

ANALYSIS OF RANSOMWARE ATTACKS IN WINDOWS OPERATING SYSTEM USING THE APPROACH OF MEMORY ANALYSIS

Muhammad Ichsan Rabani Lidanta^{*1)}, Vera Suryani²⁾, Erwid Musthofa Jadied³⁾

1. Faculty of Informatics, Telkom University, Bandung, Indonesia
2. Faculty of Informatics, Telkom University, Bandung, Indonesia
3. Faculty of Informatics, Telkom University, Bandung, Indonesia

Article Info

Keywords: Detection; Memory Analysis; Ransomware; Suspicious Activities

Article history:

Received 2 Agustus 2024

Revised 18 September 2024

Accepted 29 September 2024

Available online 1 September 2025

DOI :

<https://doi.org/10.29100/jipi.v10i3.6317>

* Corresponding author.

Muhammad Ichsan Rabani Lidanta

E-mail address:

twinxpeanut@student.telkomuniversity.ac.id

ABSTRACT

Ransomware is a growing and evolving problem in digital security. The significant losses caused by ransomware can target individuals as well as companies and organizations due to its increasingly complex and escalating threats. To address this issue, a memory analysis approach is needed to gain a better understanding of its characteristics and behavior. This research proposes a memory analysis approach as a means to detect and analyze ransomware. The memory analysis approach involves capturing the memory running on an infected operating system. This approach can also assist in detection and analyzing ransomware samples that may go undetected by traditional security tools. The result shows the memory analysis approach is capable of detecting WannaCry infections through the analysis of running processes and DLL files. However, this method was not successful in detecting other ransomware infections such as Jigsaw and Locky. These results indicate that the specific characteristics of WannaCry make it identifiable through this approach, while other types of ransoms may require different detection techniques.

I. INTRODUCTION

RANSOMWARE is a form of malware that encrypts a victim's documents and media, subsequently demanding payment from the victim to decrypt and restore access to the encrypted files [1]. As criminals advance and present new threats, significant research efforts have been dedicated to creating countermeasures that effectively shield individuals and organizations from these attacks [2] [3]. Ransomware attackers often demand ransom payments in Bitcoin due to the anonymity provided by encrypted transactions [4]. Ransomware typically involves executing a process on the target program and extracting valuable user data [5].

Unlike other cyberattacks that leave some parts functional, ransomware attacks pose a significant security threat because they can cripple the core functions of a business system. The primary reason for shifting the target from regular users to businesses and organizations is the potentially greater financial gain [6]. Ransomware can change the device PIN and demand a ransom payment to provide a new PIN when attacking mobile devices [7]. It involves examining the volatile memory of a computer to gather information about active processes and other system statuses. By analyzing system memory, valuable information about ransomware and effective detection strategies can be obtained to counteract the threat [8].

Research conducted by Hwang et al., in 2020, a two-stage mixed ransomware detection method was proposed. This approach combines the Markov Chain model for detecting ransomware and Random Forest. Markov Chain assumes that transitions depend only on the current state, not on the longer history of previous states. This makes it less effective in predicting complex patterns that may involve long-term dependencies. Random Forest is generally resistant to overfitting, it can still occur, particularly with highly imbalanced or complex datasets where individual decision trees may overfit the training data. Using techniques like Principal Component Analysis (PCA) for feature selection or dimensionality reduction can help reduce complexity and improve interpretability. For Markov Chain employing higher-order Markov Chains can capture more historical context, though this increases complexity. However, a drawback of this research is that the resulting model must be continuously trained to detect new types of ransoms that can disguise themselves as regular malware [5].

Research conducted by Manaar et al., in 2020, performance counters were used to detect ransomware. Performance counters are software-based monitoring tools that can track the performance of a computer system. A

limitation of this research is its inability to handle cases like the WannaCry ransomware, which encrypts each file with randomly generated and unique keys using AES-128 CBC (Cipher Block Chaining). Combining data from performance counters with advanced machine learning models can improve detection accuracy [9].

Research conducted by Hampton et al., in 2018 successfully identified significant differences in the usage of Windows API calls between a normal operating system and one infected with ransomware. Identifying ransomware by low-level windows API calls can be useful [10]. In 2018 research by Cabaj et al., Software Defined Networking (SDN) was used to detect ransomware. In the developed system, SDN is used to provide a rapid response to detected threats. The results demonstrated that this approach offers good effectiveness. However, SDN may not be suitable for all types of networks and other variations of ransomware. Implementing architectures that support redundancy and failover can mitigate risks if the controller is attacked [11].

This is highly problematic because most hosts or original owners connected to the internet possess valuable assets, such as financial data and confidential company information, which could become targets of attacks [12]. Therefore, memory analysis is crucial for maintaining the security and performance of computer systems. It detects ransomware by looking for suspicious DLL files, as each type of ransomware typically has its own characteristics [13].

Windows 10 is one of the most widely used operating systems globally, both in personal and corporate environments. Understanding how ransomware operates on Windows 10 is crucial for enhancing security for the majority of computer users [14] [15]. The selected ransomware samples represent common characteristics found in other ransomware variants. Therefore, the findings from this research can provide insights that can be generalized to address other ransomware strains with similar attributes [16]. Previous studies have extensively explored ransomware behavior on various operating systems, including earlier versions of Windows, focusing on attack vectors, encryption methods, and mitigation strategies. This research builds on that foundation by applying established analytical methods to Windows 10, ensuring that the findings are relevant to today's users. The aim of this research is to determine how to detect ransomware using a memory analysis on existing ransomware samples. However, the limitation of this research is that it is conducted solely on the Windows 10 operating system and only includes three ransomware samples Jigsaw, Locky, and WannaCry. This research uses a memory analysis approach to detect ransomware with the Volatility tool. Volatility is an open-source tool commonly used by security experts to detect ransomware and other suspicious activities, especially in RAM [17].

II. RESEARCH METHODOLOGY

Figure 1 illustrates several main stages that must be carried out in this research process, adapting the flow of the Analysis Memory Cycle.

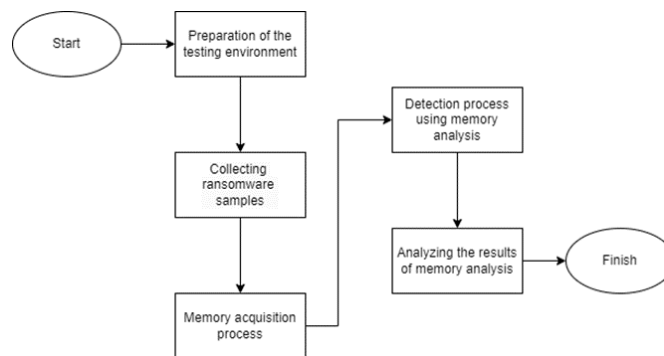


Fig. 1. Analysis Memory Cycle

A. Preparation of The Testing Environment

The following table shows the hardware specifications used in the testing environment and the software specifications used in the experimental environment.

