Memory forensics is the practice of analyzing the data stored in a computer's RAM to understand what was happening on the system at the time the memory was captured. RAM stores information that the computer is actively using, and a lot of this information never gets written to the hard drive. That makes RAM extremely valuable during an investigation because it can contain certain kinds of evidence that would normally be invisible or completely gone after the machine is shut down. This includes running processes, active network connections, open files, registry activity, data from programs the user was running, and even private information like usernames and passwords. All of this helps create a real picture of what the system was doing at that moment. This is why memory forensics is a core skill in incident response. It gives responders a quick and detailed look at what was happening. It can also reveal signs of an attack long before the attacker has time to delete anything or cover their tracks.

For this lab, I worked inside a Windows virtual machine that already had Volatility installed and ready to use. This made the entire process much easier. I did not have to worry about setting up the environment or installing different tools. I could just open the VM, launch Command Prompt, and start working immediately. The memory image file we used in the lab was already included in the VM as well, so all I had to do was run the correct Volatility commands and interpret the results. Because everything was prepared ahead of time, I was able to focus more on understanding the investigation rather than troubleshooting the tools.

The tool we used was Volatility. It is a memory analysis tool used by digital forensic analysts to extract valuable information from RAM. Investigators use it to answer questions like which programs were running, what files were open, what the user was doing, and whether any suspicious activity was taking place. Volatility breaks memory into different categories and gives you specific plugins designed to examine each one. This makes it easier to go through the memory step by step and slowly build an understanding of the system. For this lab, I used Volatility to examine a Windows XP memory image and pulled information such as process lists, loaded DLLs, user data, suspicious files, and other artifacts that showed evidence of malware activity. The purpose of the lab I think was to understand how each piece fits into the overall investigation.

The first thing I did was open Command Prompt and set up a doskey (Figure 1) alias so I could run Volatility commands faster. This helped a lot since memory analysis requires running many plugins, and typing the full Volatility command over and over again would be time consuming. With the alias in place, the first command I ran was imageinfo. This command verifies that Volatility can read the memory file and gives useful information about the operating system. It is a good starting point because it confirms that the memory file is valid and also helps Volatility decide which OS profile to use for deeper analysis. I also ran imageinfo to make sure it was working properly.

