Lab 4

By :- Faraz Ahmed

Q1. What can be extracted from memory using this profile based on the results of this command?

Ans1. We can extract (refer Figure 1 and 2):-

- Active processes
- Registry contents
- Malware indicators
- Network activity
- Kernel drivers

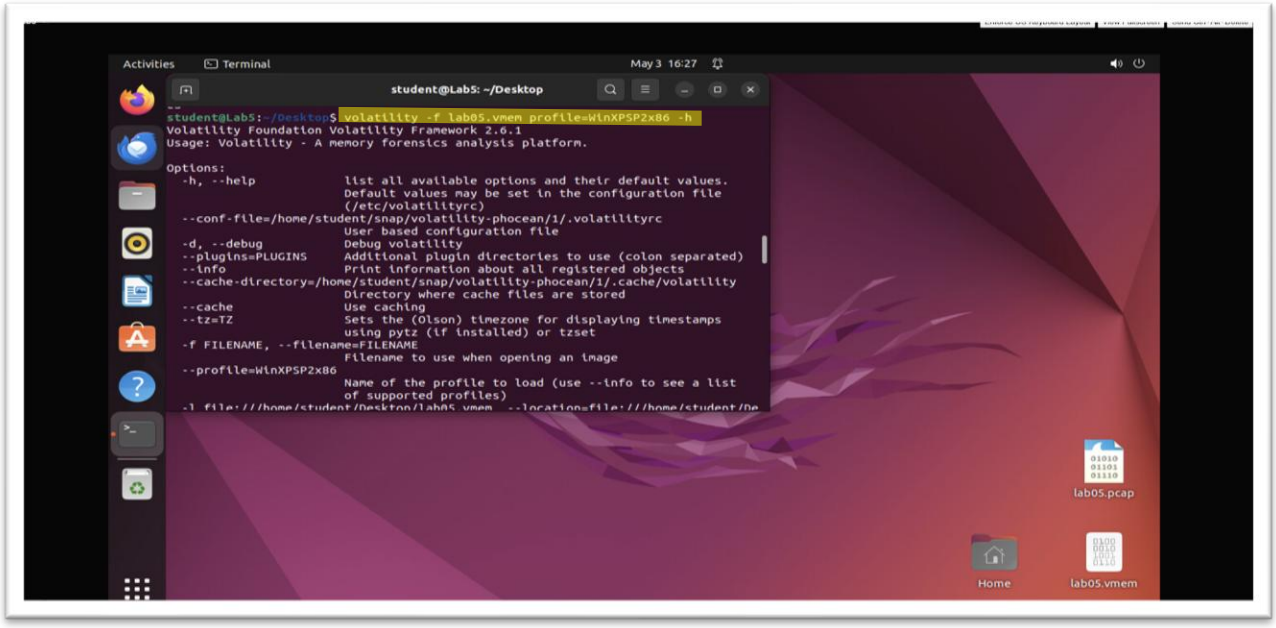


Figure 1: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 -h".

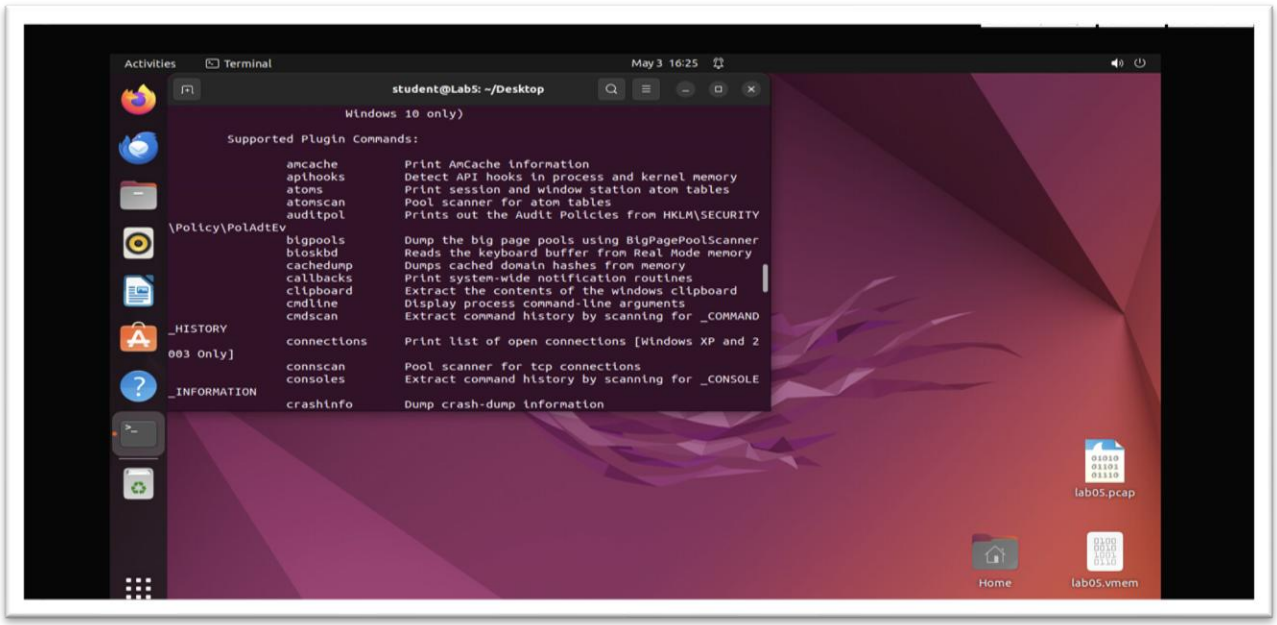


Figure 2: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 -h".

Q2. What information is the plugin pstree providing you?

Ans2. Pstree provide us with:-

- Process tree
- Process name
- Process IDs (PIDs)
- Parent Process IDs (PPIDs)
- Creation time

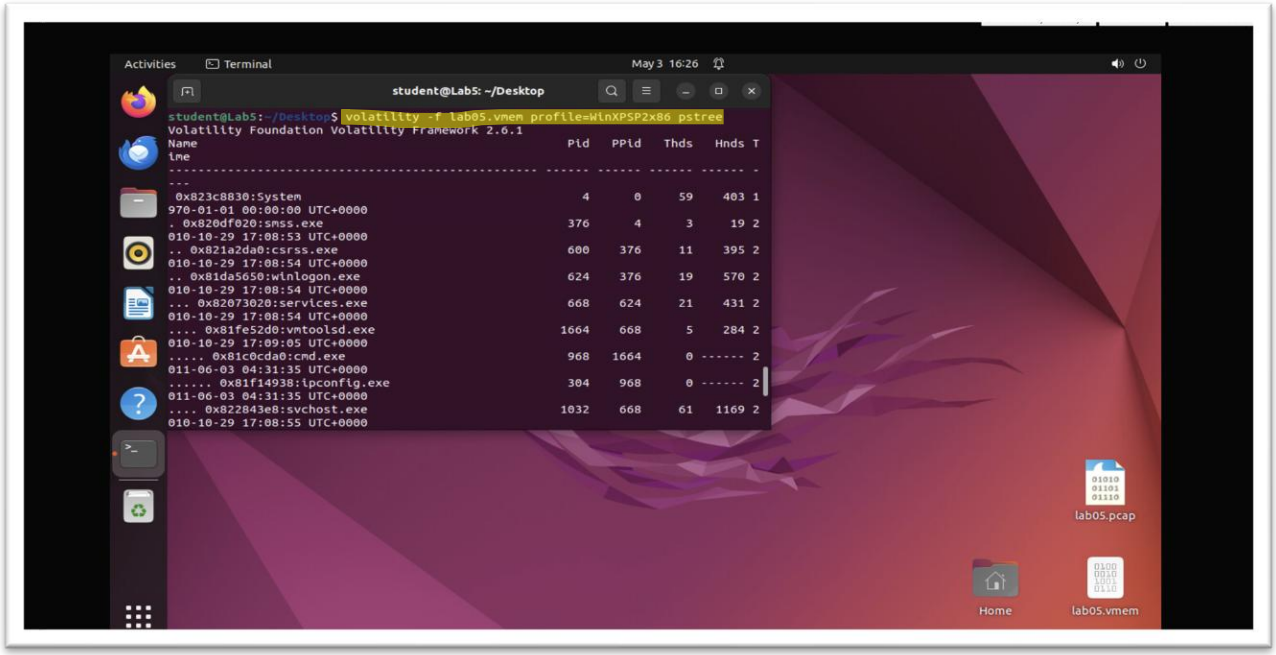


Figure 3: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 pstree".

