ISCS 3523-004

Lab 03: Conducting a network IR Investigation

Christian Barba; fgb587

November 6th, 2022

Table of Contents

List of IOCs

- Wireshark shows TCP influx between two computers indicating a download.
- Wireshark shows port 4444 being used, indicates Metasploit being used
- Process Monitor shows the .exe interacting with carious HKLM software
- Process Monitor shows .exe file accessing several DLLs
- Process Explorer shows .exe is related to the command line
- Process Explorer shows a surplus of svchost.exe applications running
- Virus Total indicated that the file was malicious
- Network Miner Hosts shows TCP Port 4444(Meterpreter)
- Network Miner DLL file found from Meterpreter
- Suricata Alert System

Introduction

The purpose of this lab is to safely conduct a simulation about someone infiltrating another system. We were able to play the role of both attacker and victim by creating malware, in this case my malware is called “Destiny_Launcher.exe,” and injecting it into a vulnerable Windows virtual machine. During this process, there was monitoring happening on several layers of software, such as Wireshark, Process Monitor, and Process Explorer to name a few. These application as well as analysis from other applications allowed us to find the indicators of compromise for the victim’s computer. Being able to see the various alerts and triggers allows us to gain more knowledge on the attack and will allow us to see what really happened.

Wireshark

Before I could use any .pcap analysis tools, I had to capture the activity occurring on the network. As soon as I started the capture, I launched the Metasploitable malware from Kali Linux in order to not waste any time. As shown in figure 1.1, the capture started out with the host computer, 172.16.2.4, being completely flooded with TCP packets coming from an “unknown” source, 172.16.2.2. This could indicate that there was a connection establishing themselves between the two computers using the port 4444 on the attacker’s computer. The usage of port 4444 coming from the attacker side could be an indicator that Metasploit was being used because this is the default port for that software, but until there is hard evidence, we cannot be certain that this is from Metasploit. This constant stream of TCP packets occurs for the first 50 seconds of the capture until I start to download the Windows SAM file from the Kali machine. Figure 1.2 shows the Input/output graph generated by Wireshark that shows a spike in transmission data, and when I looked to inspect these packets, I was shown a long list of TCP packets transferring between the two computers. I believe that this is when the downloading of the SAM file

occurred. Throughout the rest of the scan, I could still see that there was a connection between the two computers with more sub spikes happening throughout. Nevertheless, I was able to see the connection be established between the host and attacker from the very beginning. This is proof of an unwanted user trespassing on the victim's computer and can be the first indicator of compromise. Throughout this capture, there was also more indicators happening behind the scenes.

Process Monitor and Process Explorer

Process monitor and process explorer are both applications that come from the Windows Sysinternals Suite, and they are used to show all activity that is occurring on the computer. A lot of this data can be written off as background information that the computer regularly uses to function, but by digging through both applications, there are indicators that an attack has occurred. At this time, the session had ended from the Kali machine and the .exe file had been terminated on the Windows machine. Since this is only a simulation, I relaunched the file to obtain more information, but in the real world this is the last thing any person would want to do. By inspecting Process Monitor while the malware was running, I was able to see locate the malicious "Destiny_Launcher.exe" process through the Process Tree. Looking into the process further allowed me to view all of the operations this file was doing including accessing the various HKLM registries, as shown in figure 2.1. By clicking on an operation with a path specifying the WOW6432Node, I was able to see that there were several modules being utilized, as shown in figure 2.2. These all had the DLL extension, so I knew that this file was accessing these libraries and potentially tampering with them. In total, several registry keys were opened and closed by this malware we had injected, so the evidence is pointing toward something malicious happening through this file.

Process Explorer is another tool I used to see the bytes sent and received by the .exe file. I am doing this because while Wireshark did show me all of the bytes transmitted, this is giving me a condensed view to see the severity of the damages done by the file. Figure 2.3 shows the number of bytes transmitted and received over this application. This shows that 80 kilobytes have been sent to a remote address with an IP address and port of 172.16.2.2:4444. Although the byte count is not accurate in relation to the original capture, this bit of information can be useful to see how many bytes were sent to the attacker's side had the original process still ran. Another piece of information I was able to see was in the security tab. This shows me that the session was not protected, and a privilege that was enabled for this process was "SeImpersonatePrivilege", also shown in figure 2.3. This is a clear indication that this process has the ability to mimic other accounts' privileges, and this could lead to privilege escalation. This could be a very serious problem for anyone that is unaware of the dangers of this file.

