Lab #3: Hunting in Memory

Inor Wang

Department of Cyber Security, The University of Texas at San Antonio

IS 3523.004

Professor Munoz

April 10, 2025

Table of Contents

Introduction…….3

Objectives……3

Methodology……4

Story……..15

Conclusion…….17

Bibliography……..18

Introduction

In this lab, I delved into the critical field of memory analysis as a key technique for investigating malicious activity. Using the Volatility framework, I examined the KobayashiMaru.vmem memory image to extract dynamic system data that static disk forensics could easily overlook. Throughout the analysis, I focused on uncovering key artifacts, such as running processes, loaded DLLs, and transient user credentials, each providing a snapshot of the system's active state during the compromise. Working within a legacy Windows environment, with its inherent vulnerabilities and outdated software, I was able to explore common exploitation techniques, identify abnormal process behavior, and trace the forensic clues left behind by attackers. In conducting this investigation, I employed a variety of Volatility plugins to filter and interpret the volatile data, which reinforced my understanding of how ephemeral evidence can be pivotal in reconstructing attack timelines. I also ensured that I maintained rigorous chain-of-custody procedures and detailed documentation of every step, knowing that such discipline is essential for both accurate analysis and the legal defensibility of the findings. This lab not only enhanced my practical memory forensics skills but also deepened my appreciation for how essential thorough in-memory analysis is in today's evolving cybersecurity landscape.

Objectives

The main goal of Lab 3 (Hunting in Memory) is to conduct a comprehensive memory forensics investigation of the "KobayashiMaru.vmem" file using Volatility as the central analysis tool. In the process, students learn to identify and select the correct operating system profile and verify how much physical RAM is included in the capture, which provides a clear starting point for their forensic workflow. Beyond these basics, they delve into a detailed inspection of running processes, watching for anything that appears unfamiliar or atypical, such as abnormal process trees or file names that don't align with standard system operations. This methodical process also involves investigating DLLs (dynamic-link libraries), because unexpected or hidden modules can signal deeper signs of compromise. Moreover, the lab encourages students to look for user account details, potential remnants of credentials, and registry entries, helping them recognize the myriad ways in which attackers can gain and maintain unauthorized access. From an educational perspective, this lab is crucial for students because it underscores the significance of volatile data, information that exists only in the system's memory and can disappear as soon as the machine is turned off. Through hands-on exploration, students learn how to methodically uncover these fleeting artifacts, tie them to broader indicators of compromise, and correctly document every discovery. This documentation aspect is especially critical, as it nurtures professionalism and diligence: skills that are required not only for technical accuracy but also for preserving evidence integrity in potential legal scenarios. By the time students complete the lab, they have gained valuable, real-world skills that span identification, analysis, and remediation of threats, proficiencies that remain universally relevant in an ever-evolving cybersecurity landscape.

Methodology

1. What operating system is the computer using? What version?

a. In order to find out what operating system and what version the computer is using, I used Volatility and one of its plugis which is “imageinfo”. The full command is as shown in Figure 1, however it is, “volatility -f “KobayashiMaru 1.vmem” imageinfo”. This command will find out multiple information about the image, more specifically, the computer. The computer is using Windows XP with Service Pack (SP2) and a x86 architecture as shown in Figure 1. The imageinfo plugin in Volatility is designed to parse key metadata from a memory image, providing investigators with an initial overview of the system that was captured. Specifically, imageinfo analyzes headers within the dump and examines various signatures to deduce critical details such as the operating system version, service pack level, CPU architecture, and recommended profile(s) for further analysis. By reviewing this plugin’s output, investigators can confirm the precise environment they are dealing with, for instance, whether it is Windows XP SP2 on a 32-bit platform or a more recent 64-bit operating system. This clarity is invaluable it helps people select the appropriate Volatility profile, ensuring subsequent commands and plugins work properly. Also it helps people understand exactly what operating system that the .vmem file capture is of. Moreover, imageinfo may reveal additional clues like the date and time the memory was acquired or show inconsistencies that signal tampering or compression. Overall, imageinfo is an essential first step in memory forensics, as it quickly gives analysts the foundational insight they need to structure their examination, reduce guesswork, and maintain accuracy throughout the investigative process.

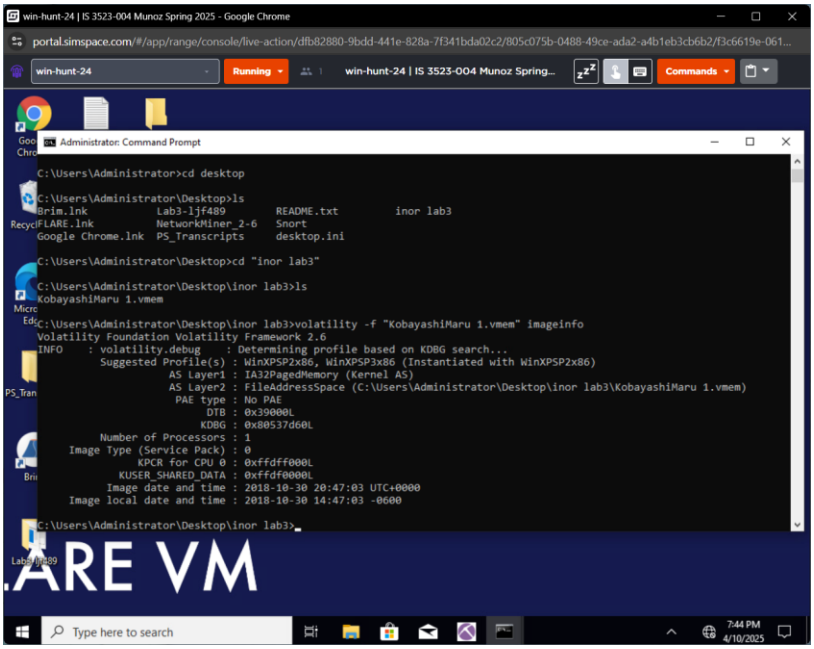

Figure 1: The terminal output of the following command, "volatility -f "KobayashiMaru 1.vmem" imageinfo"

2. How much RAM is included in the analysis?

a. It seems like there isn't a way to check how much RAM is included in the analysis however you can check the properties of the .vmem file and find the RAM included. There is 512mb of RAM included in the analysis as shown in Figure 2.

