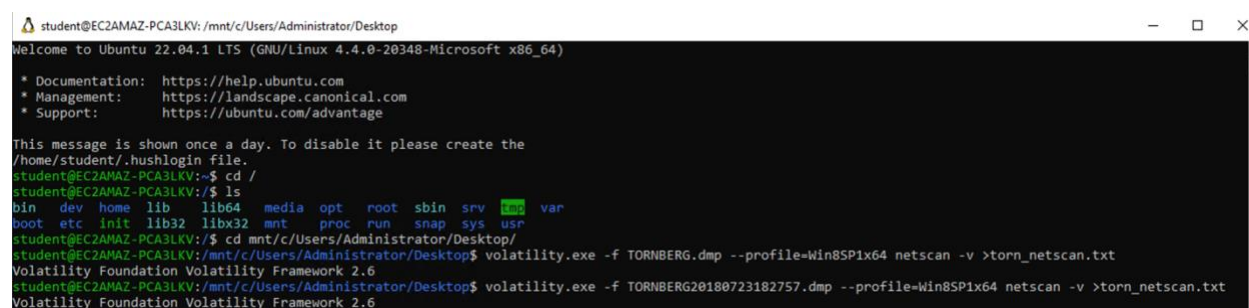


Volatility -f TORNBERG.dmp --profile=Win8SP1x64 netscan -v >torn_netscan.txt

In this task, the objective was to analyze network connections from a memory dump using the volatility tool within the Kali Linux Windows Subsystem for Linux. The assignment required extracting data from a memory file called TORNBERG.dmp to identify network connections and pinpoint the process responsible for an established connection on a specific remote port.

To carry out this process, the following command was used:

volatility -f TORNBERG.dmp --profile=Win8SP1x64 netscan -v >torn_netscan.txt



```

student@EC2AMAZ-PCA3LKV: /mnt/c/Users/Administrator/Desktop
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 4.4.0-20348-Microsoft x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/home/student/.hushlogin file.
student@EC2AMAZ-PCA3LKV:~$ cd /
student@EC2AMAZ-PCA3LKV:/$ ls
bin  dev  home  lib  lib64  media  opt  root  sbin  srv  tmp  var
boot  etc  init  lib32  libx32  mnt  proc  run  snap  sys  usr
student@EC2AMAZ-PCA3LKV:/$ cd /mnt/c/Users/Administrator/Desktop/
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBERG.dmp --profile=Win8SP1x64 netscan -v >torn_netscan.txt
Volatility Foundation Volatility Framework 2.6
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBERG20180723182757.dmp --profile=Win8SP1x64 netscan -v >torn_netscan.txt
Volatility Foundation Volatility Framework 2.6

```

This command generates a text file torn_netscan.txt containing a detailed list of all active and listening network connections that existed at the time the memory capture was taken. By opening this file in notepad++, we were able to examine the network activity and focus on