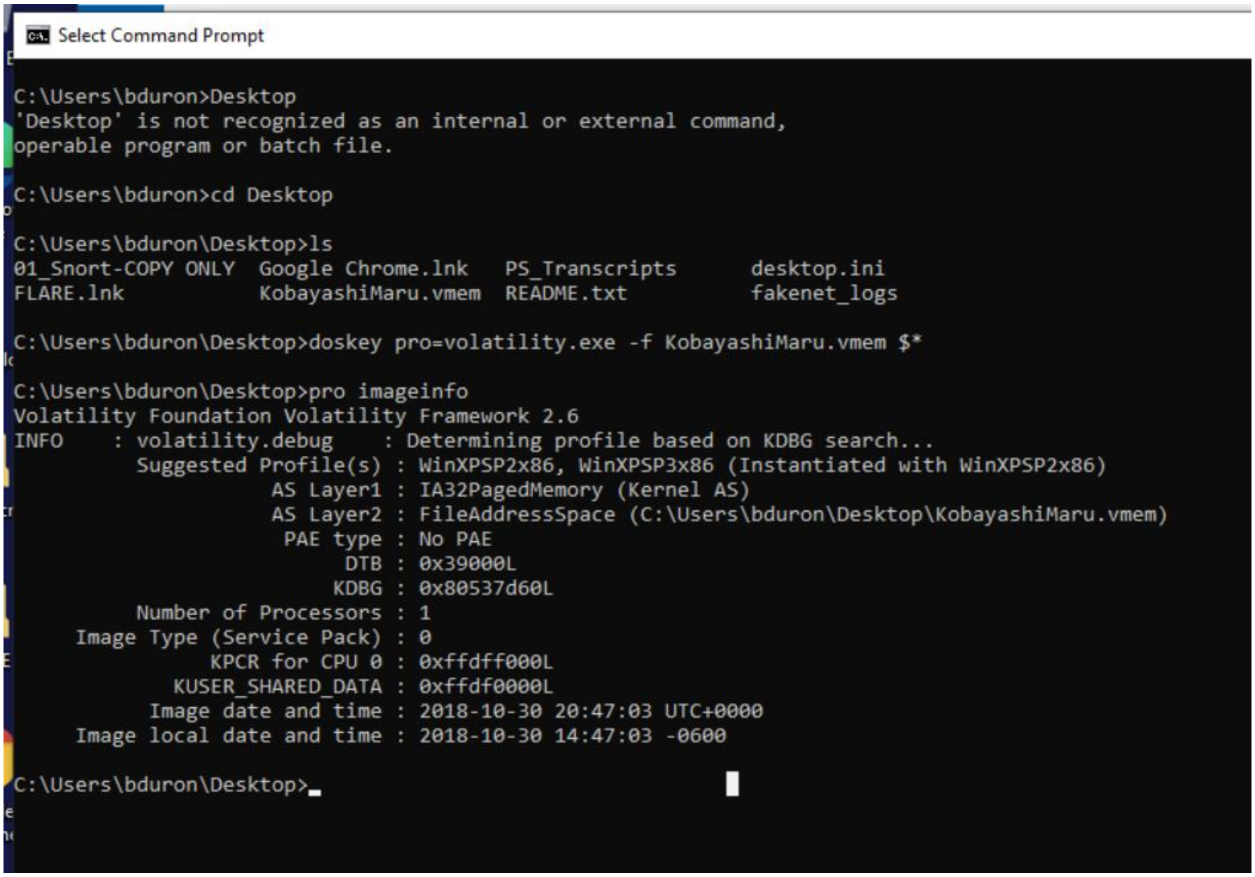


Figure 1

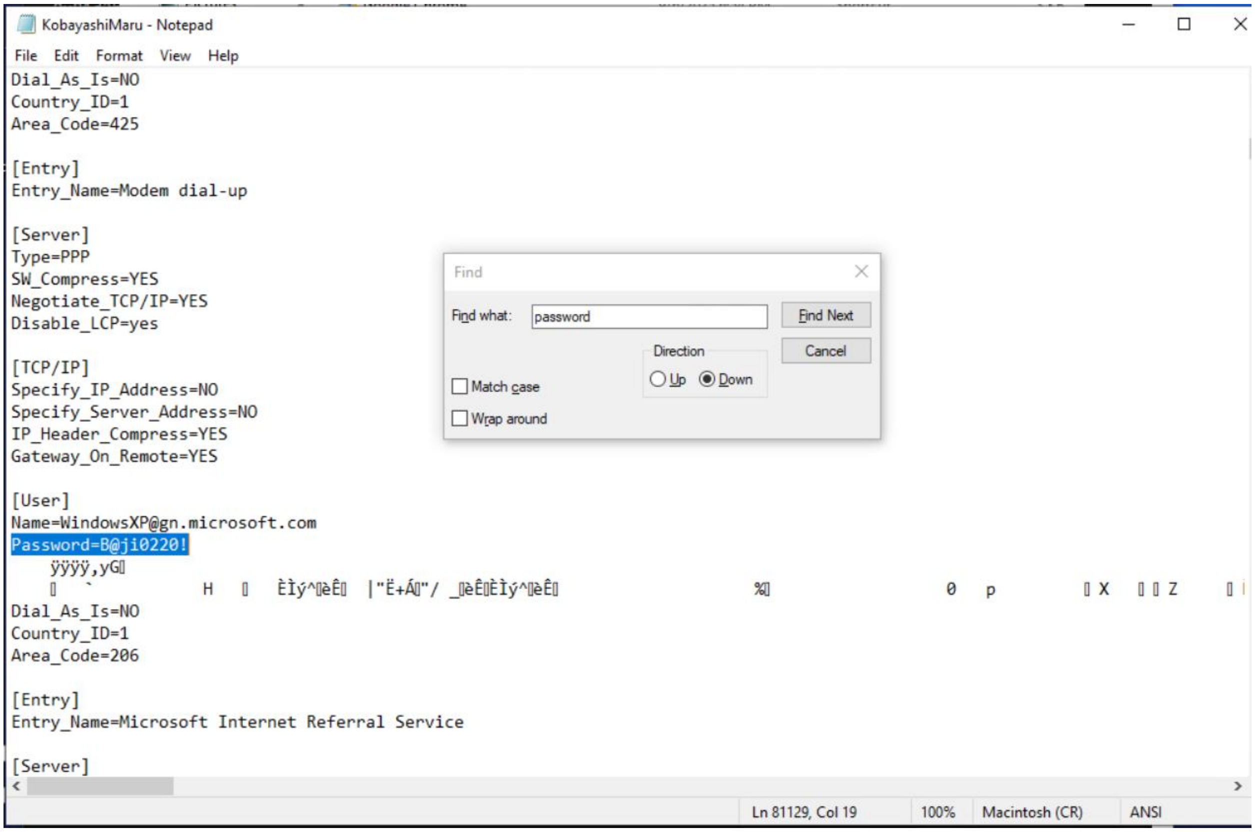
The output from imageinfo showed that the memory came from a Windows XP 32 bit system (Figure 1). The memory size was about one gigabyte which made sense for a machine from that era. This information also helped me understand the environment that the user was working in. Older systems like Windows XP have certain weaknesses and behaviors that make them easier targets for attackers since it is so old now. Knowing this made me expect that I might find clear text credentials or other personal information inside the memory. Once I had all this basic information confirmed, I could move forward with checking what was running on the system.