TABLE I
HARDWARE SPECIFICATIONS

Hardware	Information
Processor	Intel® Core™ i5-1035G1 CPU @ 1.00GHz (8CPUs)
Memory	12GB (12288 MB)
SSD	500GB
Operating System	Windows 11 Home Single Language 64-bit (10.0, Build 22631)

TABLE II
SOFTWARE SPECIFICATIONS

Software	Functions
Windows 10 Pro 64-bit (10.0, Build 19045)	As the guest operating system in VirtualBox, it was duplicated into three separate entities for testing the Jigsaw, Locky, and WannaCry ransomware. This cloning process enabled separate and specific analysis of the system's reaction to each type of ransomware.
Ubuntu 22.04.4 LTS (Jammy Jellyfish)	As an additional guest operating system in VirtualBox, it was used to run the Volatility analysis framework.
Oracle VM VirtualBox 7.0.14	As a virtual environment where both Windows 10 and Ubuntu operating systems can be installed and run independently on a single physical machine, it facilitates the evaluation of ransomware in a controlled and isolated setting.
FTK Imager 4.7.1	As a utility for capturing memory images from infected virtual systems, it provides an effective way to secure the memory of infected systems without altering the existing data.
Volatility 2.6.1	As a framework or tool used to analyze memory captured from ransomware infected systems.

After the initial installation of the Windows 10 operating system was completed, the virtual machine was cloned to create three separate virtual machines, each designated for testing the Jigsaw, Locky, and WannaCry ransomware. Each of these virtual machines was isolated and prepared for infection with their respective ransomware independently.

The Ubuntu virtual machine was allocated 4000 MB (4 GB) of RAM, 2 CPUs, and a 50 GB hard disk, just like the Windows 10 virtual machine. The Ubuntu operating system was installed on VirtualBox, along with the Volatility 2.6.1 software for memory forensic analysis. The Shared Folders configuration was adjusted to facilitate access and transfer of memory dump files generated from the ransomware-infected Windows 10 virtual machine.

Volatility is an open-source software, meaning it is freely available to anyone. This makes it accessible to the cybersecurity community, researchers, and law enforcement agencies worldwide [18]. Volatility supports a broad range of operating systems and versions, including Windows, Linux, and macOS. This allows users to analyze memory dumps from various platforms, making it a versatile tool in memory forensics. When analyzing large memory dumps, Volatility can be slow, particularly if the hardware being used is not powerful enough. This can impact the efficiency of the investigation process, especially when time is a critical factor [19]. FTK Imager is known for its reliable and accurate imaging capabilities. It creates bit-by-bit copies of digital evidence, ensuring that no data is altered or lost during the imaging process. This level of accuracy is crucial for maintaining the integrity of evidence in forensic investigations. While FTK Imager is excellent for creating forensic images, its analysis capabilities are limited compared to full-featured forensic suites. Users often need to use additional tools, such as FTK (Forensic Toolkit) or other forensic software, for in-depth analysis [20].

B. Collecting Ransomware Samples

Table III classifies the ransomware samples by family, lists the number of samples for each family and the source.

TABLE III
RANSOMWARE SAMPLES

No	Ransomware Family	Total	Source	Years
1	Jigsaw	1	https://github.com/kh4sh3i/Ransomware-Samples	2021
2	Locky	1	https://github.com/kh4sh3i/Ransomware-Samples	2021
3	WannaCry	1	https://github.com/kh4sh3i/Ransomware-Samples	2021

The ransomware samples used in this ransomware were obtained from a GitHub repository. The author of the repository packaged each sample in a zip file and added a password for extraction to prevent accidental execution.

C. Memory Acquisition Process

After obtaining the ransomware samples, they will be used with FTK Imager to capture memory image files. Each memory image is 4.5 GB, matching the RAM allocation on the virtual machine, ensuring the entire system memory is fully recorded. The user starts FTK Imager with administrative privileges to ensure it can access and capture the entire memory space. FTK Imager provides the option to capture the physical memory of the system. The investigator navigates to File > Capture Memory to initiate the memory acquisition process. All memory image files are then stored in the /home/ubuntu/Desktop/data/ directory on the Ubuntu operating system.

D. Detection Process Using Memory Analysis

The detection process will follow the workflow outlined in Figure 2.

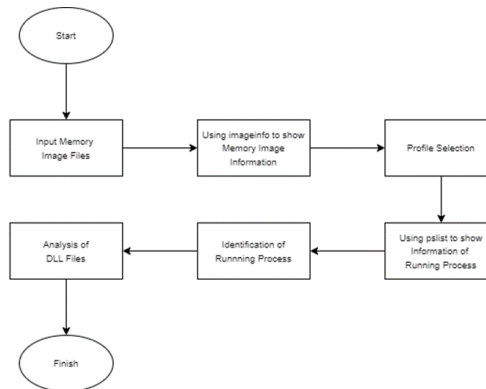


Fig. 2. Memory Analysis Procedure

1) Input Memory Image Files

List of memory image files is shown in Figure 3.

```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ ls -lh ~/Desktop/data/Jigsaw.mem
-rwxrwxrwx 1 ubuntu ubuntu 4,5G Mei  1 17:04 /home/ubuntu/Desktop/data/Jigsaw.mem
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ ls -lh ~/Desktop/data/Locky.mem
-rwxrwxrwx 1 ubuntu ubuntu 4,5G Mei  1 17:12 /home/ubuntu/Desktop/data/Locky.mem
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ ls -lh ~/Desktop/data/WannaCry.mem
-rwxrwxrwx 1 ubuntu ubuntu 4,5G Mei  1 14:17 /home/ubuntu/Desktop/data/WannaCry.mem
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$
  
```

Fig. 3. List of Memory Image Files

After completing the memory acquisition process, the memory image files are ready to be analyzed. In this research, the Volatility tool will function as the analyzer. The image files will run according to the path.