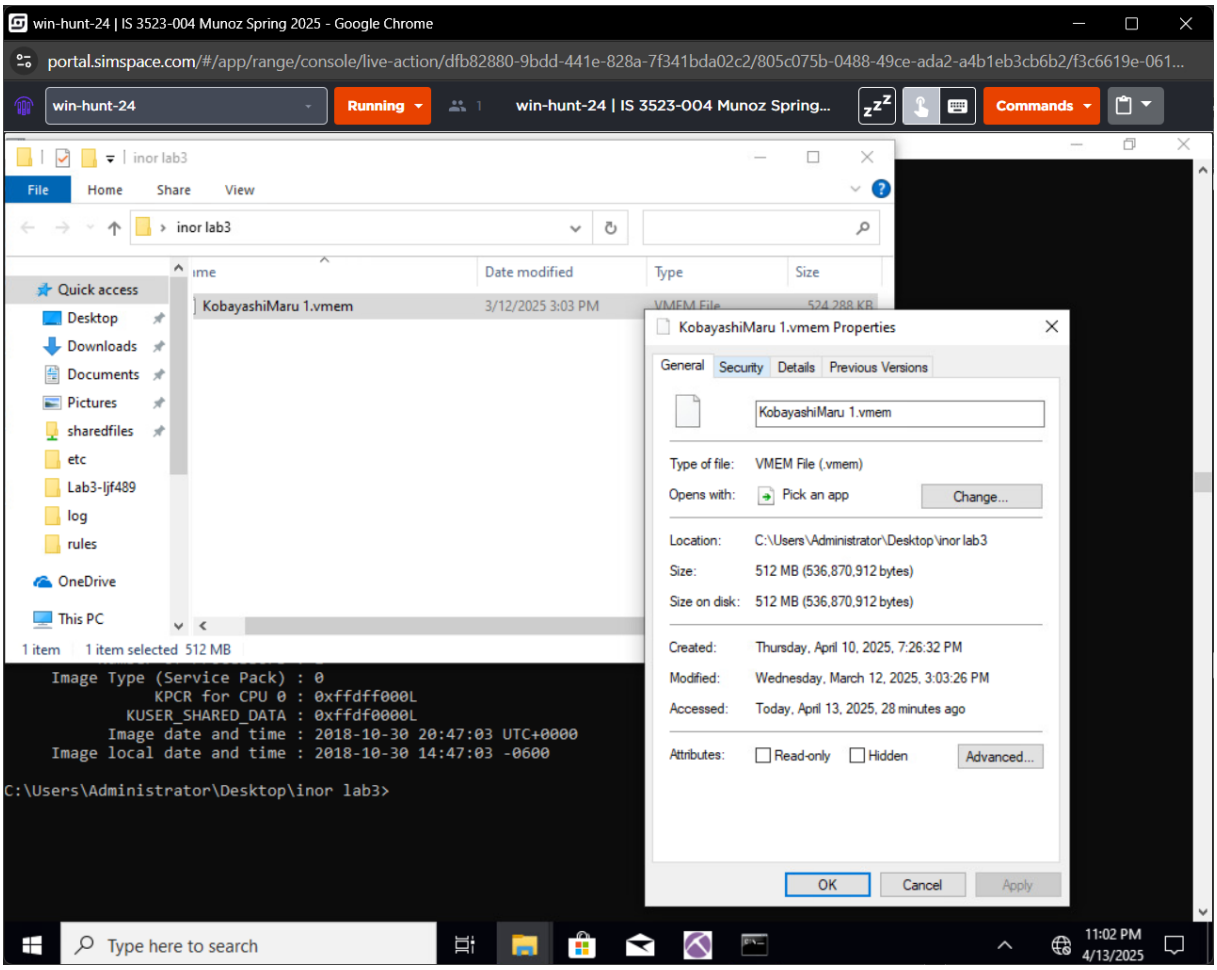

Figure 2: Properties tab if the .vmem file

3. View the running processes. Does this look like your average box? What processes look abnormal? What makes them abnormal?
    a. In order to find out the running processes on the computer, I used Volatility again. More specifically, I used "volatility -f "KobayashiMaru 1.vmem" pslist" as shown in Figure 3. There are multiple typical Windows processes such as System, smss.exe, csrss.exe, wininit.exe, services.exe, lsass.exe, spoolsv.exe, svchost.exe, etc. However, there are numerous abnormal processes. What makes an abnormal process "abnormal?" There are multiple ways, if the processes is not a recognized typical Windows process that is usually standard and default, if the process has suspicious naming, if the process is running from a non-standard path, if the processes is not part of a typical installed software, and if the processes has an unusual parent-child relationship. So, to start with the first abnormal indicator, if the process is not a typical Windows process. There are multiple processes that seem out of place such as:

hxdef100.exe, cryptcat.exe, bircd.exe, iroffer.exe, poisonivy.exe, and nc.exe. These are potentially harmful processes, so it warrants for further investigation. Next, the second abnormal indicator, if the process has suspicious naming. Now this indictor is usually mostly encapsulated in the previous indicator, so we don't need to specifically look for suspicious names. However, if you come across a process that specifically that is named, "hack.exe", or something along those lines, then it definitely warrants for further investigation. Next, the third indicator, if the process is running from a non-standard path. However, in the "pslist" command we used, we cant see if the path is abnormal so I decided to use the command, "volatility -f "KobayashiMaru 1.vmem" –profile=WinXPSP2x86 modules", to find the current loaded kernel modules. This is able to provide insights into system behavior, reveal any suspicious or maliciously injected modules, and will help find root kits. The command prints the kernel out with it's starting directory and as you can see in Figure 3, "hxdefdrv.sys" starts from "\??\C:\hxdefrootkit\" which clearly indicates that it is a root kit. So, it is important to check the paths. Last but not least, check if the process has an unusual parent-child relationship. If you look at Figure 3, you can see that explorer.exe has a PID of 303 and there are multiple processes that follow it with it's PPID of 404, which is normal. However, you can see that one of them is "poisonivy.exe" which is a RAT. Since explorer.exe is normally the Windows shell (responsible for the desktop interface, taskbar, and file browsing), seeing a malicious application like Poison Ivy running under it is suspicious. Attackers often leverage user-mode processes like explorer.exe to hide malware in plain sight, making it appear more benign at first glance. Examining the list of running processes is one of the most crucial steps in any memory forensics investigation because it provides a snapshot of the system's actual state at the time of capture. Unlike static disk images, which may not always reflect active or hidden threats, the process list reveals every program that was present in RAM, whether legitimate operating system tasks or malware masquerading under a benign name. By comparing expected processes against those that appear unusual, investigating their relationships (parent and child processes), and cross-referencing them with known file paths and command-line arguments, an analyst can quickly isolate anomalies that might indicate a compromise, making it an indispensable technique for detecting and responding to intrusions.