Next, I used the pstree (Figure 2) command to view all running processes at the time of the memory capture. This is one of the most important steps in memory forensics because it shows what programs the user and the system were executing when the memory was taken. Most of the processes looked normal for an XP machine. I saw the operating system processes that always run in the background, including winlogon, services, lsass, csrss, and explorer. These processes are expected. What stood out was a process called poisonivy.exe. This process does not belong on a Windows machine and is known to be part of a remote access trojan called Poison Ivy. This kind of malware is used by attackers to control a victim's computer remotely. The pstree output showed that explorer.exe launched poisonivy.exe which means the user executed it directly. This suggests the user probably opened a file or attachment that launched the malware. Seeing this connection was the first major sign that the system had been attacked.

```
C:\Users\bduron\Desktop>pro pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x81fcc800:System                                      4      0     54    275 1970-01-01 00:00:00 UTC+0000
. 0x81f07da8:smss.exe                                 336      4      3     21 2018-10-30 20:46:44 UTC+0000
.. 0x81d2b020:csrss.exe                               664    336     12    453 2018-10-30 20:46:45 UTC+0000
.. 0x81dc4020:winlogon.exe                            688    336     25    486 2018-10-30 20:46:45 UTC+0000
... 0x819efda8:services.exe                           732    688     18    390 2018-10-30 20:46:45 UTC+0000
.... 0x81d626a0:inetinfo.exe                         1432    732     34    540 2018-10-30 20:46:46 UTC+0000
.... 0x81db4298:hxdef100.exe                         1416    732      2     31 2018-10-30 20:46:46 UTC+0000
..... 0x81ede980:cryptcat.exe                        1472   1416      1     62 2018-10-30 20:46:47 UTC+0000
..... 0x81cada80:bircd.exe                           1480   1416      2     45 2018-10-30 20:46:47 UTC+0000
.... 0x81d32988:wmiapsrv.exe                          216    732      5    121 2018-10-30 20:46:36 UTC+0000
.... 0x819edda8:svchost.exe                           916    732      9    252 2018-10-30 20:46:45 UTC+0000
..... 0x819e83c8:wmiprvse.exe                         252    916      7    107 2018-10-30 20:46:37 UTC+0000
.... 0x81d976c8:svchost.exe                          1028    732      5     72 2018-10-30 20:46:45 UTC+0000
.... 0x81e536a0:spoolsv.exe                          1308    732     15    189 2018-10-30 20:46:46 UTC+0000
.... 0x819e2c20:jqs.exe                              1464    732      7    214 2018-10-30 20:46:47 UTC+0000
.... 0x81ee5500:svchost.exe                           960    732     70    875 2018-10-30 20:46:45 UTC+0000
.... 0x81e07da8:svchost.exe                          1108    732     12    142 2018-10-30 20:46:46 UTC+0000
.... 0x81c71508:VMwareService.e                      1624    732      2    119 2018-10-30 20:46:47 UTC+0000
.... 0x81e92418:vmacthlp.exe                          888    732      1     27 2018-10-30 20:46:45 UTC+0000
... 0x81b98da8:lsass.exe                              744    688     25    339 2018-10-30 20:46:45 UTC+0000
... 0x81b82638:logonui.exe                            636    688      4    133 2018-10-30 20:46:40 UTC+0000
... 0x81edfc18:userinit.exe                           368    688      2     34 2018-10-30 20:46:38 UTC+0000
.... 0x81a3bc18:explorer.exe                          404    368     15    252 2018-10-30 20:46:38 UTC+0000
..... 0x81d28790:VMwareTray.exe                       456    404      1     30 2018-10-30 20:46:38 UTC+0000
..... 0x81e234e8:poisonivy.exe                        480    404      1     20 2018-10-30 20:46:38 UTC+0000
..... 0x81bb3da8:VMwareUser.exe                       464    404      5    146 2018-10-30 20:46:38 UTC+0000
..... 0x81aaa708:jusched.exe                          472    404      1     24 2018-10-30 20:46:38 UTC+0000
..... 0x81cacda8:msmsgs.exe                           488    404      4    127 2018-10-30 20:46:39 UTC+0000
..... 0x81d40418:rundll32.exe                         984    404      1     81 2018-10-30 20:46:43 UTC+0000
 0x81e579f8:soffice.exe                               516    496      1     20 2018-10-30 20:46:39 UTC+0000
. 0x81ec6848:soffice.bin                              524    516      7    164 2018-10-30 20:46:39 UTC+0000
 0x81e8f9c0:iroffer.exe                              1692   1488      0 ------ 2018-10-30 20:46:47 UTC+0000
. 0x81c85420:iroffer.exe                             1728   1692      5     92 2018-10-30 20:46:47 UTC+0000
.. 0x81df6b20:iroffer.exe                            1824   1728      0 ------ 2018-10-30 20:46:47 UTC+0000
 0x81a2eb78:cmd.exe                                   560    508      1     20 2018-10-30 20:46:39 UTC+0000
 0x81eb3020:winvnc4.exe                               548    508      2     81 2018-10-30 20:46:39 UTC+0000
 0x81c6f7b8:nc.exe                                    532    508      1     62 2018-10-30 20:46:39 UTC+0000

C:\Users\bduron\Desktop>
```

Figure 2

In step four, I examined the memory file for user information. I opened the memory file with Notepad and searched for words like user and password. This can work on older systems because they often store this information in plain text inside RAM. With help from the instructions, I found the account name Daniel Faraday and the password B@ji0220! stored in the memory (Figures 3 and 4). This was a good example of why attackers often try to dump memory after exploiting a machine. If they get access to RAM, they can easily search for credential strings just like I did in the lab. Once they find these credentials, they can log into the machine or use the same password elsewhere if the victim reused it. This step showed how exposing a machine's memory can be a serious security risk and why investigators need to know how to analyze it.