2) Memory Image Information

The image info plugin from the Volatility Framework was used to identify basic information from the memory images generated during the process. The results indicated that all three memory images came from a Windows 10 64-bit operating system, version 19041, with an Intel x64-based CPU.

```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/Jigsaw.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win10x64_19041
      AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/ubuntu/Desktop/data/Jigsaw.mem)
      PAE type : No PAE
      DTB : 0x1aa002L
      KDBG : 0xf80746a00b20L
      Number of Processors : 2
      Image Type (Service Pack) : 0
      KPCR for CPU 0 : 0xfffff807429a6000L
      KPCR for CPU 1 : 0xfffff807429a6000L
      KUSER_SHARED_DATA : 0xfffff80000000000L
      Image date and time : 2024-05-01 10:04:07 UTC+0000
      Image local date and time : 2024-05-01 17:04:07 +0700
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$
  
```

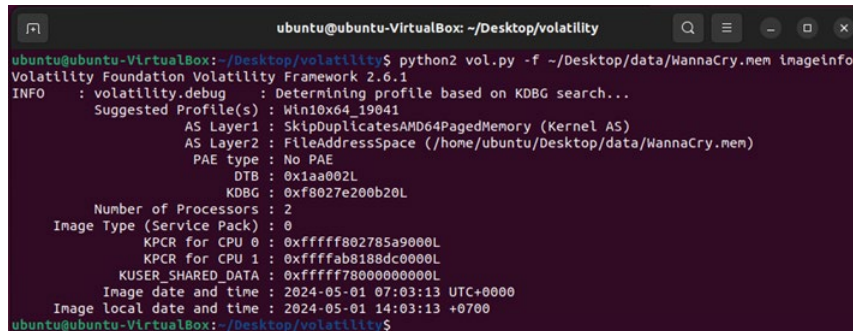
Fig. 4. Profile Information for The Jigsaw Memory Image

```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/Locky.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
      Suggested Profile(s) : Win10x64_19041
      AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
      AS Layer2 : FileAddressSpace (/home/ubuntu/Desktop/data/Locky.mem)
      PAE type : No PAE
      DTB : 0x1aa002L
      KDBG : 0xf8033f200b20L
      Number of Processors : 2
      Image Type (Service Pack) : 0
      KPCR for CPU 0 : 0xfffff8033babc000L
      KPCR for CPU 1 : 0xfffff8033babc000L
      KUSER_SHARED_DATA : 0xfffff80000000000L
      Image date and time : 2024-05-01 10:11:04 UTC+0000
      Image local date and time : 2024-05-01 17:11:04 +0700
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$
  
```

Fig. 5. Profile Information for The Locky Memory Image

As shown in Figure 4, the memory image infected with Jigsaw provides the recommended profile is Win10x64_19041, with the same date and a memory image creation time of 10:04 UTC and local time of 17:04. Figure 5 shows The Locky memory image also shows the same profile, Win10x64_19041, with the same date and a memory image creation time of 10:11 UTC and local time of 17:11.



```
ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/WannaCry.mem imageinfo
Volatility Foundation Volatility Framework 2.6.1
INFO : volatility.debug : Determining profile based on KDBG search...
Suggested Profile(s) : Win10x64_19041
AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/home/ubuntu/Desktop/data/WannaCry.mem)
PAE type : No PAE
DTB : 0x1aa002L
KDBG : 0xf8027e20b20L
Number of Processors : 2
Image Type (Service Pack) : 0
KPCR for CPU 0 : 0xfffff802785a9000L
KPCR for CPU 1 : 0xfffffab8188dc0000L
KUSER_SHARED_DATA : 0xfffff78000000000L
Image date and time : 2024-05-01 07:03:13 UTC+0000
Image local date and time : 2024-05-01 14:03:13 +0700
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$
```

Fig. 6. Profile Information for The WannaCry Memory Image

As shown in Figure 6, the memory image infected with WannaCry display the recommended profile is Win10x64_19041. This information includes memory address layers, PAE type, number of processors (2), and image type (Service Pack 0). Additional details include KPCR (Kernel Processor Control Region) for CPU 0 and 1, and KUSER_SHARED_DATA. The memory image creation date and time are May 1, 2024, at 07:03 UTC, with a local time of 14:03.

3) Profile Selection

The consistent profile across all memory images indicates a standardized environment. The Win10x64_19041 profile was chosen because it matches the operating system version used in the memory images, which is Windows 10 64-bit version 19041. Selecting the correct memory image profile is crucial as it enables in-depth analysis of system activities, including running processes, active network connections, and files accessed during the ransomware infection. A proper profile also allows for the detection of anomalies or suspicious activities that may indicate the presence of ransomware.

4) Information of Running Process

At this stage, the pslist plugin from the Volatility Framework is used to identify the processes running at the time the system memory was captured. This plugin provides a list of active processes, including details such as process names, PIDs (Process IDs), start times, and other relevant information. This analysis is crucial for understanding how the ransomware operates, and which processes are active during the infection.

5) Identification of Running Process

a. Jigsaw

The analysis of the pslist command results from the Jigsaw.mem memory image in Volatility revealed several critical processes. These include the core processes such as System (PID 4) for kernel operations, registry (PID 92) for managing the Windows registry, and smss.exe (PID 348), known as the Session Manager Subsystem.

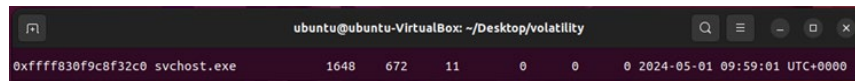
In terms of service processes, several important ones were found. The csrss.exe processes (PIDs 456 and 548) manage the user interface and console windows. The wininit.exe process (PID 532) is responsible for Windows initialization and starting various services and system components during boot. The winlogon.exe process (PID 628) handles user logins, while services.exe (PID 672) acts as the Service Control Manager. Additionally, the lsass.exe process (PID 684), which handles security policies, user logins, and password changes, was also identified.

The identified user processes include explorer.exe (PID 4496), which provides the graphical interface for file management and the desktop environment. MsMpEng.exe (PID 2832), associated with Windows Defender Antivirus, is responsible for real-time protection against ransomware and threats. SearchIndexer.exe (PID 3532) indexes content for the Windows search function, enabling quick search results for files and applications. RuntimeBroker.exe (PID 2600) manages app permissions and acts as a security intermediary for running applications.

A large number of svchost.exe processes were found, each running different services. The svchost.exe process with PID 792 has 13 threads and is responsible for running various system services shown in Figure 8. Another example is svchost.exe with PID 908 shows in Figure 9, which has 7 threads, and svchost.exe with PID 1648 shows in Figure 7, which has 11 threads, indicating that this process is running more intensive services. Other svchost.exe processes with various PIDs, such as 1092, 1128, 1332, and so on, each run different services with varying numbers of threads and handles.

