MOHAMAD ALMASR 101167438

DIGITAL FORENSICS&INCID. RESP - COMP 4071

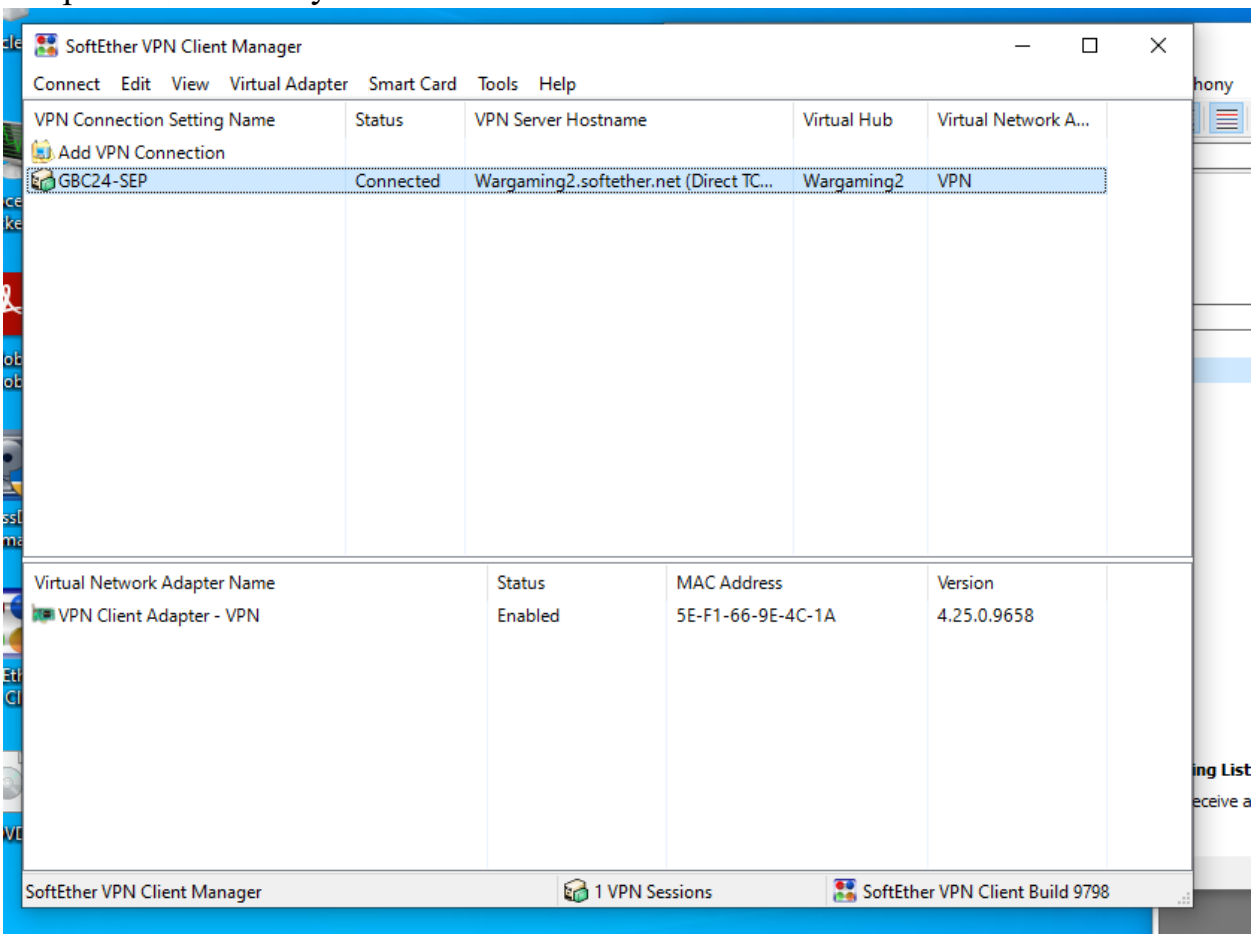
Assignment 2

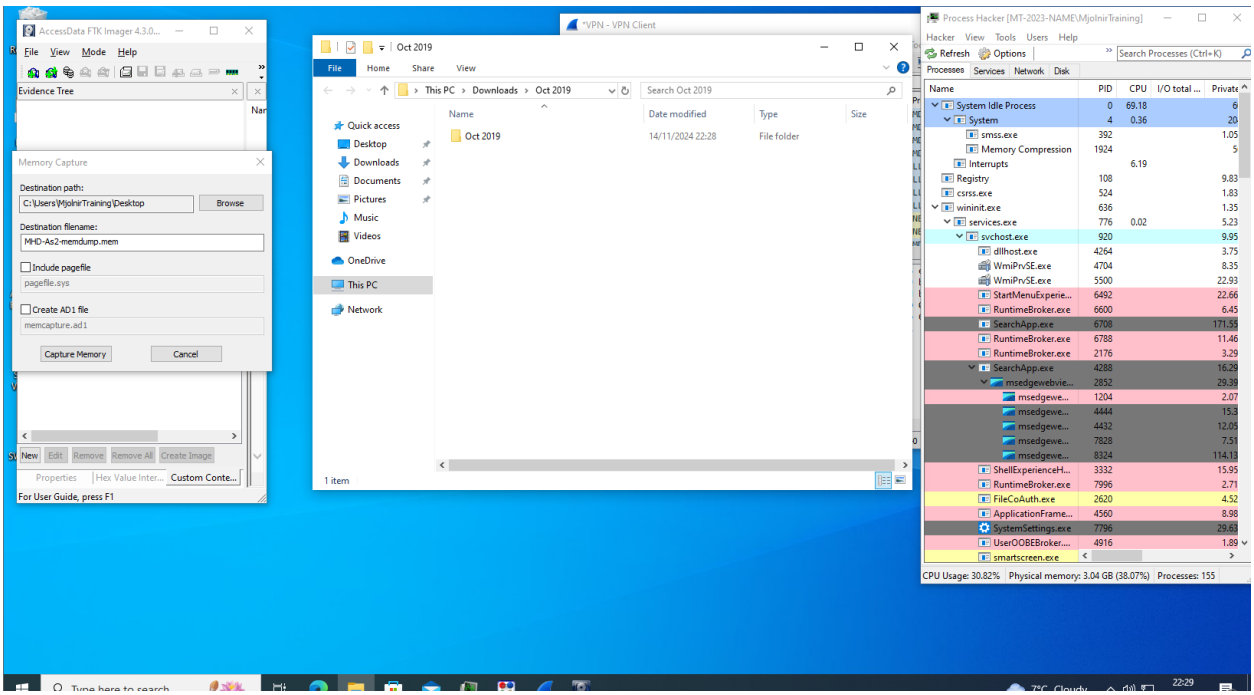
Introduction:

I executed a sample of ransomware in a virtual machine to analyze the sample in this assignment. A range of forensic tools was used in the analysis to identify malicious processes . The focus was on examining memory data, creating YARA rules related to identified ransomware and testing them with the THOR Lite scanner. This report will tell the whole story of how was the evidence collected, what was done with that evidence and finally, analysis of the evidence. Each and everything you will find in this report including a description of the tool which was used and its challenges.

At the start of the process, I prepared my VMware virtual machine and connected to the VPN. After that, I navigated to my system to access the necessary files. I opened the "October 2019" folder, which contained all the ransomware malware

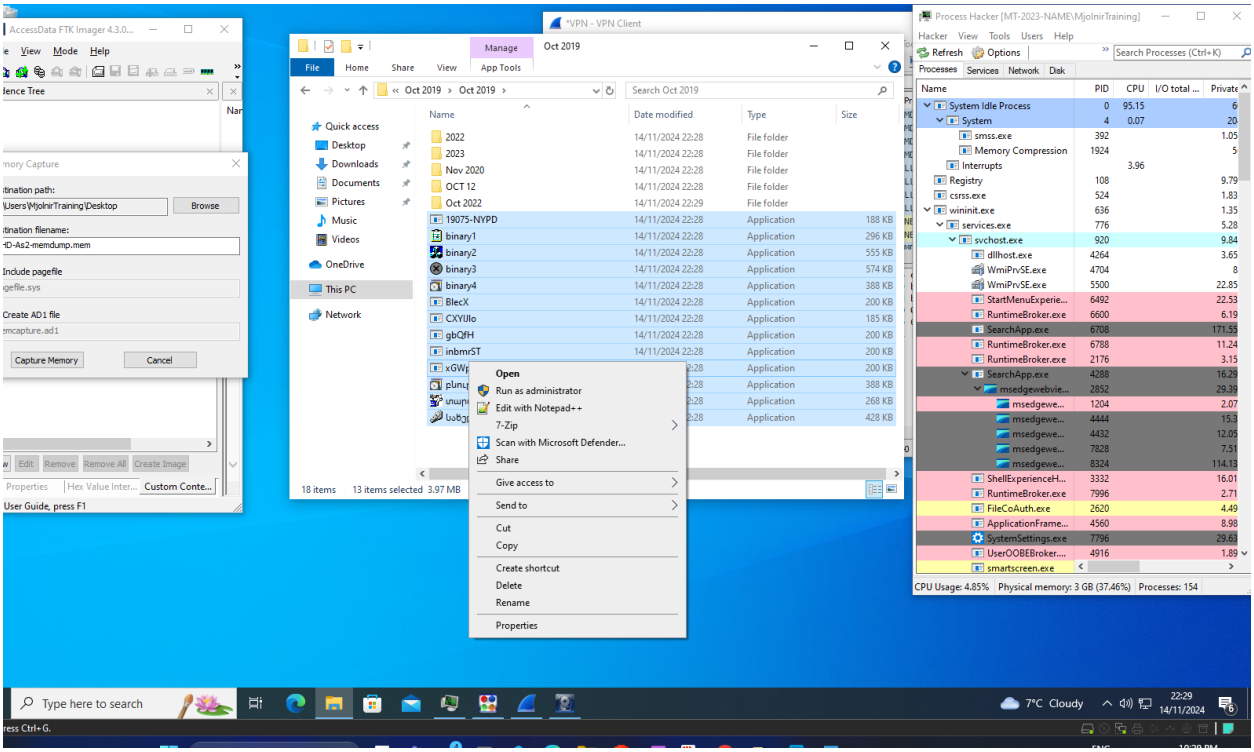
samples for this analysis.