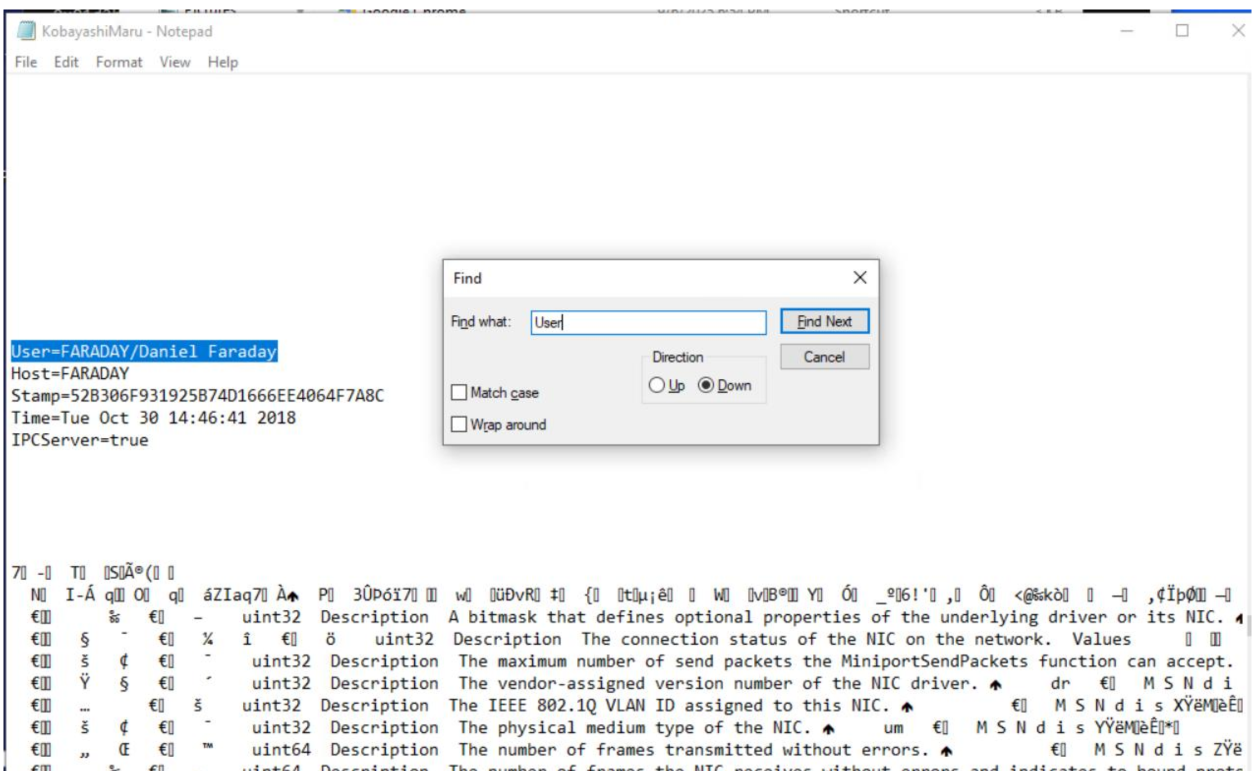


```
KobayashiMaru - Notepad
File Edit Format View Help
Dial_As_Is=NO
Country_ID=1
Area_Code=425

[Entry]
Entry_Name=Modem dial-up

[Server]
Type=PPP
SW_Compress=YES
Negotiate_TCP/IP=YES
Disable_LCP=yes

[TCP/IP]
Specify_IP_Address=NO
Specify_Server_Address=NO
IP_Header_Compress=YES
Gateway_On_Remote=YES

[User]
Name=WindowsXP@gn.microsoft.com
Password=B@ji0220!
    ÿÿÿÿ,yG
Dial_As_Is=NO
Country_ID=1
Area_Code=206

[Entry]
Entry_Name=Microsoft Internet Referral Service

[Server]
```

```
Find
Find what: password          Find Next
                             Cancel
            Direction
Match case   Up  Down
Wrap around
```

Ln 81129, Col 19 100% Macintosh (CR) ANSI

Figure 3



```
KobayashiMaru - Notepad
File Edit Format View Help
User=FARADAY/Daniel Faraday
Host=FARADAY
Stamp=52B306F931925B74D1666EE4064F7A8C
Time=Tue Oct 30 14:46:41 2018
IPCServer=true

uint32 Description A bitmask that defines optional properties of the underlying driver or its NIC.
uint32 Description The connection status of the NIC on the network. Values
uint32 Description The maximum number of send packets the MiniportSendPackets function can accept.
uint32 Description The vendor-assigned version number of the NIC driver. dr M S N d i
uint32 Description The IEEE 802.1Q VLAN ID assigned to this NIC. M S N d i s
uint32 Description The physical medium type of the NIC. um M S N d i s
uint64 Description The number of frames transmitted without errors. M S N d i s
```

```
Find
Find what: User          Find Next
                         Cancel
            Direction
Match case   Up  Down
Wrap around
```