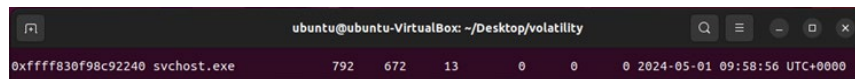
Specifically, unusual svchost.exe processes often use unexpected amounts of memory or have suspicious numbers of threads and handles. Based on these considerations, the processes identified for further examination include PID 1648, which has a high number of threads (11 threads). PIDs 792 and 908, although having a moderate number of threads, still need to be checked to ensure no suspicious activity.

Other svchost.exe PIDs appear to follow expected patterns and are thus given lower priority. By focusing on a few standout PIDs based on these criteria, the analysis is expected to be more efficient without needing to examine every svchost.exe PID in depth.



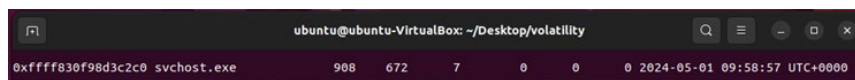
Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	1648	672	11	0	0	0	2024-05-01 09:59:01 UTC+0000

Fig. 7. Svchost.exe PID 1648 Process



Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	792	672	13	0	0	0	2024-05-01 09:58:56 UTC+0000

Fig. 8. Svchost.exe PID 792 Process



Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	908	672	7	0	0	0	2024-05-01 09:58:57 UTC+0000

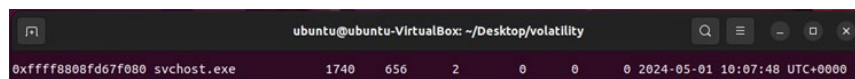
Fig. 9. Svchost.exe PID 908 Process

b. Locky

The analysis of the pslist command results from the Locky.mem memory image in Volatility revealed several critical processes. By identifying and understanding the roles of various processes such as System (PID 4), Registry (PID 92), and smss.exe (PID 348), we can better manage and secure the Windows environment. Additionally, examining suspicious svchost.exe instances based on specific criteria, such as an unusual number of threads or irregular start times, can help detect potential anomalies or malicious activities. Focusing the analysis on standout PIDs allows for a more efficient approach in ensuring the system's integrity and security.

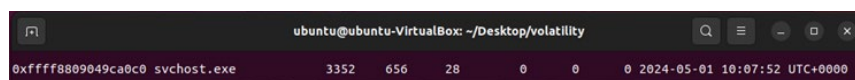
Several user-level processes, such as explorer.exe (PID 4108), provide the graphical interface for file management and the desktop environment. The MsMpEng.exe process (PID 2808) is associated with Windows Defender Antivirus. Additionally, SearchIndexer.exe (PID 3336) indexes content for the Windows search function. The svchost.exe process, which is a common host process for services running from dynamiclink libraries (DLLs), appears multiple times with various PIDs. This multiplication is normal in a Windows environment.

Although the presence of multiple svchost.exe processes is standard, some may require further investigation to detect potential anomalies or malicious activities. Based on these considerations, processes that need closer examination include PID 1740 shows in Figure 10, which has a low thread count (2 threads) that might be unusual. PID 3352 has a high thread count (28 threads) shows in Figure 11, which could be suspicious. As shown in Figure 12 and Figure 13, PIDs 5072 and 6200 started at different times (2024-05-01 10:09:57 and 10:10:01 UTC+0000) and have high thread counts (11 and 14 threads, respectively). By focusing on these standout PIDs based on these criteria, the analysis can be conducted more efficiently without having to examine each svchost.exe PID individually.



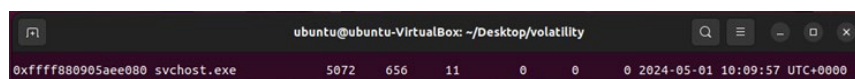
Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	1740	656	2	0	0	0	2024-05-01 10:07:48 UTC+0000

Fig. 10. Svchost.exe PID 1740 Process



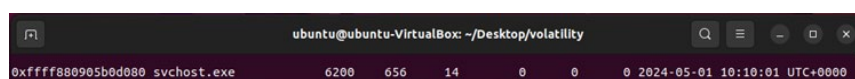
Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	3352	656	28	0	0	0	2024-05-01 10:07:52 UTC+0000

Fig. 11. Svchost.exe PID 3352 Process



Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	5072	656	11	0	0	0	2024-05-01 10:09:57 UTC+0000

Fig. 12. Svchost.exe PID 5072 Process



Process Name	PID	PPID	Threads	Private Bytes	Working Set	Start Time	Time
svchost.exe	6200	656	14	0	0	0	2024-05-01 10:10:01 UTC+0000

Fig. 13. Svchost.exe PID 6200 Process

c. WannaCry

In the analysis of the WannaCry.mem memory image using Volatility, various system and user processes active during the infection were successfully identified. The detected system processes include System (PID 4), Registry (PID 92), smss.exe (PID 344), csrss.exe (PIDs 452 and 544), wininit.exe (PID 524),

services.exe (PID 656), and lsass.exe (PID 664). These processes are core components of Windows operations, responsible for kernel management, registry operations, and security and system services.

Additionally, several important user processes were identified, including explorer.exe (PID 4368), MicrosoftEdgeU (PID 4060), RuntimeBroker.exe (PIDs 5104 and 5128), SearchApp.exe (PID 556), and SkypeBackground (PID 5396). These processes reflect user activity and applications running in the background during the infection.

Figure 14 shows the results from pslist also revealed the presence of a suspicious process, such as ed01ebfbc9eb5b (PID 1068), which has an unusual name and is likely associated with WannaCry ransomware activity. This finding strongly indicates a ransomware infection, warranting further analysis to determine its behavior and impact.

```
ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
```

0xffff880b2aa56080	ed01ebfbc9eb5b	1068	4368	7	0	1	1	2024-05-01	06:56:30	UTC+0000
--------------------	----------------	------	------	---	---	---	---	------------	----------	----------

Fig. 14. ed01ebfbc9eb5b PID 1068 Process

6) Analysis of DLL Files

After identification of the running process, we will be analyzing the DLL files loaded by a specific process that can provide valuable information about the activity and presence of ransomware in the system. By using the dlllist plugin in Volatility, we can understand the list of DLLs loaded by a particular process and how ransomware operates and interacts with the system.

