ACS EDU Evaluation Report

Date of Submission: 2025.07.17

<table>
<tr><td rowspan="2">Student</td><td>Name</td><td>[Kong Darong]</td><td rowspan="2">Course Selected</td><td rowspan="2">[Incident Response]</td></tr>
<tr><td>Group No.</td><td>[8]</td></tr>
<tr><td rowspan="2">Lecture Report</td><td>Name</td><td colspan="3">[Lecture Report – Advanced Analysis and Reflection]</td></tr>
<tr><td colspan="4">Memory Forensics and Threat Detection Using Volatility</td></tr>
<tr><td>Report Topic
(Analysis Result)</td><td colspan="4">
Summary of Key Learnings
Having taken the Advanced Cybersecurity Incident Response course as a student, I was able to get the extensive view of the whole incident handling lifecycle including the detection of the incident through recovery stages, which I was taught both theoretically and intensively in the form of lab exercises. This course has added great value by giving us a profound understanding of how attackers operate, what techniques, and tools they use, as well as their behaviors. Additionally, it instilled the technical and analytical background I need to appropriately respond to elaborate threats directed at cybersecurity.
◇ Incident Theory

I developed a good knowledge of the sequential stages of incident response such as the following:

Detection & Analysis
Recovery, Containment and Eradication
Incident Handling and Reporting after Incident
Using a framework like MITRE ATT&CK, I read the attacker trends and came to identify patterns of compromise using the real-life adversarial methods. This enabled me to implement threat intelligence practically in the work of investigation.
◇ Memory & Artifact Analysis

A major focus of the course was the collection and analysis of both volatile and non-volatile data sources. I conducted forensic analysis on a variety of system artifacts, including:
</td></tr>
</table>

- Processes and hidden process trees
- Loaded DLLs and injected code
- Command-line execution traces
- Registry keys, scheduled tasks, and AutoStart entries
- Active network connections
- I used the Volatility Framework (versions 2 and 3) to perform memory forensics and identify malicious behaviors, persistence mechanisms, and evidence of compromise.
- Key skills developed:
- Detecting signs of code injection
- Identifying process hollowing and parent spoofing
- Extracting and decoding Base64-encoded payloads
- Correlating artifacts to reconstruct attacker activity

◇ PowerShell for IR

- Practiced writing scripts to automate evidence collection and detect malicious activity.
- Gained hands-on experience with PowerShell scripting for both live response and forensic automation.
- Wrote scripts to collect volatile data, detect suspicious accounts or tasks, and extract relevant indicators of compromise (IOCs).
- Learned how attackers misuse PowerShell (fileless malware or LOLBins) and how to detect such activities.

◇ Variety of Labs

- Ransomware memory labs and malware labs
- Cross-team incident handling Joint incident handling
- Attack chains and correlating time lines Investigation
- Malicious command execution identification and effects
- These labs confirmed my skills in practicing theory in the real world and allowed building confidence in carrying out independent research.

◇ Active Directory (AD)

- Configured an AD environment and understood its vulnerabilities.
- Detected and analyzed AD-related incidents using PowerShell.
- Participated in labs involving AD abuse (privilege escalation, enumeration).

◇ Timeline Creation

- Built attack timelines using logs, memory dumps, and registry analysis to recreate an attacker's sequence of actions.
- Used correlation skills to connect multiple indicators across tools and data sources (Volatility + Event Logs + PowerShell output).

- Technical & Personal Reflections

In this course, I have acquired technical expertise as well as personal understanding that is needed in the field of incident response and digital forensics in the real world. The combination of structured learning and practice labs enabled me to learn how to cope with difficult tasks and to feel more confident in the process of applying my knowledge to practice.

Technical Knowledge

- I also got practical skills on how to use Volatility to analyze memory and was able to explore the live malware activity and the mechanism of how the processes work in volatile memory.
- I learned how to apply the use of PowerShell scripting in automating of forensic operations especially in live response and evidence collection purposes to improve accuracy and efficiency.
- I also went through an exercise in creating incident timelines off of system and memory artifacts and in tracing back attacker behavior and the chain of compromise.
- The lab activities related to memory dumps and actual ransomware excised with the help of which I was able to start recognizing and correlating forensic artifacts, including the process trees, log entries, and registry modifications, to recognize indicators of compromise.
- I enhanced my skill of employing forensic science to identify persistence, malware injections, and utilization of LOLBins to improve my threat-hunting skills.

Personal Development

- Now I am sure in how to deal with an incident and detect it to complete resolution with the collection, analysis, and reporting of evidence.
- The labs focused on the documentation and the ability of cross-team collaboration with a specific emphasis on high-stress situations and the necessity to pay attention to guarantees of accuracy and communication.
- It was a bit difficult and a very fulfilling experience to analyze real memory dumps since it played a major role in honing my investigative thought-processes and problem-solving capabilities.
- The course has equipped me to take up the role of a Security Operations Center (SOC) analyst or an Incident Response Team (IRT) member in which I can actively make contribution in the detection as well as response to cyber threat.

- References
    a. MITRE ATT&CK Framework - https://attack.mitre.org/
    b. Volatility Framework - https://www.volatilityfoundation.org/
    c. PowerShell IR Guide - https://learn.microsoft.com/en-us/powershell/
    d. CIS Controls - https://www.cisecurity.org/controls/
    e. Windows Forensics Book
    f. ACS EDU Lecture Practice Materia