Figure 4

After identifying the suspicious process and collecting the user information, I used the dlllist command to see which DLLs were loaded by poisonivy.exe (Figure 5). Understanding the DLLs loaded by a process helps determine what the program is doing. Some of the DLLs were standard Windows DLLs that almost every process loads. Others told me more about the malware's behavior. There were DLLs related to registry access, user interface interaction, and networking. These are all strongly associated with remote access trojans that need to communicate with a command and control server and possibly monitor user activity. Seeing these DLLs confirmed that poisonivy.exe was not just sitting idle on the system. It was fully active and performing tasks typical of malware that enables remote control.

```
************************************************************************
poisonivy.exe pid:    480
Command line : "C:\WINDOWS\System32\poisonivy.exe"

Base             Size  LoadCount Path
---------- ---------- ---------- ----
0x00400000     0x1c00     0xffff C:\WINDOWS\System32\poisonivy.exe
0x77f50000    0xa9000     0xffff C:\WINDOWS\System32\ntdll.dll
0x77e60000    0xe5000     0xffff C:\WINDOWS\system32\kernel32.dll
0x77dd0000    0x8b000       0x1a C:\WINDOWS\system32\advapi32.dll
0x77cc0000    0x75000        0xb C:\WINDOWS\system32\RPCRT4.dll
0x77d40000    0x8d000        0x5 C:\WINDOWS\system32\user32.dll
0x77c70000    0x40000        0x4 C:\WINDOWS\system32\GDI32.dll
0x75260000    0x27000        0x1 C:\WINDOWS\System32\advpack.dll
0x771b0000   0x11a000        0x1 C:\WINDOWS\system32\ole32.dll
0x77c00000     0x7000        0x1 C:\WINDOWS\system32\VERSION.dll
0x71ab0000    0x15000        0x5 C:\WINDOWS\System32\ws2_32.dll
0x77c10000    0x53000        0x8 C:\WINDOWS\system32\msvcrt.dll
0x71aa0000     0x8000        0x7 C:\WINDOWS\System32\WS2HELP.dll
0x71a50000    0x3b000        0x2 C:\WINDOWS\system32\mswsock.dll
0x71a90000     0x8000        0x1 C:\WINDOWS\System32\wshtcpip.dll
0x76fc0000     0x5000        0x1 C:\WINDOWS\System32\rasadhlp.dll
************************************************************************
```

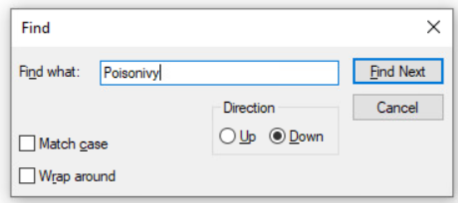


Figure 5

Step six involved using the cmdline plugin to see how the malware was executed. This plugin shows the full path that was used when the process started. The output for poisonivy.exe showed that it was running from C:\WINDOWS\System32\poisonivy.exe (Figure 6). This told me a few things. First, it confirmed again that the malware had been executed. Second, storing itself in the System32 folder is a method attackers use to make malware look more legitimate or blend in with the system. Most people trust files in that folder. Knowing that explorer.exe launched it and that it lived inside System32 helped complete the picture of how the attack happened and how the malware tried to hide.