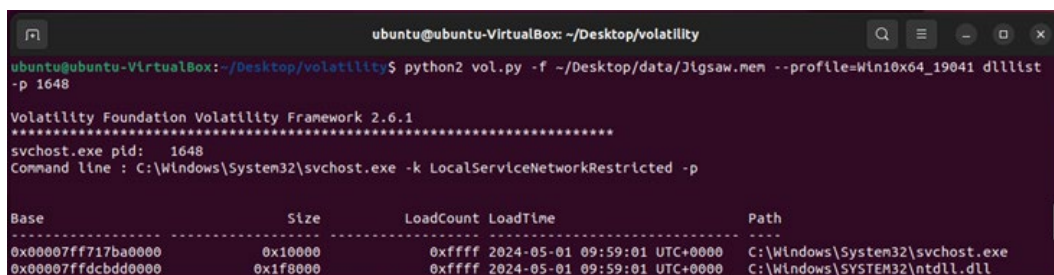
E. Analyzing the Result of Memory Analysis

Following the identification of the processes, further analysis will be conducted to examine the loaded DLLs and compare the system's performance before and after the ransomware infection. This analysis aims to understand the impact of ransomware on system resources and identify any additional malicious activities. To ensure the accuracy of our findings, we conduct meticulous monitoring of running processes and file system changes, cross-referencing these observations with baseline data to detect any anomalies or unauthorized modifications.

III. RESULT AND DISCUSSION

A. DLL File of Jigsaw

Following the identification of the processes, further analysis will be conducted to examine the loaded DLLs and compare the system's performance before and after the ransomware infection. This analysis aims to understand the impact of ransomware on system resources and identify any additional malicious activities. Figure 15 shows the svchost.exe process with PID 1648 runs services with the parameters -k LocalServiceNetworkRestricted -p. Analysis shows this process loads several critical DLL files, including ntdll.dll, KERNEL32.DLL, RPCRT4.dll, bcrypt.dll, and sechost.dll. Additionally, it loads modules related to audio management (audiosrv.dll), device configuration (cfgmgr32.dll), and power management (POWERPROF.dll). Further, it loads additional modules like MMDevAPI.DLL and AUDIOSRVPOLICYMANAGER.dll for managing audio services.



Base	Size	LoadCount	LoadTime	Path
0x00007ff717ba0000	0x10000	0xffff	2024-05-01 09:59:01 UTC+0000	C:\Windows\System32\svchost.exe
0x00007ffdcbbd0000	0x1f8000	0xffff	2024-05-01 09:59:01 UTC+0000	C:\Windows\System32\ntdll.dll

Fig. 15. Details of DLL Files from Svchost.exe Process PID 1648

Figure 16 shows the svchost.exe process with PID 792 runs services with the parameters -k DcomLaunch -p. The DLL files loaded by this process include ntdll.dll, KERNEL32.DLL, combase.dll, RPCRT4.dll, msvcrt_win.dll, and sechost.dll. This process also loads modules related to device management (umpnnpmgr.dll), power policy (umpo.dll), and network operations (IPHLPAPI.DLL). Additionally, files such as umpoext.dll and sppc.dll, which are associated with power policy management and product activation services, were also found.


```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/3lgsaw.mem --profile=Win10x64_19041 dlllist -p 792
Volatility Foundation Volatility Framework 2.6.1
*****
svchost.exe pid: 792
Command line : C:\Windows\system32\svchost.exe -k DcomLaunch -p

Base                               Size      LoadCount LoadTime                               Path
-----
0x00007ff717ba0000                 0x10000          0xffff 2024-05-01 09:58:56 UTC+0000 C:\Windows\system32\svchost.exe
0x00007ffdcbbd0000                 0x1f8000          0xffff 2024-05-01 09:58:56 UTC+0000 C:\Windows\SYSTEM32\ntdll.dll

```

Fig. 16. Details of DLL Files from Svchost.exe Process PID 792

Figure 17 shows the svchost.exe process with PID 908 runs services with the parameters -k RPCSS -p. The DLL files loaded by this process include ntdll.dll, KERNEL32.DLL, RPCRT4.dll, sspicli.dll, combase.dll, and FirewallAPI.dll. Other modules found include DNSAPI.dll, IPHLPAPI.DLL, powrprof.dll, and UMPDC.dll, all of which are related to managing network communication and security. This process also uses modules like fwpucnt.dll for packet filtering and security APIs, as well as ws2_32.dll for Windows Sockets network operations.

```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/3lgsaw.mem --profile=Win10x64_19041 dlllist -p 908
Volatility Foundation Volatility Framework 2.6.1
*****
svchost.exe pid: 908
Command line : C:\Windows\system32\svchost.exe -k RPCSS -p

Base                               Size      LoadCount LoadTime                               Path
-----
0x00007ff717ba0000                 0x10000          0xffff 2024-05-01 09:58:57 UTC+0000 C:\Windows\system32\svchost.exe
0x00007ffdcbbd0000                 0x1f8000          0xffff 2024-05-01 09:58:57 UTC+0000 C:\Windows\SYSTEM32\ntdll.dll
0x00007ffdc310000                 0xbd000          0xffff 2024-05-01 09:58:57 UTC+0000 C:\Windows\SYSTEM32\KERNEL32.DLL

```

Fig. 17. Details of DLL Files from Svchost.exe Process PID 908

B. DLL File of Locky

The analysis of the dlllist command results from Volatility for the svchost.exe processes with PIDs 1740, 3352, 5072, and 6200 provides important insights into the modules loaded by each process. As shown in Figure 18, the svchost.exe process with PID 1740 runs with the command line C:\Windows\System32\svchost.exe -k netsvcs -p -s Themes. The analysis shows that the themeservice.dll module, related to Windows theme services, is loaded, consistent with the Themes parameter in the command line. No suspicious DLLs were found loaded from the C:\Windows\System32 directory.

```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/Locky.mem --profile=Win10x64_19041 dlllist -p 1740
Volatility Foundation Volatility Framework 2.6.1
*****
svchost.exe pid: 1740
Command line : C:\Windows\System32\svchost.exe -k netsvcs -p -s Themes

Base                               Size      LoadCount LoadTime                               Path
-----
0x00007ff7e5da0000                 0x10000          0xffff 2024-05-01 10:07:48 UTC+0000 C:\Windows\System32\svchost.exe
0x00007ffc144d0000                 0x1f8000          0xffff 2024-05-01 10:07:48 UTC+0000 C:\Windows\SYSTEM32\ntdll.dll