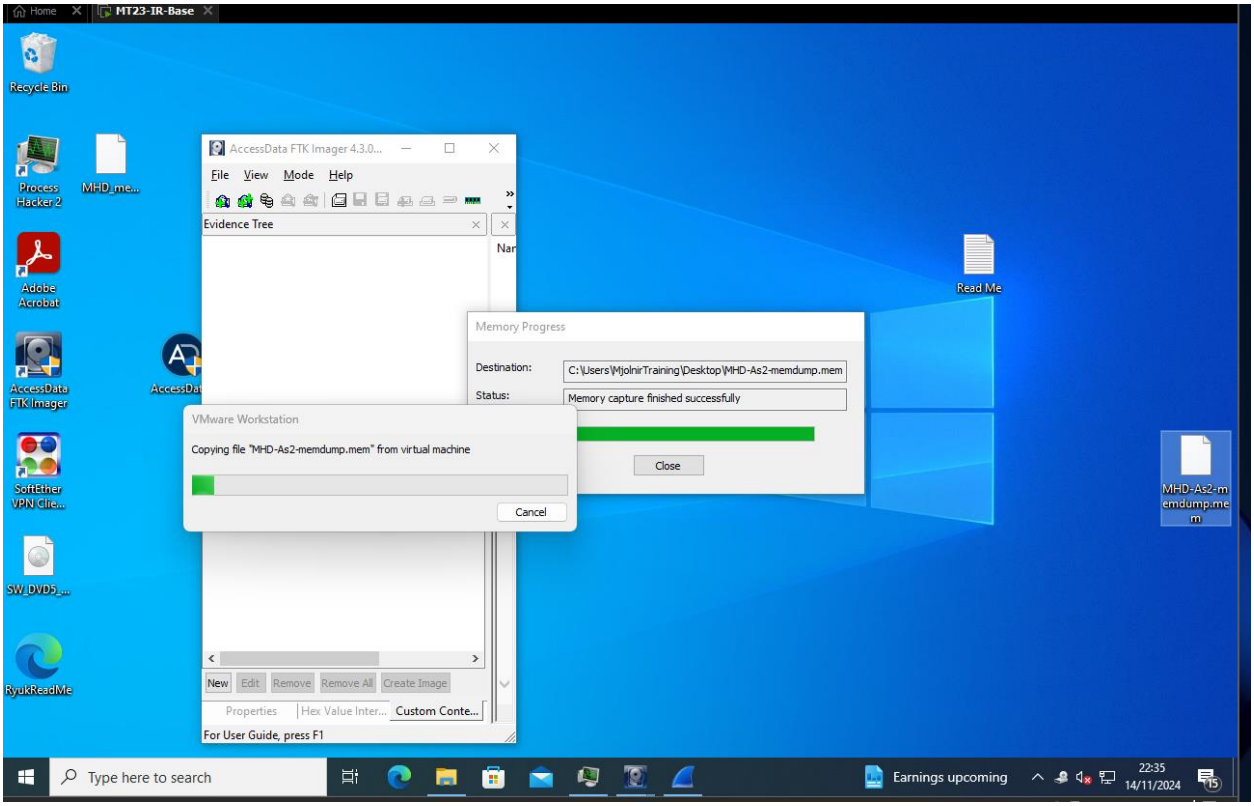


MOHAMAD ALMASR 101167438

Each ransomware sample was run as an administrator to ensure it executed correctly and fully

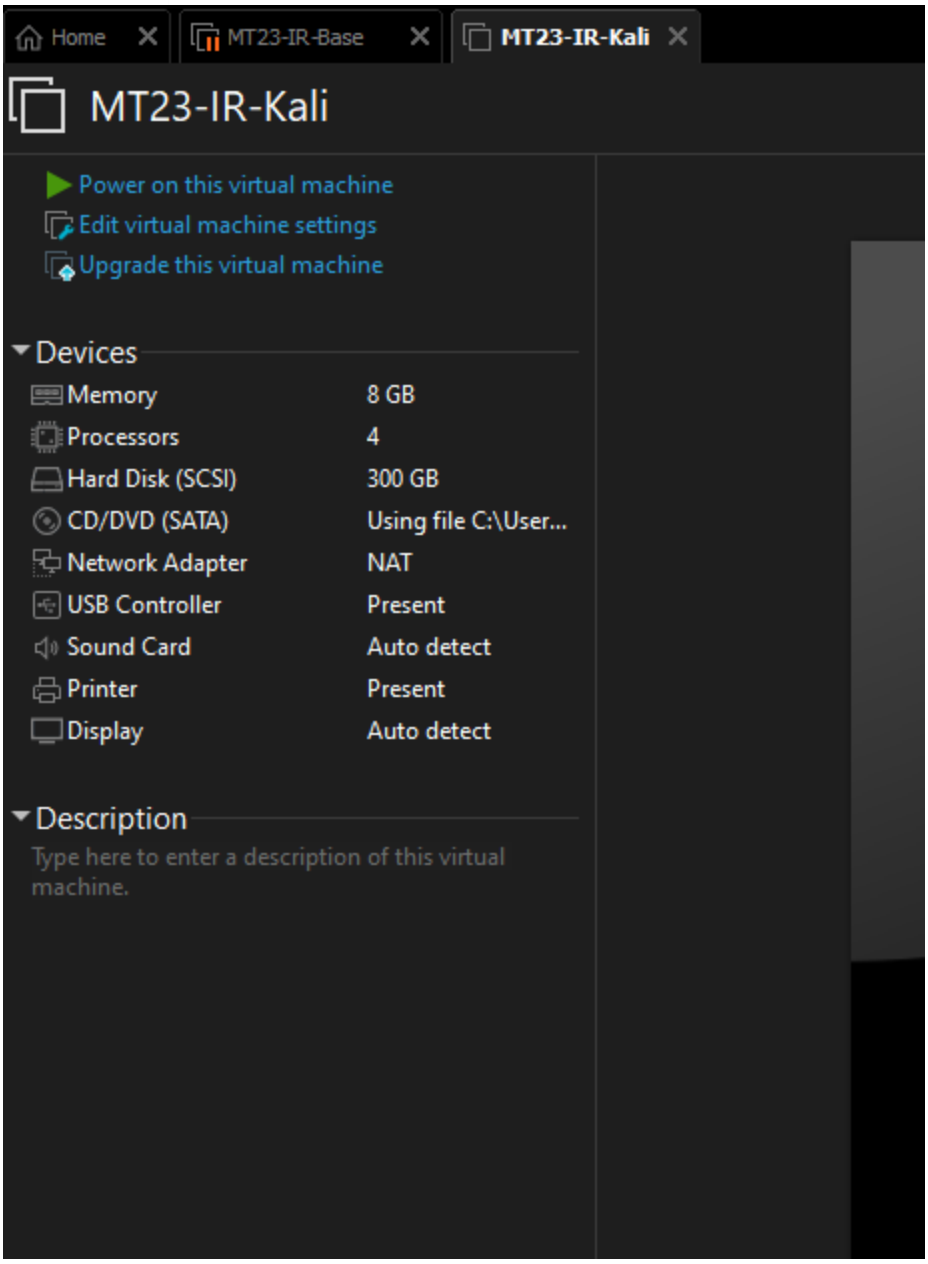


MOHAMAD ALMASR 101167438

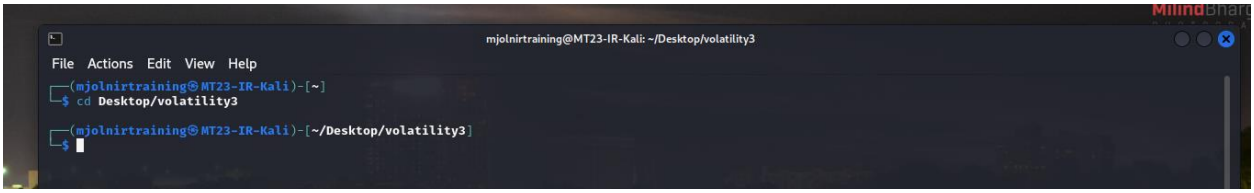


After the memory dump was complete, I copied the file to my Kali machine for analysis. The VM was paused

MOHAMAD ALMASR 101167438

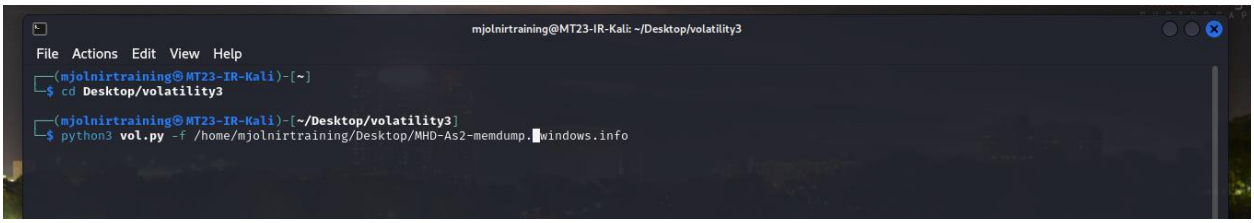


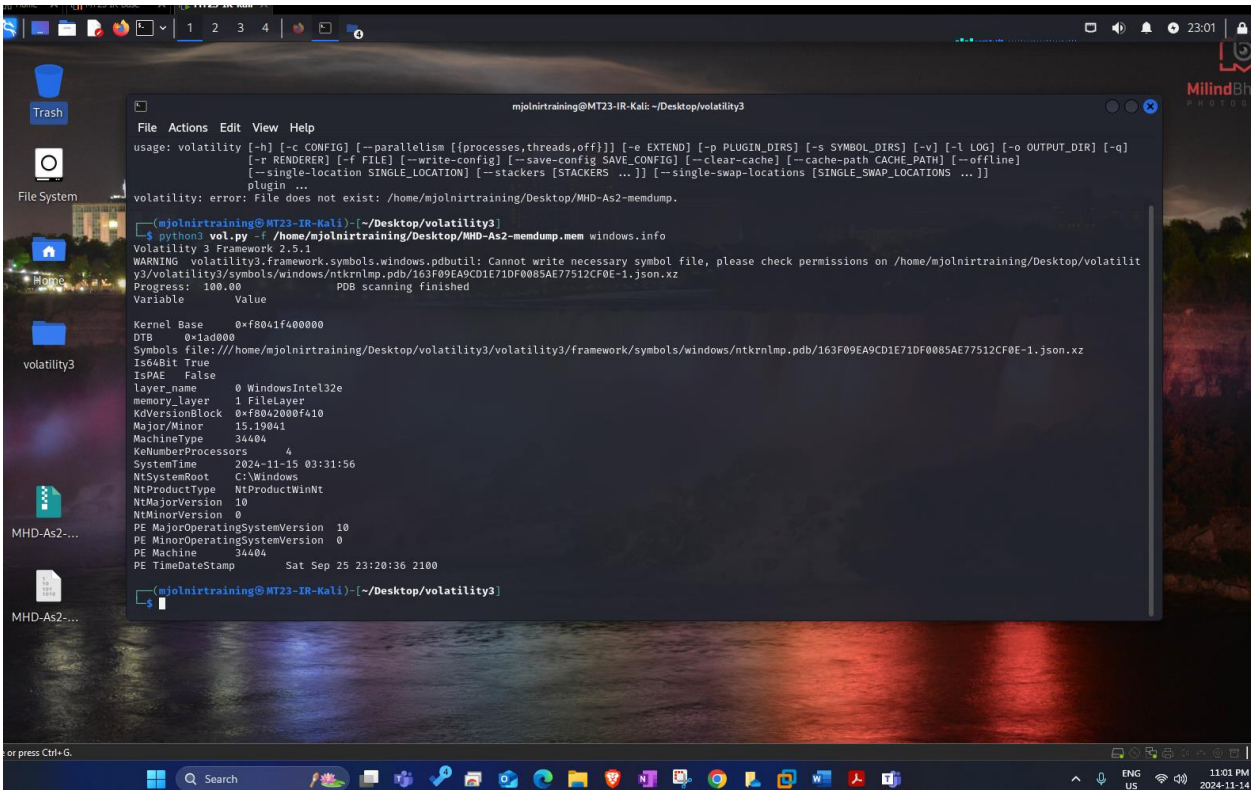
On the Kali VM, I opened the Volatility 3 framework on the terminal to analyze the memory dump.



windows.info: This command provided general information about the memory dump, such as the OS version, architecture, and more. It ensures compatibility with subsequent commands.