Q3. Research lsass.exe and briefly explain what its purpose is

- What are the 3 process IDs (PID) of these processes found within this memory capture?

Ans3. lsass.ese basically stands for Local Security Authority Subsystem Service. It helps to generate access tokens, manage password changes and confirm all user logins.

There are 3 different process IDs (PID) running as seen in Figure 4:-

- First PID= 1928
- Second PID= 868
- Third PID= 680

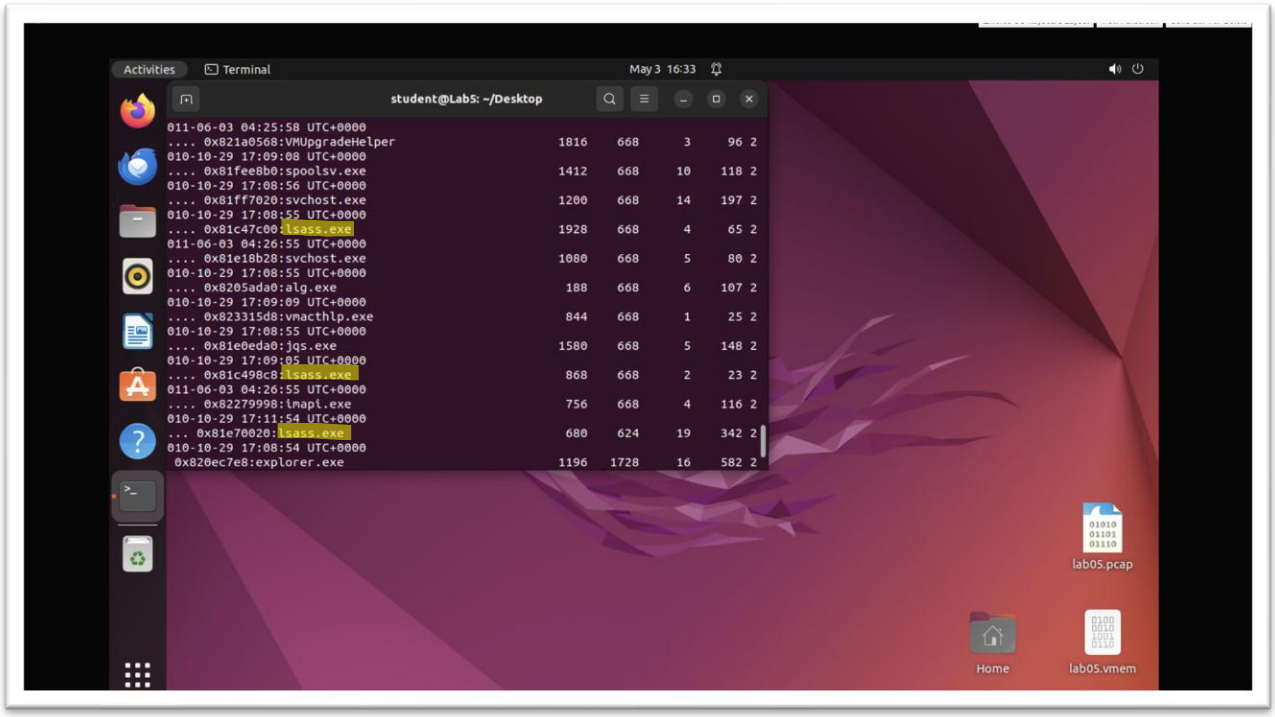


Figure 4: Screenshot of three malicious process running with PIDs.

Q4. What is a PPID?

Ans4. Parent Process ID (PPID) is the PID of process that launched the current process.

Q5. Why would 3 concurrently running versions of this process be an Indicator Of Compromise (IOC)?

Ans5. 3 concurrently running versions of lsass.exe can be IoC as windows only allows to run one lsass.exe at a time so multiple processes can be due to some kind of process injection or impersonation and also to harvest the credentials or maintain access.

Q6. What is the expected parent process of svchost.exe?

Ans6. Service.exe is the expected parent process of svchost.exe

Q7. What is the expected parent process of lsass.exe?

Ans7. Wininit.exe is the expected parent process of lsass.exe. I didn't find any wininit.exe which is a critical red flag for lsass.exe.

Q8. What does the plugin malfind do?

Ans8. The malfind basically helps to search for memory regions that are either scans executables or processes, contain suspicious code or may have been malicious code injected.

Q9. Did any of these PIDs return anything? If so, which ones?

Ans9. Yes, 1928 and 868 are the once which return something while 680 returns nothing as observed in Figure 5,6 and 7 respectively.

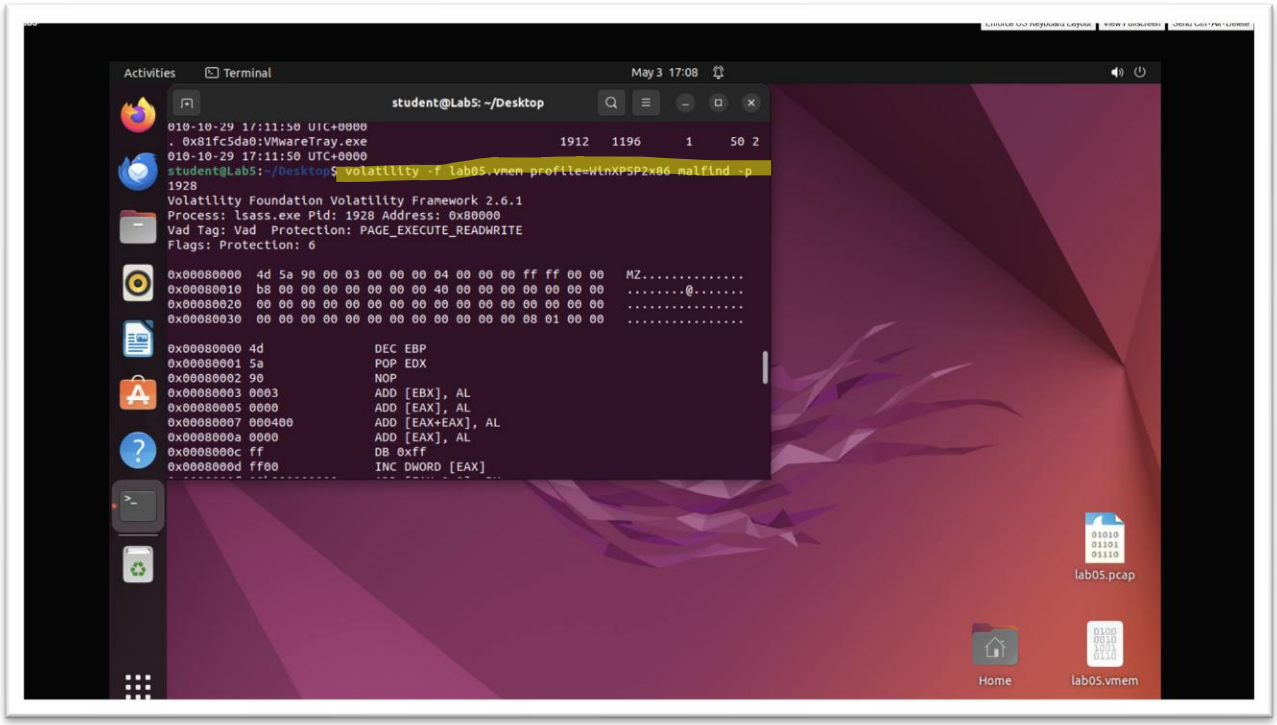


Figure 5: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 malfind -p 1928".