```

Fig. 18. Details of DLL Files from Svchost.exe Process PID 1740

Figure 19 shows the svchost.exe process with PID 3352 runs with the command line C:\Windows\system32\svchost.exe -k wsappx -p -s AppXSvc. Modules such as appxdeploymentservice.dll and AppXDeploymentClient.dll, which are related to the AppX service, are loaded in accordance with the AppXSvc parameter. No suspicious DLLs were found. All loaded DLLs originate from the C:\Windows\SYSTEM32 or C:\Windows\system32 directories.

```

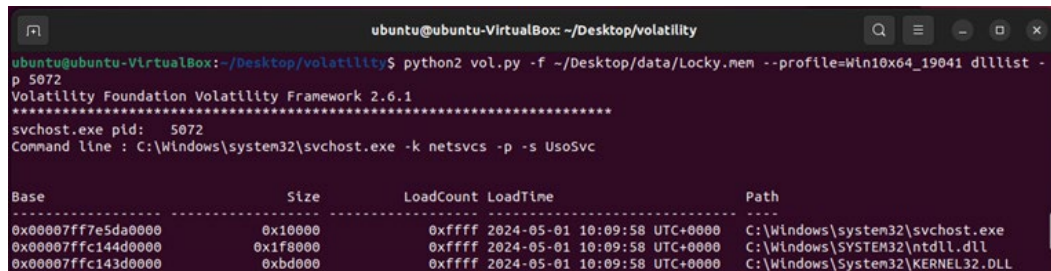
ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/Locky.mem --profile=Win10x64_19041 dlllist -p 3352
Volatility Foundation Volatility Framework 2.6.1
*****
svchost.exe pid: 3352
Command line : C:\Windows\system32\svchost.exe -k wsappx -p -s AppXSvc

Base                               Size      LoadCount LoadTime                               Path
-----
0x00007ff7e5da0000                 0x10000          0xffff 2024-05-01 10:07:52 UTC+0000 C:\Windows\system32\svchost.exe
0x00007ffc144d0000                 0x1f8000          0xffff 2024-05-01 10:07:52 UTC+0000 C:\Windows\SYSTEM32\ntdll.dll
0x00007ffc143d0000                 0xbd000          0xffff 2024-05-01 10:07:52 UTC+0000 C:\Windows\SYSTEM32\KERNEL32.DLL

```

Fig. 19. Details of DLL Files from Svchost.exe Process PID 3352

Next, Figure 20 shows the svchost.exe process with PID 5072 runs with the command line C:\Windows\system32\svchost.exe -k netsvcs -p -s UsoSvc. The analysis shows that the module usosvc.dll, related to the Windows Update service, is loaded in accordance with the UsoSvc parameter. All loaded DLLs originate from the C:\Windows\System32 or C:\Windows\system32 directories, with no suspicious DLLs found.



```

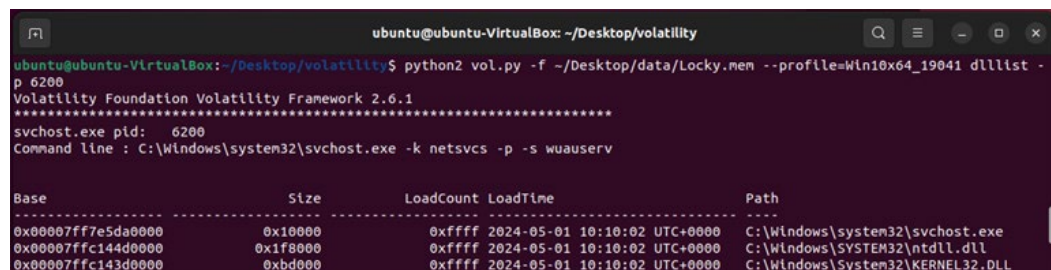
ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/Locky.mem --profile=Win10x64_19041 dlllist -p 5072
Volatility Foundation Volatility Framework 2.6.1
*****
svchost.exe pId: 5072
Command line : C:\Windows\system32\svchost.exe -k netsvcs -p -s UsoSvc
*****

```

Base	Size	LoadCount	LoadTime	Path
0x00007fff7e5da0000	0x10000	0xffff	2024-05-01 10:09:58 UTC+0000	C:\Windows\system32\svchost.exe
0x00007ffc144d0000	0x1f8000	0xffff	2024-05-01 10:09:58 UTC+0000	C:\Windows\SYSTEM32\ntdll.dll
0x00007ffc143d0000	0xbd000	0xffff	2024-05-01 10:09:58 UTC+0000	C:\Windows\System32\KERNEL32.DLL

Fig. 20. Details of DLL Files from Svchost.exe Process PID 5072

Finally, Figure 21 shows the svchost.exe process with PID 6200 runs with the command line C:\Windows\system32\svchost.exe -k netsvcs -p -s wuauserv. The module wuaueng.dll, related to the Windows Update Agent, is loaded in accordance with the wuauserv parameter. As with the other processes, no suspicious DLLs were found. All loaded DLLs originate from the C:\Windows\System32 or C:\Windows\system32 directories.



```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/Locky.mem --profile=Win10x64_19041 dlllist -p 6200
Volatility Foundation Volatility Framework 2.6.1
*****
svchost.exe pId: 6200
Command line : C:\Windows\system32\svchost.exe -k netsvcs -p -s wuauserv
*****

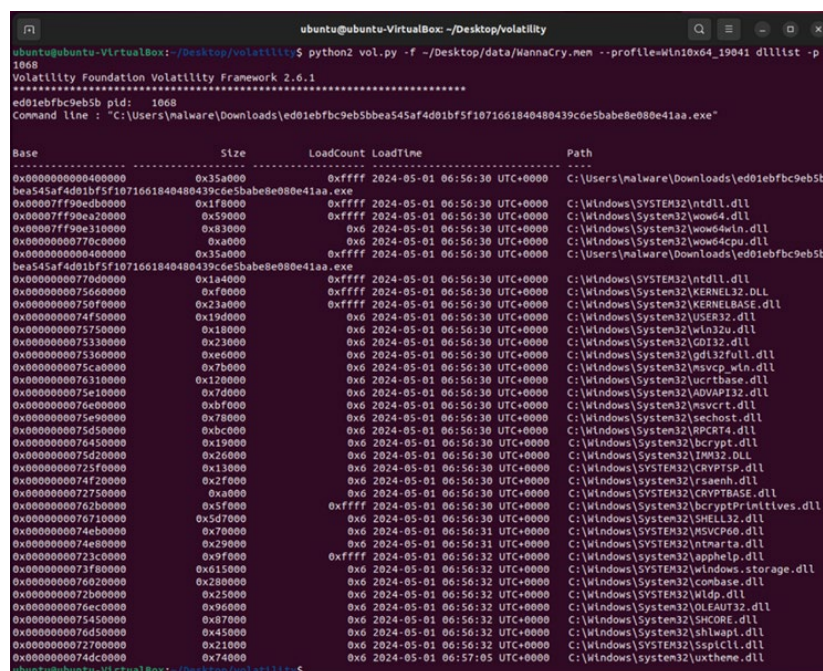
```