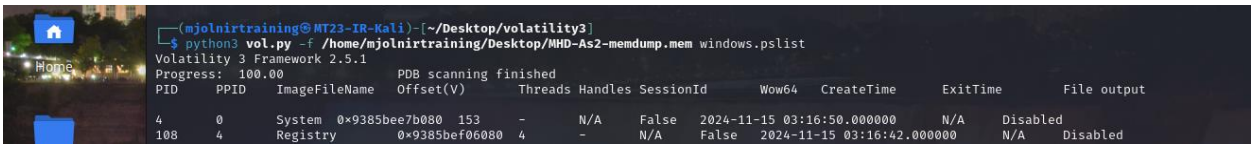
As you see the screenshot Shows the metadata about the memory dump, validating the environment details.

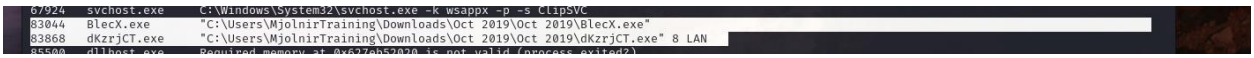


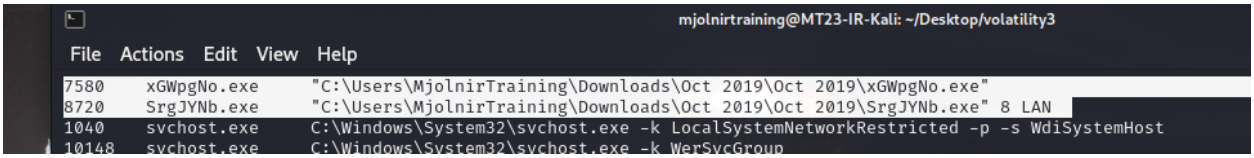


Next, I used Volatility's windows.pslist to list all active processes in the memory dump. This helped US decide which process PID to analyze. As you see the screenshot Highlights two processes identified as suspicious. Another two were found in subsequent results.

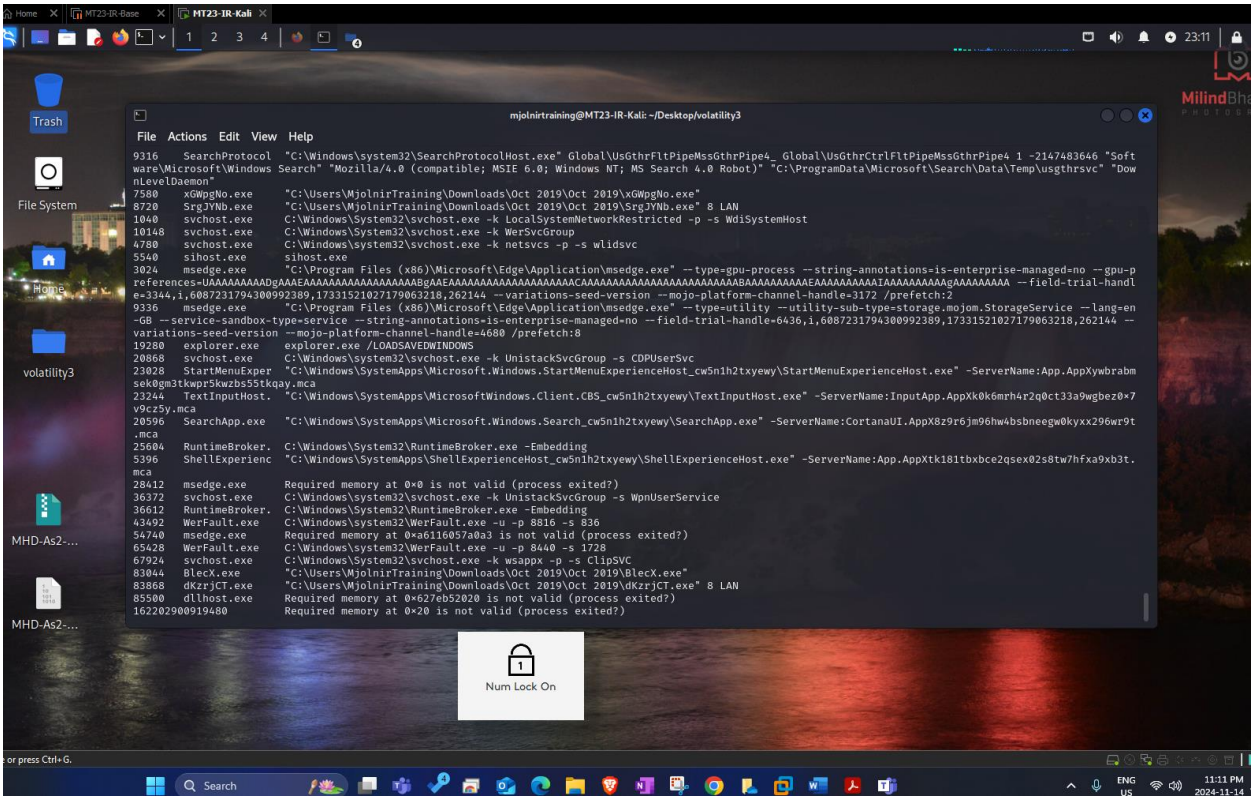
MOHAMAD ALMASR 101167438







windows.cmdline : Identifies how the process was executed and any parameters passed



Select and Dump Process: With a specific PID chosen, I used windows.dumpfiles to extract the file associated with that process. This step gave US a copy of the file in memory

Copy

MOHAMAD ALMASR 101167438

```
┌──(mjolnirtraining㉿MT23-IR-Kali)-[~/Desktop/volatility3]
└─$ python3 vol.py -f /home/mjolnirtraining/Desktop/MHD-As2-memdump.mem windows.dumpfiles.DumpFiles --pid 7580
Volatility 3 Framework 2.5.1
Progress:  100.00               PDB scanning finished
Cache   FileObject      FileName        Result
```

```
ImageSectionObject      0×9385c5221290  mpr.dll file.0×9385c5221290.0×9385c5415d30.ImageSectionObject.mpr.dll.img
DataSectionObject       0×9385c7db5ae0  xGWpgNo.exe     file.0×9385c7db5ae0.0×9385c77aba10.DataSectionObject.xGWpgNo.exe.dat
ImageSectionObject      0×9385c7db5ae0  xGWpgNo.exe     file.0×9385c7db5ae0.0×9385c73e1a20.ImageSectionObject.xGWpgNo.exe.img
ImageSectionObject      0×9385c594ecb0  virtdisk.dll    file.0×9385c594ecb0.0×9385c59a9050.ImageSectionObject.virtdisk.dll.img
```

Copy and Extract Dumped File: I copied the dumped file and saved it in a new location. This made it easier to manage and examine the file separately.

```
-rw------- 1 mjolnirtraining mjolnirtraining       0 Nov 14 23:14 file.0×9385c5221290.0×9385c7744d10.DataSectionObject.mpr.dll.dat
-rw------- 1 mjolnirtraining mjolnirtraining  127488 Nov 14 23:14 file.0×9385c5946910.0×9385befaaa20.ImageSectionObject.edputil.dll.img
-rw------- 1 mjolnirtraining mjolnirtraining   57856 Nov 14 23:14 file.0×9385c594ecb0.0×9385c59a9050.ImageSectionObject.virtdisk.dll.img
-rw------- 1 mjolnirtraining mjolnirtraining   26112 Nov 14 23:14 file.0×9385c59535d0.0×9385c59a6740.ImageSectionObject.fltLib.dll.img
-rw------- 1 mjolnirtraining mjolnirtraining  462336 Nov 14 23:14 file.0×9385c5f681e0.0×9385c51afa20.ImageSectionObject.OneCoreCommonProxyStub
-rw------- 1 mjolnirtraining mjolnirtraining  583680 Nov 14 23:14 file.0×9385c6613140.0×9385c1127750.ImageSectionObject.AppResolver.dll.img
-rw------- 1 mjolnirtraining mjolnirtraining 3398656 Nov 14 23:14 file.0×9385c732ac80.0×9385c63b98a0.ImageSectionObject.OneCoreUAPCommonProxyS
-rw------- 1 mjolnirtraining mjolnirtraining  206848 Nov 14 23:14 file.0×9385c7db5ae0.0×9385c73e1a20.ImageSectionObject.xGWpgNo.exe.img
-rw------- 1 mjolnirtraining mjolnirtraining  204800 Nov 14 23:14 file.0×9385c7db5ae0.0×9385c77aba10.DataSectionObject.xGWpgNo.exe.dat
-rw-r--r-- 1 mjolnirtraining mjolnirtraining      79 Sep 13  2023 mypy.ini
-rw-r--r-- 1 mjolnirtraining mjolnirtraining    1854 Sep 13  2023 setup.py
drwxr-xr-x 2 mjolnirtraining mjolnirtraining    4096 Sep 13  2023 test
drwxr-xr-x 2 mjolnirtraining mjolnirtraining    4096 Oct  8 06:26 tools
-rwxr-xr-x 1 mjolnirtraining mjolnirtraining     290 Sep 13  2023 vol.py
-rw-r--r-- 1 mjolnirtraining mjolnirtraining    5450 Sep 13  2023 vol.spec
drwxr-xr-x 8 mjolnirtraining mjolnirtraining    4096 Sep 13  2023 volatility3
drwxr-xr-x 2 root            root               4096 Sep 13  2023 volatility3.egg-info
-rwxr-xr-x 1 mjolnirtraining mjolnirtraining     297 Sep 13  2023 volshell.py
-rw-r--r-- 1 mjolnirtraining mjolnirtraining    2963 Sep 13  2023 volshell.spec
-rw------- 1 mjolnirtraining mjolnirtraining  206848 Nov 14 23:15 xGWpgNo.exe

┌──(mjolnirtraining㉿MT23-IR-Kali)-[~/Desktop/volatility3]
└─$
```