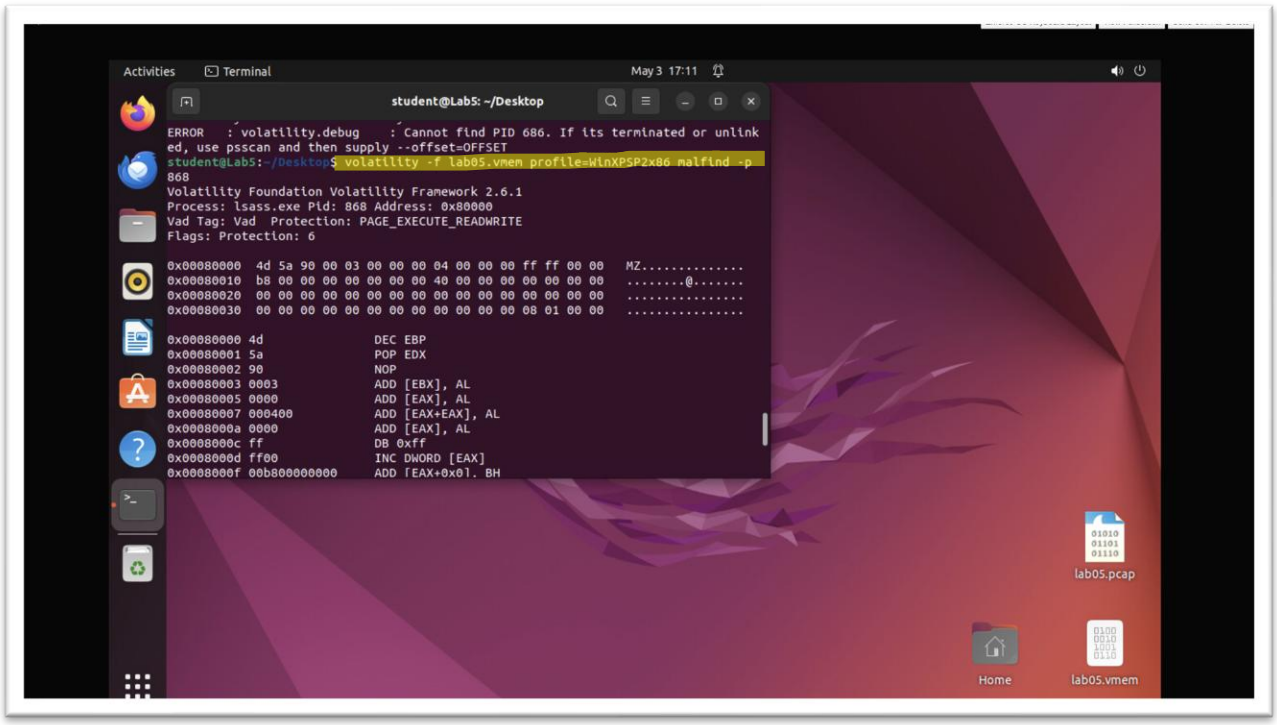


Figure 6: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 malfind -p 868".

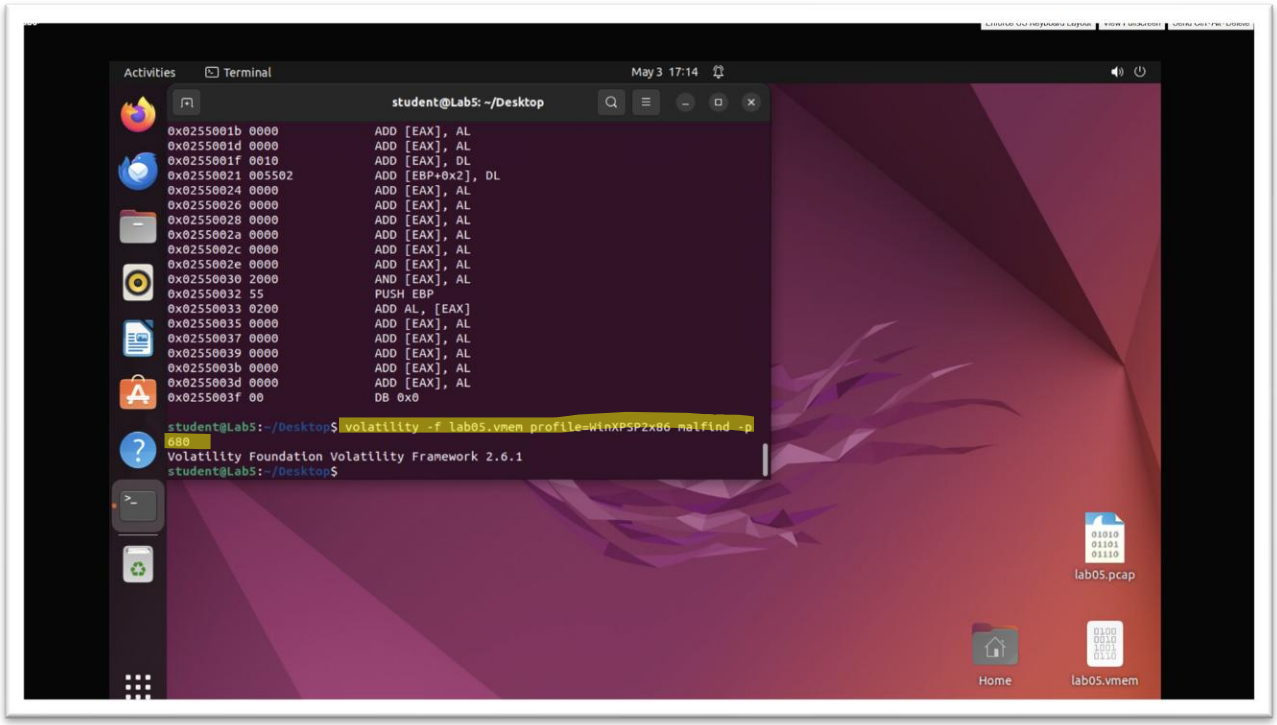


Figure 7: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 malfind -p 680".

Q10. What does "PAGE_EXECUTE_READWRITE" mean in this context?

- Why is this bad?

Ans10. "PAGE_EXECUTE_READWRITE" means that a memory region can Be read, Be written or Be executed.
This could be a bad thing because normal processes don't require all of those three permissions altogether. So, there might be some level of shellcode execution or malicious code injection which made it flag out.

Q11. Should lsass.exe ever have this permission on its memory region?

- What are hollowed-out processes?
- Why would a bad actor use this method?

Ans11. No, it shouldn't have any type of write or executable access in memory. If it does then it's a strong indicator of compromise.

So, Hollowed-Out process happens when a legit process is started in a suspended state but then its memory is unmapped and replaced by a malicious code which appears like a normal process. So, attacker use this to avoid any type of detection by AV or EDR and also maintain hidden status under a legit process (like in this case was lsass.exe).

Q12. What are the hashes of the malicious processes?

Ans12. We can dump these malicious processes using “volatility -f lab05.vmem profile==WinXPSP2x86 procdump -p 1928,868 -D ~/Desktop” and got details of two processes as in Figure 8. So, the hash value of the malicious processes are:-

- 1928- 20a3c5f02b6b79bcac9adaef7ee138763054bbedc298fb2710b5adaf9b74a47d (Figure 9)
- 868- 6f293f095e960461d897b688bf582a0c9a3890935a7d443a929ef587ed911760 (Figure 9)