Virus Total, Network Miner, and Suricata

It was very evident at this point that the file was malicious, so I took the download and submitted it to VirusTotal.com to understand how deadly it could possibly be. The results of this scan came back with 54 detected pieces of malware including with the main one being a trojan, shown in figure 3.1. There were also indications of the virus containing a trojan backdoor-shell, and this would mean that the attacker could have accesses the shell and try to escalate their privileges or access files that they could have access to. As seen with Process Explorer, the malware had the ability to mimic privileges, so it is clear that with these two combined the malware could have all of the privileges they wanted. When an attacker is able to gain root privileges, they would be able to copy, delete, or alter any files they would like and the system would essentially be theirs for however long they wanted.

While using Network Miner, I had confirmed my suspicions that the port 4444 was an indication of Metasploit. I loaded the .pcap file into Network Miner to obtain more information about the session that occurred on Wireshark, and I saw the attacking computer on the list of hosts that interacted with the machine. By inspecting the IP address more closely, I was shown that the open TCP port used was 4444 with Meterpreter in parentheses, as shown in figure 3.2. This figure also shows the original sent and received byte count with the victim computer sending out 438 kilobytes. Another alarming piece of information obtained from network Miner is that there was a file that was downloaded early on in the session. The file downloaded was named meterpreter.dll, and the DLL extension could be evidence of a DLL injection happening on the victim machine. A DLL injection is when an attacker attempts to "evade process-based defenses as well as possibly elevate privileges" (Process injection). If this injection were to happen on this machine, the hacker would have full access to do what they want, and since I was the attacker, I was able to take out sensitive information.

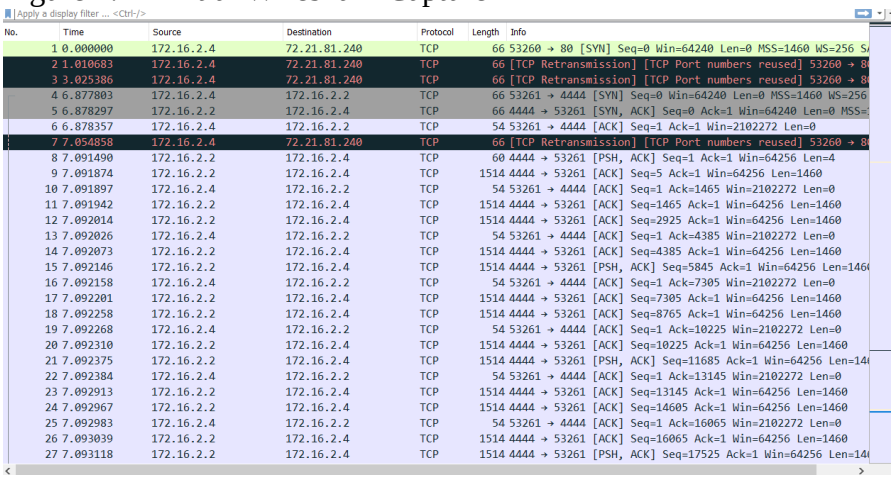
The last resource I used to obtain indicators of compromise was Suricata. Since I was able to capture the network traffic at the time of attack, I was able to load the .pcap file into Suricata to see if any alerts were triggered. Figure 3.3 shows that 1168 alerts were found by Suricata. When trying to inspect these alerts more closely, I was able to see that several alerts triggered were destined to travel through port 4444 to reach the attacking computer, shown in figure 3.4. It is very concerning to see that several thousands of bytes of data were being transmitted during the time of the alert because these bytes could be the attacker downloading bits of sensitive data from the victim's computer. Suricata is

Password Cracking

This part of the lab gave me the most trouble when trying to complete it using “John the Ripper”. By downloading the “SAM” and “SYSTEM” files, I should be able to obtain the hashes for the password I would like to crack. In order to get the hashes onto a file I can use for John, I ran the command highlighted in figure 4.1 to output the hashes into a file labeled sys.txt. I then tried to run several commands while using John, shown in figure 4.2, to try and figure out how to crack the hash file. After hours of attempts and research, I could not list the cracked password correctly although the terminal stated the password hashes were cracked.