Base	Size	LoadCount	LoadTime	Path
0x00007fff7e5da0000	0x10000	0xffff	2024-05-01 10:10:02 UTC+0000	C:\Windows\system32\svchost.exe
0x00007ffc144d0000	0x1f8000	0xffff	2024-05-01 10:10:02 UTC+0000	C:\Windows\SYSTEM32\ntdll.dll
0x00007ffc143d0000	0xbd000	0xffff	2024-05-01 10:10:02 UTC+0000	C:\Windows\System32\KERNEL32.DLL

Fig. 21. Details of DLL Files from Svchost.exe Process PID 6200

C. DLL File of WannaCry

An in-depth analysis of the WannaCry.mem memory image using Volatility revealed important details about the Dynamic Link Libraries (DLLs) loaded by the suspicious process ed01ebfbc9eb5b with PID 1068. This process runs with the command line C:\Users\Sample\Downloads\ed01ebfbc9eb5b-bea545af4d01b-f5f1071661840-480439c6e5babe8e080e41aa.exe. It loads a large number of DLLs from the Windows system directory, including cryptographic and security DLLs. Figure 22 shows the list of DLL files loaded by the ed01ebfbc9eb5b process.



```

ubuntu@ubuntu-VirtualBox: ~/Desktop/volatility
ubuntu@ubuntu-VirtualBox:~/Desktop/volatility$ python2 vol.py -f ~/Desktop/data/WannaCry.mem --profile=Win10x64_19041 dlllist -p 1068
Volatility Foundation Volatility Framework 2.6.1
*****
ed01ebfbc9eb5b pId: 1068
Command line : "C:\Users\malware\Downloads\ed01ebfbc9eb5b-bea545af4d01b-f5f1071661840-480439c6e5babe8e080e41aa.exe"
*****

```

Base	Size	LoadCount	LoadTime	Path
0x0000000000000000	0x35a000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Users\malware\Downloads\ed01ebfbc9eb5b-bea545af4d01b-f5f1071661840-480439c6e5babe8e080e41aa.exe
0x00007fff90e0b0000	0x1f8000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Windows\SYSTEM32\ntdll.dll
0x00007fff90e020000	0x59000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\wow64.dll
0x00007fff90e030000	0x83000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\wow64cpu.dll
0x00000000770c0000	0xa000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\wow64cpu.dll
0x0000000000000000	0x35a000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Users\malware\Downloads\ed01ebfbc9eb5b-bea545af4d01b-f5f1071661840-480439c6e5babe8e080e41aa.exe
0x00000000770d0000	0x1a4000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Windows\SYSTEM32\ntdll.dll
0x0000000075600000	0xf0000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\KERNEL32.DLL
0x00000000750f0000	0x23a000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\USER32.dll
0x0000000074f50000	0x19d000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\USER32.dll
0x0000000075750000	0x18000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\USER32.dll
0x0000000075330000	0x23000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\GDI32.dll
0x0000000075360000	0xe6000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\GDI32Full.dll
0x0000000075ca0000	0x7b000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\msvc_p_wln.dll
0x0000000076310000	0x120000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\USER32.dll
0x0000000075e10000	0x7d000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\ADVAPI32.dll
0x0000000076e00000	0xbff000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\msvcrt.dll
0x0000000075e90000	0x78000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\sechost.dll
0x0000000075d00000	0xbcb000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\RPCRT4.dll
0x0000000076450000	0x19000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\bcrypt.dll
0x0000000075d20000	0x26000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\IMM32.dll
0x00000000725f0000	0x13000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\SYSTEM32\CRYPTSP.dll
0x0000000074f20000	0x2f000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\system32\rsaenh.dll
0x0000000072750000	0xa000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\SYSTEM32\CRYPTBASE.dll
0x00000000762b0000	0x3f000	0xffff	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\bcryptPrimitives.dll
0x0000000076710000	0x5d7000	0x6	2024-05-01 06:56:30 UTC+0000	C:\Windows\System32\SHELL32.dll
0x0000000074eb0000	0x70000	0x6	2024-05-01 06:56:31 UTC+0000	C:\Windows\SYSTEM32\MSVCPE06.dll
0x0000000074e80000	0x29000	0x6	2024-05-01 06:56:31 UTC+0000	C:\Windows\SYSTEM32\ntmarta.dll
0x00000000723c0000	0x9f000	0xffff	2024-05-01 06:56:32 UTC+0000	C:\Windows\system32\apphelp.dll
0x0000000073f80000	0x615000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\SYSTEM32\Windows.storage.dll
0x0000000076020000	0x280000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\System32\combase.dll
0x0000000072b00000	0x25000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\SYSTEM32\Wldp.dll
0x0000000076ec0000	0x96000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\System32\OLEAUT32.dll
0x0000000073450000	0x37000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\System32\SHCORE.dll
0x0000000076d50000	0x45000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\System32\shlwapi.dll
0x0000000072700000	0x21000	0x6	2024-05-01 06:56:32 UTC+0000	C:\Windows\SYSTEM32\SspiCli.dll
0x0000000074dc0000	0x74000	0x6	2024-05-01 06:57:05 UTC+0000	C:\Windows\system32\uxtheme.dll

Fig. 22. Details of DLL Files from ed01ebfbc9eb5b Process PID 1068

This analysis reveals several important points. First, the presence of a suspicious process with a long and unusual file name is likely part of the WannaCry ransomware infection. Second, the large number of DLL files loaded from the SYSTEM32 directory indicates that this process has extensive access to various core Windows functions, enabling it to perform potentially harmful system operations, including memory manipulation, kernel interaction, and user management.

Furthermore, the presence of DLLs such as wow64.dll, wow64win.dll, and wow64cpu.dll indicates that this process is running in 32-bit compatibility mode on a 64-bit operating system (WOW64). This is a common technique used by ransomware to evade detection and exploit weaknesses in the compatibility subsystem. The process also loads several cryptographic and security-related DLLs, such as bcrypt.dll, bcryptPrimitives.dll, and CRYPTSP.dll, suggesting the possible use of cryptographic functions to encrypt data or communicate with command-and-control (C2) servers. Additionally, DLLs like uxtheme.dll and IMM32.DLL indicate that this process might be attempting to manipulate the user interface, which could be used for activities such as information theft or further ransomware propagation through user interaction, as shown in Figure 23 below. The analysis of these DLL files confirms that the process ed01ebfbc9eb5b is part of the WannaCry ransomware infection.