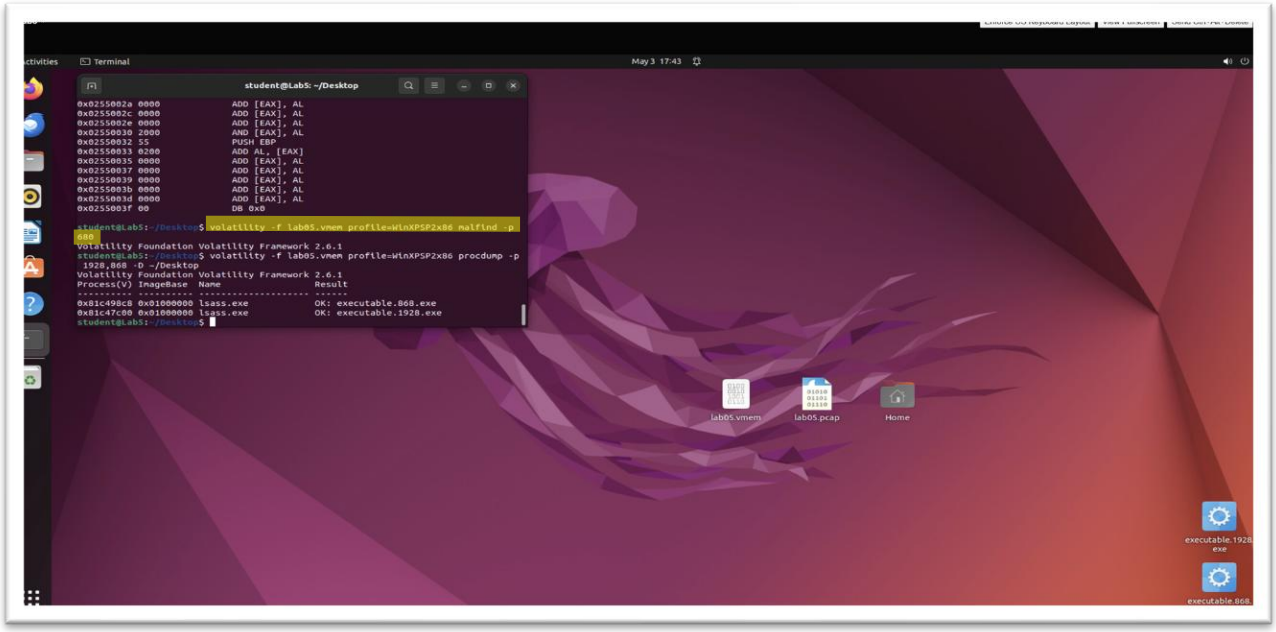


Figure 8: Screenshot of executing “volatility -f lab05.vmem profile==WinXPSP2x86 procdump -p 1928,868 -D ~/Desktop”.

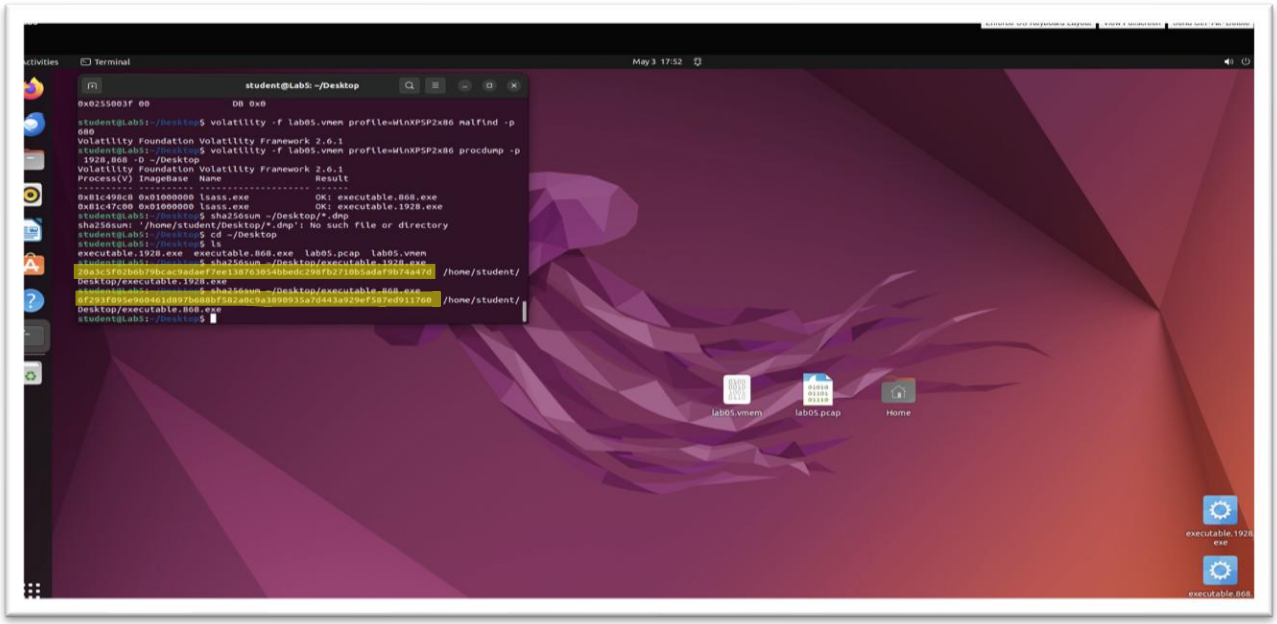


Figure 9: Screenshot of SHA256sum values of 1928 and 868.

Q13. Does Virustotal return any information on the name of what malware is running?

Ans13.

- 1928- Figure 10
- 868- Figure 11

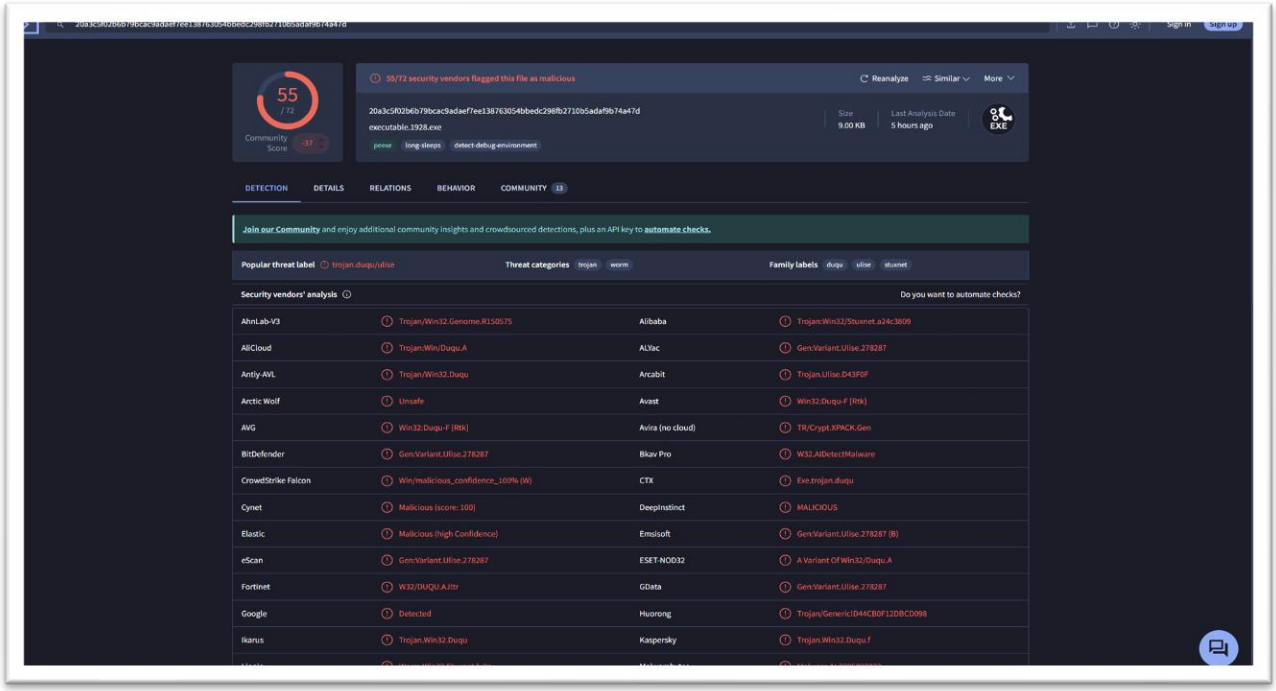


Figure 10: Screenshot of Virustotal result for 1928.