- Screenshots

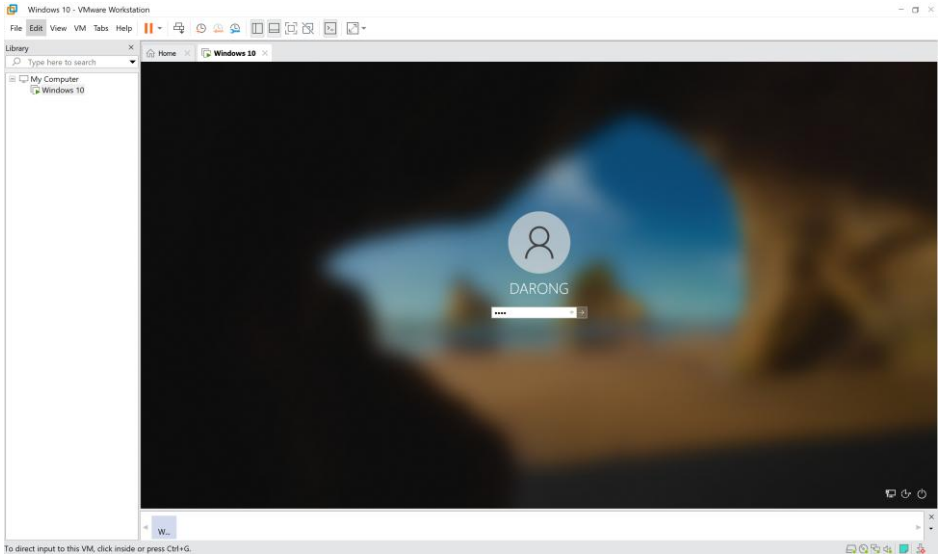


Figure 1: Window10 Installation

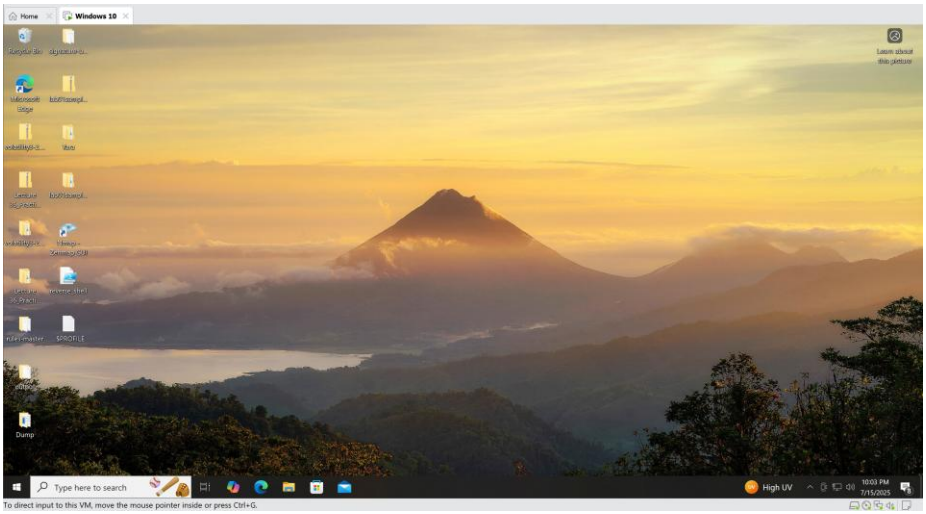


Figure 2: Window10

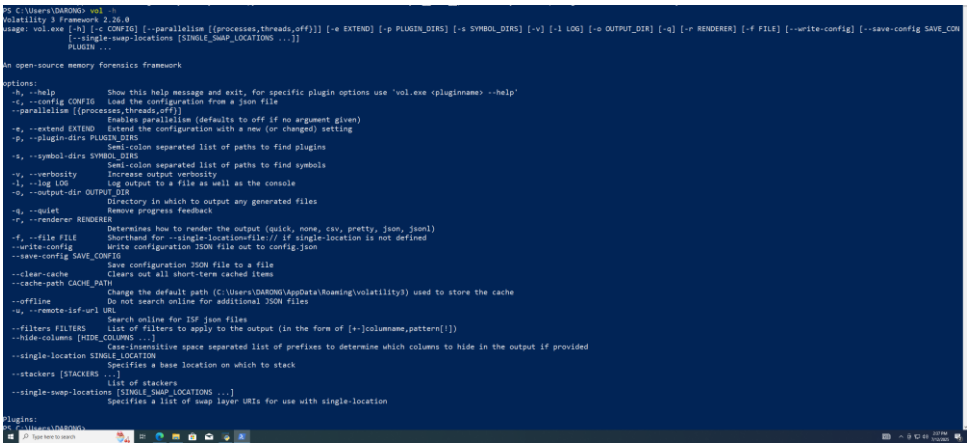


Figure 3: Volalitity2 Tool on Window10

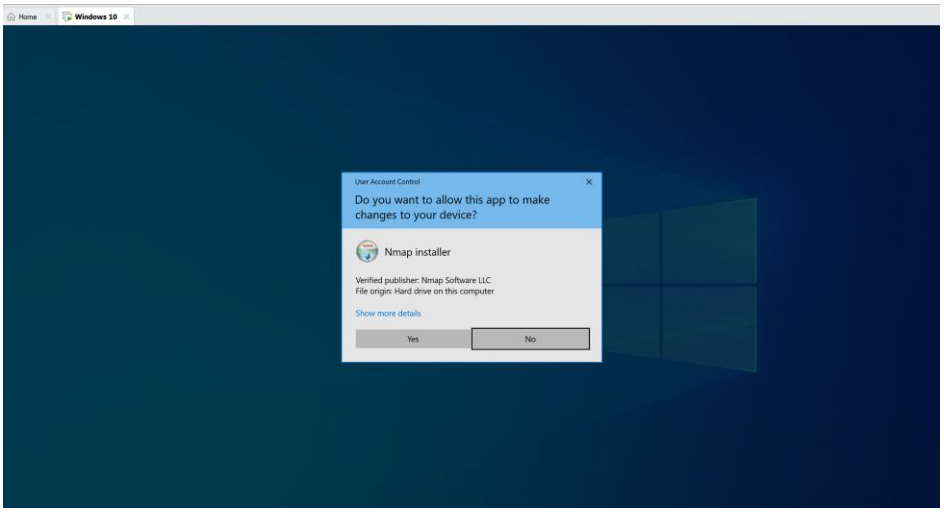


Figure 4: Nmap Scaning

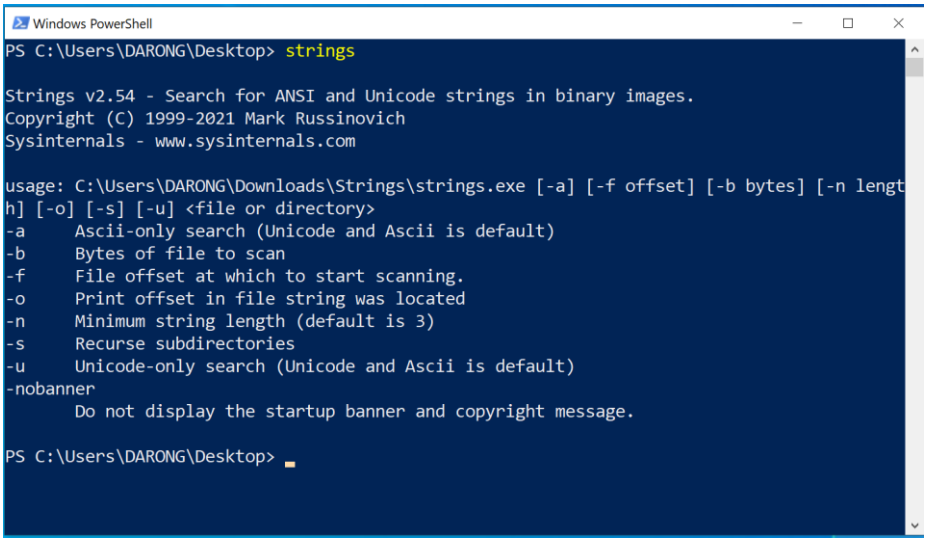


Figure 5: String Tools

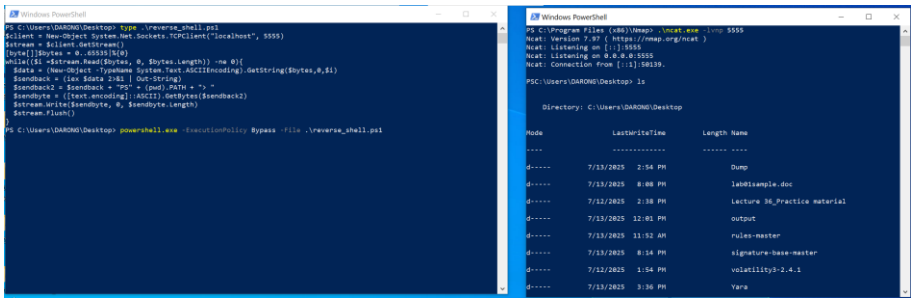


Figure 6: PowerShell Automation Script

Lab	• Summary of Key Learnings Lab01: Windows Memory Forensics & Process Investigation Using Volatility This lab I analyzed a windows 7 memory dump (win7.vmem) using Volatility 2. I became conversant with: • Determining system data and selecting the right Volatility profile • Listing and examination of running, concealed or orphaned processes • Probing of suspicious behavior with the help of process tree analysis • Parsing the command line arguments, and process memory • Noticing the evidence of an attacker like: o LOLBins (regsvr32.exe) o Scripting (timeout.exe, cmd.exe) o Persisting or payload staging

Step-by-Step Forensic Workflow

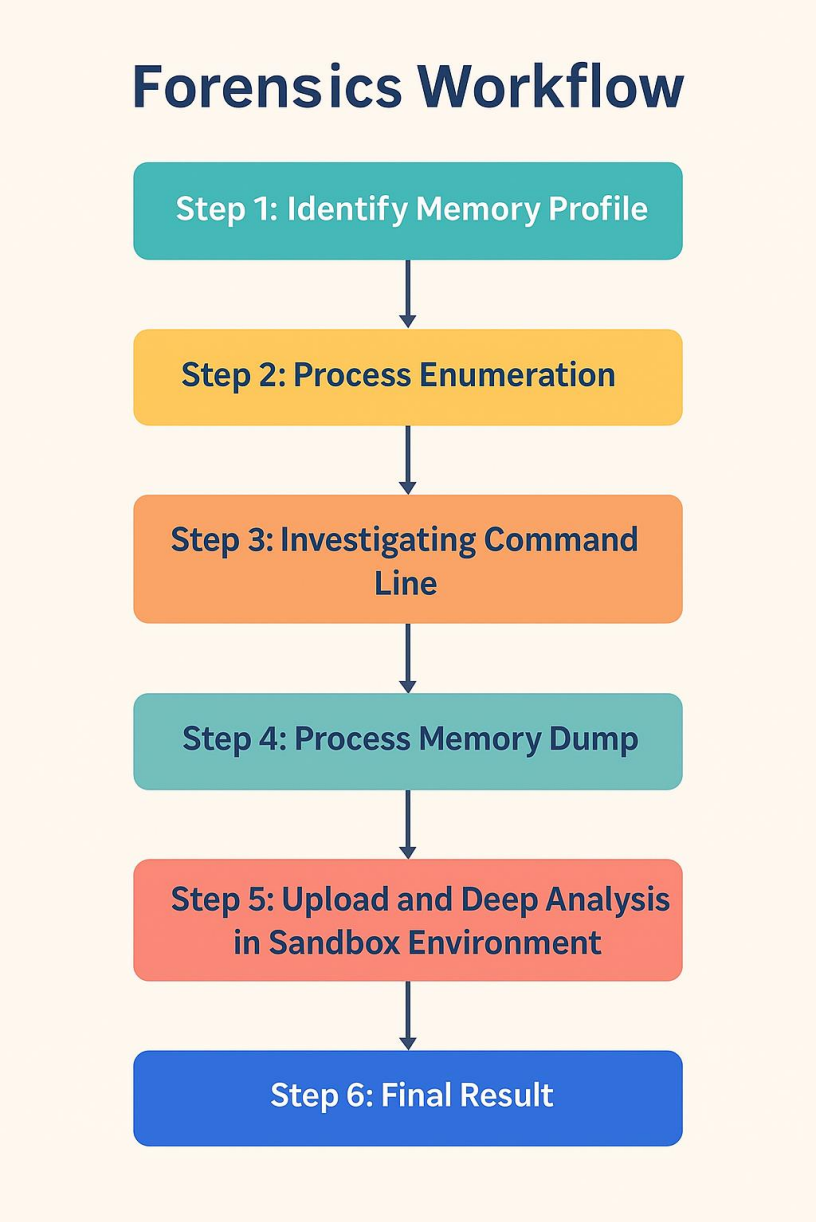


Figure 7: Work Flow

To begin the lab, I selected one memory file for analysis. I started with win7.vmem

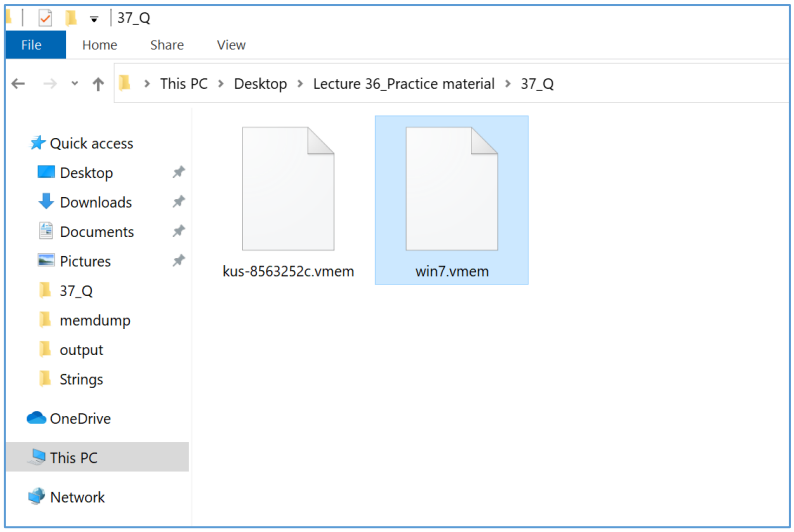


Figure 8: lab File path

- Now I will use the forensic tool Volatility 2 for my exercise. First, I tried to gather information about the memory file. Please refer to my screenshot below

☑ Step 1: Identify Memory Profile

Use this command to get OS information:

```
Vol2 -f win7.vmem imageinfo
```

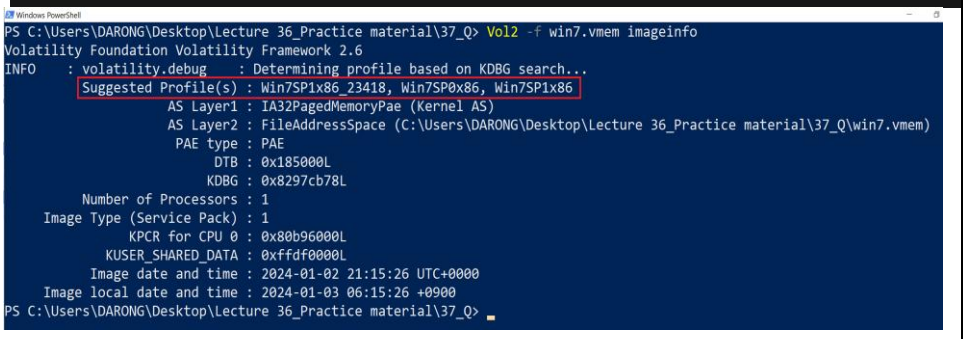


Figure 9: ImageInfo

Field	Details
Suggested Profile(s)	Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
OS, Version	Windows, 7
PAE Type	Physical Address Extension enabled
Image Date/Time (UTC)	2024-01-02 21:15:26 UTC
Local Time (KST)	2024-01-03 06:15:26 +0900
DTB (Directory Table Base)	0x185000
KDBG (Kernel Debug Block)	0x8297cb78
Number of CPUs	1

KPCR (Kernel Processor Control Region)	0x80b96000
KUSER_SHARED_DATA	0xffdf0000

Table 1: Info field

Step 2: Process Enumeration

Based on the information retrieved from the imageinfo plugin, I proceeded to the next step using the suggested profile Win7SP1x86_23418. My goal was to enumerate all running processes in memory and identify any hidden, orphaned, or suspicious processes using various Volatility plugins.

To achieve this, I used the following commands:

- Pslist-Displays active processes based on the EPROCESS list (standard process view).
- Pstree-Visualizes the process hierarchy, helping identify parent-child relationships.
- Psxview-Performs cross-checks using multiple methods to detect hidden or unlinked processes.