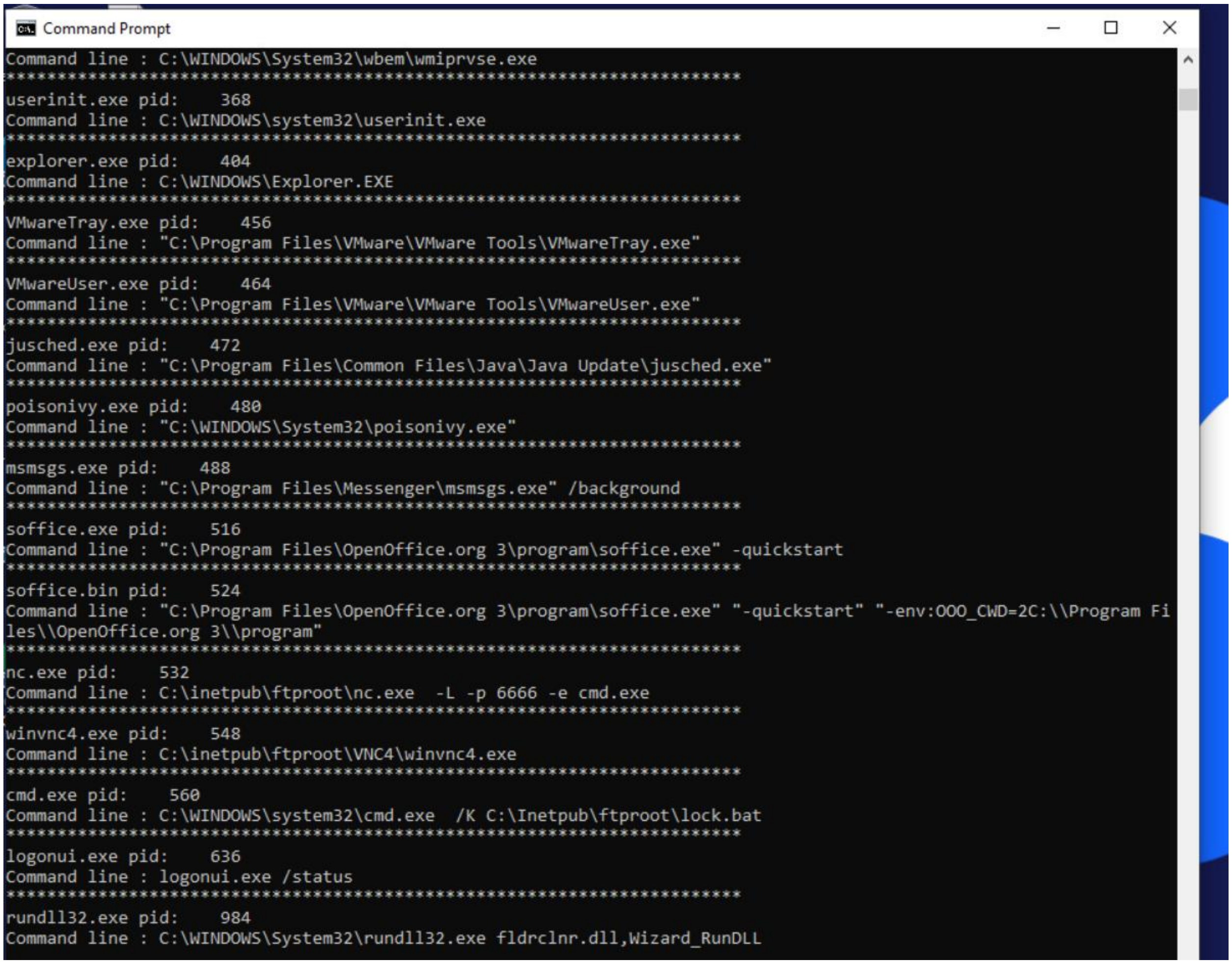


Figure 6

In step seven, I used the filescan command which searches the memory for file objects (Figures 7 and 8). This command shows files that were open or used by the system at the time the memory was captured. I found several references to poisonivy.exe which made sense. I also found two suspicious files located in C:\inetpub\ftproot. These files were nc.exe and lock.bat. Netcat is a tool used by attackers to create backdoors or transfer data. A random batch file in this directory is also strange because batch scripts are commonly used to automate malicious commands. These two files showed that the attacker likely did more than just run Poison Ivy. They were probably trying to maintain access or expand their control over the system by adding more tools. This strengthened the conclusion that is was intentional and planned.