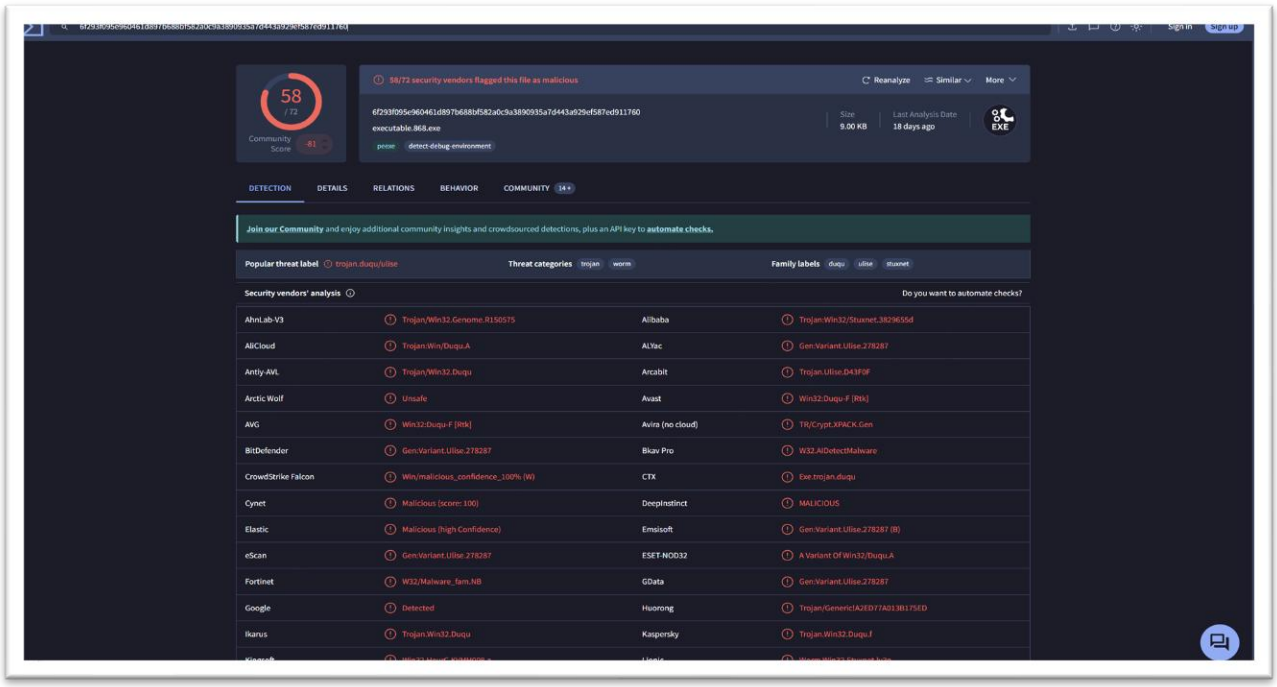


Figure 11: Screenshot of Virustotal result for 868.

Q14. What does the connections plugin do?

- Are there any connections?

Ans14. Connections plugin helps to identify which processes had active network connections so, it shows active TCP connections from memory. It could be crucial to find some of the Indicator of Compromise (IoCs).

No, there is no connections as observed in Figure 12.

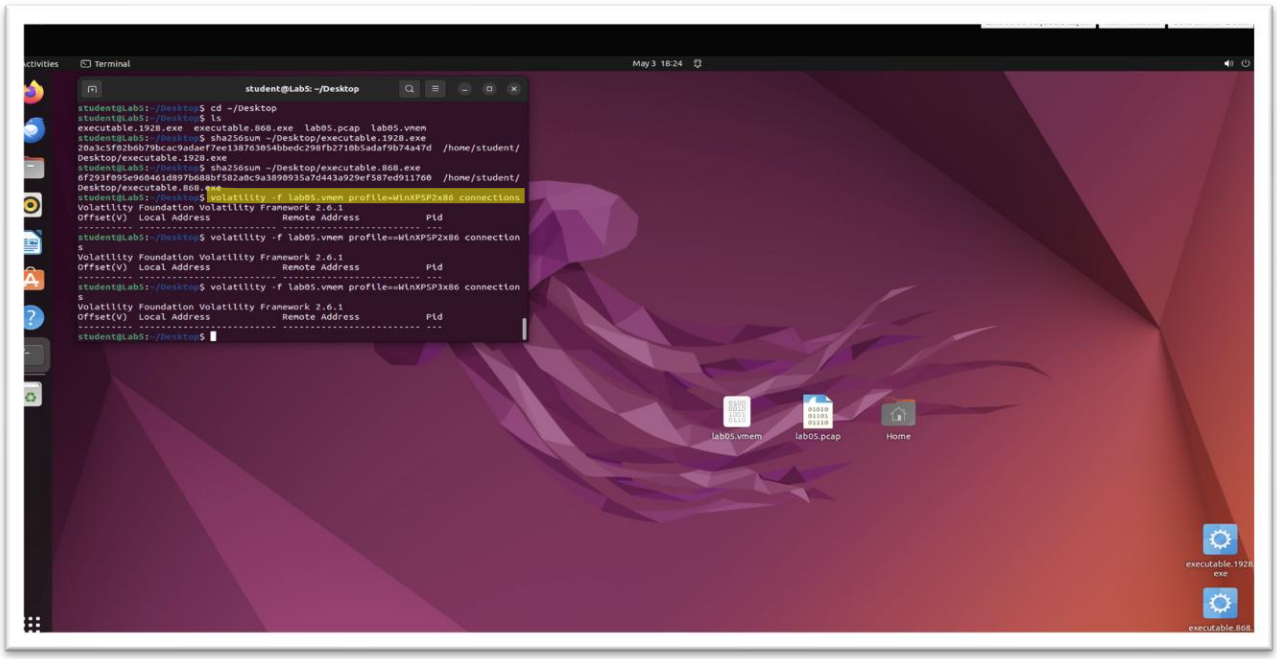


Figure 12: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 connections".

Q15. What does the callbacks plugin do?

Ans15. Callbacks plugin list functions that made to be notified by the kernal when certain system events occur which are:-

- Registry modifcations
- Process creation and deletion
- Active drivers

Q16. What are the names of the modules associated with the two malicious callbacks?

Ans16. There are two malicious module names which are- mrxcls.sys and mrxnet.sys as highlighted in Figure 13.

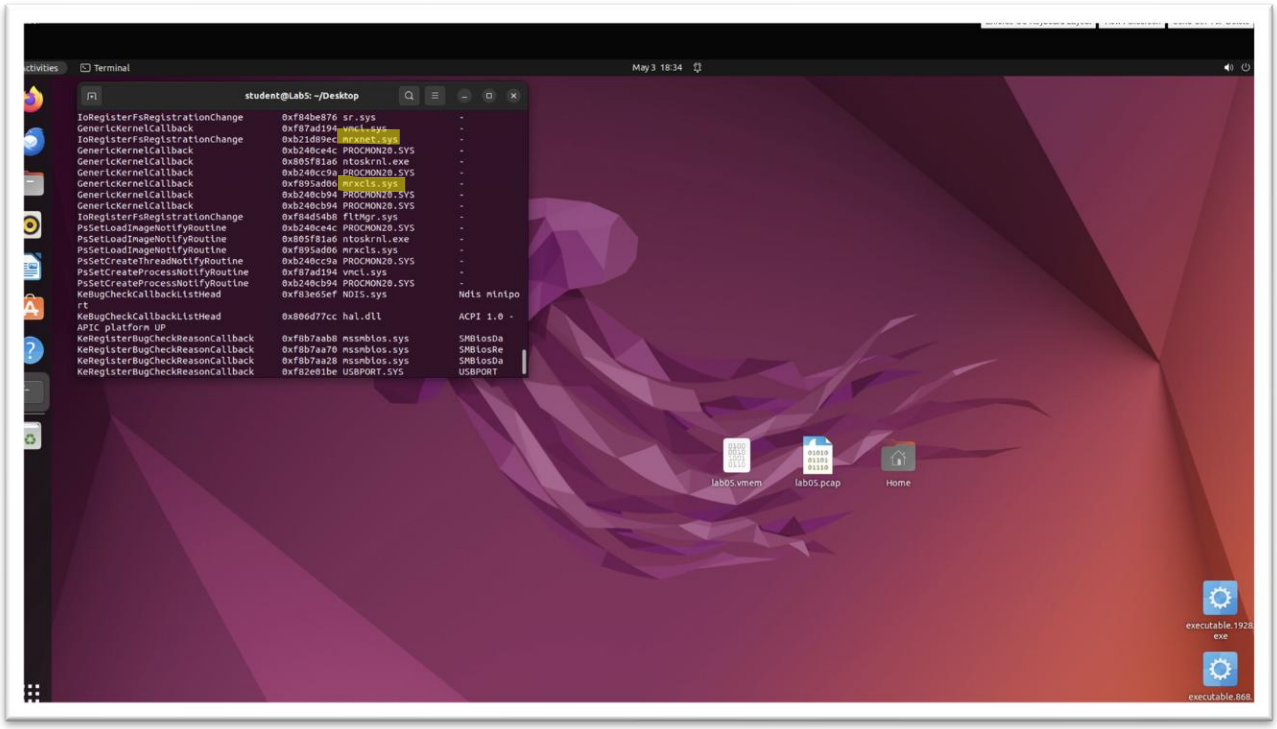


Figure 13: Screenshot of two malicious modules- mrxcls.sys and mrxnet.sys.