Commands Used:

Vol2 -f win7.vmem --profile=Win7SP1x86_23418 pslist

Vol2 -f win7.vmem --profile=Win7SP1x86_23418 pstree

Vol2 -f win7.vmem --profile=Win7SP1x86_23418 psxview

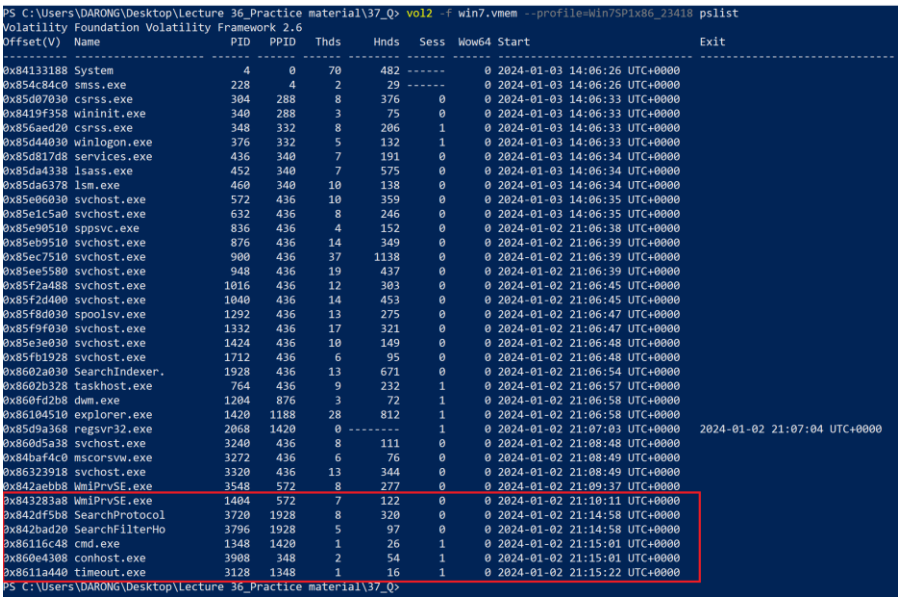


Figure 10: Pslist

During my analysis of running processes using pslist, pstree, and psxview, I observed several processes that, while typically legitimate, may have been involved in suspicious activity:

- WmiPrvSE.exe (PID 1404) is a legitimate Windows process used for WMI operations. However, it is commonly abused by attackers for stealthy execution or persistence. Therefore, I noted it for further inspection to verify that it was not being misused in this case.
- SearchFilterHost.exe (PID 3796) is part of the Windows Search service and is usually spawned by SearchIndexer.exe. In this case, it launched shortly before suspicious activity involving cmd.exe, suggesting the possibility of attacker manipulation, such as process hollowing. I plan to investigate its loaded modules and memory content to confirm its integrity.
- SearchProtocolHost.exe (PID 3720) also launched at the same time as SearchFilterHost.exe and could potentially be involved in the same suspicious behavior. It's another component of the Windows Search framework, but due to the timing, I will analyze it further for possible DLL injection or hijacking.
- cmd.exe (PID 1348) was executed under unusual circumstances, with its parent process being explorer.exe (PID 1420). This relationship could be legitimate, but in combination with other indicators, it warrants closer scrutiny.
- timeout.exe (PID 3128) was launched 21 seconds after cmd.exe. This pattern suggests that the attacker may have used it to delay execution commonly seen in malware staging or script-based payloads. The presence of timeout.exe in this context strongly implies that a second-stage payload might have been deployed. I will perform memory dumping and analysis on timeout.exe to identify its contents and confirm whether it was part of a malicious sequence.

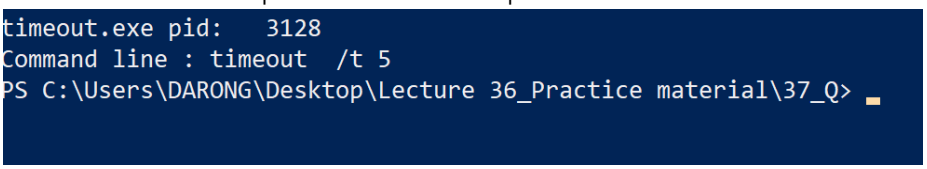


Figure 11: Timeout.exe

After analyzing the processes, I confirmed that the behavior is consistent with attacker activity. timeout.exe **(PID 3128)** was executed by cmd.exe **(PID 1348)** using the command timeout /t 5, indicating a scripted delay before the next stage of execution. This, combined with the suspicious launch of cmd.exe and possible WMI-based execution, supports evidence of malicious activity in the memory image.

➢ Now I will move on to analyze the full process history used by the attacker. Please refer to the process tree below.

```
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q> Vol2 -f win7.vmem --profile=Win7SP1x86_23418 pstree
Volatility Foundation Volatility Framework 2.6
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0x84133188:System                                      4      0     70    482 2024-01-03 14:06:26 UTC+0000
. 0x854c84c0:smss.exe                                 228      4      2     29 2024-01-03 14:06:26 UTC+0000
 0x85d07030:csrss.exe                                 304    288      8    376 2024-01-03 14:06:33 UTC+0000
 0x8419f358:wininit.exe                               340    288      3     75 2024-01-03 14:06:33 UTC+0000
. 0x85d817d8:services.exe                             436    340      7    191 2024-01-03 14:06:34 UTC+0000
.. 0x8602a030:SearchIndexer.                         1928    436     13    671 2024-01-02 21:06:54 UTC+0000
... 0x842df5b8:SearchProtocol                        3720   1928      8    320 2024-01-02 21:14:58 UTC+0000
... 0x842bad20:SearchFilterHo                        3796   1928      5     97 2024-01-02 21:14:58 UTC+0000
.. 0x85f8d030:spoolsv.exe                            1292    436     13    275 2024-01-02 21:06:47 UTC+0000
.. 0x85f2d400:svchost.exe                            1040    436     14    453 2024-01-02 21:06:45 UTC+0000
.. 0x85e90510:sppsvc.exe                              836    436      4    152 2024-01-02 21:06:38 UTC+0000
.. 0x85ec7510:svchost.exe                             900    436     37   1138 2024-01-02 21:06:39 UTC+0000
.. 0x860d5a38:svchost.exe                            3240    436      8    111 2024-01-02 21:08:48 UTC+0000
.. 0x85fb1928:svchost.exe                            1712    436      6     95 2024-01-02 21:06:48 UTC+0000
.. 0x85f9f030:svchost.exe                            1332    436     17    321 2024-01-02 21:06:47 UTC+0000
.. 0x85ee5580:svchost.exe                             948    436     19    437 2024-01-02 21:06:39 UTC+0000
.. 0x85e06030:svchost.exe                             572    436     10    359 2024-01-03 14:06:35 UTC+0000
... 0x842aebb8:WmiPrvSE.exe                          3548    572      8    277 2024-01-02 21:09:37 UTC+0000
... 0x843283a8:WmiPrvSE.exe                          1404    572      7    122 2024-01-02 21:10:11 UTC+0000
.. 0x84baf4c0:mscorsvw.exe                           3272    436      6     76 2024-01-02 21:08:49 UTC+0000
.. 0x85f2a488:svchost.exe                            1016    436     12    303 2024-01-02 21:06:45 UTC+0000
.. 0x85e3e030:svchost.exe                            1424    436     10    149 2024-01-02 21:06:48 UTC+0000
.. 0x85eb9510:svchost.exe                             876    436     14    349 2024-01-02 21:06:39 UTC+0000
... 0x860fd2b8:dwm.exe                               1204    876      3     72 2024-01-02 21:06:58 UTC+0000
.. 0x85e1c5a0:svchost.exe                             632    436      8    246 2024-01-03 14:06:35 UTC+0000
.. 0x8602b328:taskhost.exe                            764    436      9    232 2024-01-02 21:06:57 UTC+0000
.. 0x86323918:svchost.exe                            3320    436     13    344 2024-01-02 21:08:49 UTC+0000
. 0x85da4338:lsass.exe                                452    340      7    575 2024-01-03 14:06:34 UTC+0000
. 0x85da6378:lsm.exe                                  460    340     10    138 2024-01-03 14:06:34 UTC+0000
 0x86104510:explorer.exe                             1420   1188     28    812 2024-01-02 21:06:58 UTC+0000
. 0x85d9a368:regsvr32.exe                            2068   1420      0 ------ 2024-01-02 21:07:03 UTC+0000
. 0x86116c48:cmd.exe                                 1348   1420      1     26 2024-01-02 21:15:01 UTC+0000
.. 0x8611a440:timeout.exe                            3128   1348      1     16 2024-01-02 21:15:22 UTC+0000
 0x85d44030:winlogon.exe                              376    332      5    132 2024-01-03 14:06:33 UTC+0000
 0x856aed20:csrss.exe                                 348    332      8    206 2024-01-03 14:06:33 UTC+0000
. 0x860e4308:conhost.exe                             3908    348      2     54 2024-01-02 21:15:01 UTC+0000
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q>
```

Figure 12: pstree result

Suspicious Chain Identified:

explorer.exe (PID 1420)

├── regsvr32.exe (PID 2068) – Likely LOLBin abuse

└── cmd.exe (PID 1348) – Launched test.bat

    └── timeout.exe (PID 3128) – Script delay (5s)

Memory analysis reveals a suspicious process chain starting from explorer.exe. The attacker likely used regsvr32.exe (a known LOLBin) for fileless execution, followed by a delayed script via cmd.exe and timeout.exe. This pattern is consistent with staged or persistent malware behavior.

- Let me quickly move on to check for hidden processes using the psxview plugin.

```
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q> vol2 -f win7.vmem --profile=Win7SP1x86_23418 psxview
Volatility Foundation Volatility Framework 2.6
Offset(P)  Name                    PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- -------------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x3e12d400 svchost.exe            1040 True   False  True     True   True  True    True
0x3e0e5580 svchost.exe             948 True   False  True     True   True  True    True
0x3e12a488 svchost.exe            1016 True   False  True     True   True  True    True
0x3e0b9510 svchost.exe             876 True   False  True     True   True  True    True
0x3e090510 sppsvc.exe              836 True   False  True     True   True  True    True
0x3ff5f358 wininit.exe             340 True   False  True     True   True  True    True
0x3dd23918 svchost.exe            3320 True   False  True     True   True  True    True
0x3df04510 explorer.exe           1420 True   False  True     True   True  True    True
0x3e01c5a0 svchost.exe             632 True   False  True     True   True  True    True
0x3e18d030 spoolsv.exe            1292 True   False  True     True   True  True    True
0x3e006030 svchost.exe             572 True   False  True     True   True  True    True
0x3e1b1928 svchost.exe            1712 True   False  True     True   True  True    True
0x3defd2b8 dwm.exe                1204 True   False  True     True   True  True    True
0x3f5af4c0 mscorsvw.exe           3272 True   False  True     True   True  True    True
0x3e3a4338 lsass.exe               452 True   False  True     True   True  True    False
0x3de2a030 SearchIndexer.         1928 True   False  True     True   True  True    True
0x3dee4308 conhost.exe            3908 True   False  True     True   True  True    True
0x3df1a440 timeout.exe            3128 True   False  True     True   True  True    True
0x3fcdf5b8 SearchProtocol         3720 True   False  True     True   True  True    True
0x3e03e030 svchost.exe            1424 True   False  True     True   True  True    True
0x3e19f030 svchost.exe            1332 True   False  True     True   True  True    True
0x3fcbad20 SearchFilterHo         3796 True   False  True     True   True  True    True
0x3df16c48 cmd.exe                1348 True   False  True     True   True  True    True
0x3de2b328 taskhost.exe            764 True   False  True     True   True  True    True
0x3ded5a38 svchost.exe            3240 True   False  True     True   True  True    True
0x3e344030 winlogon.exe            376 True   False  True     True   True  True    True
0x3e3817d8 services.exe            436 True   False  True     True   True  True    False
0x3e0c7510 svchost.exe             900 True   False  True     True   True  True    True
0x3fd283a8 WmiPrvSE.exe           1404 True   False  True     True   True  True    True
0x3fcaebb8 WmiPrvSE.exe           3548 True   False  True     True   True  True    True
0x3e3a6378 lsm.exe                 460 True   False  True     True   True  True    False
0x3fff3188 System                    4 True   False  True     True   False False   False
0x3e8aed20 csrss.exe               348 True   False  True     True   False True    True
0x3e307030 csrss.exe               304 True   False  True     True   False True    True
0x3eac84c0 smss.exe                228 True   False  True     True   False False   False
0x3e39a368 regsvr32.exe           2068 True   False  False    True   False True    False    2024-01-02 21:07:04 UTC+0000
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q>
```

Figure 13: psxview result

Based on the psxview results, all listed processes have pslist=True, indicating they are visible in the standard Windows process list and not hidden. This suggests that no rootkit-style hiding techniques were used for these processes.