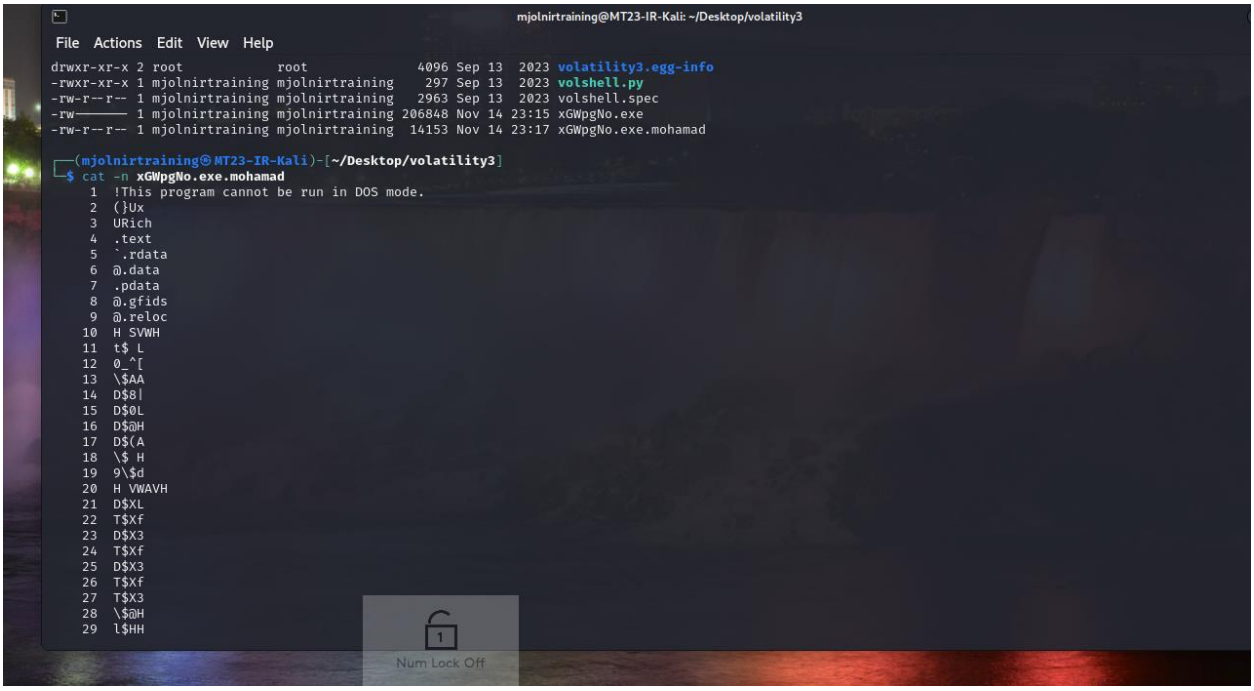
```
┌──(mjolnirtraining㉿MT23-IR-Kali)-[~/Desktop/volatility3]
└─$ ls -l
total 232
-rw-r--r-- 1 mjolnirtraining mjolnirtraining   5947 Sep 13  2023 README.md
drwxr-xr-x 4 root            root              4096 Sep 13  2023 build
drwxr-xr-x 2 mjolnirtraining mjolnirtraining   4096 Oct  8 06:26 config
drwxr-xr-x 3 mjolnirtraining mjolnirtraining   4096 Sep 13  2023 development
drwxr-xr-x 2 root            root              4096 Sep 13  2023 dist
drwxr-xr-x 3 mjolnirtraining mjolnirtraining   4096 Sep 13  2023 doc
drwxr-xr-x 2 mjolnirtraining mjolnirtraining   4096 Oct  8 06:26 docs
-rw-r--r-- 1 mjolnirtraining mjolnirtraining     79 Sep 13  2023 mypy.ini
-rw-r--r-- 1 mjolnirtraining mjolnirtraining   1854 Sep 13  2023 setup.py
drwxr-xr-x 2 mjolnirtraining mjolnirtraining   4096 Sep 13  2023 test
drwxr-xr-x 2 mjolnirtraining mjolnirtraining   4096 Oct  8 06:26 tools
-rwxr-xr-x 1 mjolnirtraining mjolnirtraining    290 Sep 13  2023 vol.py
-rw-r--r-- 1 mjolnirtraining mjolnirtraining   5450 Sep 13  2023 vol.spec
drwxr-xr-x 8 mjolnirtraining mjolnirtraining   4096 Sep 13  2023 volatility3
drwxr-xr-x 2 root            root              4096 Sep 13  2023 volatility3.egg-info
-rwxr-xr-x 1 mjolnirtraining mjolnirtraining    297 Sep 13  2023 volshell.py
-rw-r--r-- 1 mjolnirtraining mjolnirtraining   2963 Sep 13  2023 volshell.spec
-rw------- 1 mjolnirtraining mjolnirtraining 206848 Nov 14 23:15 xGWpgNo.exe
-rw-r--r-- 1 mjolnirtraining mjolnirtraining  14153 Nov 14 23:17 xGWpgNo.exe.mohamad

┌──(mjolnirtraining㉿MT23-IR-Kali)-[~/Desktop/volatility3]
└─$
```

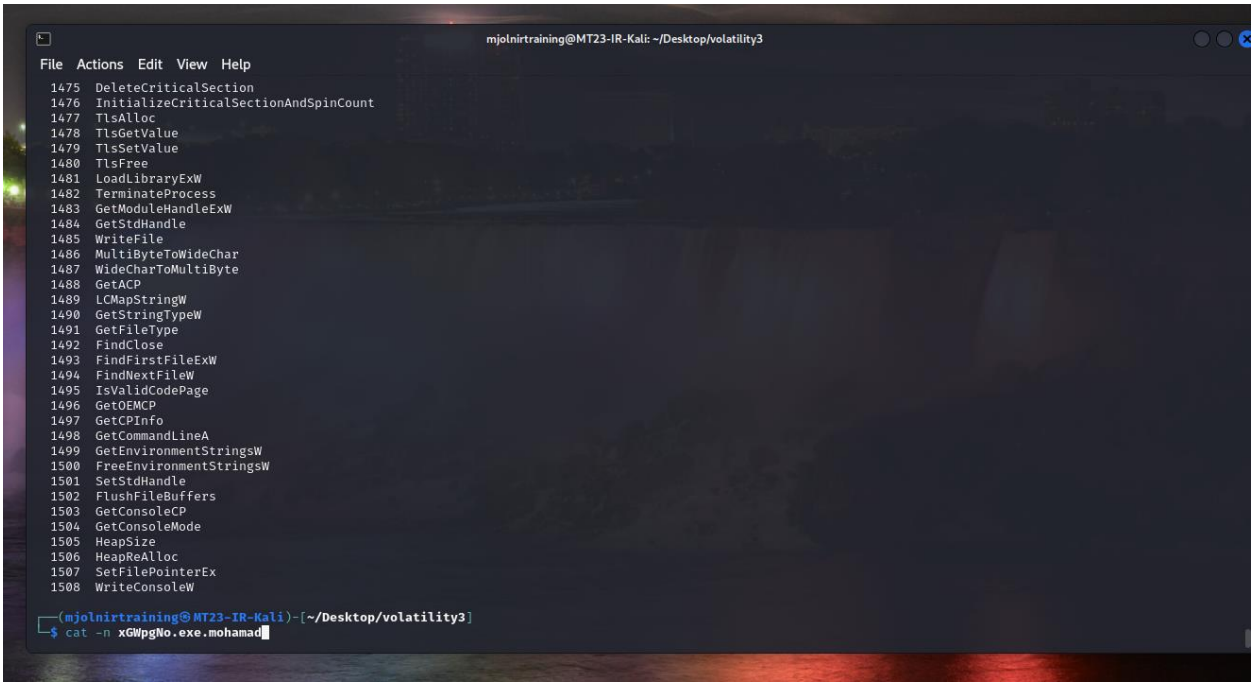
Analyze with Strings Command: To look for readable text within the dumped file, I used the strings command. This helped US identify any suspicious patterns, keywords, or clues within the file.

MOHAMAD ALMASR 101167438

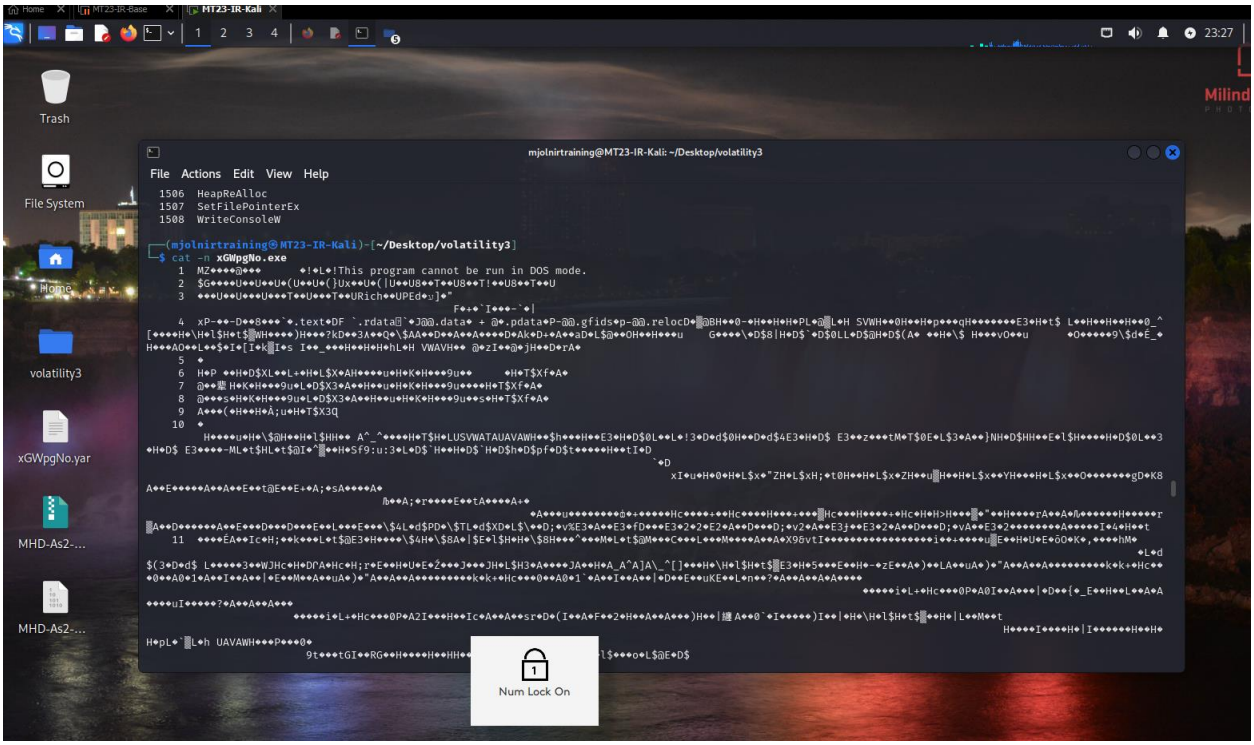
Save Strings Output: I saved the strings output with ny name filename . This way, i could keep track of it and refer back to it later.

Open File with cat Command: Using cat, i viewed the contents of the saved strings output file. This allowed us to review the extracted text from the dumped file easily.