Q17. What does the modules plugin do?

- What is a module in the context of memory analysis?
- What is the base memory address for each of these?

Ans17. The modules plugin lists all of the loaded kernel modules that were running when the memory was captured. So, we can observe details like:-

- Memory base address
- Module name and size
- Path of the module file.

So, in memory analysis, a module refers to a driver in which .sys file is loaded into memory which could be potentially malicious.

The base memory address for each are:-

- Mrxnet- 0xb21d8000 (refer Figure 14)
- Mrxcls- 0xf895a000 (refer Figure 14)

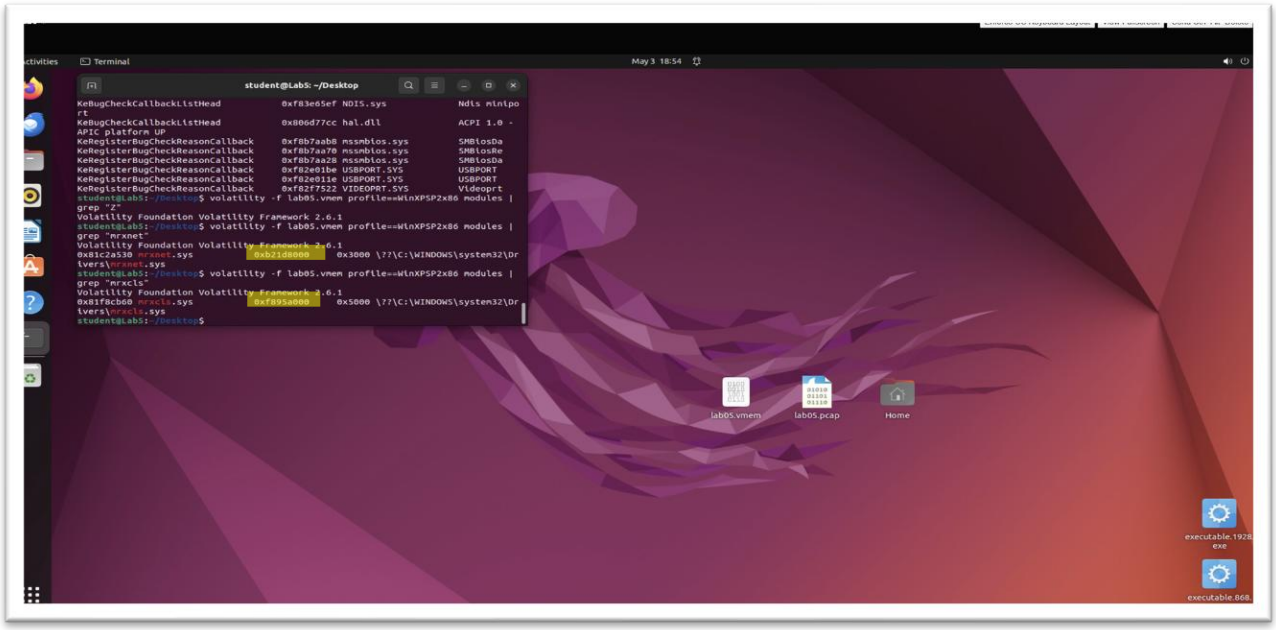


Figure 14: Screenshot of executing "volatility -f lab05.vmem profile==WinXPSP2x86 modules | grep "Z"".

Q18. What is the SHA256 hash of these two new files created?

Ans18. The SHA256 hash value of two new files are:-

- driver.b21d8000.sys- 6aa1f54fbd8c79a3109bfc3e7274f212e5bf9c92f740d5a194167ea940c3d06c (refer Figure 15)
- driver.f895a000.sys- 6bc86d3bd3ec0333087141215559aec5b11b050cc49e42fc28c2ff6c9c119dbd (refer Figure 15).

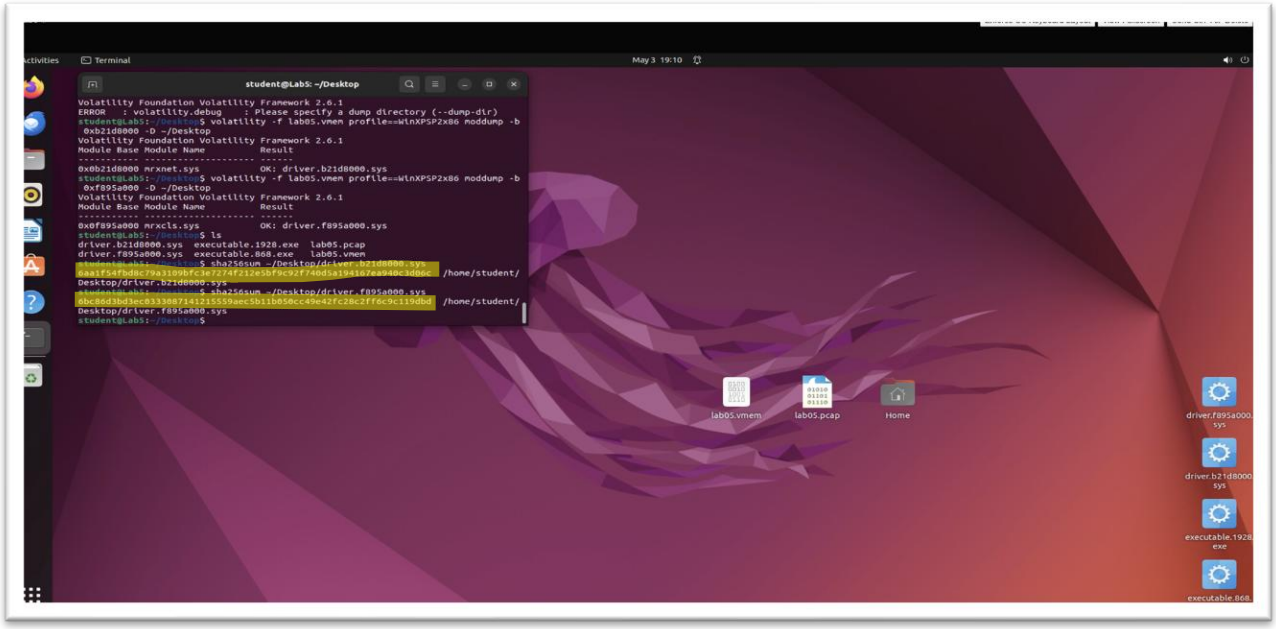


Figure 14: Screenshot of SHA256sum values of two new files.

Q19. What is a registry in this context?

- What is a registry hive?

Ans19. Registry basically refers to Windows Registry which is a database that stores configuration settings and options for the Windows operating system and also installed application.

And registry hive is a major section or file of the Windows Registry where each hive consist of the registry and map to physical disk suck as SYSTEM, SOFTWARE and SECURITY.

Q20. What is the file path of the lsass.exe file?

Ans20. The file path of lsass.exe is C:\Windows\System32\lsass.exe (which came from %SystemRoot%\system32\lsass.exe) as observed in Figure 15.