Step 3: Investigating Command Line

- Next, I analyzed the command-line arguments used by processes with the following command:

```
vol2 -f win7.vmem --profile=Win7SP1x86_23418 cmdline
**************************************************************************
cmd.exe pid:      1348
Command line : C:Windows\system32\cmd.exe /c ""C:\Users\kusti\Desktop\test.bat" "
**************************************************************************
```

The cmd.exe process was launched to run a batch file located at C:\Users\kusti\Desktop\test.bat

It appears the attacker used this batch file to execute malicious actions quietly. The script likely contains a loop involving tasklist and timeout, which suggests system monitoring or a delay tactic before delivering the actual payload. I will try to find file scan of the process and content in the file test.bat.

```
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q> vol2 -f win7.vmem --profile=Win7SP1x86_23418 filescan | findstr "test.bat"
Volatility Foundation Volatility Framework 2.6
0x000000003fcab4e0      8      0 RW-rw- \Device\HarddiskVolume1\Users\kusti\Desktop\test.bat
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q> vol2 -f win7.vmem --profile=Win7SP1x86_23418 dumpfiles -Q 0x000000003fcab4e0 -D DUMP
Volatility Foundation Volatility Framework 2.6
DataSectionObject 0x3fcab4e0   None   \Device\HarddiskVolume1\Users\kusti\Desktop\test.bat
PS C:\Users\DARONG\Desktop\Lecture 36_Practice material\37_Q>
```

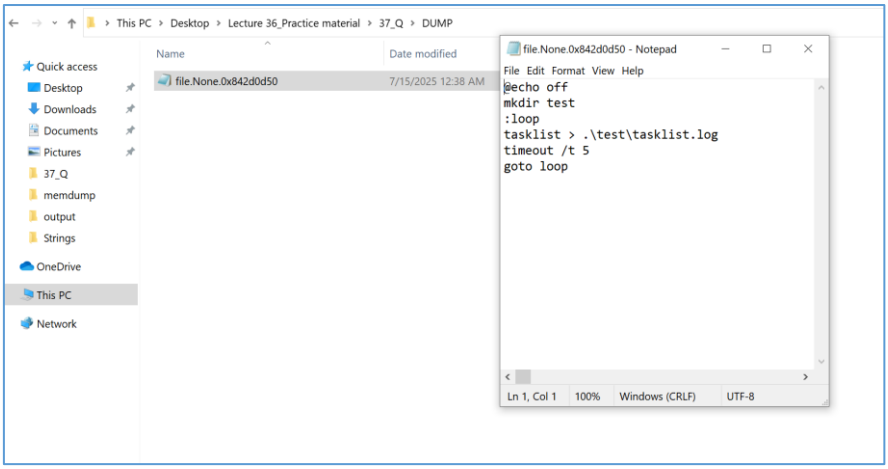


Figure 14: script extract

After dumping and reviewing the contents of the test.bat file, I found that the script is designed to run in an infinite loop.

- Hiding terminal output using @echo off.
- Creating a folder named test, if it doesn't already exist.
- Collecting running process information using tasklist, which logs the output to .tasklist.log.
- Delaying execution with timeout /t 5.
- Repeating the process with goto loop

➢ Let me check the network connectivity related to the file.

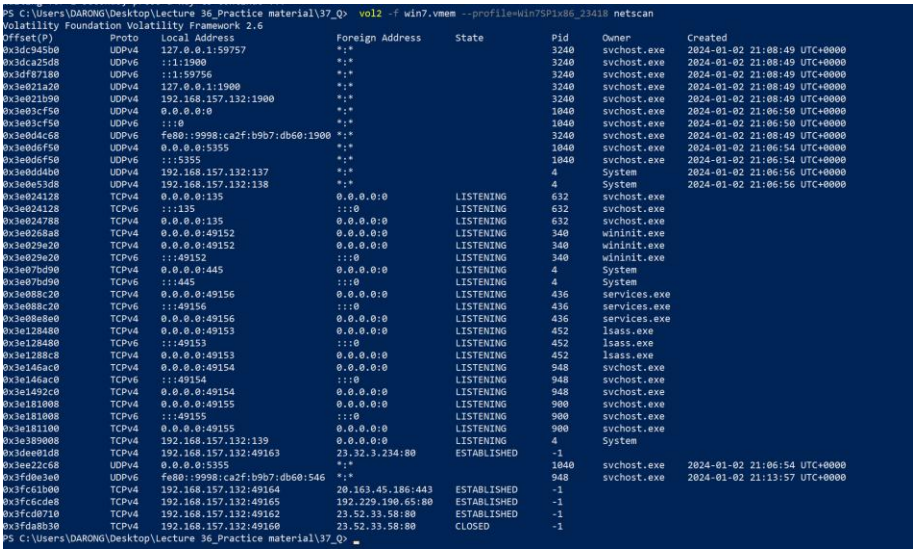


Figure 15: Network Connectivity

I did a scan of a network using windows 7 dump file (win7.vmem) to see open networks and ports at the time of its capture. The netscan plugin displayed all the necessary data on the current state of the network with software and hardware in use, local and foreign IP addresses and ports, connection states, owning processes, and timestamps.

I have captured Established outbound TCP connections to external IP address on

typical ports such as HTTPS (443) and HTTP (80). As an example, the connection with IPs like 20.163.45.186:443 are 23.52.33.58:80 are standard external communications. Network connections and listening ports detected do not seem unusual indicating the system was operating as expected. Based on this output, there were no suspicious or unauthorized network connections spotted, immediately or at all.

☑ Step 4: Process Memory Dump

In this step, I dumped the memory of the memory locked inside processes called the cmd.exe to examine the I/O operations of the process and gain more knowledge regarding what the command prompt was doing coincidentally of its memory capture.

Going through the strings obtained, there were no visible suspicious behavior and malicious indicators. The full executable files that are related to the mentioned processes would be extracted with the help of the procdump plugin in order to explore the matter to the full.

These executables will be sent to a sandboxed environment that will further analyze their behavior and can identify concealed or sophisticated threats not visible concurrently through a static analysis of memory strings.

```
vol2 -f win7.vmem --profile=Win7SP1x86_23418 memdump -p 1348 -D DUMP
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing cmd.exe [ 1348] to 1348.dmp
vol2 -f win7.vmem --profile=Win7SP1x86_23418 memdump -p 3128 -D DUMP
Volatility Foundation Volatility Framework 2.6
************************************************************************
Writing cmd.exe [ 1348] to 3128.dmp
```

Command used

```
vol2 -f win7.vmem --profile=Win7SP1x86_23418 procdump -p 3128 -D DUMP
vol2 -f win7.vmem --profile=Win7SP1x86_23418 procdump -p 3128 -D DUMP
```

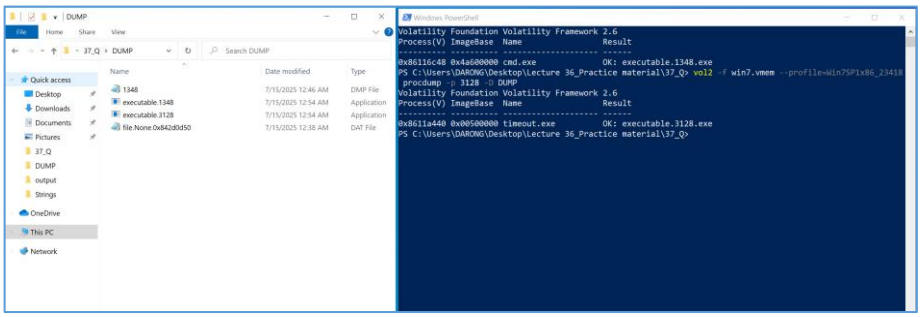


Figure 16: exe file

☑ Step5 : Upload and Deep Analysis in Sandbox Environment

The second thing would be to upload the two extracted files that are in the form of executable files and perform a detailed dynamic analysis in a sandbox environment. Using this method, one can observe the runtime behavior of the files in a confined environment, so it is therefore easier to detect any malicious or covert activities which may not have been picked by the static analysis.

The sandbox detection proved that both executables hide malicious and suspicious elements, and such elements are even basic to the memory dumps found above. This result suggests that indeed malware contained in these processes did infect the system.