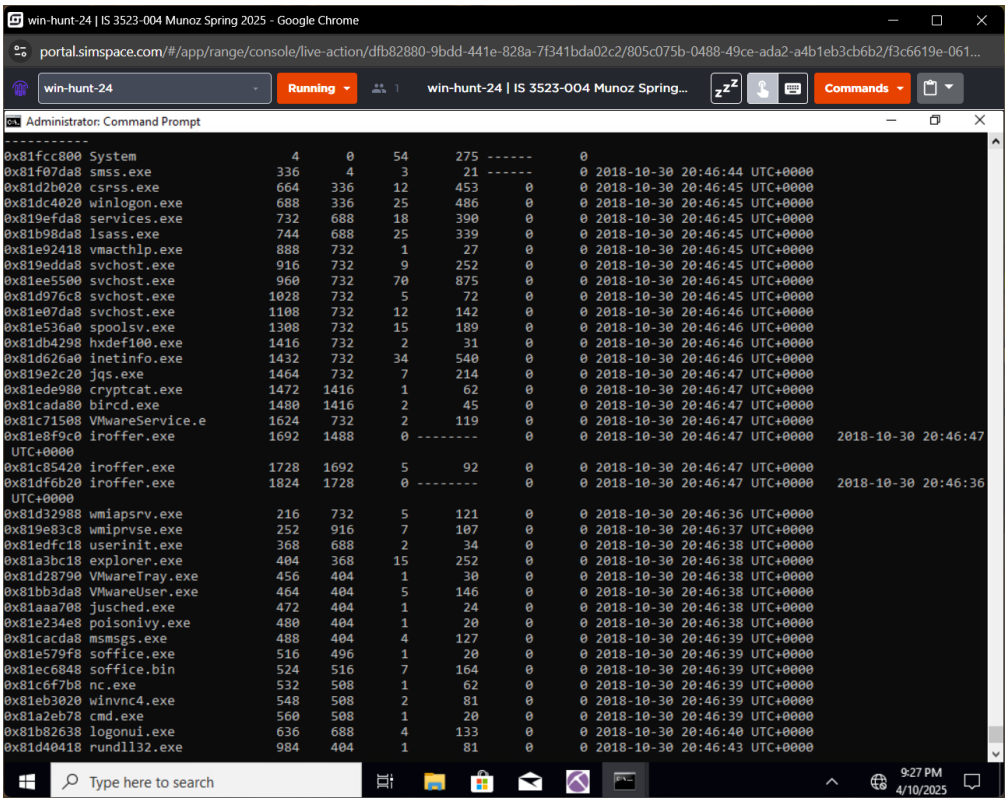

Figure 3: pslist command

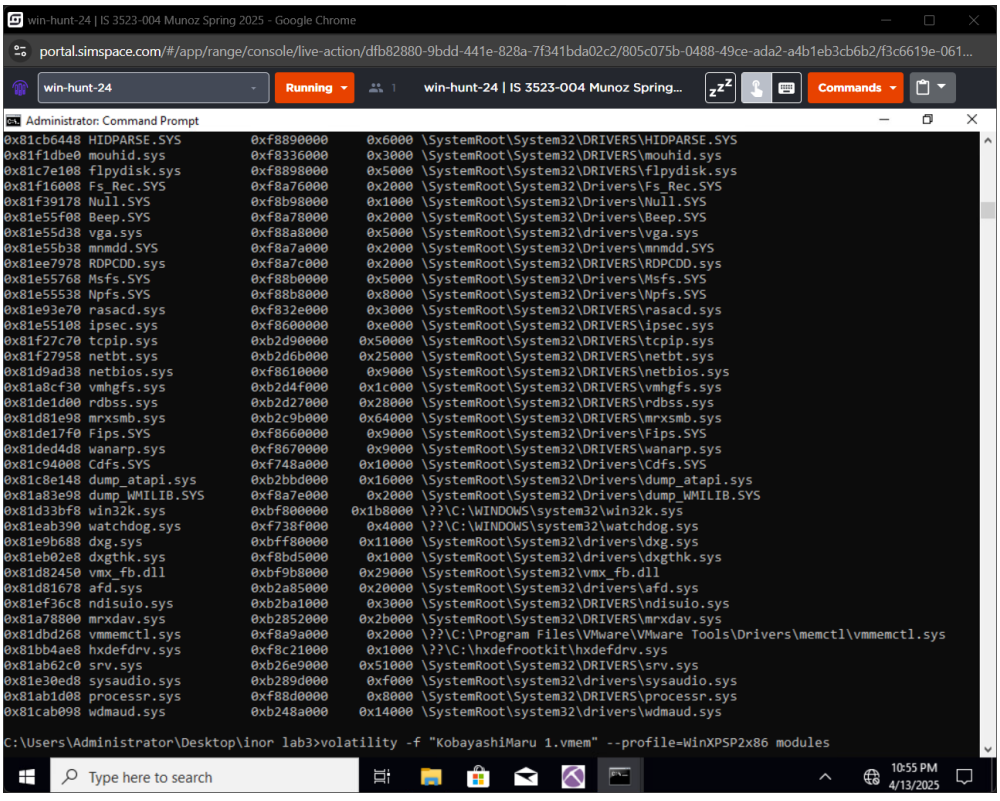

Figure 4: modules plugin

4. Can you find user account names? Passwords?
    a. Yes, I can find the user account names which are: “Administrator”, “Daniel Faraday”, “Guest”, “HelpAssistant”, “IUSR_FARADAY”, “IWAM_FARADAY”, and “SUPPORT_388945a0” as shown in Figure 5. However, I cannot find the passwords using the hashdump option as shown in Figure 2, I researched throughout Google however I could not find a way to find the password.