identifying a connection made to remote port 4444.

```

103 0x21f84ec0 UDPv4 0.0.0.0:0 *:* 980 svchost.exe 2018-07-23 18:27:35 UTC+0000
104 0x21f84ec0 UDPv6 :::0 *:* 980 svchost.exe 2018-07-23 18:27:35 UTC+0000
105 0x2440aec0 UDPv4 0.0.0.0:0 *:* 2416 Traffic.Client 2018-07-11 00:45:01 UTC+0000
106 0x2440aec0 UDPv6 :::0 *:* 2416 Traffic.Client 2018-07-11 00:45:01 UTC+0000
107 0x2441ea20 UDPv4 0.0.0.0:0 *:* 2416 Traffic.Client 2018-07-11 00:45:01 UTC+0000
108 0x24421010 UDPv4 0.0.0.0:0 *:* 2012 EmailGenerator 2018-07-11 00:45:01 UTC+0000
109 0x24421010 UDPv6 :::0 *:* 2012 EmailGenerator 2018-07-11 00:45:01 UTC+0000
110 0x24453270 UDPv4 0.0.0.0:0 *:* 980 svchost.exe 2018-07-23 18:27:35 UTC+0000
111 0x3d074100 TCPv4 10.252.9.131:49633 10.252.1.74:445 CLOSED 4 System
112 0x3f0f5170 UDPv4 0.0.0.0:0 *:* 980 svchost.exe 2018-07-23 18:27:35 UTC+0000
113 0x3f19c930 UDPv4 0.0.0.0:0 *:* 5012 CarbonSpector. 2018-07-23 18:27:24 UTC+0000
114 0x3f111270 TCPv4 0.0.0.0:49153 0.0.0.0:0 LISTENING 864 svchost.exe
115 0x3f111270 TCPv6 :::49153 :::0 LISTENING 864 svchost.exe
116 0x3f1113b0 TCPv4 0.0.0.0:49153 0.0.0.0:0 LISTENING 864 svchost.exe
117 0x3f1afed0 TCPv4 0.0.0.0:445 0.0.0.0:0 LISTENING 4 System
118 0x3f1afed0 TCPv6 :::445 :::0 LISTENING 4 System
119 0x3f1abc40 TCPv4 10.252.9.131:49633 10.139.205.204:4444 ESTABLISHED 1748 powershell.exe
120

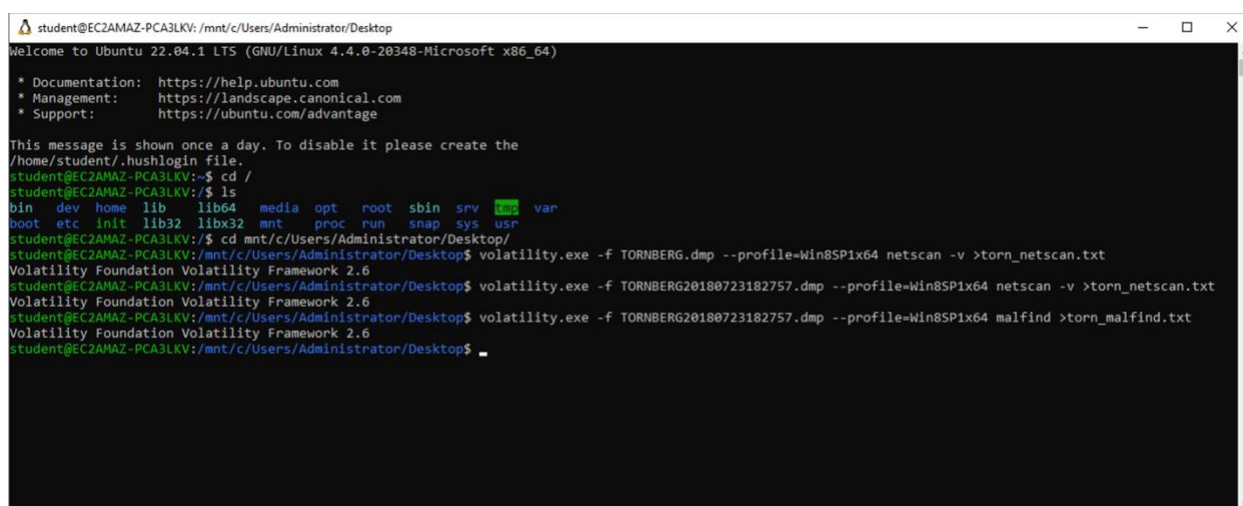
```

Upon reviewing the file, the relevant connection was found on line 119. This shows that the local IP address 10.252.9.131 on port 49633 was connected to the remote IP address 10.139.205.204 on port 4444. The connection is marked as established, and it is linked to the process powershell.exe.

The Process ID or PID for this particular connection is **1748**. This identifier is key to further analysis, allowing for tracking of the specific process that was responsible for the network activity. Powershell.exe is often used in system operations but can also be leveraged for nefarious purposes, making it an important detail in any network or system investigation.

Question 11: Volatility in the Kali Linux Windows Subsystem for Linux

In this task, the objective is to investigate a potential case of process injection using a memory dump file, with the help of the volatility toolset in a Kali Linux environment. Specifically, we need to identify the process responsible for injecting malicious code into a powershell.exe process with PID 1748. The correct answer, which emerges after carefully analyzing the memory artifacts, is 5012. The steps involved in reaching this conclusion are detailed below.



```

student@EC2AMAZ-PCA3LKV: /mnt/c/Users/Administrator/Desktop
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 4.4.0-20348-Microsoft x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

This message is shown once a day. To disable it please create the
/home/student/.hushlogin file.
student@EC2AMAZ-PCA3LKV:~$ cd /
student@EC2AMAZ-PCA3LKV:/$ ls
bin  dev  home  lib  lib64  media  opt  root  sbin  srv  sys  usr  var
boot  etc  init  lib32  libx32  mnt  proc  run  snap  sys  usr
student@EC2AMAZ-PCA3LKV:/$ cd /mnt/c/Users/Administrator/Desktop/
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBERG.dmp --profile=Win8SP1x64 netscan -v >torn_netscan.txt
Volatility Foundation Volatility Framework 2.6
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBERG20180723182757.dmp --profile=Win8SP1x64 netscan -v >torn_netscan.txt
Volatility Foundation Volatility Framework 2.6
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBERG20180723182757.dmp --profile=Win8SP1x64 malfind >torn_malfind.txt
Volatility Foundation Volatility Framework 2.6
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ _

```

The first step in the investigation is to run the malfind plugin on the memory dump. This command allows us to detect processes that show signs of being manipulated or injected with malicious code. After searching the output for entries related to PID 1748, we locate two relevant memory addresses. One of these addresses stands out because the data begins with the "MZ" signature, which indicates a Windows executable header. This signature is important because it suggests that the memory contains executable code, possibly the result of process injection.

```

C:\Users\Administrator\Desktop\tom_malfind.txt - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

tom_netscan.txt tom_malfind.txt tom_apihooks.txt tom_idmodules.txt

344 0x05740029 f07405 JZ 0x5740031
345 0x0574002c 0b00 OR EAX, [EAX]
346 0x0574002e 0000 ADD [EAX], AL
347 0x05740030 0100 ADD [EAX], EAX
348 0x05740032 0000 ADD [EAX], AL
349 0x05740034 0000 ADD [EAX], AL
350 0x05740036 0000 ADD [EAX], AL
351 0x05740038 f03f AAS
352 0x0574003a 7405 JZ 0x5740041
353 0x0574003c f03f AAS
354 0x0574003e 7405 JZ 0x5740045
355
356 Process: powershell.exe Pid: 1748 Address: 0x6db0000
357 Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
358 Flags: PrivateMemory: 1, Protection: 6
359
360 0x06db0000 4d 5a e8 00 00 00 5b 52 45 55 89 e5 81 c3 62 MZ.....[REU....b
361 0x06db0010 17 00 00 ff d3 81 c3 97 82 0e 00 89 3b 53 6a 04 .....:S).
362 0x06db0020 50 ff d0 00 00 00 00 00 00 00 00 00 00 00 00 F.....
363 0x06db0030 00 00 00 00 00 00 00 00 00 00 00 00 f8 00 00 00 .....
364
365 0x06db0000 4d DEC EBP
366 0x06db0001 5a POP EDX
367 0x06db0002 e800000000 CALL 0x6db0007
368 0x06db0007 5b POP EBX
369 0x06db0008 52 PUSH EDX
370 0x06db0009 45 INC EBP
371 0x06db000a 55 PUSH EBP
372 0x06db000b 89e5 MOV EBP, ESP
373 0x06db000d 81c362170000 ADD EBX, 0x1762
374 0x06db0013 ffd3 CALL EBX
375 0x06db0015 81c397820e00 ADD EBX, 0xe8297
376 0x06db001b 893b MOV [EBX], EDI
377 0x06db001d 53 PUSH EBX
378 0x06db001e 6a04 PUSH 0x4

Normal text file length: 30,236 lines: 689 Ln: 360 Col: 49 Pos: 15,803 Windows (CR LF) UTF-8 INS