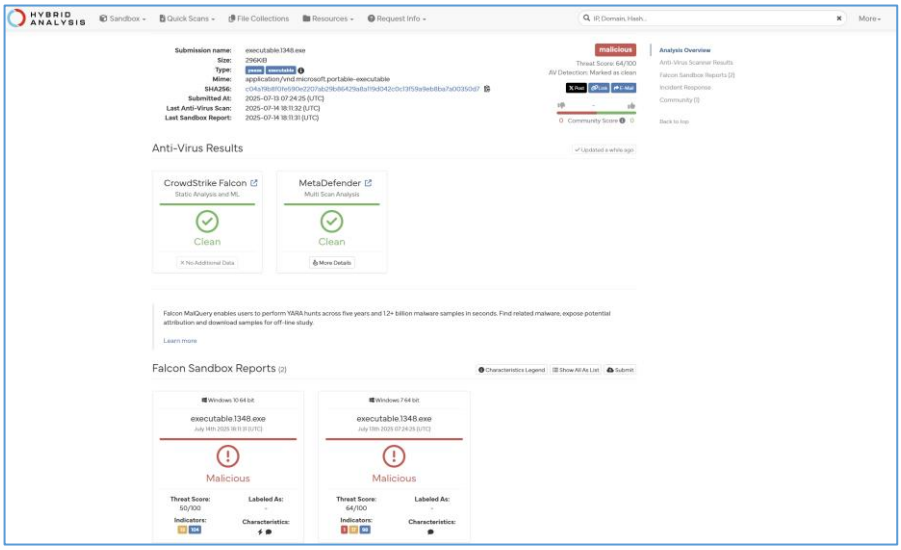


Figure 17:1348.exe file

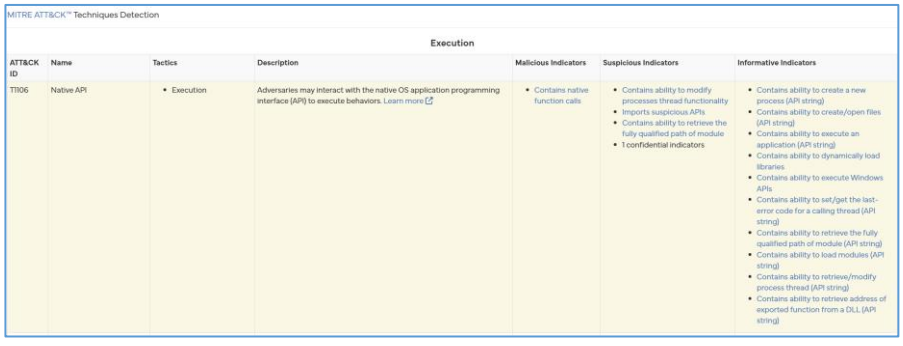


Figure 18: MITRE ATT&CK

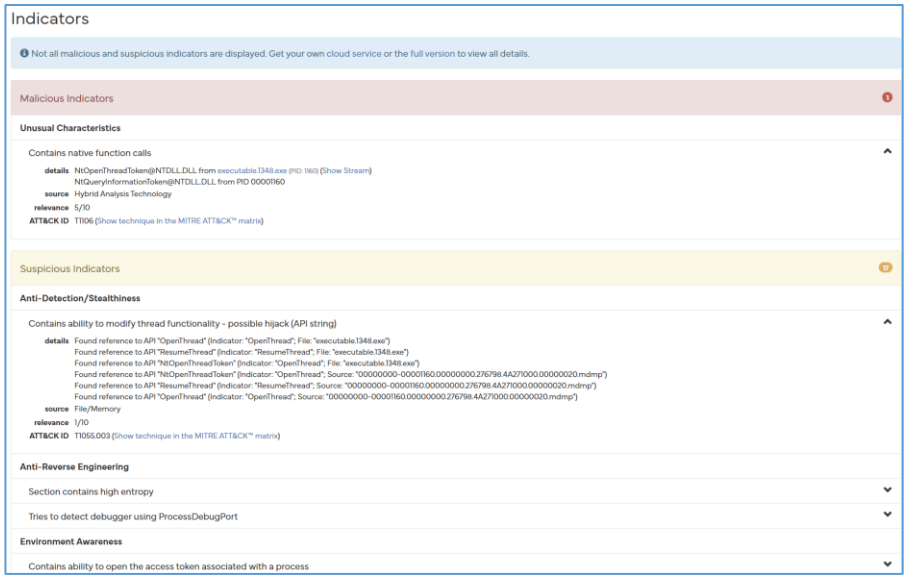


Figure 19: Indicator

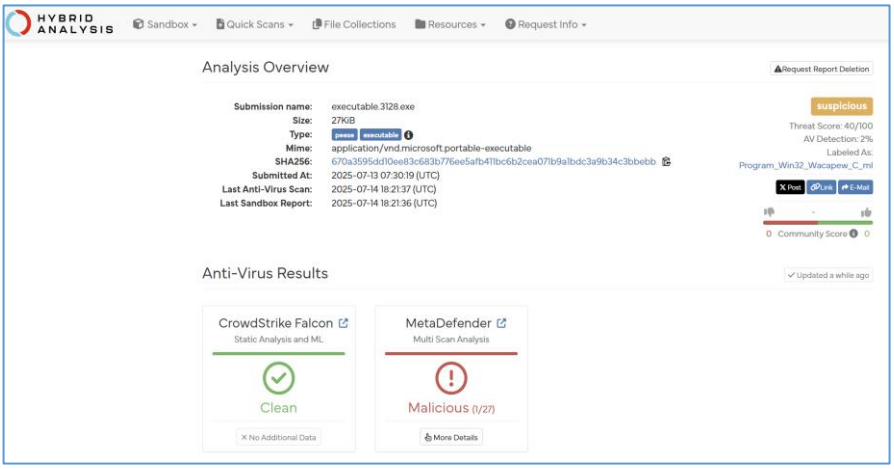


Figure 20: 3128.exe

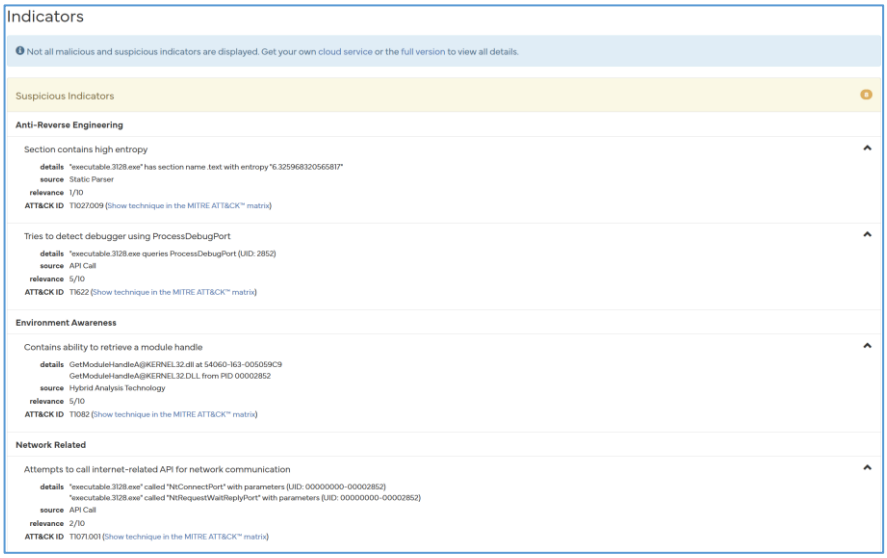


Figure 21: Indicator

☑ Step 6: Final Result

This analysis of the memory of the Windows 7 system with the help of Volatility showed the active network connections and several processes of the command prompt, which were active at the time of capturing. The first analysis of the memory dumps via string analysis did not reveal anything suspicious in the cmd.exe. Nevertheless, the extraction of the accompanied executables, as well as sandbox research, proved the existence of malicious components embedded.

This illustrates the need of integrating various methods of forensics such as memory dumping, use of strings and dynamic sandbox testing so as to successfully identify advanced malware. In general, the research emphasizes the role of memory forensics in exposing the hidden threats that static analysis may not detect and supports its usefulness on incident responses and digital forensics.

- Technical & Personal Reflections

Memory forensics using Volatility Framework was introduced well in this lab. With the help of practical tasks, I acquired an actual experience of the ways to correlate memory artifacts, listing of processes, command followed arguments, and memory injections to recreate an image of a compromised system.

I discovered that the sophisticated tactics which attackers commonly use include:

- Obfuscation as a form of hiding the process,
- Memory intercession to insert and run pernicious codes straight inside RAM.
- Exploitation of Living Off the Land Binaries (LOLBins) in which legitimate system tools are re-used and abused in order to execute malicious actions.

The insights of the stealthy tactics made me understand that attacks can be made to seem ordinary by just looking at them, but they have major effects. This observation stressed the need to make extensive analyzes and not to look at superficial signs.

The fact that this investigation allowed the critical analysis of suspicious behaviors was probably what made it especially valuable, as it was not concerned merely with symptoms. My mindset was trained into asking questions regarding the existence of a process, how it was initiated, and what it is attempting to attain because these are the same skill sets required in practical incident handling.

On the personal level, I:

- Become less frightened of command-line tools and raw memory dumps files,
- Improved my thinking skills and analysis,
- Was able to learn how to tackle malware in a methodical way with an intermingling of both tools and observations.
- Acquired the understanding of the strategies of the real-world adversaries to hide in the memory.
- In general, I learned more about the behavior of advanced malware in the volatile memory and the need to apply a multi-layered approach to the forensic investigation procedure undertaken. It has also encouraged me to learn more about the area of threat hunting and incident response.

- References

a. CrowdStrike. (n.d.). Falcon Sandbox Docs. July 14, 2025, https://www.crowdstrike.com/resources/?fwp_resource_type=documentation
b. MITRE ATT&CK. (n.d.). T1027.009: Obfuscated Files or Information: Embedded Payloads. Accessed July 14, 2025, https://attack.mitre.org/techniques/T1027/009/
c. SANS Institute. (n. d.). Cheat Sheets Volatility Memory Forensics. Accessed July 14, 2025, https://digital-forensics.sans.org/community/downloads
d. The Volatility Foundation. (n.d.). Volatility Framework. Retrieved July 14, 2025, on https://github.com/volatilityfoundation/volatility/wiki.

Lab02: CTF-Based Memory Forensics and Threat Hunting with Volatility

I was able to undertake a detailed memory extraction of a compromised windows computer using a single memory dump (kus-8563252c.vmem) in this forensic lab. Lab is modelled on Capture the Flag (CTF) nature of investigation where I used Volatility and other forensic tools to find evidence of suspicious processes, malware injections, persistence mechanisms, and indicators of compromise (IoCs).

☑ Memory Forensics Workflow

This section outlines the step-by-step workflow I followed during the forensic investigation of the memory image (kus-8563252c.vmem) using Volatility 2.

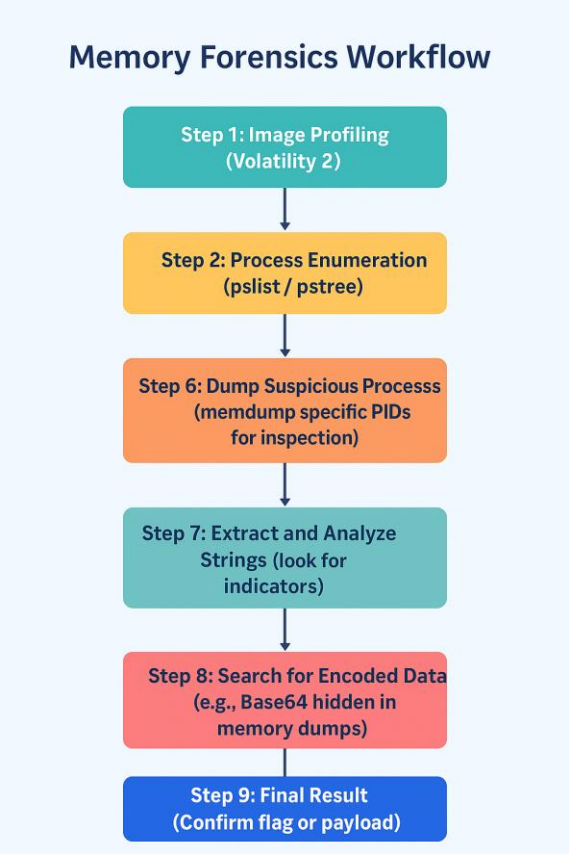


Figure 22: Work Flow

☑ Step 1-Image Profiling

Starting the information of the image in memory dump.