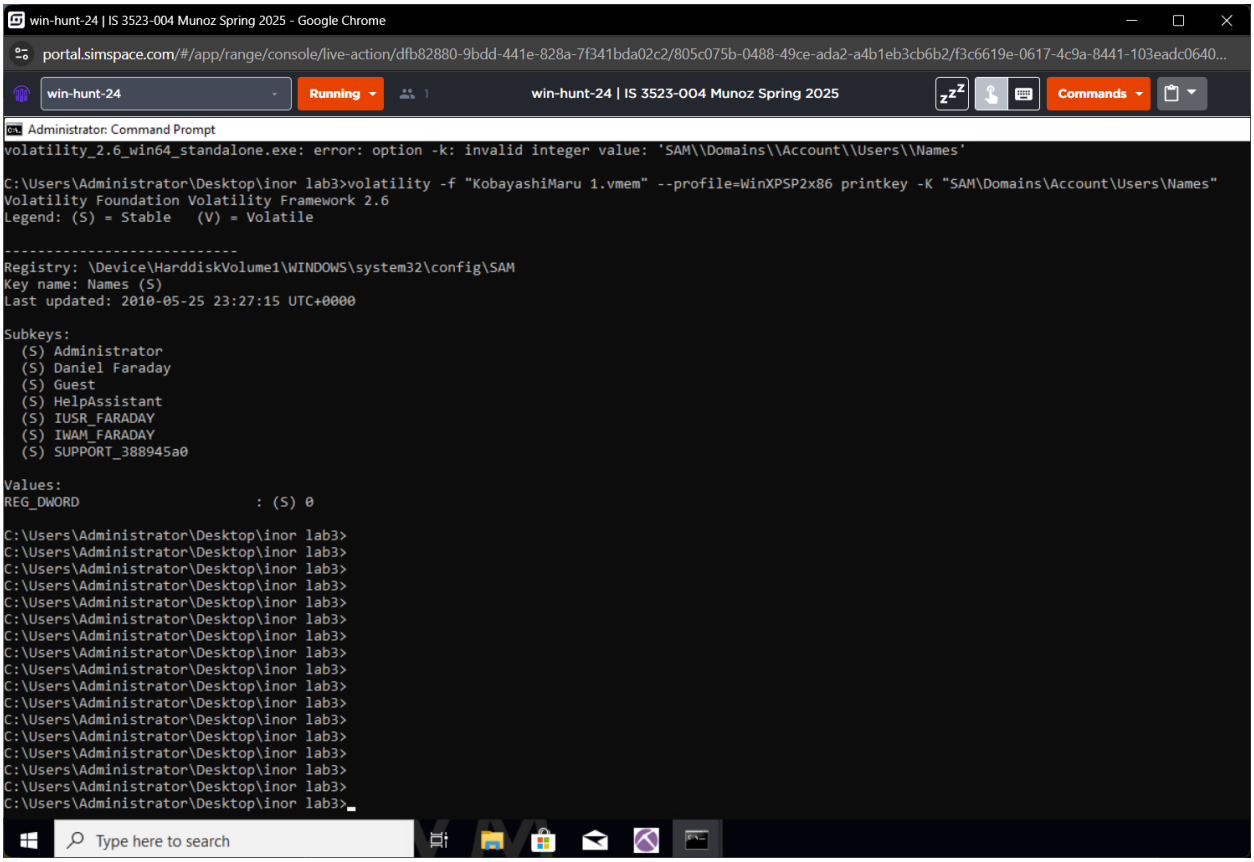

Figure 5: Showing the usernames

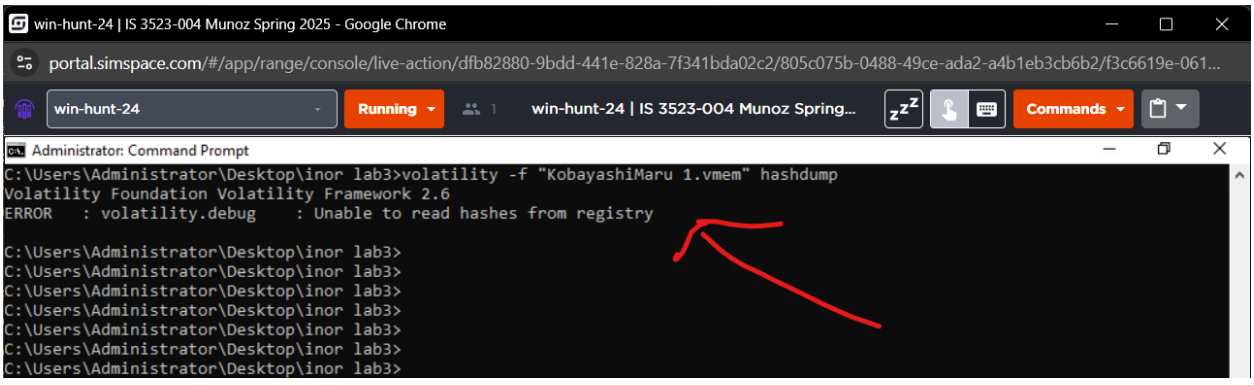

Figure 6: hashdump command

5. View the Dynamically Linked Libraries. Does this look like your average box?
    a. This does not look like my average box because there are multiple abnormal DLLs. I viewed the DLLs of the .vmem file using the command, “volatility -f “KobayasiMaru 1.vmem” –profile=WinXPSP2x86 dllllist”. However, I piped it into a .txt file so I can read it more easily as shown in Figure 7.

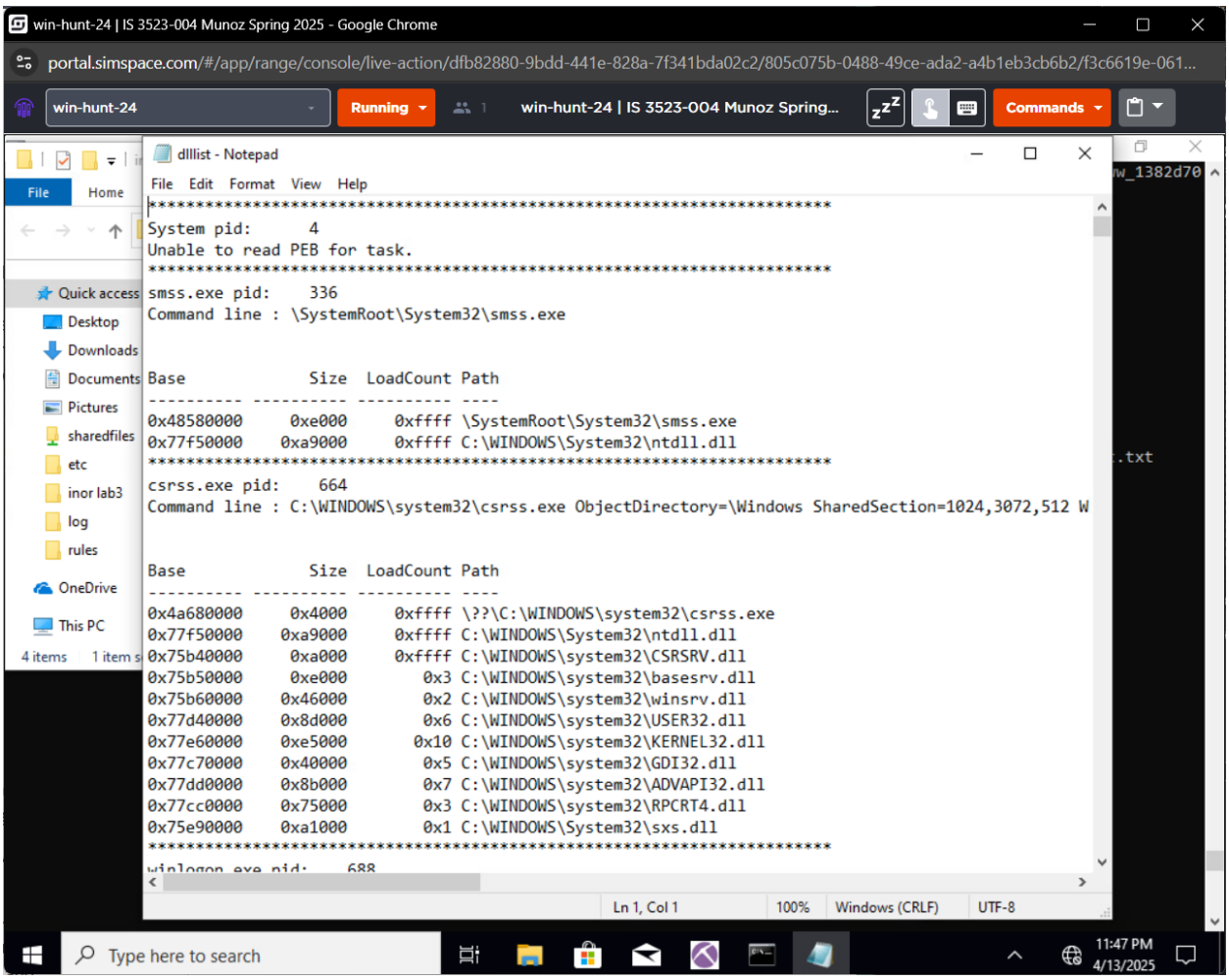

Figure 7: dlllist command into .txt file

    b. I specficailly searched for the PID of 1416 because I knew that it was a malicious root kit process, which is shown in Figure 8. I also specifically searched for multiple other PIDs such as 1472 which is shown in Figure 9. I searched for PID of 1480 which is as shown in Figure 10. I searched for PID of 1728 which is shown in Figure 11. I also searched for PID of 532. I also searched for PID of 480. It seems like most of the DLLs are normal however there are two that caught my eye, “crygcrypt-o.dll” and “cygwin1.dll” as shown in Figure 11.