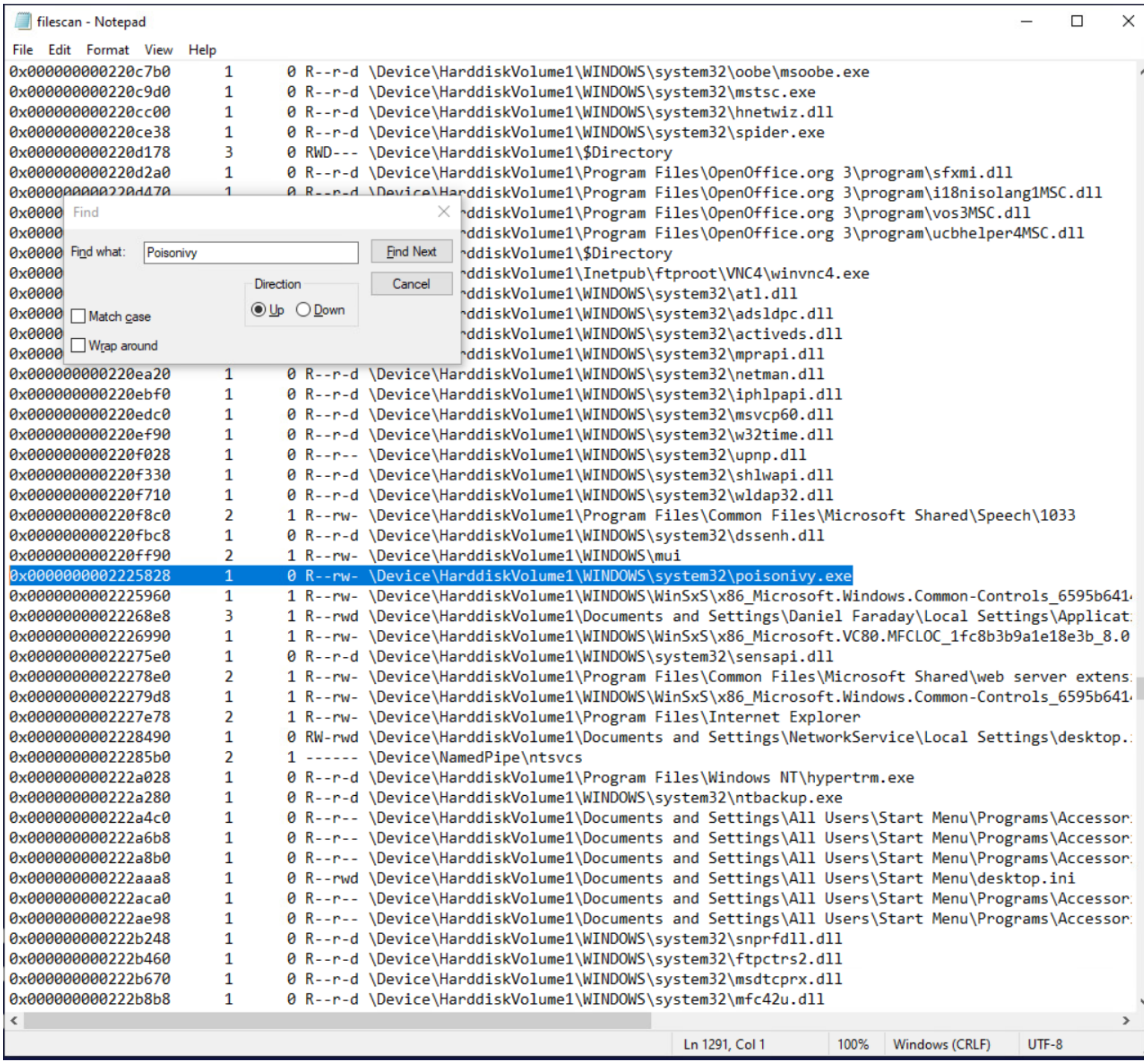

filescan - Notepad
File Edit Format View Help
0x000000000220c7b0 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\oobe\msoobe.exe
0x000000000220c9d0 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\mstsc.exe
0x000000000220cc00 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\hnetwiz.dll
0x000000000220ce38 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\spider.exe
0x000000000220d178 3 0 RWD--- \Device\HarddiskVolume1\$Directory
0x000000000220d2a0 1 0 R--r-d \Device\HarddiskVolume1\Program Files\OpenOffice.org 3\program\sfxmi.dll
ddiskVolume1\Program Files\OpenOffice.org 3\program\i18nisolang1MSC.dll
ddiskVolume1\Program Files\OpenOffice.org 3\program\vos3MSC.dll
ddiskVolume1\Program Files\OpenOffice.org 3\program\ucbhelper4MSC.dll
ddiskVolume1\$Directory
ddiskVolume1\Inetpub\ftproot\VNC4\winvnc4.exe
ddiskVolume1\WINDOWS\system32\atl.dll
ddiskVolume1\WINDOWS\system32\adsldpc.dll
ddiskVolume1\WINDOWS\system32\activeds.dll
ddiskVolume1\WINDOWS\system32\mprapi.dll
Find
Find what: Poisonivy
Find Next
Cancel
Direction
Up Down
Match case
Wrap around
0x00000000220ea20 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\netman.dll
0x00000000220ebf0 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\iphlpapi.dll
0x00000000220edc0 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\msvcp60.dll
0x00000000220ef90 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\w32time.dll
0x00000000220f028 1 0 R--r-- \Device\HarddiskVolume1\WINDOWS\system32\upnp.dll
0x00000000220f330 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\shlwapi.dll
0x00000000220f710 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\wldap32.dll
0x00000000220f8c0 2 1 R--rw- \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\Speech\1033
0x00000000220fbc8 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\dssenh.dll
0x00000000220ff90 2 1 R--rw- \Device\HarddiskVolume1\WINDOWS\mui
0x0000000002225828 1 0 R--rw- \Device\HarddiskVolume1\WINDOWS\system32\poisonivy.exe
0x0000000002225960 1 1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b641
0x00000000022268e8 3 1 R--rwd \Device\HarddiskVolume1\Documents and Settings\Daniel Faraday\Local Settings\Applicat
0x0000000002226990 1 1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.VC80.MFCLOC_1fc8b3b9a1e18e3b_8.0
0x00000000022275e0 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\sensapi.dll
0x00000000022278e0 2 1 R--rw- \Device\HarddiskVolume1\Program Files\Common Files\Microsoft Shared\web server extens
0x00000000022279d8 1 1 R--rw- \Device\HarddiskVolume1\WINDOWS\WinSxS\x86_Microsoft.Windows.Common-Controls_6595b641
0x0000000002227e78 2 1 R--rw- \Device\HarddiskVolume1\Program Files\Internet Explorer
0x0000000002228490 1 0 RW-rwd \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\desktop.
0x00000000022285b0 2 1 ------ \Device\NamedPipe\ntsvcs
0x000000000222a028 1 0 R--r-d \Device\HarddiskVolume1\Program Files\Windows NT\hypertrm.exe
0x000000000222a280 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\ntbackup.exe
0x000000000222a4c0 1 0 R--r-- \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Accessor
0x000000000222a6b8 1 0 R--r-- \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Accessor
0x000000000222a8b0 1 0 R--r-- \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Accessor
0x000000000222aaa8 1 0 R--rwd \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\desktop.ini
0x000000000222aca0 1 0 R--r-- \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Accessor
0x000000000222ae98 1 0 R--r-- \Device\HarddiskVolume1\Documents and Settings\All Users\Start Menu\Programs\Accessor
0x000000000222b248 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\snprfdll.dll
0x000000000222b460 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\ftpctrs2.dll
0x000000000222b670 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\msdtcprx.dll
0x000000000222b8b8 1 0 R--r-d \Device\HarddiskVolume1\WINDOWS\system32\mfc42u.dll
Ln 1291, Col 1
100%
Windows (CRLF)
UTF-8