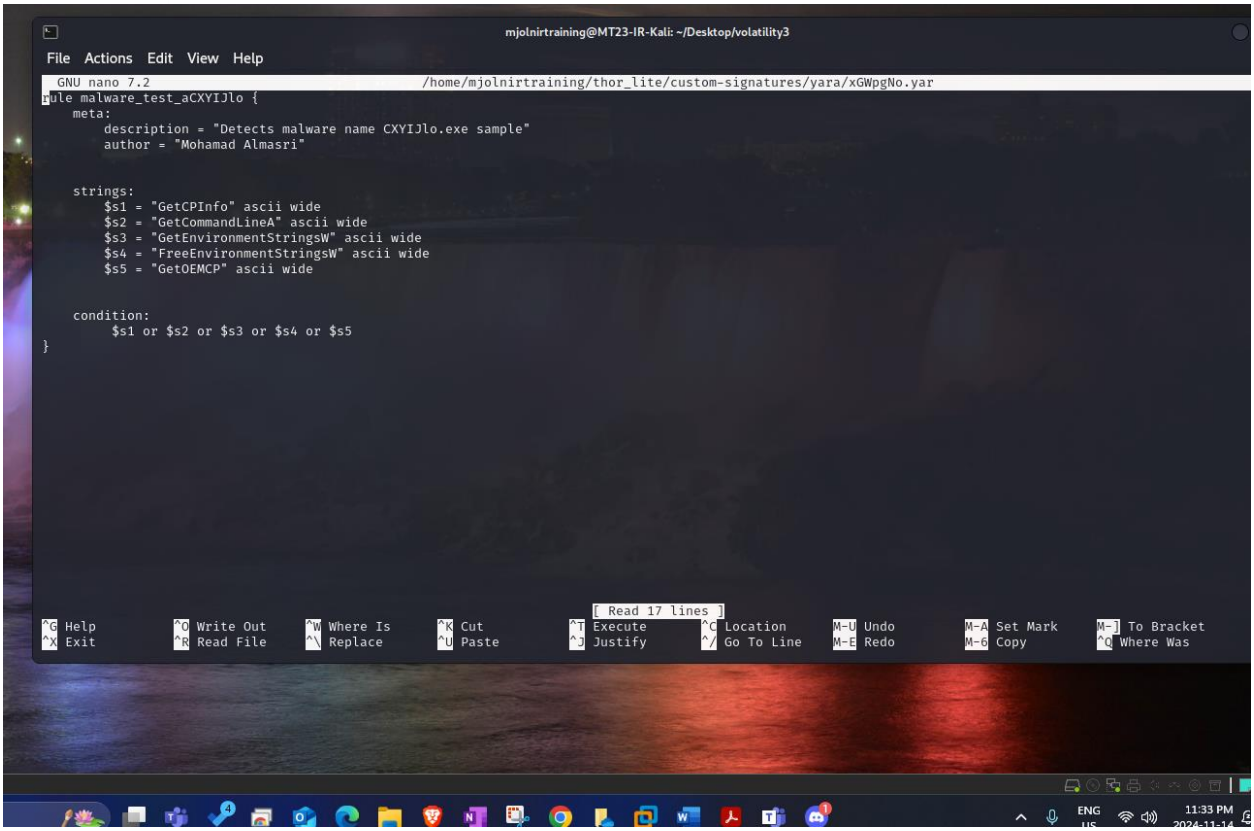


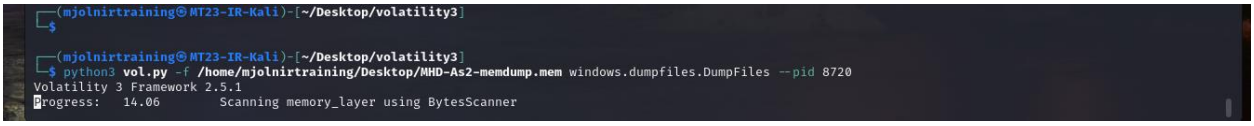
MOHAMAD ALMASR 101167438



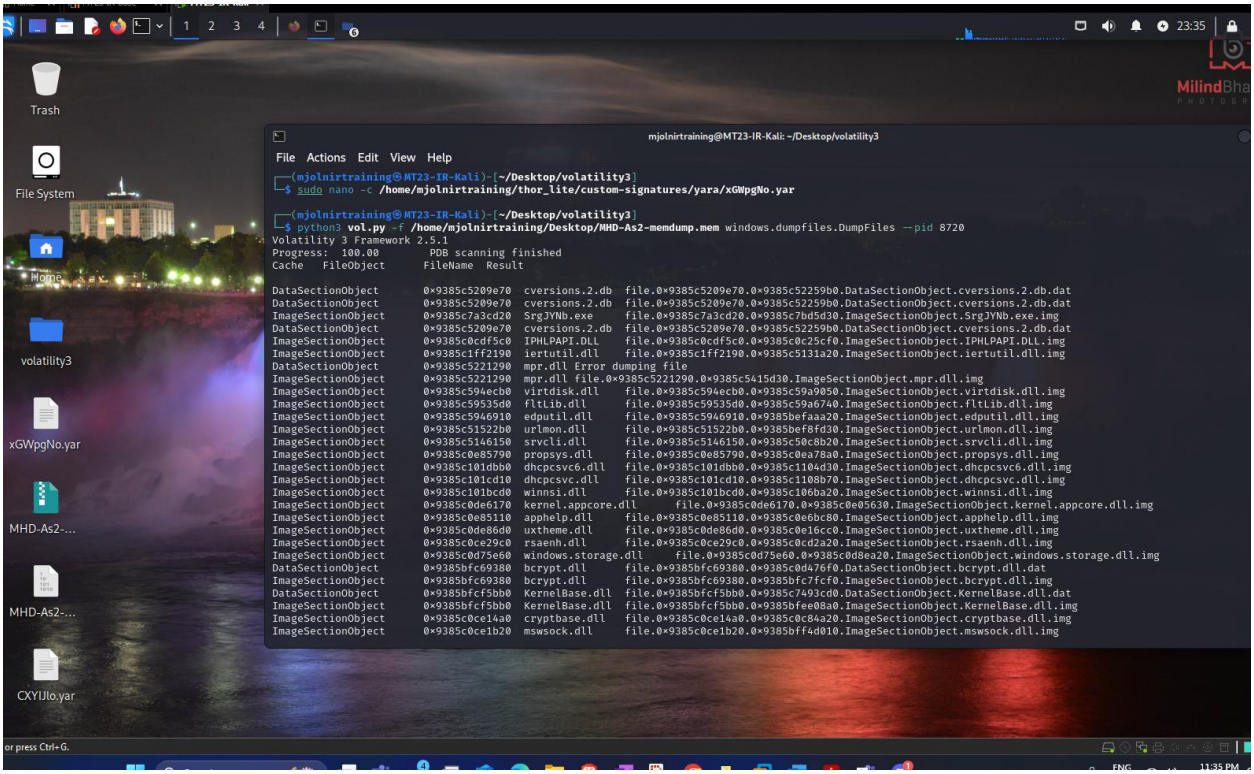
Create YARA Rules: After analyzing the file, I created YARA rules that would help detect patterns or behaviors identified in the memory dump. These rules are tailored to look for specific types of malware.