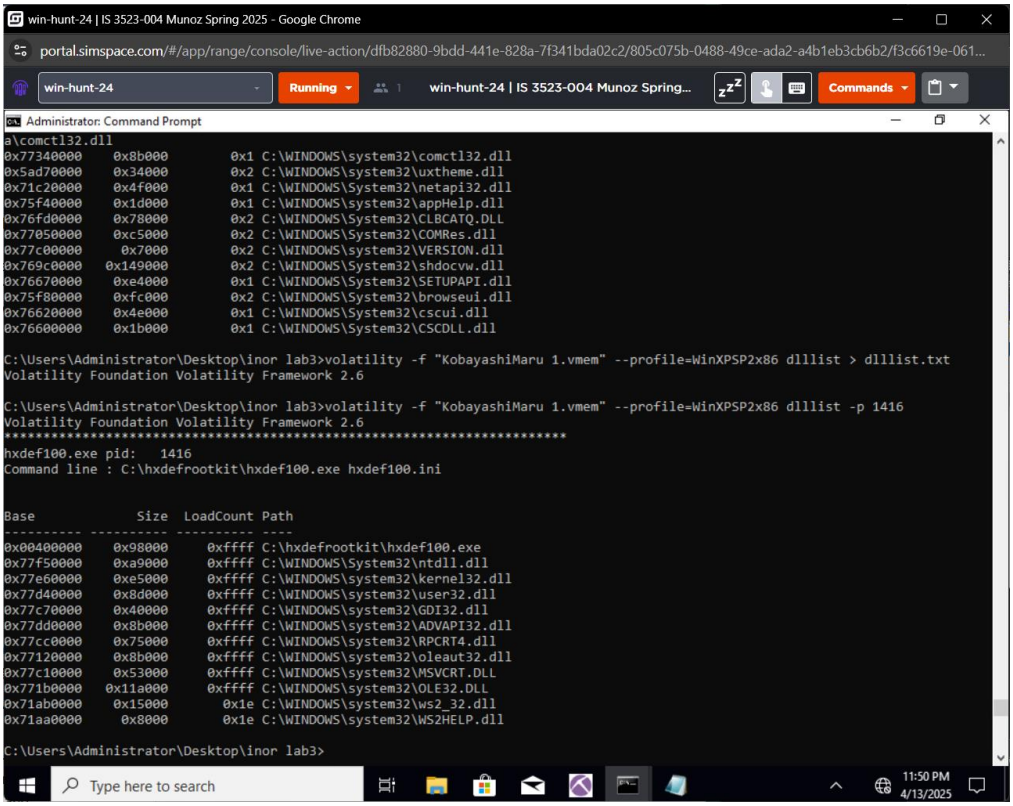

Figure 8: PID 1416

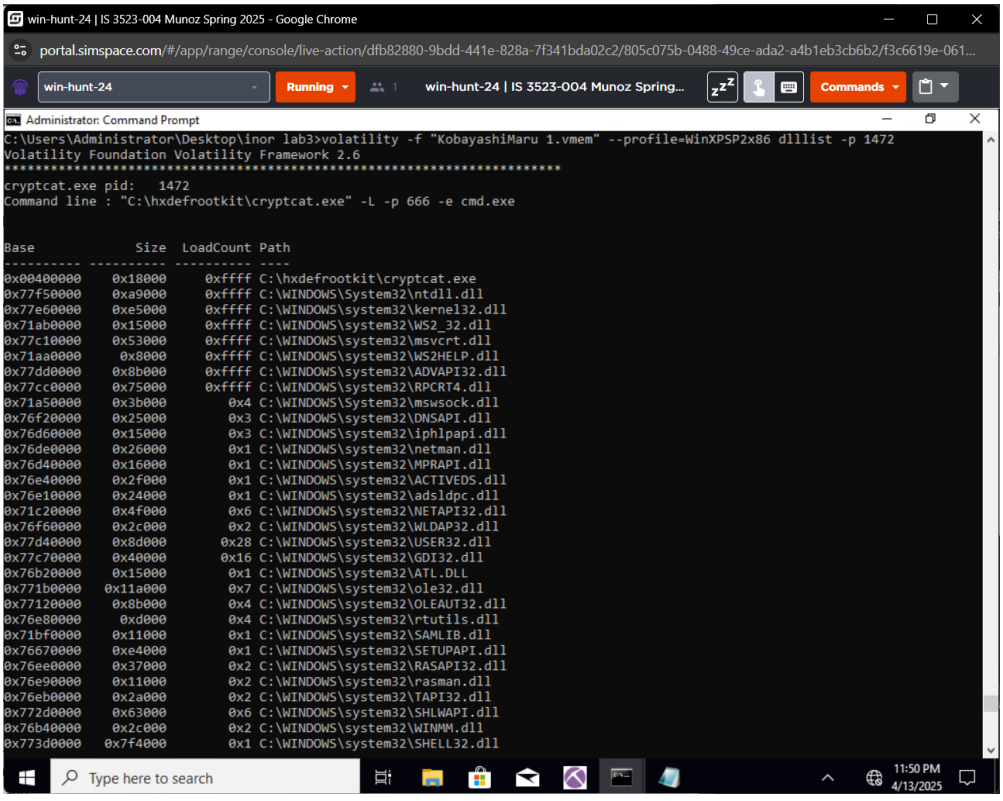

Figure 9: PID 1472

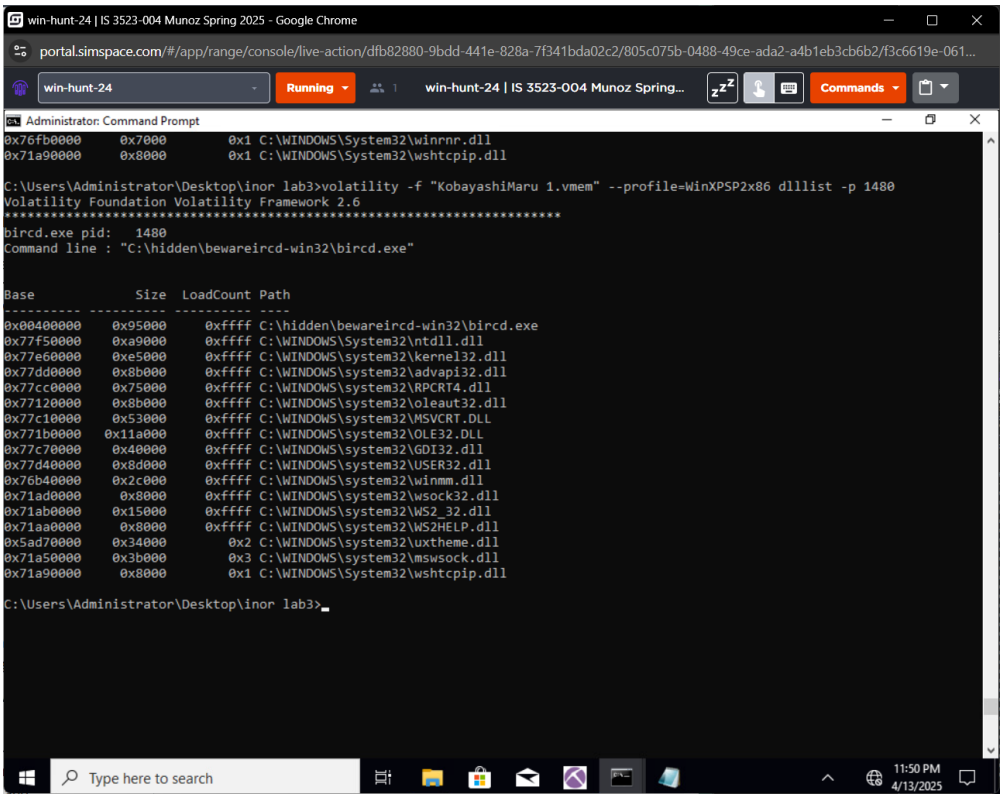

Figure 10: PID 1480

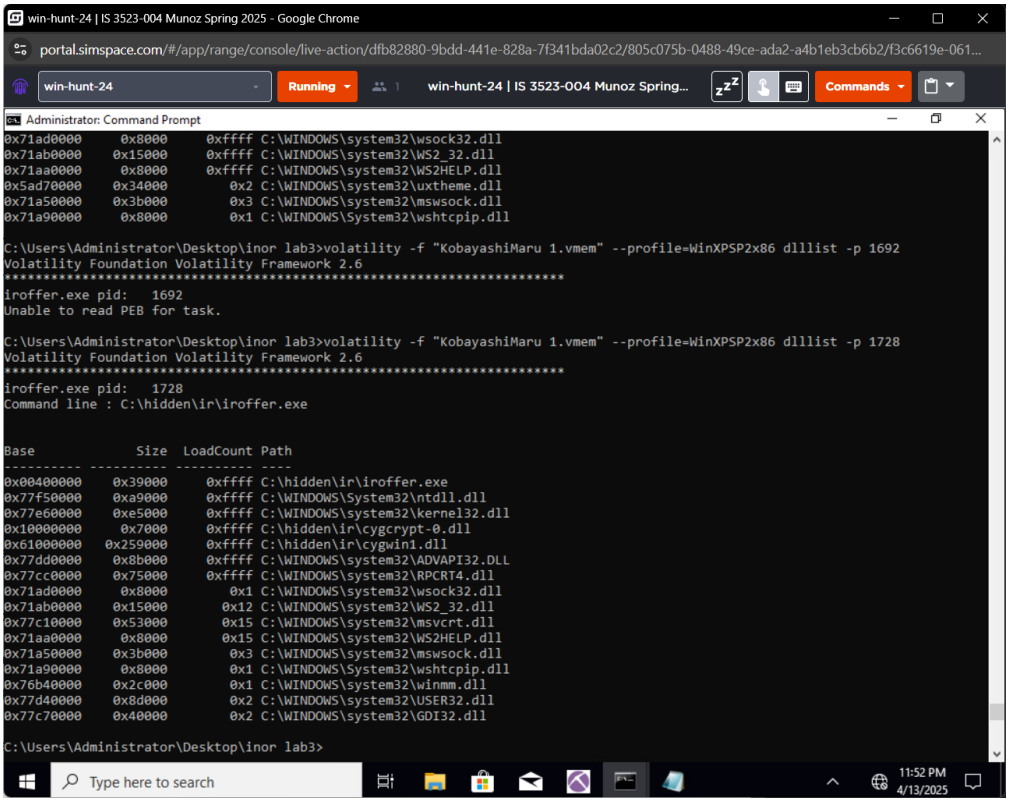

Figure 11: PID 1728

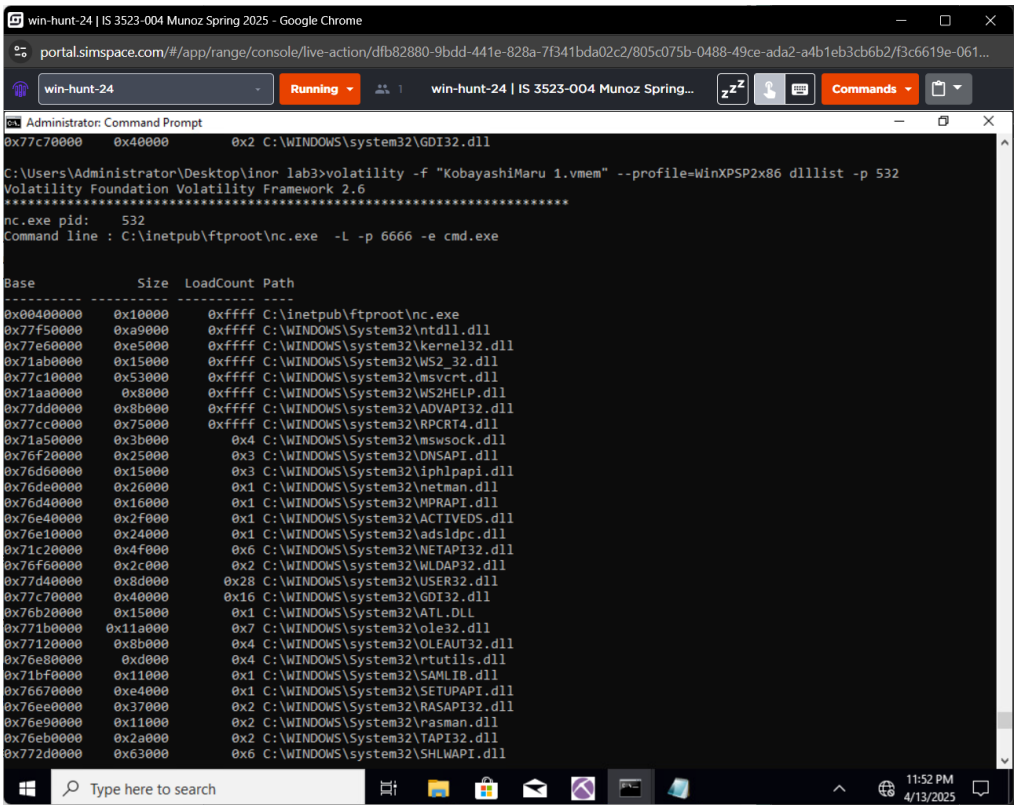

Figure 12: PID 532

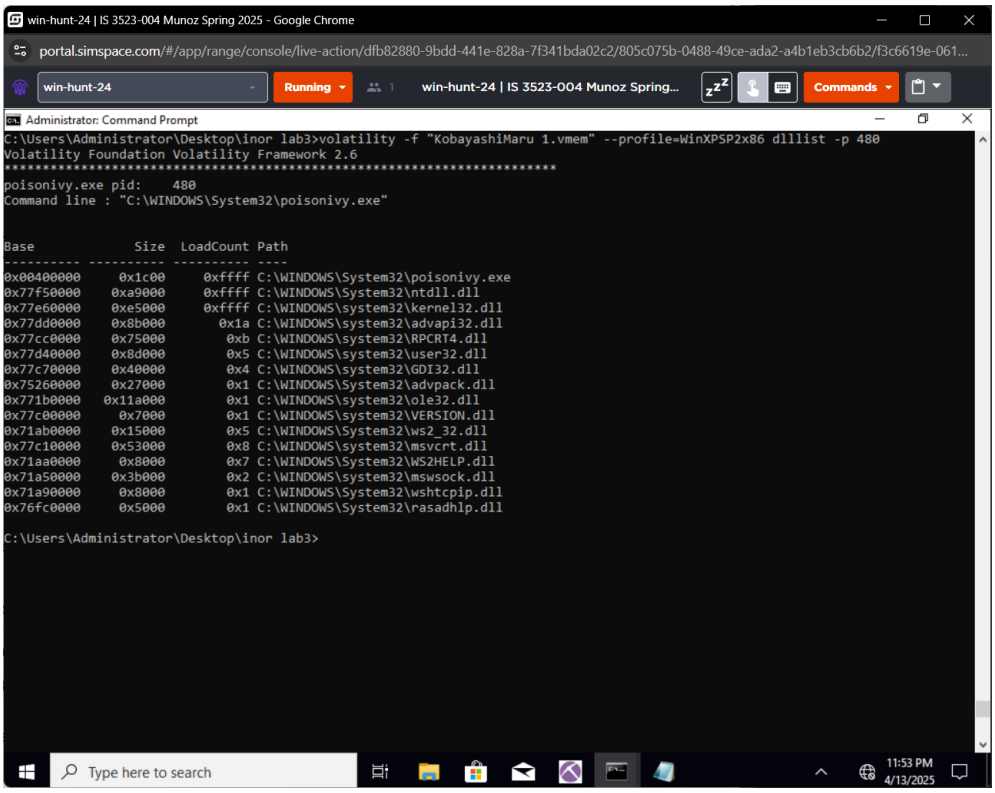

Figure 13: PID 480

6. Can you associate any Processes (PIDs), DLLs, and executables?
    a. Of course, I can associate the PID of 1416 with the hxdef100.exe executable because it clearly states it in Figure 8. Every PID has their corresponding .exe as shown in Figures 8-13. I can also associate the DLLs, “cygcrypt-o.dll” and “cygwin1.dd”, with cryptcat.
7. View the files associated with the processes. Do any files or file paths look abnormal? Reference the file path if available.
    a. All of the processes that I identified that were abnormal have file paths that look abnormal. For hxdef100.exe (PID: 1416), the file path is “C:\hxdefrootkit\hxdef100.exe”, which shows that it is a rootkit as shown in Figure 8. For cryptcat.exe (PID: 1472), the file path is “C:\hxdefrootkit\cryptcat.exe”, which shows that it is apart of the rootkit as shown in Figure 9. For bircd.exe (PID: 1480”, the file path is “C:\hidden\bewareircd-win32\bircd.exe” as shown in Figure 10. This seems abnormal because it is in a hidden directory. For iroffer.exe (PID: 1728), the file path is “C:\hidden\ir\iroffer.exe” as shown in Figure 11 which is abnormal since it is in a hidden directory. For nc.exe (PID: 532), the file path is “c:\inetpub\ftproot\nc.exe” as shown in Figure 12 which shows that it is abnormal since it is in a ftp server’s directory which mean it may not supposed to be on the system. For poisonivy.exe (PID: 480), the file path is “C:\WINDOWS\System32\poisonivy.exe” which shows that the file path isn’t abnormal as shown in Figure 13. However, the name of the program itself warrants further investigation.
8. Explain what you think happened to this box.
    a. I am going to explain what I think happened to this box in the Story section of my lab report however I wanted to use this section to incorporate more information. I used the malfind option to look for malicious code within the processes. I went through each process and then found some malicious code within the PID: 480 as shown in Figure 14. I then used the command, “volatility -f “KobayashiMaru 1.vmem” –profile=WinXPSP2x86 procdump -p 480 –dump-dir=.”, to create an executable. I then used the md5sum command to get the md5 and then I put the hash into virustotal as shown in Figure 15.