```

```

C:\Users\Administrator\Desktop\tom_malfind.txt - Notepad++ [Administrator]
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

tom_netscan.txt tom_malfind.txt tom_apihooks.txt tom_idmodules.txt

463 0x06cc002b 0000 ADD [EAX], AL
464 0x06cc002d 0000 ADD [EAX], AL
465 0x06cc002f 0000 ADD [EAX], AL
466 0x06cc0031 0000 ADD [EAX], AL
467 0x06cc0033 0000 ADD [EAX], AL
468 0x06cc0035 0000 ADD [EAX], AL
469 0x06cc0037 0000 ADD [EAX], AL
470 0x06cc0039 0000 ADD [EAX], AL
471 0x06cc003b 00f8 ADD AL, BH
472 0x06cc003d 0000 ADD [EAX], AL
473 0x06cc003f 00 DB 0x0
474
475 Process: powershell.exe Pid: 1748 Address: 0x7130000
476 Vad Tag: VadS Protection: PAGE_EXECUTE_READWRITE
477 Flags: PrivateMemory: 1, Protection: 6
478
479 0x07130000 4d 5a 90 00 03 00 00 04 00 00 00 ff ff 00 00 MZ.....
480 0x07130010 b8 00 00 00 00 00 00 40 00 00 00 00 00 00 00 .....@.....
481 0x07130020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
482 0x07130030 00 00 00 00 00 00 00 00 00 00 00 00 f8 00 00 00 .....
483
484 0x07130000 4d DEC EBP
485 0x07130001 5a POP EDX
486 0x07130002 90 NOP
487 0x07130003 0003 ADD [EBX], AL
488 0x07130005 0000 ADD [EAX], AL
489 0x07130007 000400 ADD [EAX+EAX], AL
490 0x0713000a 0000 ADD [EAX], AL
491 0x0713000c ff DB 0xff
492 0x0713000d ff00 INC DWORD [EAX]
493 0x0713000f 00b800000000 ADD [EAX+0x0], BH
494 0x07130015 0000 ADD [EAX], AL
495 0x07130017 004000 ADD [EAX+0x0], AL
496 0x0713001a 0000 ADD [EAX], AL
497 0x0713001c 0000 ADD [EAX], AL

Normal text file length: 30,236 lines: 689 Ln: 479 Col: 65 Sel: 2 | 1 Windows (CR LF) UTF-8 INS

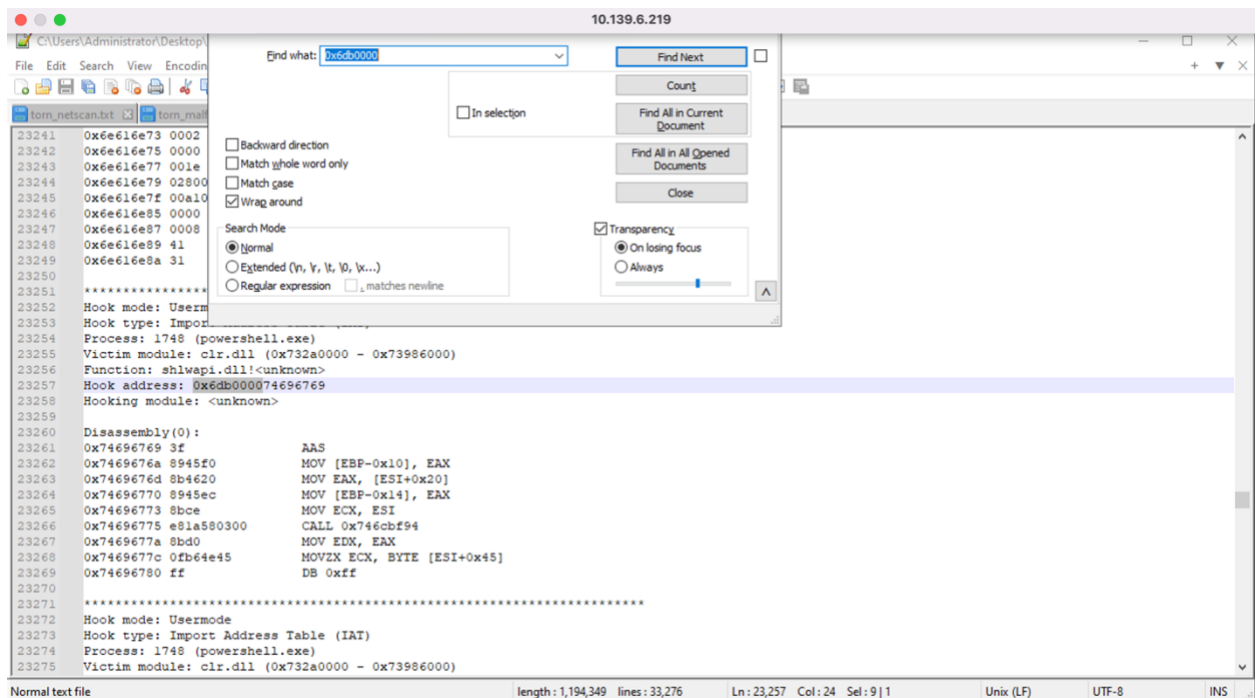
```

Next, we analyze this memory region further by using the apihooks plugin. This plugin scans for API hooking activity, a technique commonly used in malicious operations to intercept system function calls.

```

student@EC2AMAZ-PCA3LKV: /mnt/c/Users/Administrator/Desktop
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBORG20180723182757.dmp --profile=Win8S
P1x64 apihooks >torn_apihooks.txt
Volatility Foundation Volatility Framework 2.6