Appendix

Figure 1.1 - Initial Wireshark Capture



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.2.4	72.21.81.240	TCP	66	53260 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 S
2	1.010683	172.16.2.4	72.21.81.240	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 53260 → 8
3	3.025386	172.16.2.4	72.21.81.240	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 53260 → 8
4	6.877803	172.16.2.4	172.16.2.2	TCP	66	53261 → 4444 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
5	6.878297	172.16.2.2	172.16.2.4	TCP	66	4444 → 53261 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=
6	6.878357	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
7	7.054858	172.16.2.4	72.21.81.240	TCP	66	[TCP Retransmission] [TCP Port numbers reused] 53260 → 8
8	7.091490	172.16.2.2	172.16.2.4	TCP	60	4444 → 53261 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=4
9	7.091874	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=5 Ack=1 Win=64256 Len=1460
10	7.091897	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=1465 Win=2102272 Len=0
11	7.091942	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=1465 Ack=1 Win=64256 Len=1460
12	7.092014	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=2925 Ack=1 Win=64256 Len=1460
13	7.092026	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=4385 Win=2102272 Len=0
14	7.092073	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=4385 Ack=1 Win=64256 Len=1460
15	7.092146	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [PSH, ACK] Seq=5845 Ack=1 Win=64256 Len=146
16	7.092158	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=7305 Win=2102272 Len=0
17	7.092201	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=7305 Ack=1 Win=64256 Len=1460
18	7.092258	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=8765 Ack=1 Win=64256 Len=1460
19	7.092268	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=10225 Win=2102272 Len=0
20	7.092310	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=10225 Ack=1 Win=64256 Len=1460
21	7.092375	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [PSH, ACK] Seq=11685 Ack=1 Win=64256 Len=14
22	7.092384	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=13145 Win=2102272 Len=0
23	7.092913	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=13145 Ack=1 Win=64256 Len=1460
24	7.092967	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=14605 Ack=1 Win=64256 Len=1460
25	7.092983	172.16.2.4	172.16.2.2	TCP	54	53261 → 4444 [ACK] Seq=1 Ack=16065 Win=2102272 Len=0
26	7.093039	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [ACK] Seq=16065 Ack=1 Win=64256 Len=1460
27	7.093118	172.16.2.2	172.16.2.4	TCP	1514	4444 → 53261 [PSH, ACK] Seq=17525 Ack=1 Win=64256 Len=14