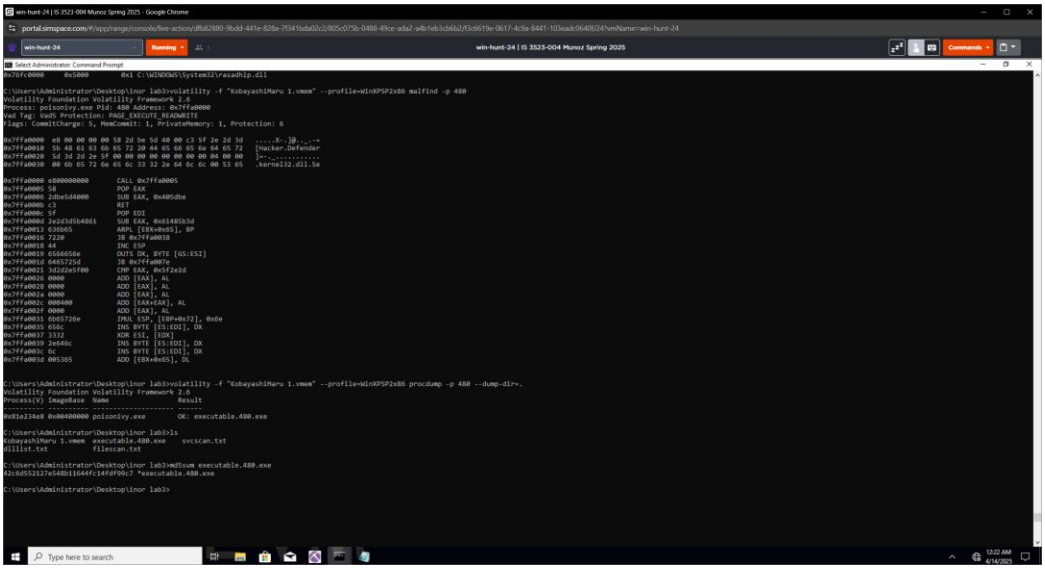

Figure 14: Commands for malfind for PID: 480

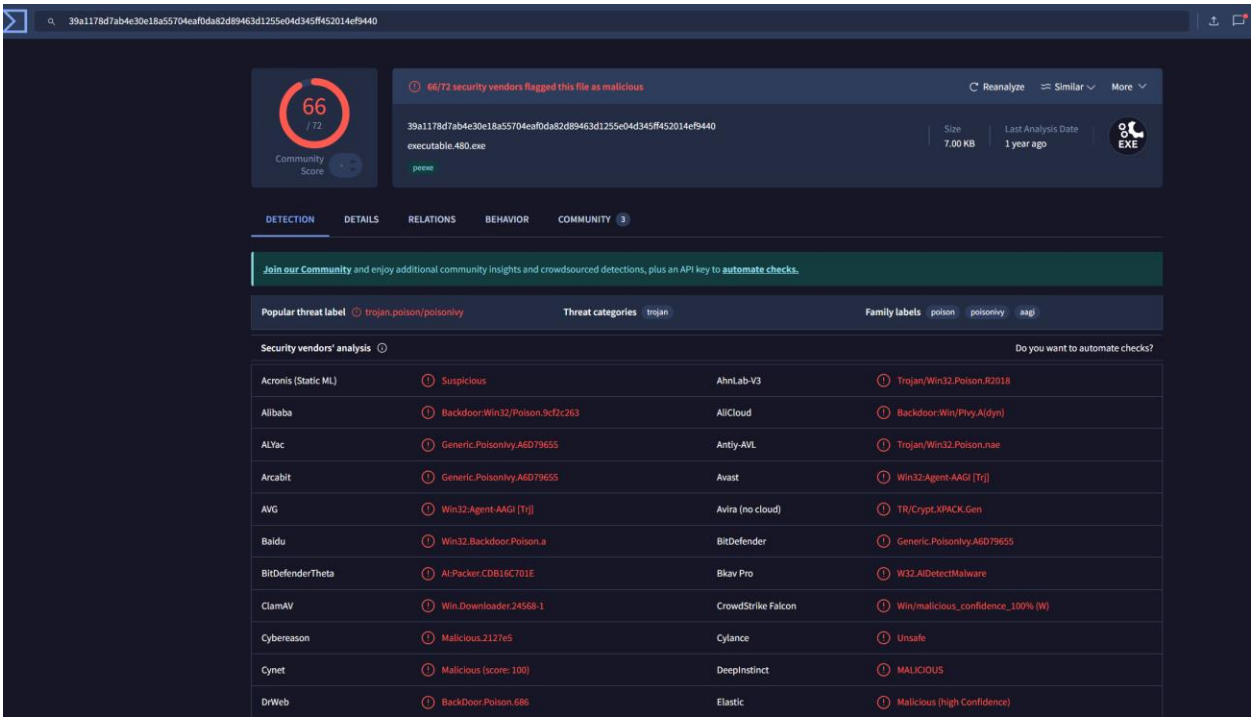

Figure 15: md5hash into virustotal

Story

The memory forensics investigation of this Windows XP SP2 system uncovered strong evidence that it was compromised and actively used for malicious purposes. Initially, the Volatility imageinfo plugin confirmed the operating system to be Windows XP SP2 on a 32-bit platform, which helped guide the rest of the analysis. From there, the process list (pslist) quickly revealed multiple suspicious executables beyond the standard Windows processes. Specifically, hxdef100.exe, cryptcat.exe, bircd.exe, iroffer.exe, poisonivy.exe, and nc.exe were running. Each one stood out, since these filenames are not part of a typical Windows environment and they strongly suggest malicious intent. Digging deeper, the rootkit located in the folder C:\hxdefrootkit\ appeared to be the center of the stealth mechanisms on this machine. The rootkit driver, hxdefdrv.sys, was identified through the modules plugin. This driver was mapped from the \??\C:\hxdefrootkit\ path, an immediate indication that attackers had gained high-level access and installed specialized software to conceal or elevate their operations. The hashes of the rootkit files, if extracted, could further confirm their malicious nature via malware databases. Additionally, the presence of Netcat (nc.exe) and Cryptcat (cryptcat.exe) is a telltale sign of an attacker's desire to create backdoors or tunnels. Both utilities allow for covert data transfer and remote command execution, making them favored tools among adversaries. Evidence of these utilities was found in nonstandard or hidden directories, with Cryptcat residing in the same rootkit folder and Netcat appearing in c:\inetpub\ftproot\nc.exe. Placing Netcat in the FTP root directory strongly indicates that the attackers intended to maintain ongoing external access or to transfer data, perhaps bypassing security controls and leaving fewer conventional forensic traces on the system. The system also had IRC-related tools, specifically bircd.exe and iroffer.exe, installed in C:\hidden\bewareircd-win32\ and C:\hidden\ir\ respectively. IRC (Internet Relay Chat) platforms are commonly abused by attackers for stealthy command-and-control channels or botnet orchestration, often under the radar of standard network monitoring. That these directories were deliberately hidden further underscores the operators' efforts to evade detection. Of particular concern was poisonivy.exe in the C:\WINDOWS\System32\ directory. Poison Ivy is a well-known Remote Access Trojan (RAT) capable of granting attackers full remote access to the compromised system. Volatility's malfind plugin provided further evidence of malicious code injection in this process (PID: 480). By dumping this process and examining it in VirusTotal, investigators confirmed that it matched known malicious signatures, reinforcing the conclusion that the RAT was actively installed and operational. Although user account details were retrieved by examining memory structures (usernames such as "Administrator," "Daniel Faraday," and "IUSR_FARADAY"), no clear-text passwords were discovered. Attempts to obtain plaintext credentials via hashdump did reveal hashed passwords, which could still be prone to offline cracking attempts. However, the lack of immediately available plaintext passwords reduced the potential for real-time credential compromise. Overall, the evidence, rootkit software, remote access tools like Poison Ivy, network tunneling programs such as Netcat and Cryptcat, and IRC services, clearly indicates a coordinated attack designed to maintain persistent, stealthy, and highly flexible control over this system. The hidden directories, unusual process paths, and malicious DLLs all point to purposeful obfuscation techniques. By capturing and analyzing the memory image, investigators were able to piece together the attackers' toolkit,