Command used

vol2 -f kus-8563252c.vmem imageinfo

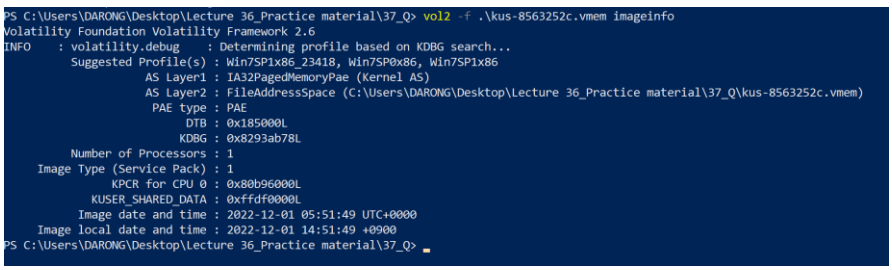


Figure 23: Imageinfo

Output Summary:

- Suggested Profiles: Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
- Architecture: 32-bit with PAE enabled (IA32PagedMemoryPae)
- DTB (Directory Table Base): 0x185000
- KDBG (Kernel Debugger Block): 0x8293ab78
- Processors: 1 CPU

- OS Build / Version: Windows 7 SP1 (Service Pack 1)
- Image Date/Time: 2022-12-01 05:51:49 UTC
- Local Timezone: +0900 (Korea/Japan timezone)

☑ Step 2: Process Enumeration

Based on the profile information obtained from the previous step (Win7SP1x86_23418), the next phase involved enumerating all running processes in the memory image. The goal was to identify any hidden, or suspicious processes that could indicate malicious activity.

To achieve this, I used multiple Volatility plugins to gain different perspectives on the process state:

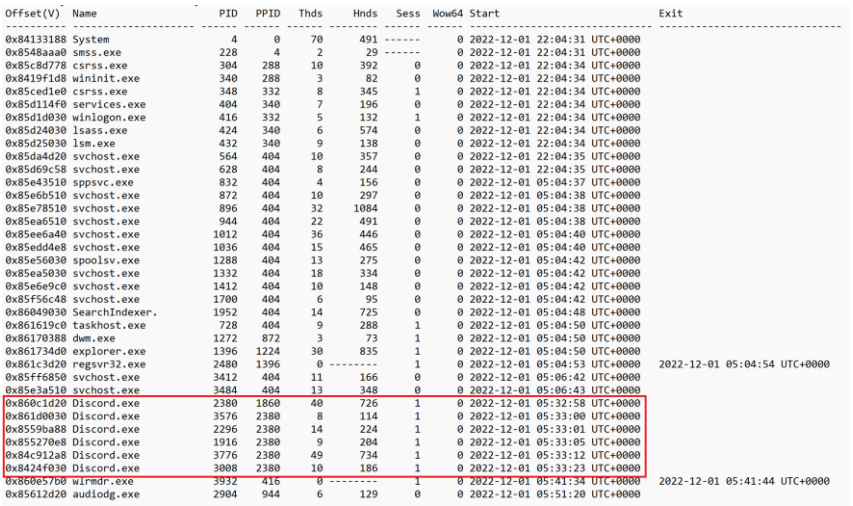
- pslist: Lists all running processes as reported by the active process list.
- pstree: Displays processes in a parent-child tree structure to understand process relationships.
- psxview: Cross-verifies process visibility across multiple listings to detect hidden or injected processes.

Commands executed:

vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 pslist

vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 pstree

vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 psxview



Offset(V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start	Exit
0x84133188	System	4	0	70	491	------	0	2022-12-01 22:04:31 UTC+0000	
0x8548aaa0	smss.exe	228	4	2	29	------	0	2022-12-01 22:04:31 UTC+0000	
0x85c8d778	csrss.exe	304	288	10	392	0	0	2022-12-01 22:04:34 UTC+0000	
0x8419f1d8	wininit.exe	340	288	3	82	0	0	2022-12-01 22:04:34 UTC+0000	
0x85ced1e0	csrss.exe	348	332	8	345	1	0	2022-12-01 22:04:34 UTC+0000	
0x85d114f0	services.exe	404	340	7	196	0	0	2022-12-01 22:04:34 UTC+0000	
0x85d1d030	winlogon.exe	416	332	5	132	1	0	2022-12-01 22:04:34 UTC+0000	
0x85d24030	lsass.exe	424	340	6	574	0	0	2022-12-01 22:04:34 UTC+0000	
0x85d25030	lsm.exe	432	340	9	138	0	0	2022-12-01 22:04:34 UTC+0000	
0x85da4d20	svchost.exe	564	404	10	357	0	0	2022-12-01 22:04:35 UTC+0000	
0x85d69c58	svchost.exe	628	404	8	244	0	0	2022-12-01 22:04:35 UTC+0000	
0x85e43510	sppsvc.exe	832	404	4	156	0	0	2022-12-01 05:04:37 UTC+0000	
0x85e6b510	svchost.exe	872	404	10	297	0	0	2022-12-01 05:04:38 UTC+0000	
0x85e78510	svchost.exe	896	404	32	1084	0	0	2022-12-01 05:04:38 UTC+0000	
0x85ea6510	svchost.exe	944	404	22	491	0	0	2022-12-01 05:04:38 UTC+0000	
0x85ee6a40	svchost.exe	1012	404	36	446	0	0	2022-12-01 05:04:40 UTC+0000	
0x85edd4e8	svchost.exe	1036	404	15	465	0	0	2022-12-01 05:04:40 UTC+0000	
0x85e56030	spoolsv.exe	1288	404	13	275	0	0	2022-12-01 05:04:42 UTC+0000	
0x85ea5030	svchost.exe	1332	404	18	334	0	0	2022-12-01 05:04:42 UTC+0000	
0x85e6e9c0	svchost.exe	1412	404	10	148	0	0	2022-12-01 05:04:42 UTC+0000	
0x85f56c48	svchost.exe	1700	404	6	95	0	0	2022-12-01 05:04:42 UTC+0000	
0x86049030	SearchIndexer.	1952	404	14	725	0	0	2022-12-01 05:04:48 UTC+0000	
0x861619c0	taskhost.exe	728	404	9	288	1	0	2022-12-01 05:04:50 UTC+0000	
0x86170388	dwm.exe	1272	872	3	73	1	0	2022-12-01 05:04:50 UTC+0000	
0x861734d0	explorer.exe	1396	1224	30	835	1	0	2022-12-01 05:04:50 UTC+0000	
0x861c3d20	regsvr32.exe	2480	1396	0	--------	1	0	2022-12-01 05:04:53 UTC+0000	2022-12-01 05:04:54 UTC+0000
0x85ff6850	svchost.exe	3412	404	11	166	0	0	2022-12-01 05:06:42 UTC+0000	
0x85e3a510	svchost.exe	3484	404	13	348	0	0	2022-12-01 05:06:43 UTC+0000	
0x860c1d20	Discord.exe	2380	1860	40	726	1	0	2022-12-01 05:32:58 UTC+0000	
0x861d0030	Discord.exe	3576	2380	8	114	1	0	2022-12-01 05:33:00 UTC+0000	
0x8559ba88	Discord.exe	2296	2380	14	224	1	0	2022-12-01 05:33:01 UTC+0000	
0x855270e8	Discord.exe	1916	2380	9	204	1	0	2022-12-01 05:33:05 UTC+0000	
0x84c912a8	Discord.exe	3776	2380	49	734	1	0	2022-12-01 05:33:12 UTC+0000	
0x8424f030	Discord.exe	3008	2380	10	186	1	0	2022-12-01 05:33:23 UTC+0000	
0x860e57b0	wirmdr.exe	3932	416	0	--------	1	0	2022-12-01 05:41:34 UTC+0000	2022-12-01 05:41:44 UTC+0000
0x85612d20	audiodg.exe	2904	944	6	129	0	0	2022-12-01 05:51:20 UTC+0000	

Figure 24: pslist

Referring to the output of the pslist plugin it is possible to note that the Discord.exe process is the most interesting one and it should be examined further. Following is the overview of all running processes found in the memory image; particular emphasis is placed on the numerous occurrences of the Discord.exe process as it might have played a pivotal role in the presented forensic study

I rebuilt the parent child relationship between processes in the memory image kus-8563252c.vmem. In this visualization, it was possible to find the hierarchy of

the processes and even detect any anomalies or suspicious activity.

Below is a summarized output of the process tree:

```
Volatility Foundation Volatility Framework 2.6
Name                                          Pid   PPid   Thds   Hnds Time
--------------------------------------------- ------ ------ ------ ------ ----
 0x85c8d778:csrss.exe                          304    288     10    392 2022-12-01 22:04:34 UTC+0000
 0x8419f1d8:wininit.exe                        340    288      3     82 2022-12-01 22:04:34 UTC+0000
. 0x85d114f0:services.exe                      404    340      7    196 2022-12-01 22:04:34 UTC+0000
.. 0x85e78510:svchost.exe                      896    404     32   1084 2022-12-01 05:04:38 UTC+0000
.. 0x85e6e9c0:svchost.exe                     1412    404     10    148 2022-12-01 05:04:42 UTC+0000
.. 0x85e56030:spoolsv.exe                     1288    404     13    275 2022-12-01 05:04:42 UTC+0000
.. 0x85edd4e8:svchost.exe                     1036    404     15    465 2022-12-01 05:04:40 UTC+0000
.. 0x85e3a510:svchost.exe                     3484    404     13    348 2022-12-01 05:06:43 UTC+0000
.. 0x85ea6510:svchost.exe                      944    404     22    491 2022-12-01 05:04:38 UTC+0000
... 0x85612d20:audiodg.exe                    2904    944      6    129 2022-12-01 05:51:20 UTC+0000
.. 0x85f56c48:svchost.exe                     1700    404      6     95 2022-12-01 05:04:42 UTC+0000
.. 0x85ea5030:svchost.exe                     1332    404     18    334 2022-12-01 05:04:42 UTC+0000
.. 0x85da4d20:svchost.exe                      564    404     10    357 2022-12-01 22:04:35 UTC+0000
.. 0x85ee6a40:svchost.exe                     1012    404     36    446 2022-12-01 05:04:40 UTC+0000
.. 0x85e43510:sppsvc.exe                       832    404      4    156 2022-12-01 05:04:37 UTC+0000
.. 0x86049030:SearchIndexer.                  1952    404     14    725 2022-12-01 05:04:48 UTC+0000
.. 0x861619c0:taskhost.exe                     728    404      9    288 2022-12-01 05:04:50 UTC+0000
.. 0x85e6b510:svchost.exe                      872    404     10    297 2022-12-01 05:04:38 UTC+0000
... 0x86170388:dwm.exe                        1272    872      3     73 2022-12-01 05:04:50 UTC+0000
.. 0x85d69c58:svchost.exe                      628    404      8    244 2022-12-01 22:04:35 UTC+0000
.. 0x85ff6850:svchost.exe                     3412    404     11    166 2022-12-01 05:06:42 UTC+0000
. 0x85d24030:lsass.exe                         424    340      6    574 2022-12-01 22:04:34 UTC+0000
. 0x85d25030:lsm.exe                           432    340      9    138 2022-12-01 22:04:34 UTC+0000
 0x860c1d20:Discord.exe                       2380   1860     40    726 2022-12-01 05:32:58 UTC+0000
. 0x84c912a8:Discord.exe                      3776   2380     49    734 2022-12-01 05:33:12 UTC+0000
. 0x861d0030:Discord.exe                      3576   2380      8    114 2022-12-01 05:33:00 UTC+0000
. 0x8559ba88:Discord.exe                      2296   2380     14    224 2022-12-01 05:33:01 UTC+0000
. 0x8424f030:Discord.exe                      3008   2380     10    186 2022-12-01 05:33:23 UTC+0000
. 0x855270e8:Discord.exe                      1916   2380      9    204 2022-12-01 05:33:05 UTC+0000
 0x84133188:System                               4      0     70    491 2022-12-01 22:04:31 UTC+0000
. 0x8548aaa0:smss.exe                          228      4      2     29 2022-12-01 22:04:31 UTC+0000
 0x85d1d030:winlogon.exe                       416    332      5    132 2022-12-01 22:04:34 UTC+0000
. 0x860e57b0:wlrmdr.exe                       3932    416      0 ------ 2022-12-01 05:41:34 UTC+0000
 0x85ced1e0:csrss.exe                          348    332      8    345 2022-12-01 22:04:34 UTC+0000
 0x861734d0:explorer.exe                      1396   1224     30    835 2022-12-01 05:04:50 UTC+0000
. 0x861c3d20:regsvr32.exe                     2480   1396      0 ------ 2022-12-01 05:04:53 UTC+0000
```