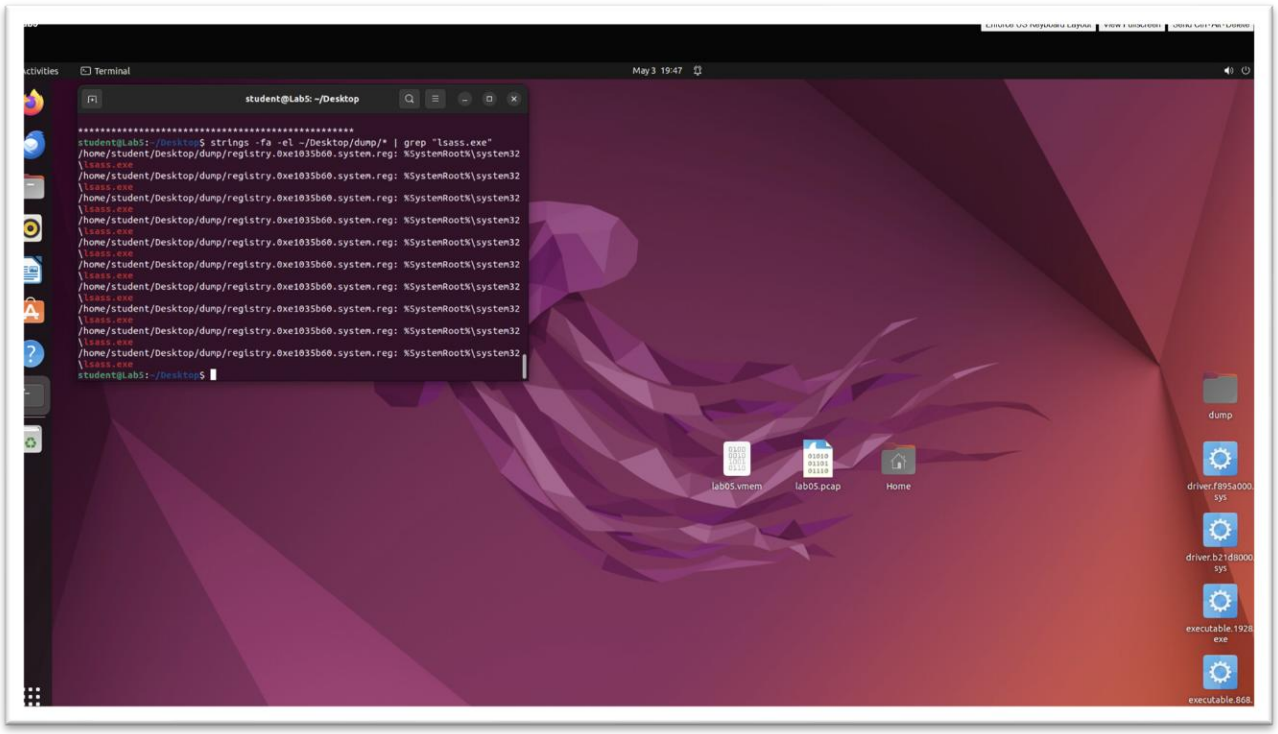


Figure 15: Screenshot of file location or path of "lsass.exe".

Q21. What is the file path of these two modules?

Ans21. File path of these two modules are:-

- Mrxcls.sys- C:\WINDOWS\system32\drivers\mrxcls.sys (refer Figure 16)
- Mrxnet.sys- C:\WINDOWS\system32\drivers\mrxnet.sys (refer Figure 16).

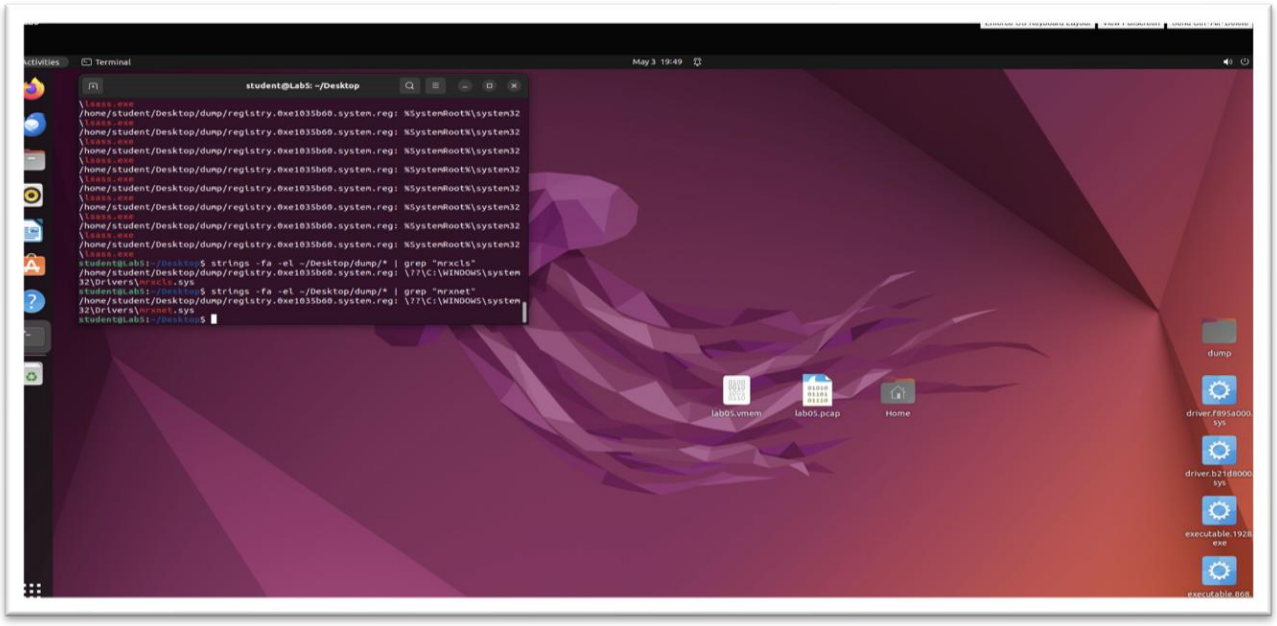


Figure 16: Screenshot of file location or path of both malicious modules.

Q22. What is the value of Start?

- What does this value of Start in the registry mean for the malware and its persistence?

Ans22. The START value of both of the malicious systems (mrxcls and mrxnet) is 1.

So, START in the registry means that the service will automatically start when the system boots up which means that malware is designed to persist on the infected system by launching itself automatically when the computer system is started. This ensures that the computer malware will resume even when the overall system reboots.

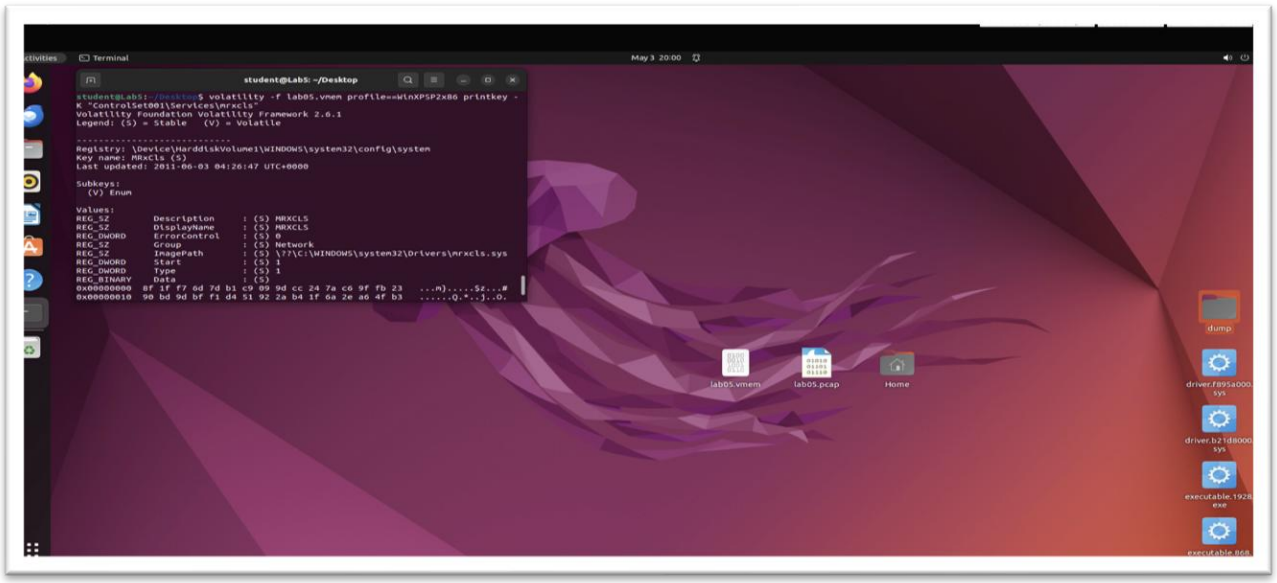


Figure 17: Screenshot of "volatility -f lab05.vmem profile==WinXPSP2x86 printkey -K "ControlSet001\Services\MRxCls"".