Figure 1.2 -Wireshark Input/Output Graph

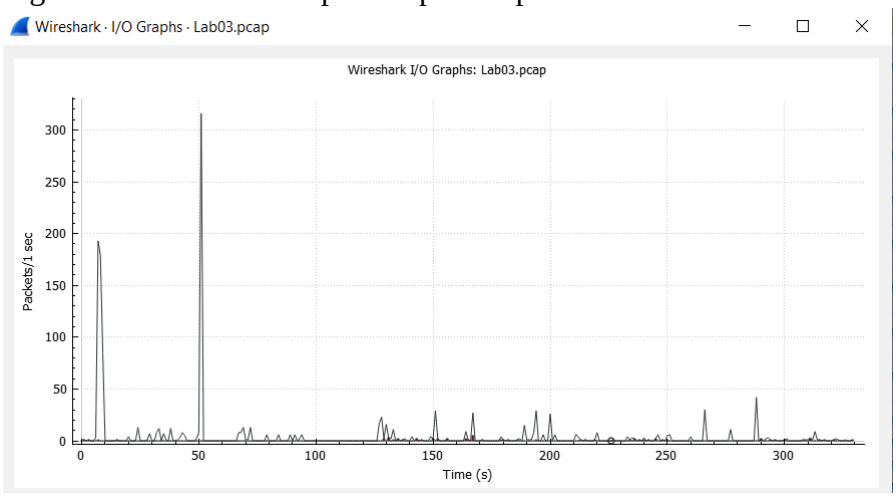


Figure 2.1 -Process Monitor list of “Destiny_Launcher.exe” operations

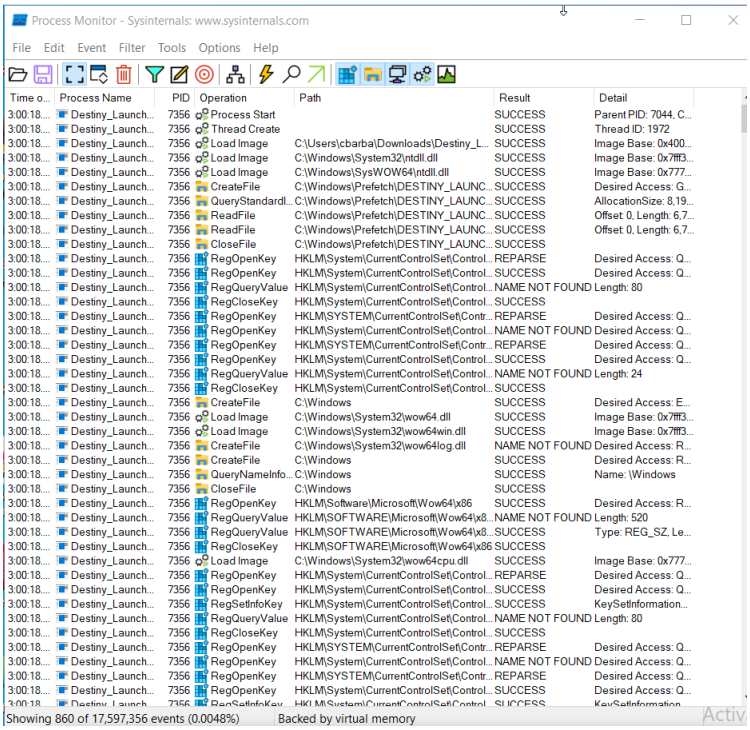


Figure 2.2 -Modules in use by “Destiny_Launcher.exe”

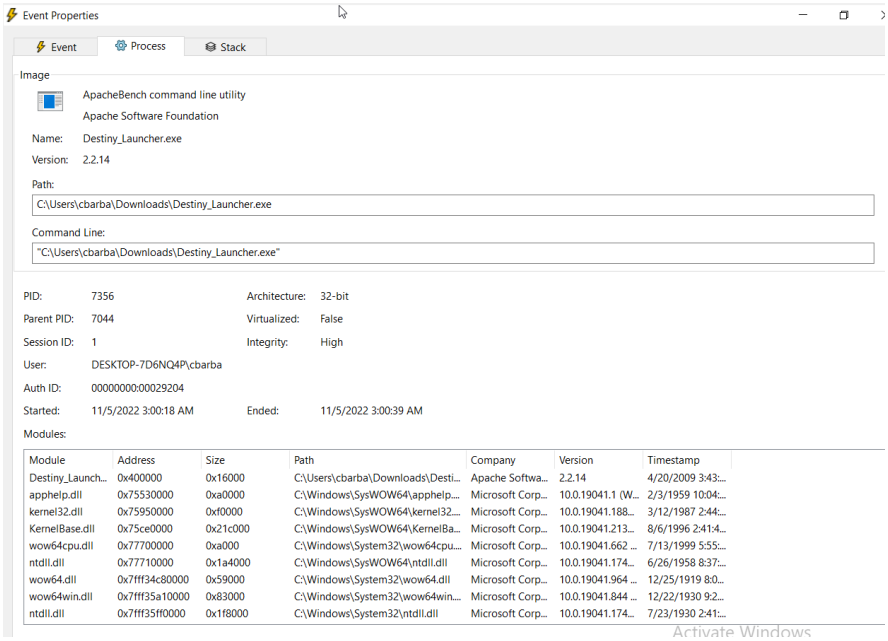


Figure 2.3 -Security and Disk & Network statistics from Process Monitor.