Figure 25:pstree

```
Offset(P)  Name                PID pslist psscan thrdproc pspcid csrss session deskthrd ExitTime
---------- ---------------- ------ ------ ------ -------- ------ ----- ------- -------- --------
0x3fc4f030 Discord.exe       3008 True   False  True     True   True  True    True
0x3e0e6a40 svchost.exe       1012 True   False  True     True   True  True    True
0x3dfd0030 Discord.exe       3576 True   False  True     True   True  True    True
0x3df70388 dwm.exe           1272 True   False  True     True   True  True    True
0x3df734d0 explorer.exe      1396 True   False  True     True   True  True    True
0x3e06b510 svchost.exe        872 True   False  True     True   True  True    True
0x3e078510 svchost.exe        896 True   False  True     True   True  True    True
0x3ff5f1d8 wininit.exe        340 True   False  True     True   True  True    True
0x3e056030 spoolsv.exe       1288 True   False  True     True   True  True    True
0x3e3a4d20 svchost.exe        564 True   False  True     True   True  True    True
0x3e324030 lsass.exe          424 True   False  True     True   True  True    False
0x3e0a5030 svchost.exe       1332 True   False  True     True   True  True    True
0x3f2912a8 Discord.exe       3776 True   False  True     True   True  True    True
0x3dec1d20 Discord.exe       2380 True   False  True     True   True  True    True
0x3e325030 lsm.exe            432 True   False  True     True   True  True    False
0x3eb9ba88 Discord.exe       2296 True   False  True     True   True  True    True
0x3e31d030 winlogon.exe       416 True   False  True     True   True  True    True
0x3eb270e8 Discord.exe       1916 True   False  True     True   True  True    True
0x3df619c0 taskhost.exe       728 True   False  True     True   True  True    True
0x3e812d20 audiodg.exe       2904 True   False  True     True   True  True    True
0x3e156c48 svchost.exe       1700 True   False  True     True   True  True    True
0x3e1f6850 svchost.exe       3412 True   False  True     True   True  True    True
0x3e06e9c0 svchost.exe       1412 True   False  True     True   True  True    True
0x3e369c58 svchost.exe        628 True   False  True     True   True  True    True
0x3e0a6510 svchost.exe        944 True   False  True     True   True  True    True
0x3e043510 sppsvc.exe         832 True   False  True     True   True  True    True
0x3de49030 SearchIndexer.    1952 True   False  True     True   True  True    True
0x3e3114f0 services.exe       404 True   False  True     True   True  True    False
0x3e0dd4e8 svchost.exe       1036 True   False  True     True   True  True    True
0x3e03a510 svchost.exe       3484 True   False  True     True   True  True    False
0x3fff3188 System               4 True   False  True     True   False False   False
0x3ea8aaa0 smss.exe           228 True   False  True     True   False False   False
0x3e2ed1e0 csrss.exe          348 True   False  True     True   False True    True
0x3dee57b0 wlrmdr.exe        3932 True   False  False    True   False True    False    2022-12-01 05:41:44 UTC+0000
0x3e28d778 csrss.exe          304 True   False  True     True   False True    True
0x3dfc3d20 regsvr32.exe      2480 True   False  False    True   False True    False    2022-12-01 05:04:54 UTC+0000
0x3dec9250                      1 False  True   False    False  False False   False
```

Figure 26:psxview

Based on the psxview results, all listed processes have pslist=True, indicating they are visible in the standard Windows process list and not hidden. This suggests that no rootkit-style hiding techniques were used for these processes.

Let check netscan for find the network connection on the file revealed potential indicators of compromise

```
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 netscan
Volatility Foundation Volatility Framework 2.6
Offset(P)          Proto    Local Address                  Foreign Address      State          Pid      Owner          Created
0x3dc1cf50         UDPv4    0.0.0.0:5355                   *:*                                 1036     svchost.exe    2022-12-01 05:44:09 UTC+0000
0x3dd2d3c8         UDPv6    fe80::6d62:4b42:5c6c:78eb:546  *:*                                 944      svchost.exe    2022-12-01 05:51:16 UTC+0000
0x3dd46968         UDPv6    ::1:1900                       *:*                                 3412     svchost.exe    2022-12-01 05:28:34 UTC+0000
0x3de82370         UDPv4    0.0.0.0:5353                   *:*                                 2296     Discord.exe    2022-12-01 05:35:02 UTC+0000
0x3de82370         UDPv6    :::5353                        *:*                                 2296     Discord.exe    2022-12-01 05:35:02 UTC+0000
0x3de9b008         UDPv4    0.0.0.0:58082                  *:*                                 2296     Discord.exe    2022-12-01 05:51:32 UTC+0000
0x3debf008         UDPv4    192.168.157.142:138            *:*                                 4        System         2022-12-01 05:44:09 UTC+0000
0x3dee8290         UDPv6    fe80::6d62:4b42:5c6c:78eb:2177 *:*                                 3412     svchost.exe    2022-12-01 05:44:06 UTC+0000
0x3df07400         UDPv4    127.0.0.1:53729                *:*                                 3412     svchost.exe    2022-12-01 05:28:34 UTC+0000
0x3dfb46a0         UDPv4    127.0.0.1:1900                 *:*                                 3412     svchost.exe    2022-12-01 05:28:34 UTC+0000
0x3e036cd0         UDPv4    192.168.157.142:2177           *:*                                 3412     svchost.exe    2022-12-01 05:44:06 UTC+0000
0x3e2fd5f8         UDPv4    0.0.0.0:0                      *:*                                 1036     svchost.exe    2022-12-01 05:44:09 UTC+0000
0x3e2fd5f8         UDPv6    :::0                           *:*                                 1036     svchost.exe    2022-12-01 05:44:09 UTC+0000
0x3e32eaf8         UDPv6    ::1:53728                      *:*                                 3412     svchost.exe    2022-12-01 05:28:34 UTC+0000
0x3e3e1138         UDPv4    192.168.157.142:1900           *:*                                 3412     svchost.exe    2022-12-01 05:28:34 UTC+0000
0x3dee9a90         TCPv4    192.168.157.142:139            0.0.0.0:0            LISTENING      4        System
0x3e044d70         TCPv4    127.0.0.1:6463                 0.0.0.0:0            LISTENING      3776     Discord.exe
0x3e0b2d68         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING      404      services.exe
0x3e0b2d68         TCPv6    :::49155                       :::0                 LISTENING      404      services.exe
0x3e0b4ab8         TCPv4    0.0.0.0:49155                  0.0.0.0:0            LISTENING      404      services.exe
0x3e0b6848         TCPv4    0.0.0.0:445                    0.0.0.0:0            LISTENING      4        System
0x3e0b6848         TCPv6    :::445                         :::0                 LISTENING      4        System
0x3e0bc008         TCPv4    0.0.0.0:49156                  0.0.0.0:0            LISTENING      424      lsass.exe
0x3e0bc008         TCPv6    :::49156                       :::0                 LISTENING      424      lsass.exe
0x3e11d660         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING      944      svchost.exe
0x3e11d660         TCPv6    :::49153                       :::0                 LISTENING      944      svchost.exe
0x3e1531f0         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING      896      svchost.exe
0x3e1531f0         TCPv6    :::49154                       :::0                 LISTENING      896      svchost.exe
0x3e1536b8         TCPv4    0.0.0.0:49154                  0.0.0.0:0            LISTENING      896      svchost.exe
0x3e355820         TCPv4    0.0.0.0:49153                  0.0.0.0:0            LISTENING      944      svchost.exe
0x3e3c80f8         TCPv4    0.0.0.0:135                    0.0.0.0:0            LISTENING      628      svchost.exe
0x3e3c80f8         TCPv6    :::135                         :::0                 LISTENING      628      svchost.exe
0x3e3c84a8         TCPv4    0.0.0.0:135                    0.0.0.0:0            LISTENING      628      svchost.exe
0x3e3d42f0         TCPv4    0.0.0.0:49152                  0.0.0.0:0            LISTENING      340      wininit.exe
0x3e3d42f0         TCPv6    :::49152                       :::0                 LISTENING      340      wininit.exe
0x3e3d44a0         TCPv4    0.0.0.0:49152                  0.0.0.0:0            LISTENING      340      wininit.exe
0x3e8179a0         UDPv4    0.0.0.0:68                     *:*                                 944      svchost.exe    2022-12-01 05:50:56 UTC+0000
0x3ea69e98         UDPv6    fe80::6d62:4b42:5c6c:78eb:1900 *:*                                 3412     svchost.exe    2022-12-01 05:28:34 UTC+0000
0x3eb653f8         UDPv4    0.0.0.0:5353                   *:*                                 2296     Discord.exe    2022-12-01 05:35:02 UTC+0000
0x3ebeaac8         UDPv4    192.168.157.142:68             *:*                                 944      svchost.exe    2022-12-01 05:50:29 UTC+0000
0x3ec691f8         TCPv4    0.0.0.0:49156                  0.0.0.0:0            LISTENING      424      lsass.exe
```

Figure 27: Network Connectivity

☑ Step 6-Dump Suspicious Processes for Further Review

During the analysis of the memory image (kus-8563252c.vmem), I identified a parent process Discord.exe (PID 2380) and multiple child instances running under it. This was visible from the process tree and pslist output.

List of Detected Discord Processes

Name	Pid	PPid	Thds	Hnds	Time
0x860c1d20:Discord.exe			2380	1860	40
726 2022-12-01 05:32:58 UTC+0000					
. 0x84c912a8:Discord.exe			3776	2380	49
734 2022-12-01 05:33:12 UTC+0000					
. 0x861d0030:Discord.exe			3576	2380	8
114 2022-12-01 05:33:00 UTC+0000					
. 0x8559ba88:Discord.exe			2296	2380	14
224 2022-12-01 05:33:01 UTC+0000					
. 0x8424f030:Discord.exe	3008	2380	10	186	
2022-12-01 05:33:23 UTC+0000					
. 0x855270e8:Discord.exe			1916	2380	9
204 2022-12-01 05:33:05 UTC+0000					

This activity places a possibility that Discord was either launching several subprocesses or modules which can come as normal to Electron applications such as Discord, or it may be used by the attackers as a living-off- the-land approach where attackers add malicious code to an existing legitimate and trusted process. This is not necessarily malicious considering that Discord is popular and is likely to have several helper processes running. However, Discord process has been reported

to be abused by attackers to evade detection and get lost in the midst of usual activity.