Figure 7

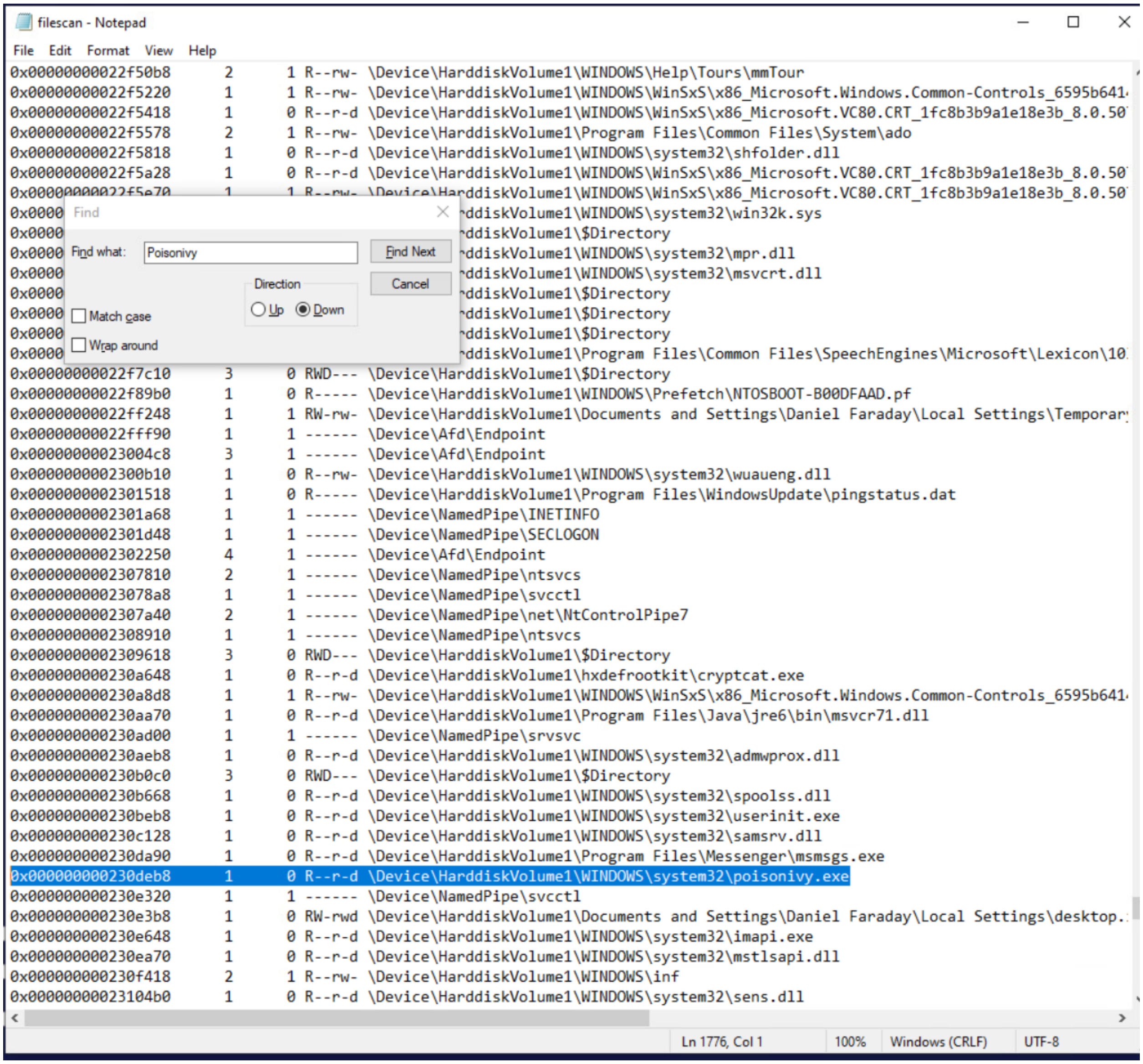


Figure 8

Looking at all the evidence, the system was clearly compromised by an attacker who used the Poison Ivy remote access trojan. It is likely that the user opened something that looked trustworthy. Once the malware executed, it placed itself into the System32 folder, loaded DLLs, and possibly communicated with a remote server. The attacker also placed Netcat and a batch script on the system which means they intended to maintain access or run additional commands. The presence of the username and password in memory shows that the attacker could have easily stolen the credentials and used them elsewhere. Everything found in the memory created a consistent story of a compromised system.

After completing this lab, I feel like I understand memory forensics much better. I was able to see how each piece of information helps explain what happened during an attack. Every command I ran gave me another clue. By the time I finished the investigation, I understood how the attacker got in, what they did, and what tools they used. I also gained more confidence in using Volatility. At first the tool seemed complicated because there are many different commands, but each one has a specific purpose and helps answer a specific question about the system. Being able to break a memory file down into processes, DLLs, files, user data, and command lines helped me understand how everything in a system connects during an attack.

This lab also taught me how important it is to act quickly during an incident. Memory does not last forever. If a machine is turned off, everything stored in RAM disappears. That means responders need to capture memory as soon as possible to preserve evidence. This lab helped me understand why memory forensics is such a big part of modern cybersecurity. Attackers often use techniques that hide their activity from logs or the hard drive, but they cannot hide what is in memory while their tools are running. Learning how to read memory gives investigators a strong advantage because they can see the system as it really was in the moment of the attack.

Overall, I learned a lot from this assignment. It gave me hands on practice with real memory analysis. It helped me understand how malware behaves and how investigators can uncover what happened by looking at RAM. I came away from this lab feeling better prepared for future work in intrusion detection and incident response because I now know how to use memory forensics to trace an attack and figure out the steps involved. It was interesting to see how everything in memory fits together and how those pieces reveal the full story of what happened on a compromised system.

References

- MalwareAnalysis. (2024, February 26). *Introduction to memory forensics with Volatility 3* [Video]. YouTube. https://www.youtube.com/watch?v=Uk3DEgY5Ue8

- The Volatility Foundation. (n.d.). *Home of the Volatility Foundation | Volatility memory forensics* [Website]. https://volatilityfoundation.org/

- Microsoft. (2022, May 31). *Dynamic-Link Libraries (DLLs)*. Microsoft Docs. https://learn.microsoft.com/en-us/windows/win32/dlls/dynamic-link-libraries

- Booz Allen Hamilton. (2021, June 15). *Volatility is an essential DFIR tool—Here's why*. https://www.boozallen.com/insights/cyber/tech/volatility-is-an-essential-dfir-tool-here-s-why.html

- New Jersey Cybersecurity & Communications Integration Cell (NJCCIC). (n.d.). *Poison Ivy (Remote Access Trojan)*. NJ Cybersecurity & Communications Integration Cell. https://www.cyber.nj.gov/threat-landscape/malware/trojans/poison-ivy

- GeeksforGeeks. (2023, April 25). *Introduction to Netcat*. GeeksforGeeks. https://www.geeksforgeeks.org/computer-networks/introduction-to-netcat/

-