```



```

Find what: 0x6db0000
Find Next
Count
Find All in Current Document
Find All in All Opened Documents
Close
In selection
Backward direction
Match whole word only
Match case
Wrap around
Search Mode
Normal
Extended (n, y, l, b, v...)
Regular expression
Transparency
On losing focus
Always

```

```

23241 0x6e616e73 0002
23242 0x6e616e75 0000
23243 0x6e616e77 001e
23244 0x6e616e79 02800
23245 0x6e616e7f 00a10
23246 0x6e616e85 0000
23247 0x6e616e87 0008
23248 0x6e616e89 41
23249 0x6e616e8a 31
23250
23251 *****
23252 Hook mode: Usermode
23253 Hook type: Import Address Table (IAT)
23254 Process: 1748 (powershell.exe)
23255 Victim module: clr.dll (0x732a0000 - 0x73986000)
23256 Function: shlwapi.dll!<unknown>
23257 Hook address: 0x6db000074696769
23258 Hooking module: <unknown>
23259
23260 Disassembly(0):
23261 0x74696769 3f      AAS
23262 0x7469676a 8945f0    MOV [EBP-0x10], EAX
23263 0x7469676d 8b4620    MOV EAX, [ESI+0x20]
23264 0x74696770 8945ec    MOV [EBP-0x14], EAX
23265 0x74696773 8bce     MOV ECX, ESI
23266 0x74696775 e81a580300 CALL 0x746cbf94
23267 0x7469677a 8bd0     MOV EDX, EAX
23268 0x7469677c 0fb64e45  MOVZX ECX, BYTE [ESI+0x45]
23269 0x74696780 ff      DB 0xff
23270
23271 *****
23272 Hook mode: Usermode
23273 Hook type: Import Address Table (IAT)
23274 Process: 1748 (powershell.exe)
23275 Victim module: clr.dll (0x732a0000 - 0x73986000)

```

In this case, one of the memory addresses from the malfind output is flagged in the apihooks results, confirming that API hooking has occurred at this location. This is a critical finding, as it points to the likelihood that malicious activity is taking place within the hooked

memory region. The apihooks output further reveals that the hooked address is associated with a DLL located at 0x732a0000.

```

student@EC2AMAZ-PCA3LKV: /mnt/c/Users/Administrator/Desktop
student@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBORG20180723182757.dmp --profile=Win8S
P1x64 apihooks >torn_apihooks.txt
Volatility Foundation Volatility Framework 2.6

ls
^Cstudent@EC2AMAZ-PCA3LKV:/mnt/c/Users/Administrator/Desktop$ volatility.exe -f TORNBORG20180723182757.dmp --profile=Win
8SP1x64 Idrmodules >torn_Idrmodules.txt

```

The screenshot shows a Windows desktop with a Notepad++ window open. The window title is "C:\Users\Administrator\Desktop\ldrmodulesTORN.txt - Notepad++ [Administrator]". The window contains a list of loaded modules, with the following columns: Address, Name, Architecture, Type, and Path. The list is sorted by address, and the module "CarbonSpectator" is highlighted. A Find dialog box is open over the window, with the search text "2020000" and the "Find Next" button highlighted. The dialog box also includes options for "Match whole word only", "Match case", "Wrap around", "Search Mode" (Normal, Extended, Regular expression), and "Transparency" (On losing focus, Always).