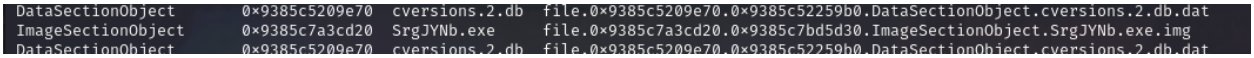
MOHAMAD ALMASR 101167438



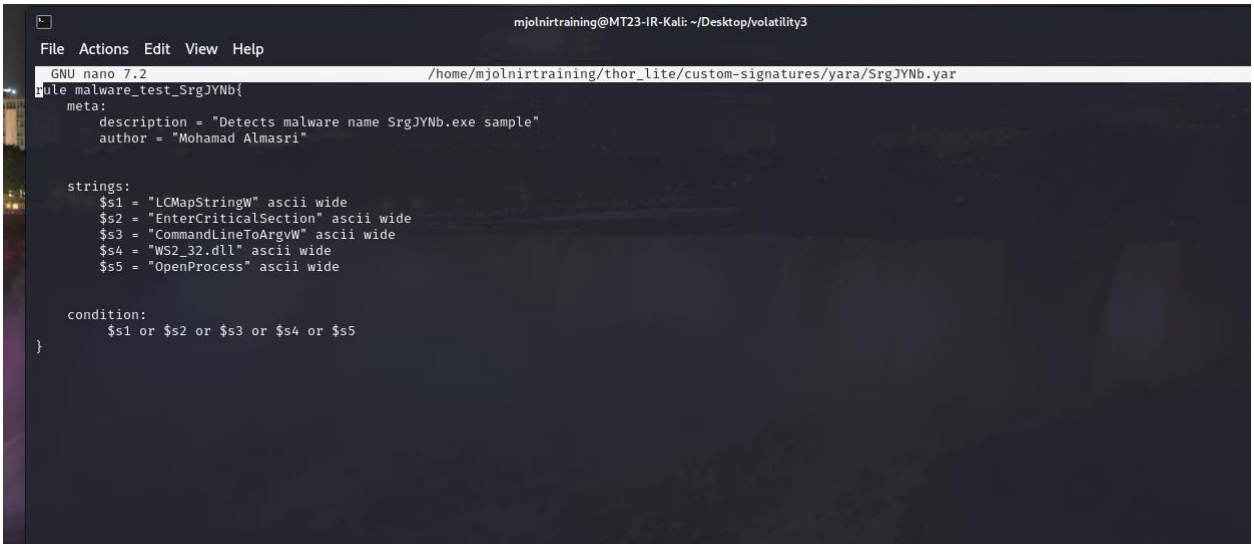


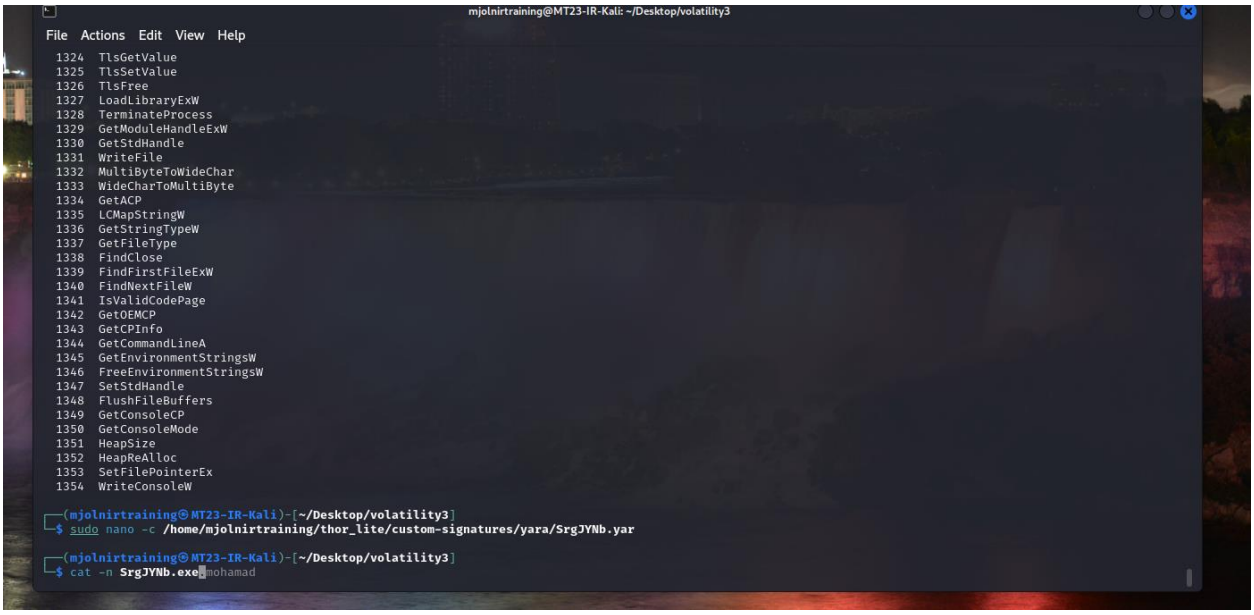
MOHAMAD ALMASR 101167438



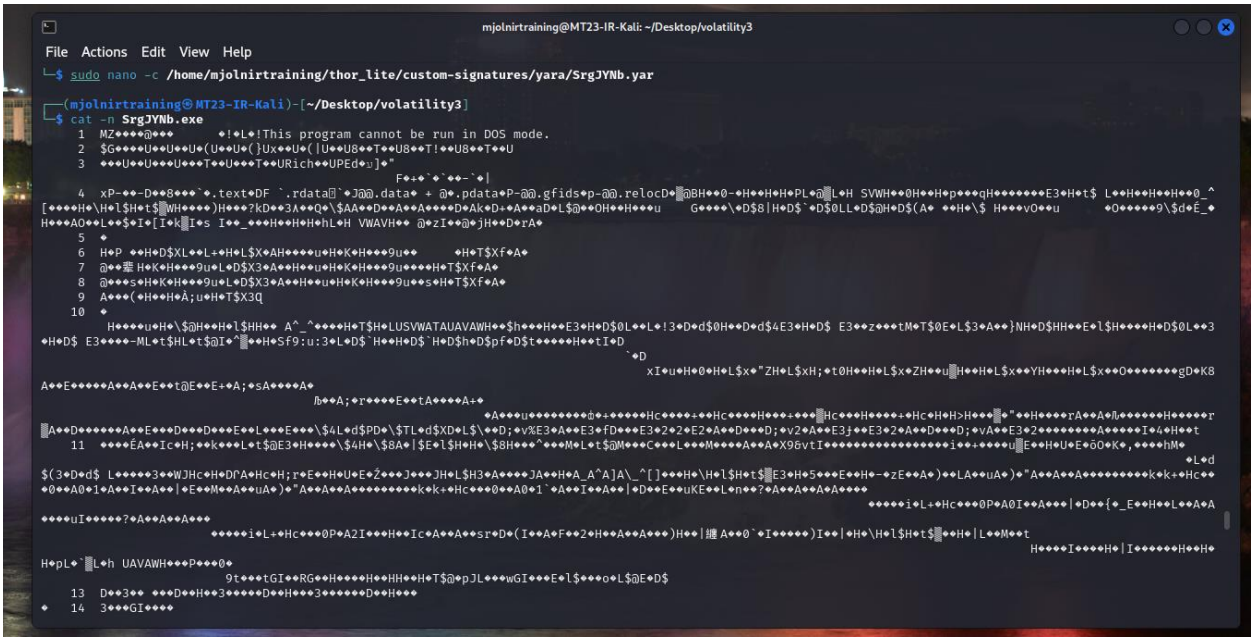


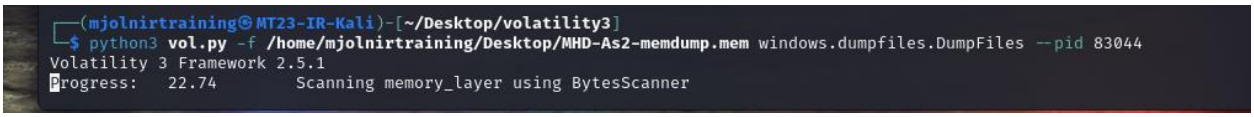
MOHAMAD ALMASR 101167438

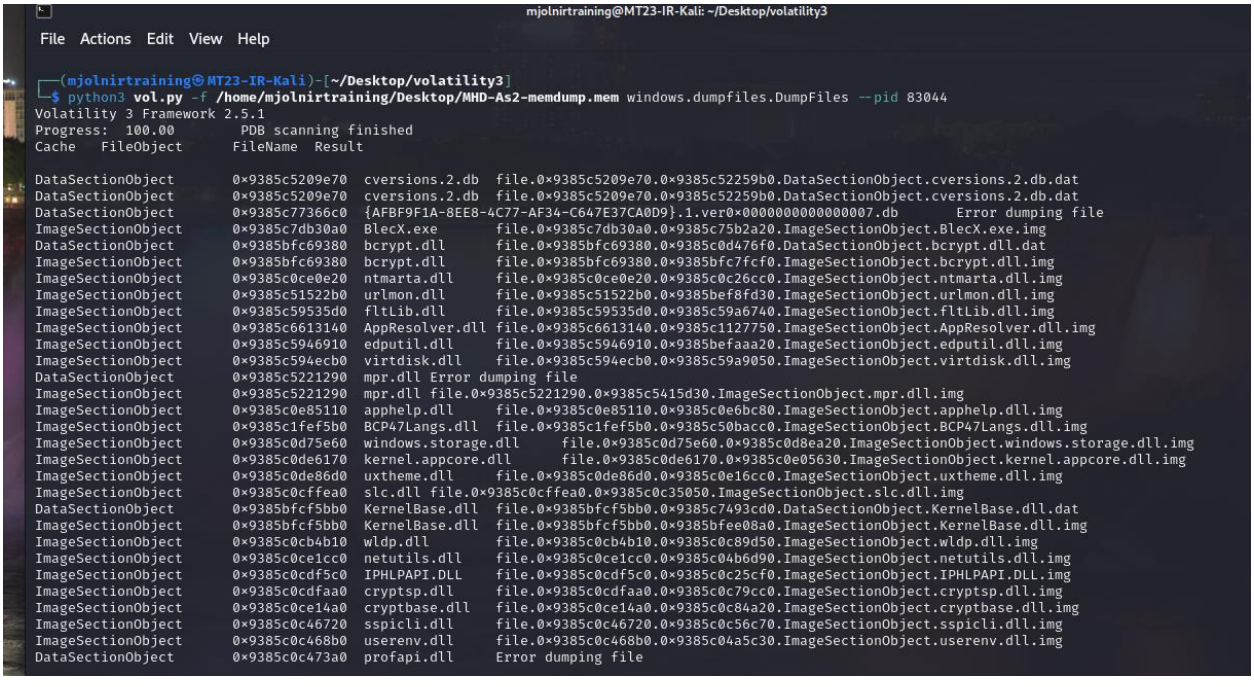




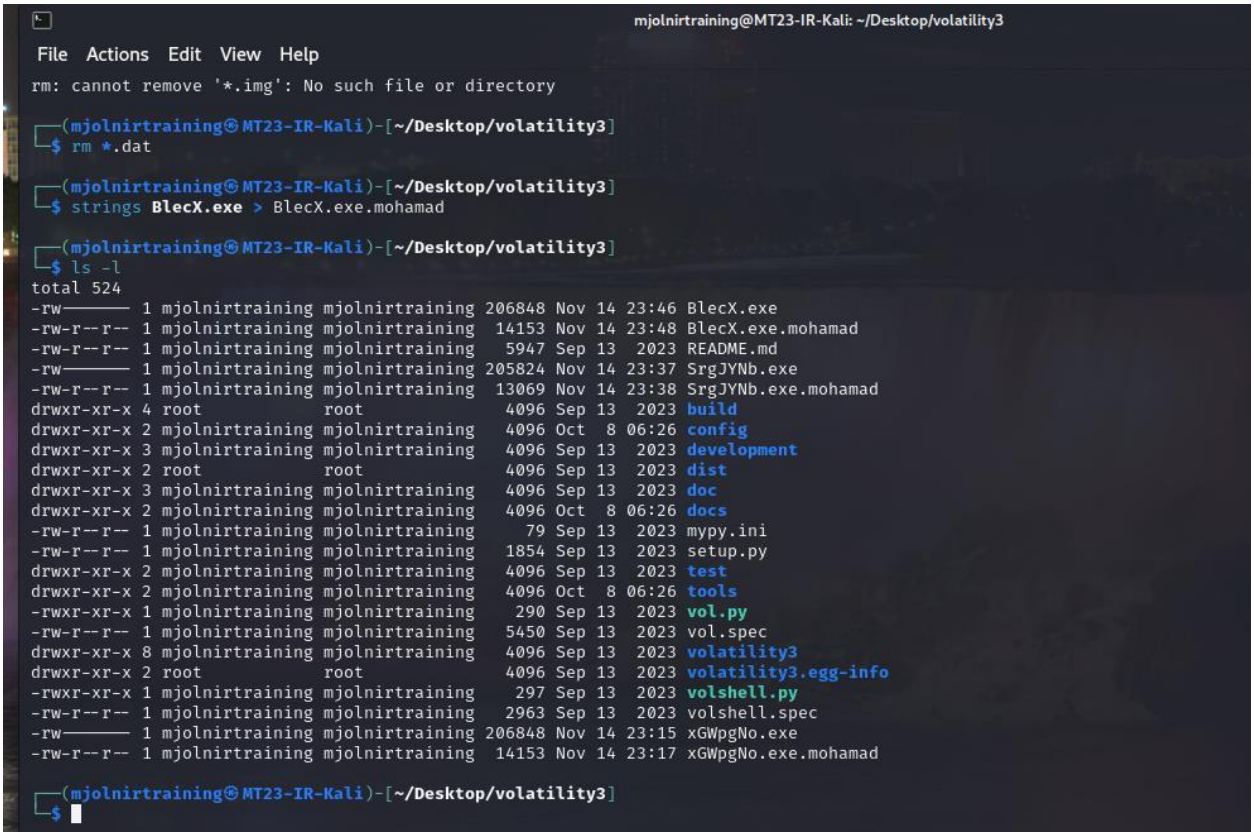
MOHAMAD ALMASR 101167438

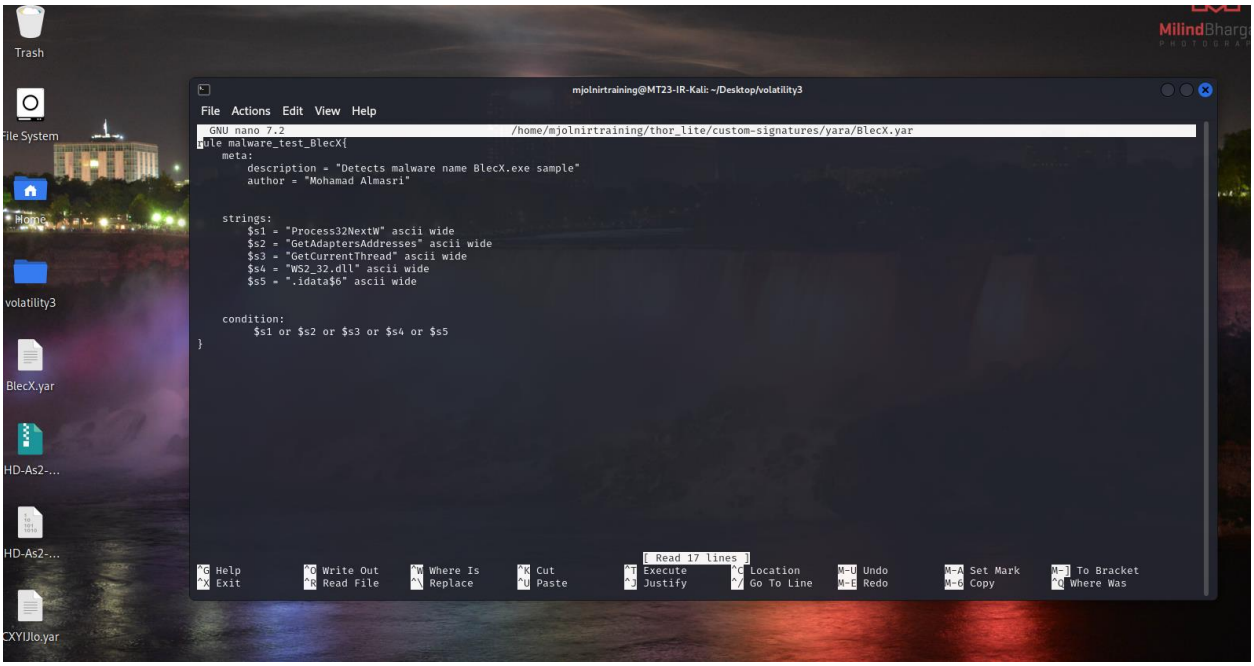


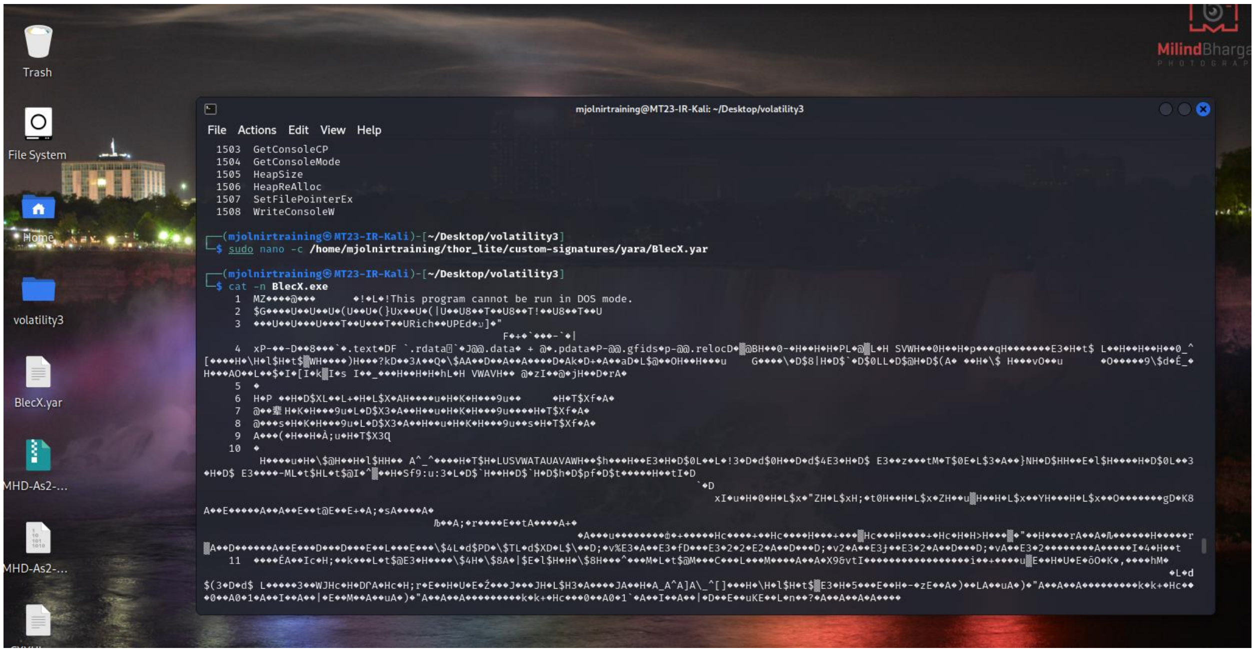


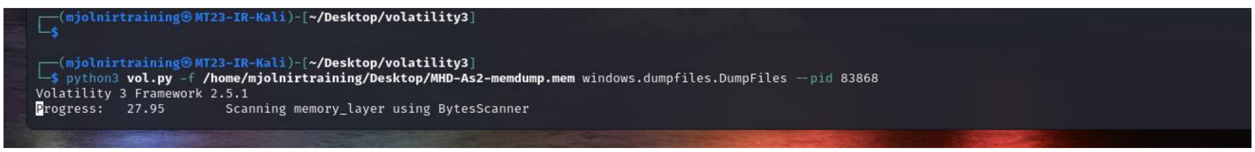


MOHAMAD ALMASR 101167438







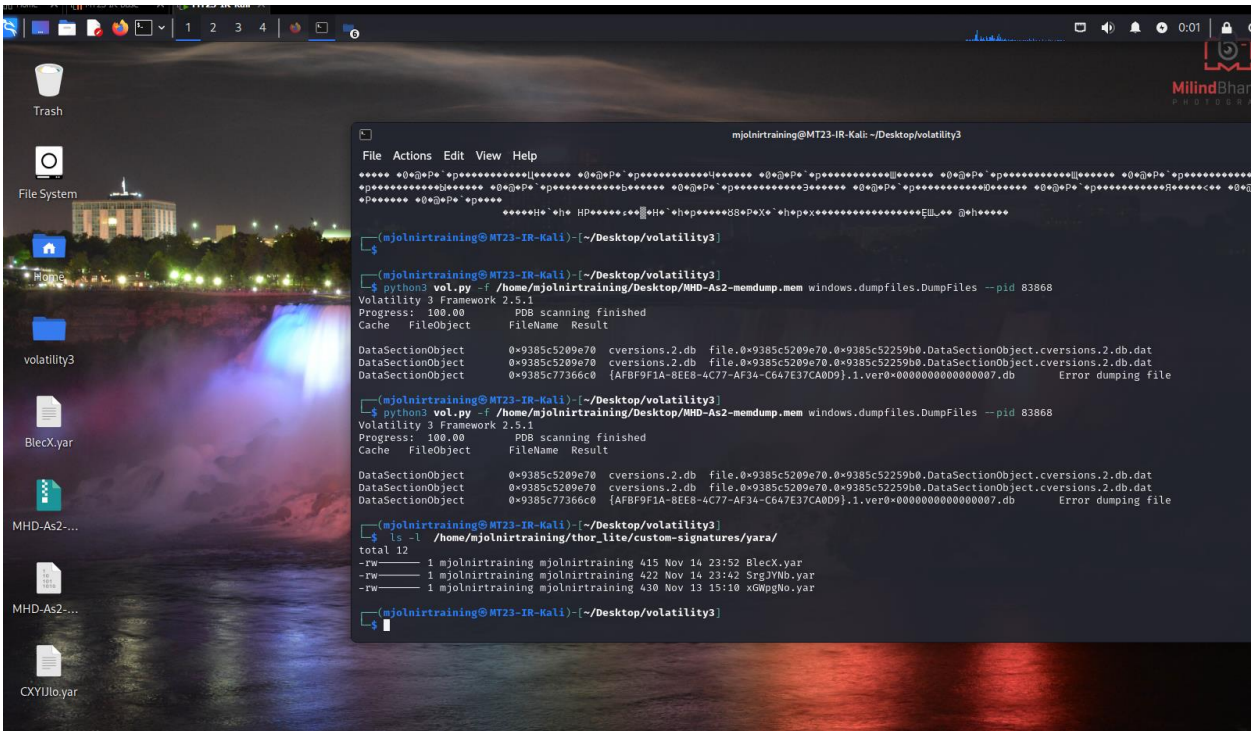


83868 83044 dKzrjCT.exe

For the last piece of malware, I was unable to locate the executable file because it was generated by another malware process. This parent process is clearly shown in the screenshot, along with its process ID (PID). The parent-child relationship between the processes is evident, and both processes share the same PID structure, making it clear that the child process was by the parent.

83044	19280	BlecX.exe	0x9385ce187080	14	-	1	False	2024-11-15 03:31:54.000000	N/A	Disabled
83868	83044	dKzrjCT.exe	0x9385ce277080	236	-	1	False	2024-11-15 03:31:54.000000	N/A	Disabled