Next in Analysis

To search further, I applied memo dump plugin in volatility to obtain memory contents of each discovered discord process.

Command used

```
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 memdump -p 1916 -D DUMP
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 memdump -p 2296 -D DUMP
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 memdump -p 2380 -D DUMP
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 memdump -p 3008 -D DUMP
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 memdump -p 3576 -D DUMP
vol2 -f kus-8563252c.vmem --profile=Win7SP1x86_23418 memdump -p 3776 -D DUMP
```

☑ Step 7. Extract and Analyze Strings

The presence of multiple Discord.exe processes is unusual and worth deeper inspection. By extracting their memory, I have set the stage for the next steps in analysis, including string searching.

It was unusual that numerous processes of Discord.exe were present on the system at the same time, and it justified a closer investigation. The only issue is that it is not unusual when Discord can create some helper processes, however, the number of helpers (during the time) and the parent-child relations seemed suspicious sometimes, including the time these processes were started- probably because of a Capture the Flag (CTF) game.

In the next step, to find out more, I applied the memdump plugin of Volatility in order to dump the memory of all the Discord.exe processes. This gave raw memory dumps which were offline analyzed and more specific inspection of in-memory artifacts was possible.

After extracting the memory, I used the strings tool to list human readable data of the dumped memory files. This measure allowed me to detect some possible clues like:

- Path names of files to be suspicious, command line or registry keys,
- IP addresses or domain names or hardcoded URLs,
- Access tokens, user IDs or other sensitive material,
- Secret text or codes.

Base64-encoded string search was also employed besides generic string searches, as the strings are usually used to encode data in memory. These strings are usually encoded payload, concealed commands or stolen data. I excluded possible Base64 strings (they are usually character pattern and length) and decoded them to get any plaintext hiding behind.

Command Used

strings -n 6 1916.dmp > 1916_strings.txt

strings -n 6 2296.dmp > 2296_strings.txt

strings -n 6 2380.dmp > 2380_strings.txt

strings -n 6 3008.dmp > 3008_strings.txt

strings -n 6 3576.dmp > 3576_strings.txt

strings -n 6 3776.dmp > 3776_strings.txt

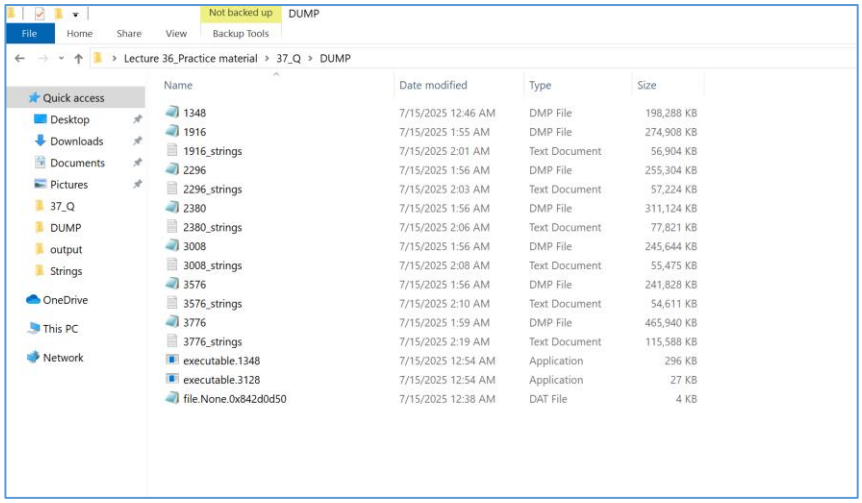


Figure 28: Result Strings

In the process of extracting memory strings and artifacts, I came across the HTML tags and encoding of the base64 data in them and they are hidden using the CSS style like display: none;. It was important to include this particular technique in the report since it is one of the most widely applied techniques to placate data invisibly in the HTML material. It is a known technique to put Base64 in the invisible HTML element in CTF challenge or even the CTF malware or phishing attack. It allows storing the data in a plain view in the HTML files or chat logs that cannot be instantly recognized by the user, making the attacker remain unnoticed.

By noting the hidden tags, I demonstrated that I could not only extract and analyze visible artifacts but also detect covert techniques used to disguise key data. This highlights the need for detailed forensic analysis in memory and file investigations.

☑ Step 8- Search for Encoded Data (Base64)

Use PowerShell with regex to extract base64 inside tags. Base64-encoded CTF flag was hidden in memory and extracted from dumped processes

Run this in PowerShell prompt:

```
# Read all content from the strings file
$content = Get-Content -Path "1916_strings.txt" -Raw
# Regex to find base64 inside <span> tags (with optional attributes)
$pattern = '<span[^>]*>([A-Za-z0-9+/=]{6,})</span>'
# Find all matches
[regex]::Matches($content, $pattern) | ForEach-Object {
    $_.Groups[1].Value
} | Set-Content -Path "base64_in_spans.txt"
```

This will create base64_in_spans.txt with all base64 strings found inside tags. After that the content of the file as the base64 format, let me decode it.

Decoding that base64 string on Powershell

```
[System.Text.Encoding]::UTF8.GetString([Convert]::FromBase64String("ZGVidWdDVEZ7dl90aGlzX2lzX2U0NXlfdn0="))
```

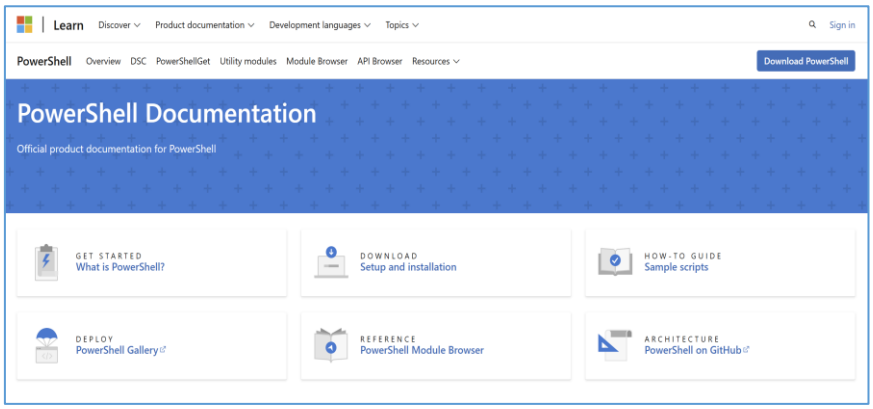


Figure 30: PowerShell Document

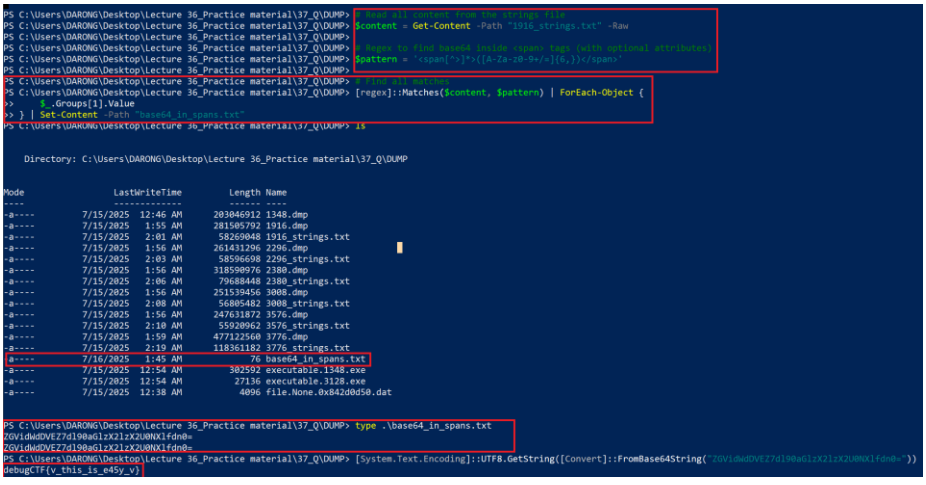


Figure 29: Extract Data

The flag **debugCTF{v_this_is_e45y_v}** confirms the presence of a challenge element in this forensic memory image. Its identification demonstrates the success of using memory analysis to extract valuable indicators and supports the learning objective

of detecting payloads in compromised environments.

☑ Step 9- Final Result

The flag was known to be:

debugCTF{v_this_is_e45y_v}

This flag was in place in several files extracted by using the strings utility, and this is an indication that there was a CTF challenge payload present in the memory dump.

Link GitHub: https://github.com/Darong-cyber/CTF

- Technical & Personal Reflections

This lab involved the usage of more sophisticated memory analysis tools, like Volatility and PowerShell to unearth masked artifacts in the system memory. The location of the embedded flag demonstrated the way digital evidence could be discoverable in volatile memory as well as be recovered by analytical procedure.

  - CTF format stimulated the application of several investigative approaches, such as
  - Analysis of raw memory using strings to take out indicators
  - Memory artifacts Correlation (process, handle, module)
  - Rational thinking to join the disparate dots

It strengthened an argument concerning the usefulness of memory forensics in the discovery of post-exploitation activity and detection of data that might be not found during file based or disk analysis.

This lab did not only involve my technical skills but it tested my investigative mind as well. It involved the thinking of a malicious user, intensive analysis of system artifacts, and putting together pieces of scattered evidence that enable me to create a comprehensive image of what happened. The life in every point of the investigation supported the importance of system approach, and nonetheless, permitted the innovative issue resolution.

Capture the Flag (CTF) format did a good job in fostering explorations. It challenged me to employ a diversity of tools including and Volatility and PowerShell, and the log correlation and apply these tools to a practical situation.

Consequently, I am now more confident about conducting memory forensic investigation particularly in an incident response. The exercise has made me further understand the place of memory analysis of modern cybersecurity processes, as well as skills I can apply directly and practically that will not be knowledge merely in theory.

	• Reference a. Volatility Foundation. Volatility Framework: An advanced memory forensics framework URL: https://www.volatilityfoundation.org b. Windows Processes Microsoft Docs. In order to know the main processes of the window like smss.exe, csrss.exe, winlogon.exe, etc. URL: https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands c. SANS Digital Forensics and Incident Response (DFIR). Techniques of Memory Forensics and Analysis. Link to the source: https://www.sans.org/cyber-security-courses/advanced-digital-forensics-incident-response-threat-hunting/ d. Process In retail setting, Process Injection, Persistence and Command & Control techniques in MITRE ATT&CK Compendium. Link: https://attack.mitre.org/ e. The GCHQ cyberChef. Base64 Decoder and CTF-style Data Recovery Text Analyzers. URL: https:// gchq.github.io/CyberChef / f. PowerShell Base64 Decoder PowerShell method of extracting and decoding base64 encrypted text. https://learn.microsoft.com/en-us/powershell/