Address	Name	Architecture	Type	Path
1378	5012 CarbonSpectator.	0x00007ffb5d150000	True	True
1379	5012 CarbonSpectator.	0x00000000739a0000	True	True
1380	5012 CarbonSpectator.	0x00000000731a0000	True	True
1381	5012 CarbonSpectator.	0x0000000074940000	True	True
1382	5012 CarbonSpectator.	0x00000000712c0000	True	True
1383	5012 CarbonSpectator.	0x00000000711e0000	True	True
1384	5012 CarbonSpectator.	0x000000006f200000	True	True
1385	5012 CarbonSpectator.	0x0000000075410000	True	True
1386	5012 CarbonSpectator.	0x00000000711d0000	True	True
1387	5012 CarbonSpectator.	0x000000006f300000	True	True
1388	5012 CarbonSpectator.	0x0000000074c30000	True	True
1389	5012 CarbonSpectator.	0x0000000074a40000	True	True
1390	5012 CarbonSpectator.	0x0000000077240000	True	True
1391	5012 CarbonSpectator.	0x000000006f260000	True	True
1392	5012 CarbonSpectator.	0x0000000075680000	True	True
1393	5012 CarbonSpectator.	0x00000000772b0000	True	True
1394	5012 CarbonSpectator.	0x00000000732a0000	True	True
1395	5012 CarbonSpectator.	0x00000000750b0000	True	True
1396	5012 CarbonSpectator.	0x0000000077130000	True	True
1397	5012 CarbonSpectator.	0x00000000706c0000	True	True
1398	5012 CarbonSpectator.	0x000000006f0d0000	True	True
1399	5012 CarbonSpectator.	0x00000000712e0000	True	True
1400	5012 CarbonSpectator.	0x0000000075140000	True	True
1401	5012 CarbonSpectator.	0x0000000074aa0000	True	True
1402	5012 CarbonSpectator.	0x0000000071300000	True	True
1403	2456 DumpIt.exe	0x00007ffb5a7f0000	True	True
1404	2456 DumpIt.exe	0x00007ffb58f10000	True	True
1405	2456 DumpIt.exe	0x00007ffb5a230000	True	True
1406	2456 DumpIt.exe	0x00007ffb5c0b0000	True	True
1407	2456 DumpIt.exe	0x00007ffb5ab60000	True	True
1408	2456 DumpIt.exe	0x00007ffb59d70000	True	True
1409	2456 DumpIt.exe	0x00007ffb5cfc0000	True	True
1410	2456 DumpIt.exe	0x00007ffb54990000	True	True
1411	2456 DumpIt.exe	0x00007ffb58f20000	True	True

Find dialog box details:

- Find what: 2020000
- Find Next button highlighted
- Search Mode: Normal (selected)
- Transparency: On losing focus (selected)

The next step involves determining which processes have loaded this suspicious DLL. To do this, we run the ldrmodules plugin, which lists all modules (such as DLLs) loaded into the memory of each process. By searching for 0x732a0000, we find that this DLL is present in two processes: PID 1748, the target process powershell.exe, and another process PID 5012. This duplication strongly suggests that PID **5012** played a role in injecting the DLL into PID 1748.

Question 12: Netscan Module on the Tornberg File.

In this task, we examined the memory image using the Volatility netscan module to analyze the network activity that took place at the time the memory was captured. The goal was to interpret the data and determine which statements accurately reflect the information found within the network scan results. Three specific statements have been identified as correct, based on the evidence provided by the analysis.

The first correct observation is that the IP address of the host system is 10.252.9.131. By reviewing the output of the netscan module, we can see this IP address associated with the local machine in several network connections. These connections indicate that the host's internal IP address within the network environment at the time of the capture was 10.252.9.131. This IP consistently appears as the local address in various communication sessions, solidifying it as the IP of the host system.

The second accurate statement relates to the process responsible for handling the remote connection, which is svchost.exe. In Windows environments, svchost.exe is a critical process that manages many essential system services. In this case, it was responsible for managing the network service tied to the remote connection. The netscan results reveal that svchost.exe was

linked to a network connection commonly associated with Remote Desktop Protocol (RDP), confirming that it handled the session. This is a typical role for svchost.exe, which often manages remote service protocols such as RDP.

Lastly, the analysis confirms that the host machine received a remote desktop connection. This conclusion is drawn from the presence of an established connection on port 3389, which is the standard port used by RDP. The netscan output clearly shows that the host system was listening on this port and that a remote session was initiated, indicating that the host was acting as the RDP server. This connection implies that an external machine accessed the host system via RDP, confirming the statement that the host received a remote desktop connection.

Conclusion

In summary, we used the Volatility tool to examine a memory dump and investigate network and memory activity. The primary goal was to identify active network connections, processes involved, and detect possible cases of process injection. The netscan module revealed that the host system's IP address was 10.252.9.131, and the process svchost.exe was responsible for managing a remote desktop connection. An established connection on port 3389 confirmed that the host had received a remote desktop session. Additionally, using the malfind and apihooks plugins, we identified that the process powershell.exe (PID 1748) had been injected with suspicious code, as seen by the presence of an executable code signature. Further investigation through the ldrmodules plugin showed that another process, PID 5012, had loaded the same suspicious DLL, suggesting that PID 5012 was responsible for the injection. By

combining these findings, we were able to trace key network activities and uncover the process injection, providing a detailed understanding of the system's state during the memory capture.