confirm the breach, and gain critical insight into how adversaries established footholds, escalated privileges, and prepared for data exfiltration or additional lateral movement.

Conclusion

In conclusion, this case underlines the critical value of memory forensics in uncovering the full scope of malicious activity on a compromised system. Traditional disk-based analysis alone might overlook transient or volatile data, such as actively running processes, injected code, and in-memory credentials, which only memory captures can preserve. By examining the system's live state, investigators gain unparalleled insight into the sequence of events and the adversary's tactics, techniques, and procedures (TTPs). For instance, memory analysis allowed us to detect the presence of a rootkit, backdoor RATs, IRC servers, and other hidden processes that an attacker specifically configured to run in stealth or hide from conventional file scans. Without this deep inspection, these components might never have surfaced, leaving incident responders and security teams unaware of the full extent of the breach and thus less able to respond effectively or remediate vulnerabilities. Furthermore, memory forensics highlights the dynamic and often fleeting nature of digital attacks, where malicious code can be injected into legitimate processes, run without leaving clear footprints on the disk, and vanish if the system is rebooted. By capturing and analyzing the entire state of the system's RAM, analysts can preserve crucial evidence that demonstrates how attackers gained access, maintained persistence, moved laterally, or exfiltrated data. This kind of forensic detail is not only instrumental for securing an organization's infrastructure against future threats but also bears significant weight when presenting findings to legal authorities or during cyber insurance evaluations. Ultimately, in a threat landscape where adversaries are increasingly sophisticated and capable of bypassing traditional protections, comprehensive memory analysis stands as one of the most reliable methods of unearthing the hidden clues necessary to fully understand an intrusion, contain it, and prevent further compromise.

Bibliography

Bircd.org. (n.d.). https://www.bircd.org/

Chatgpt. (n.d.-a). https://chatgpt.com/

Claudiouzelac. (n.d.). *ROOTKIT.COM/HF/HXDEF100R/READMEEN.TXT at master · Claudiouzelac/rootkit.com*. GitHub. https://github.com/claudiouzelac/rootkit.com/blob/master/hf/hxdef100r/readmeen.txt

Cryptcat: Kali linux tools. Kali Linux. (2024, March 11). https://www.kali.org/tools/cryptcat/

Munoz, J. (n.d.). personal.

Wikimedia Foundation. (2025a, February 27). *IRC*. Wikipedia. https://en.wikipedia.org/wiki/IRC

Wikimedia Foundation. (2025c, March 7). *Rootkit*. Wikipedia. https://en.wikipedia.org/wiki/Rootkit

Wikimedia Foundation. (2025e, March 16). *Internet information services*. Wikipedia. https://en.wikipedia.org/wiki/Internet_Information_Services#:~:text=Microsoft%20IIS%20(Internet%20Information%20Services,coding%20optimization%2C%20sitemaps%20/%20robots.

Yasar, K. (2022, October 20). *What is a rat (Remote Access Trojan)?: Definition from TechTarget*. Search Security. https://www.techtarget.com/searchsecurity/definition/RAT-remote-access-Trojan

Yasar, K., & Lockhart, E. (2022, June 15). *What is the windows registry editor? - definition from techtarget.com*. Search Enterprise Desktop. https://www.techtarget.com/searchenterprisedesktop/definition/Windows-Registry-Editor