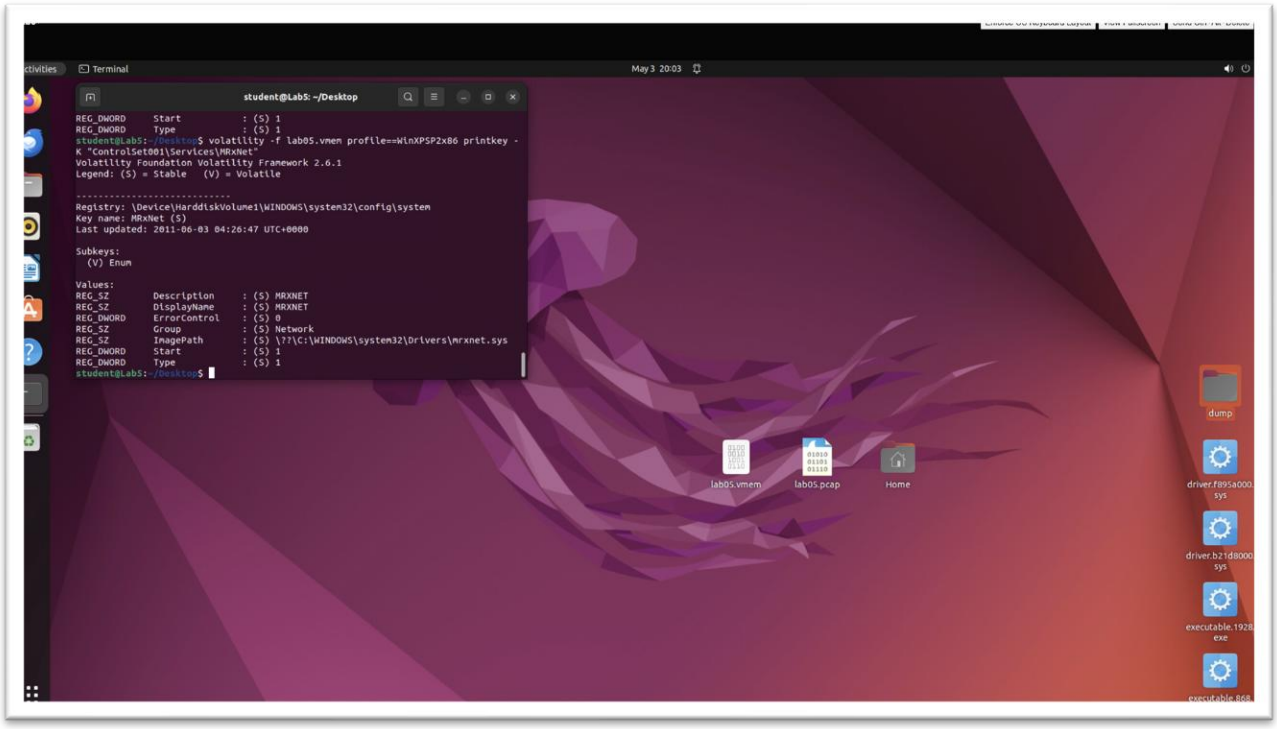
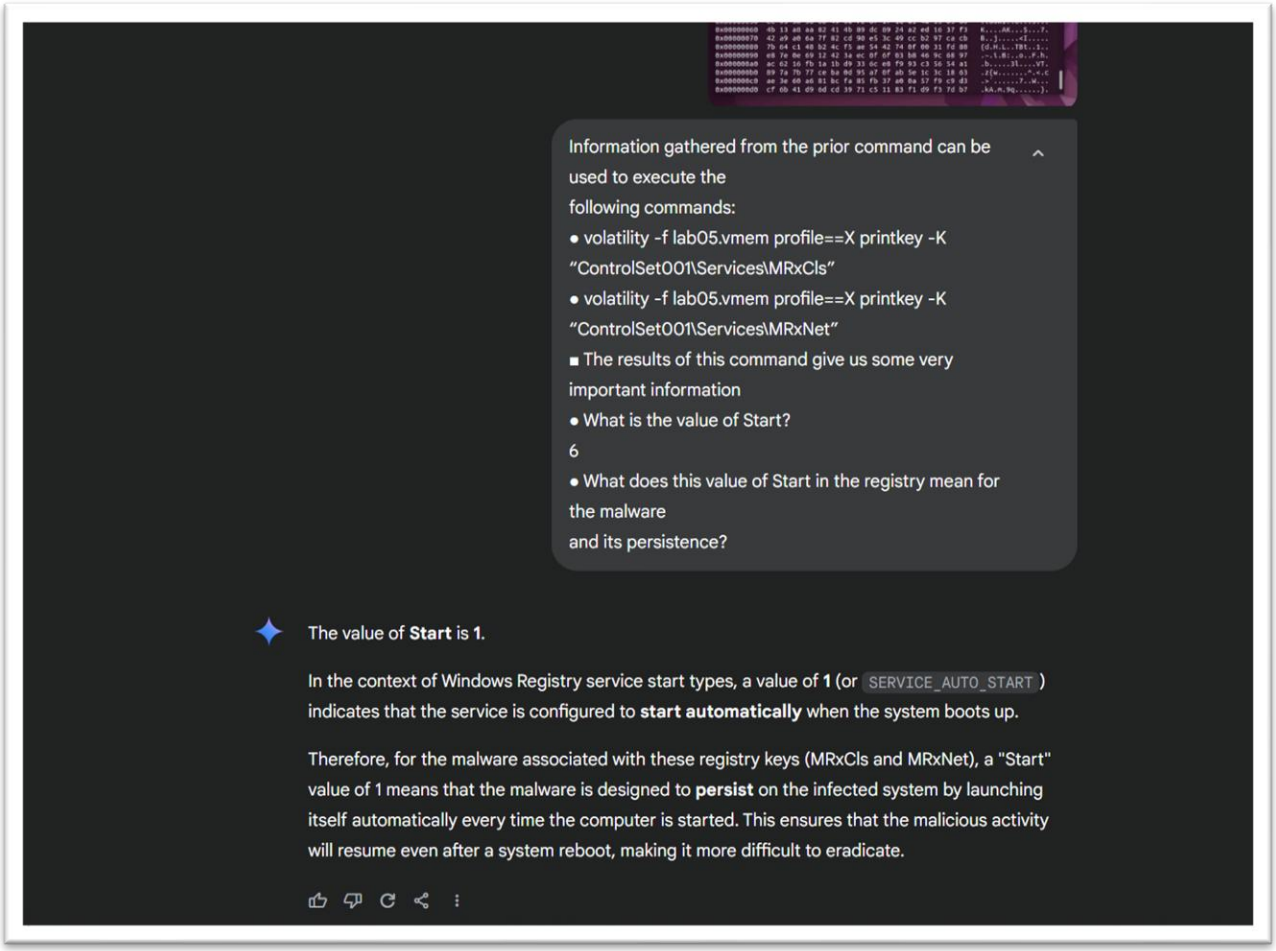


Figure 18: Screenshot of "volatility -f lab05.vmem profile==WinXPSP2x86 printkey -K "ControlSet001\Services\MRxNet"".

References


Information gathered from the prior command can be used to execute the
following commands:
• volatility -f lab05.vmem profile==X printkey -K "ControlSet001\Services\MRxCls"
• volatility -f lab05.vmem profile==X printkey -K "ControlSet001\Services\MRxNet"
■ The results of this command give us some very important information
• What is the value of Start?
6
• What does this value of Start in the registry mean for the malware
and its persistence?
The value of Start is 1.
In the context of Windows Registry service start types, a value of 1 (or SERVICE_AUTO_START) indicates that the service is configured to start automatically when the system boots up.
Therefore, for the malware associated with these registry keys (MRxCls and MRxNet), a "Start" value of 1 means that the malware is designed to persist on the infected system by launching itself automatically every time the computer is started. This ensures that the malicious activity will resume even after a system reboot, making it more difficult to eradicate.