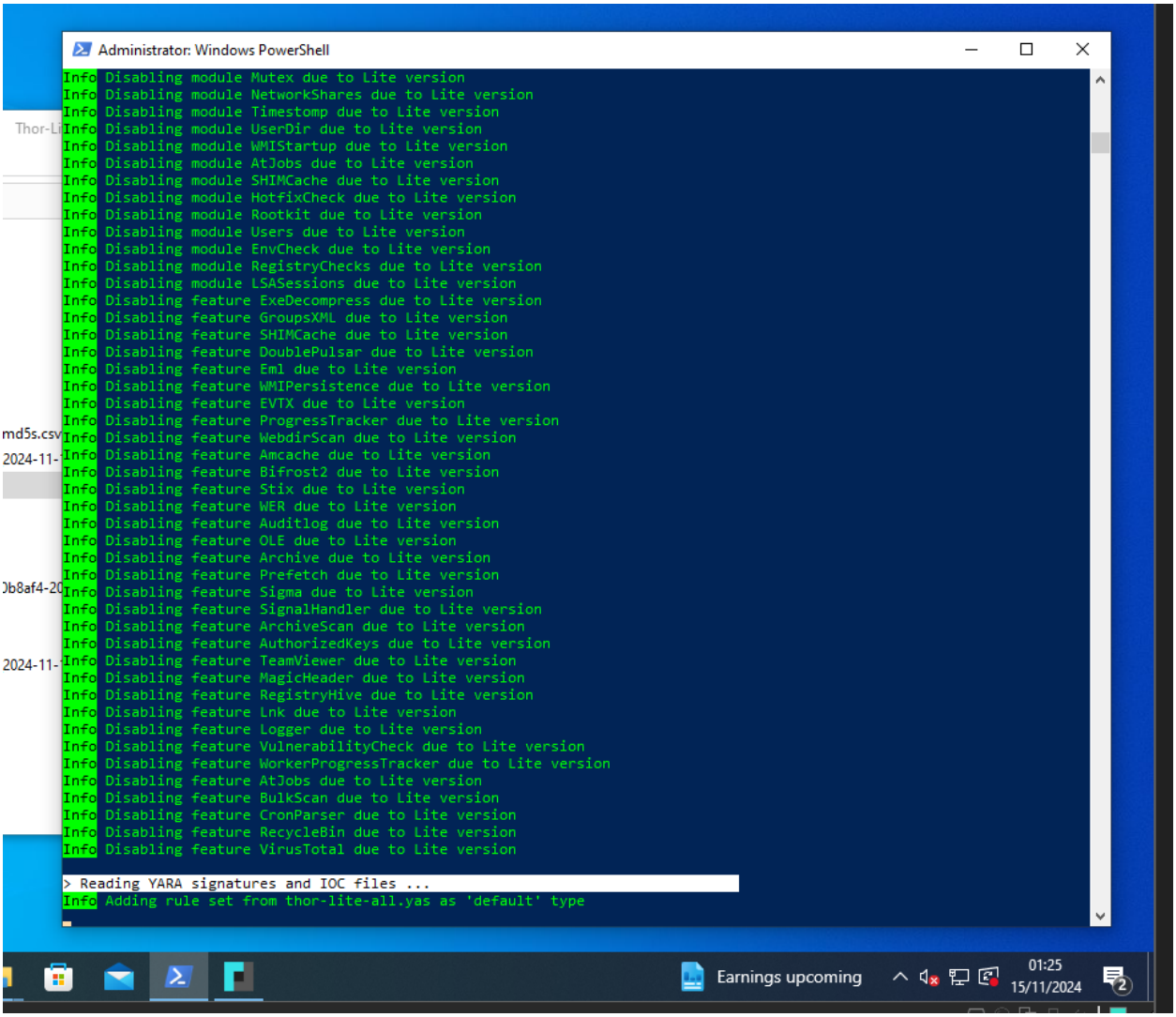
MOHAMAD ALMASR 101167438

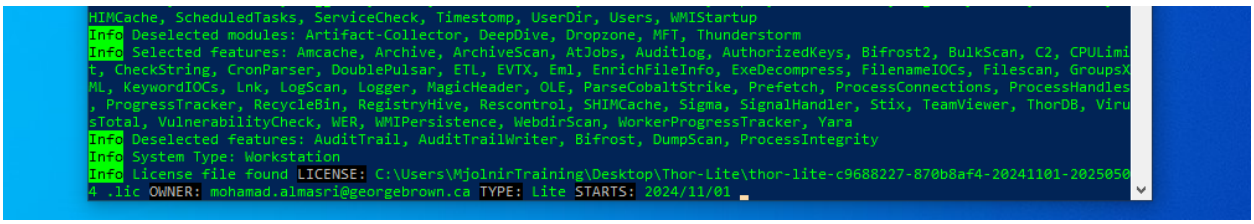


Transferred the created YARA rules to the **THOR Lite** custom signature folder on the Windows VM.

Downloaded and configured THOR Lite for scanning

MOHAMAD ALMASR 101167438

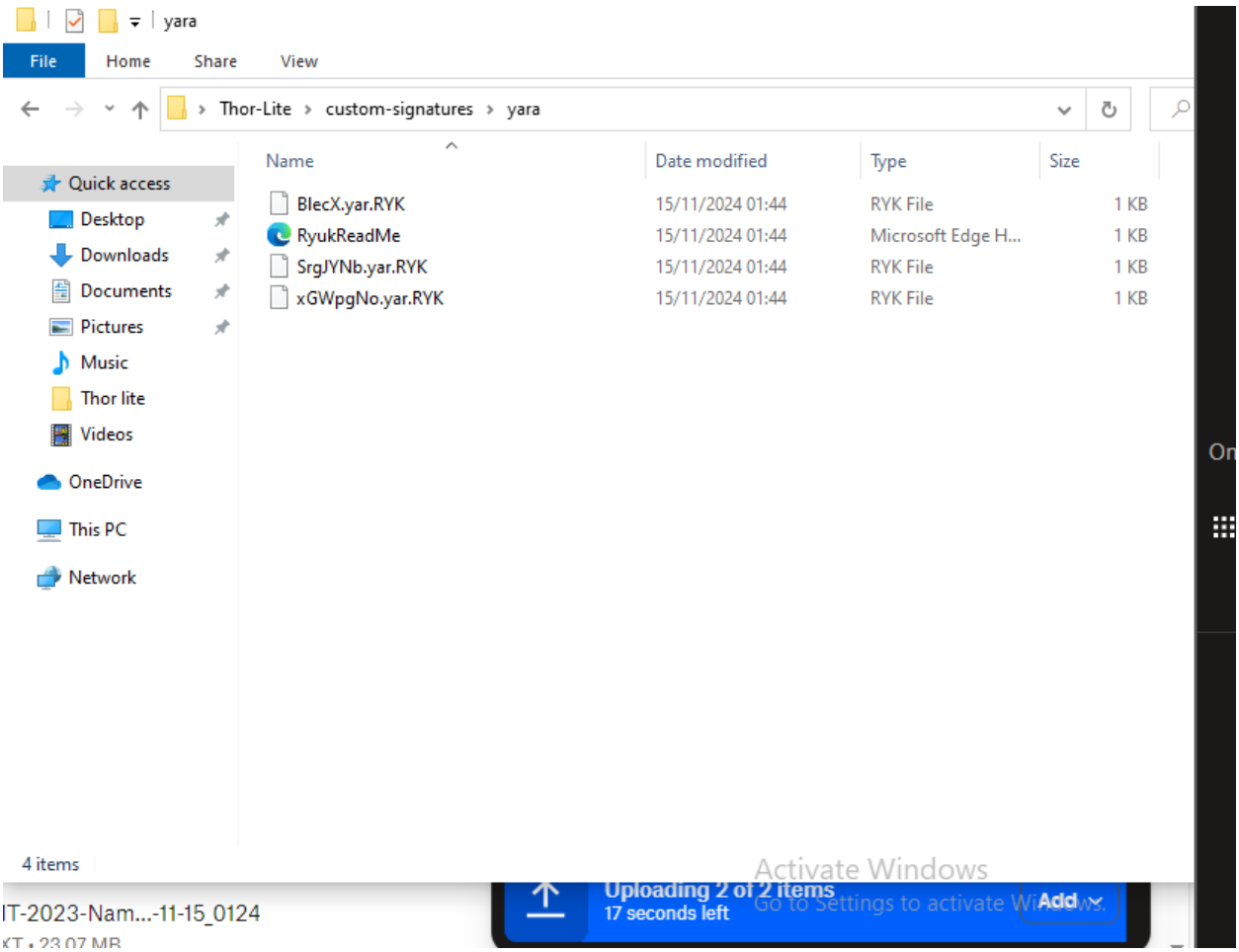




MOHAMAD ALMASR 101167438

```
Info Successfully compiled 0 malware and 0 false positive hashes TYPE: IOC
Info Successfully compiled 0 file type signatures TYPE: IOC
Info Successfully compiled 0 malware domains TYPE: IOC
Info Successfully compiled 0 named pipe ioc strings and 0 named pipe ioc regexs TYPE: IOC
Info Successfully compiled 0 malicious handles and 0 regex malicious handles TYPE: IOC
Warning No file type signatures compiled, file type detection can't be done. Because of this, many files won't be sc
d.
Info No custom directory excludes defined
Info No custom registry excludes defined
Info No custom eventlog excludes defined
Info No custom process excludes defined
Info Successfully compiled 5714 default YARA rules TYPE: YARA
Info Successfully compiled 0 custom default YARA rules TYPE: YARA
Info Successfully compiled 0 meta YARA rules TYPE: YARA
Info Successfully compiled 0 process YARA rules TYPE: YARA
Info Successfully compiled 0 log YARA rules TYPE: YARA
Info Successfully compiled 0 registry YARA rules TYPE:
```

Ran the scanner, but the rules were encrypted during scanning, which resulted in detection failure.



MOHAMAD ALMASR 101167438

What Evidence I Started With

1. Ransomware samples: October 2019 folder containing multiple ransomware samples. This is performed on a Windows virtual machine (VM).

2. Memory dump: A VM's memory dump, which is performed after executing all ransomware samples. It records the state of the system during malware activity

3. Process Monitoring: Process Hacker Observation is a tool used to monitor the processes running during malware execution. It identifies suspicious processes.

4. Screenshots: Screenshots taken during the process, including:

- VM state paused
- Active processes and their relationships.
- Memory analysis output

What Analysis Was Done

1. **Memory Dump Analysis**:
   - Transfer the memory dump file from the Windows VM to the Kali Linux machine for analysis
   - Used Volatility 3 to extract forensic data, focusing on processes and their behaviors.
   - Run windows.info, windows.pslist, windows.procdump and other commands to collect details about active processes. Identify malicious processes and delete the executable file
2. **Process Relationship Examination**:
   - Analyze parent-child relationships between processes using tools such as windows.pstree and windows.cmdline. To follow up on other processes How is it dynamically generated by some malware.
3. **Executable Extraction**:
   - Extracted malicious executables from memory using the cp command.

- For one malware sample, the executable couldn't be directly extracted as it was dynamically generated by a parent process, a behavior noted in the process tree.

4. **String Analysis**:

- Conducted string analysis on dumped executables using the strings command.
- Identified unique markers or patterns to create custom YARA rules for each piece of malware.

5. **YARA Rule Development**:

- Create YARA rules based on findings from string analysis.
- Each rule targets a specific pattern or behavior of the identified malware.

6. **THOR Lite Testing**:

- Loaded the custom YARA rules into the **THOR Lite** scanner and configured it for testing.
- Ran the scanner on the Windows VM to detect malware using the created rules.
- Observed that some YARA rules failed due to encryption issues during runtime.

Did your yara rules work? If no, why?

One possible reason is that the rules were encoded or changed when I copied them. After running all the ransomware files from the "October 2019 folder on the VM, it is possible that one of the ransomware programs encrypted or modified the YARA rules file.

Details on the Encryption Issue

• File extension changes: YARA rules normally have the extension .yar, but after copying Rules now have the extension .RYK. This suggests that the ransomware encrypted or scrambled the file, converting it from a readable YARA rule into an encrypted format that THOR Lite cannot detect.

• **Impact on THOR Lite:** THOR Lite expects YARA rules to be in a specific format (.yar or .yara). When files are encrypted or modified by ransomware, THOR Lite is unable to detect those threats. causing the scan to fail.

Why did this happen?

Perhaps the ransomware was programmed to encrypt files on the system, including my custom YARA rules. Because I ran the ransomware without restoring the VM, it affected all accessible files. Including my rules.

Conclusion

This work gave me valuable hands-on experience in memory forensics and malware analysis. I learned how to take memory from a virtual machine and analyze it using Volatility 3, an important tool in digital forensics. Using commands such as windows.info, windows.pslist, windows.pstree, and windows.cmdline I can identify suspicious processes, find PIDs, and examine relationships. Understanding the parent and child structure of malware processes helps me understand how some malicious processes replicate other processes. Create relevant threat groups... I also performed string analysis on the discarded files to identify specific patterns and behavior for each malware sample. This analysis is critical in shaping YARA regulations to identify similar hazards in the future. Although I faced challenges such as encoding my YARA rules to prevent ransomware from running on THOR Lite, I gained valuable insight into the risks of running malware in this environment. Undamaged environment.