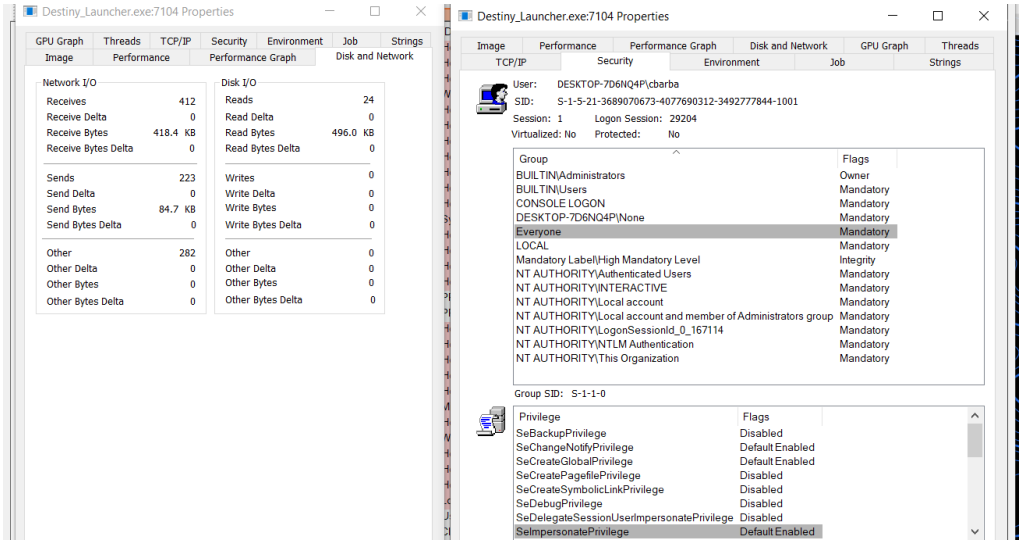


Figure 3.1 -Virus Total Results

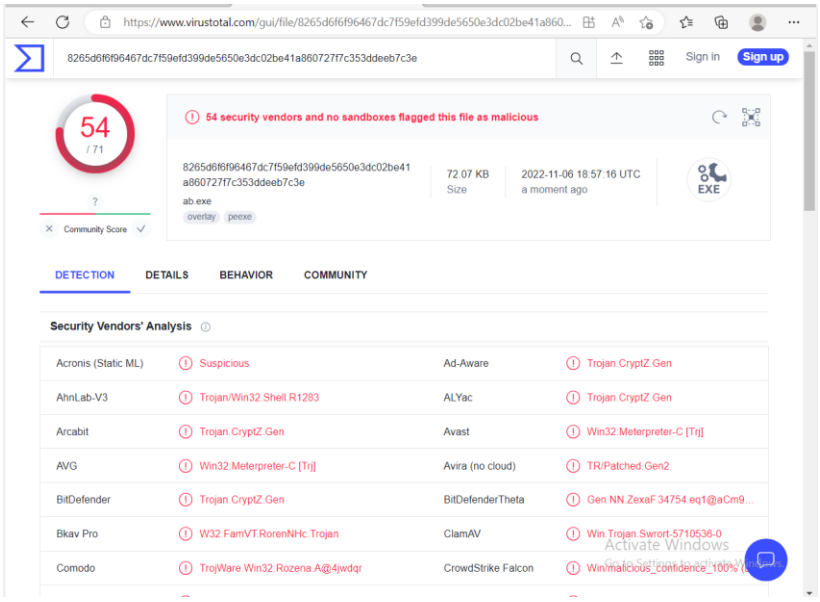


Figure 3.2 – Network Miner Hosts

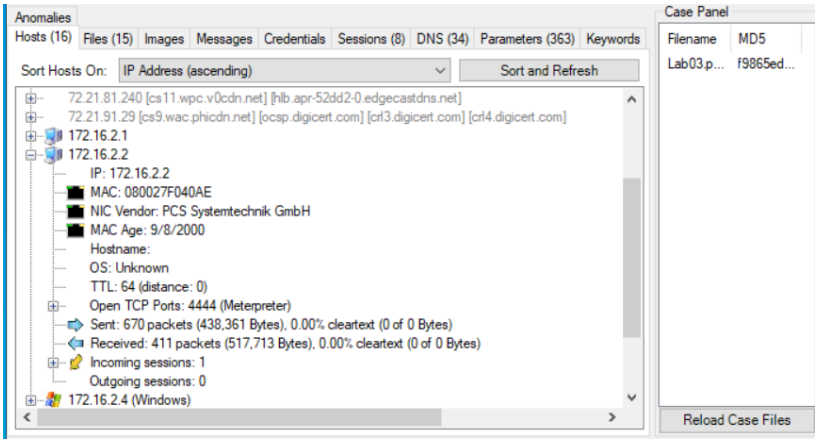


Figure 3.3 – Network Miner Files

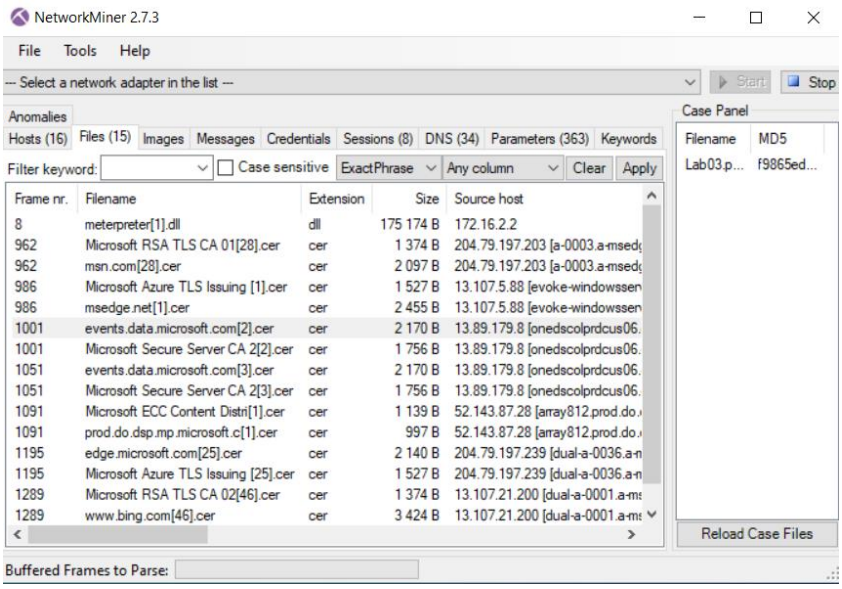


Figure 4.1 – Saving the Hashes to a sys.txt file

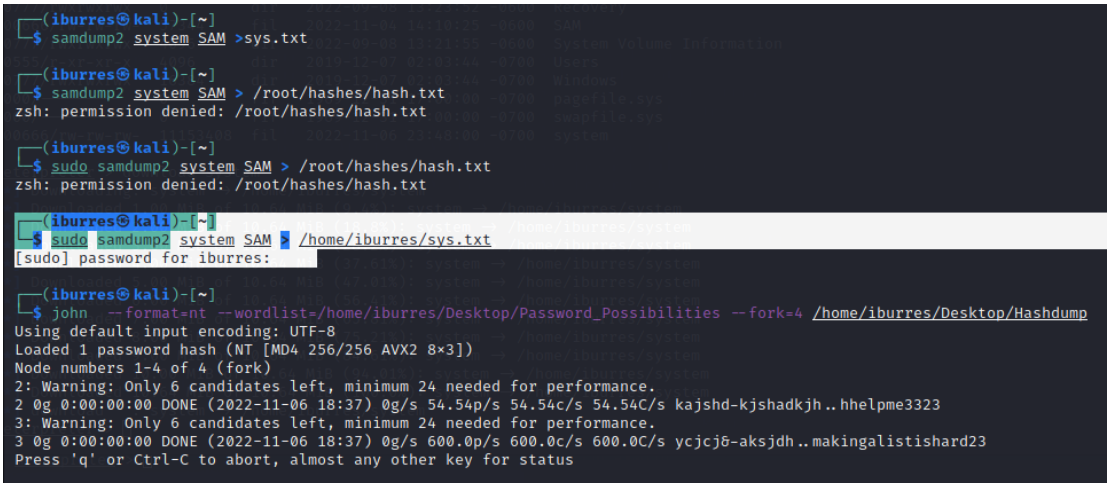
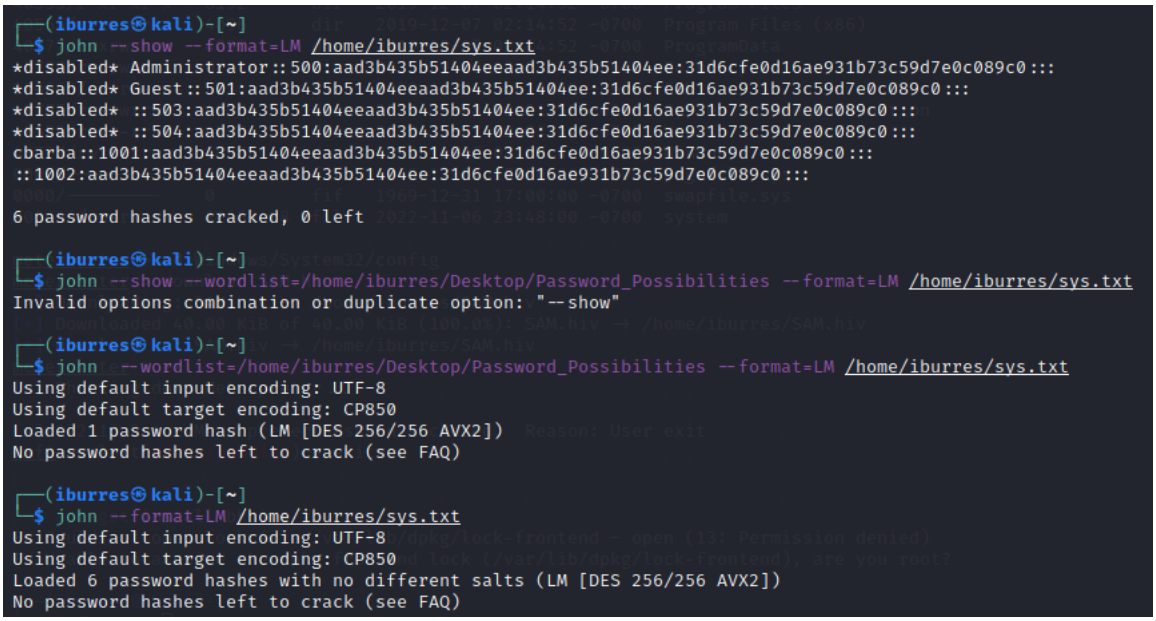


Figure 4.2 – Commands used to try to Crack Passwords



Works Cited

Jim GillCyber Security Researcher. Information security specialist. (2021, July 5). *Crack windows password with john the Ripper. A helpful tool.* Information Security Newspaper | Hacking News. Retrieved November 6, 2022, from https://www.securitynewspaper.com/2018/11/27/crack-windows-password-with-john-the-ripper/

Prasad, P. (n.d.). *Windows password cracking using john the ripper - prakhar prasad*. Retrieved November 7, 2022, from https://prakharprasad.com/blog/windows-password-cracking-using-john-the-ripper/

Process injection: Dynamic-Link Library Injection. Process Injection: Dynamic-link Library Injection, Sub-technique T1055.001 - Enterprise | MITRE ATT&CK®. (n.d.). Retrieved November 5, 2022, from https://attack.mitre.org/techniques/T1055/001/#:~:text=Other%20sub%2Dtechniques%20of%20Process%20Injection%20(12)&text=DLL%20injection%20is%20a%20method,by%20invoking%20a%20new%20thread.