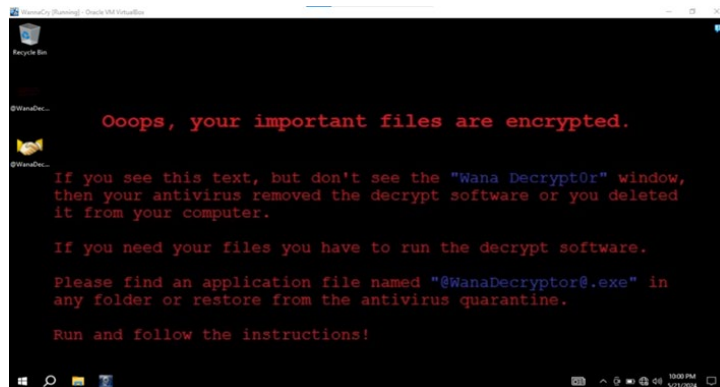


Fig. 23. Background Display on WannaCry Infected Virtual Machine

D. Ransomware Samples Categorization

After conducting the analysis, we can categorize which threats can be detected using memory analysis.

TABLE IV
 RANSOMWARE SAMPLES CATEGORIZATION

Ransomware Samples	Memory Analysis	
	Running Process	DLL Files
Jigsaw	X	X
Locky	X	X
WannaCry	√	√

Based on the data obtained from analysis, for Locky and Jigsaw, they could not be detected through memory analysis of running processes and DLL files. In both samples, only common processes typical of an operating system were displayed. In the case of WannaCry, it has been demonstrated that only WannaCry could be detected through the analysis memory of running processes and DLL files. This is evidenced by the presence of the suspicious process ed01ebfbc9eb5b. Additionally, the detailed examination of the DLL files loaded by this process further confirmed its malicious. Overall, the type of ransomware that can be detected through memory analysis involves those that exhibit suspicious activities identifiable in running processes and load specific DLLs related to data encryption or communication with command-and-control (C2) servers.

In [8], the research focuses on Windows 7 as the primary platform for conducting memory analysis related to malware. In this research, Windows 10 was selected as the primary platform, given its status as a current operating system with a broad user base, ensuring that the analysis results are highly relevant and applicable to the security threats encountered by modern users.

E. Computer System Performance

Before an infection, CPU and memory usage is usually stable and directly related to user activity and system services. CPU usage spikes when heavy applications or processes are run but returns to normal levels when activity decreases. One of the signs of a ransomware infection is a sudden and sustained spike in CPU usage. Ransomware utilizes the CPU to encrypt a large number of files and carry out other malicious activities, such as connecting to

Command and Control (C2) servers. Ransomware may run hidden processes or use obfuscation techniques to avoid detection, leading to unusual memory usage that is difficult to explain based on previous usage patterns. A significant increase in memory utilization and CPU usage was observed when comparing system performance before and after the ransomware infection. This indicates that ransomware infection substantially impacts the system's resources, as shown in Table V.

TABLE V
 MEMORY UTILIZATION BEFORE AND AFTER RANSOMWARE INFECTION

No	Type of Ransomware	Memory Before Infection	Memory After Infection
1	Jigsaw	4.5 GB	5.6 GB
2	Locky	4.5 GB	5.4 GB
3	WannaCry	4.5 GB	6.2 GB

Based on the memory utilization data before and after the malware infection, the comparison can be seen in the graph in Figure 24.

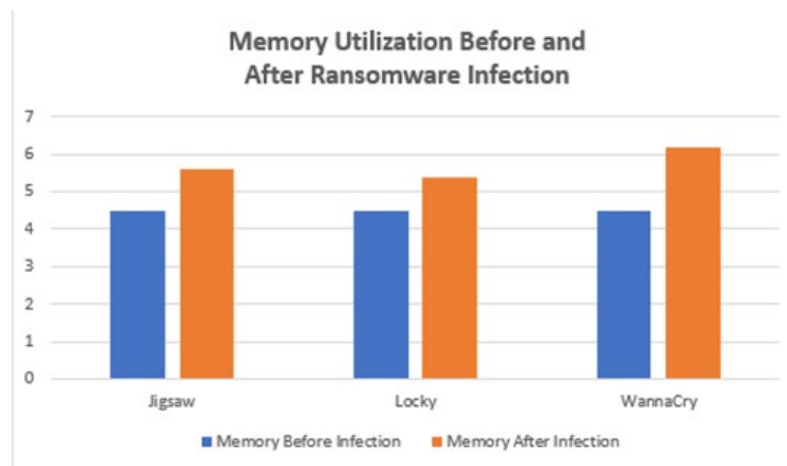


Fig. 24. Memory Utilization Before and After Ransomware Infection

Examining the computer's memory performance before and after the ransomware infection reveals an increase in CPU usage. Consequently, the computer fell fast before the infection but experienced significantly longer loading times after becoming infected. This is evident in Table VI and Figure 25.

TABLE VI
 CPU USAGE BEFORE AND AFTER RANSOMWARE INFECTION

No	Type of Ransomware	CPU Before Infection	CPU After Infection
1	Jigsaw	20%	72%
2	Locky	18%	69%
3	WannaCry	23%	83%

Based on the memory utilization data before and after the malware infection, the comparison can be seen in the graph in Figure 25.

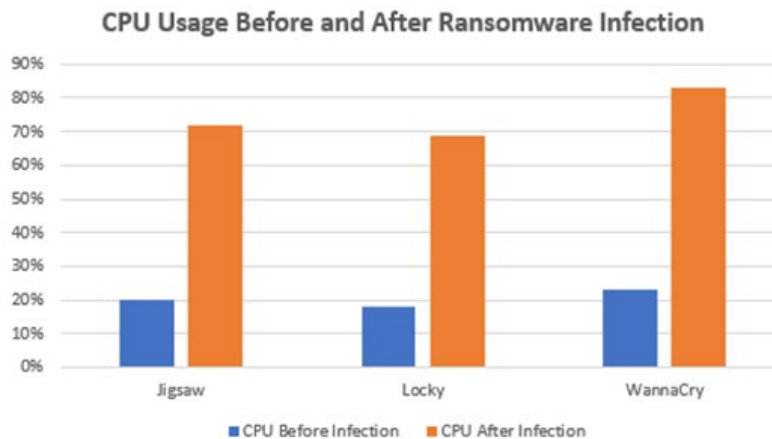


Fig. 25. CPU Usage Before and After Ransomware Infection

IV. CONCLUSION

Analysis of the system and user processes in the three memory images shows that many legitimate system processes and load a large number of DLL files essential for core Windows operations. These processes handle various critical functions, including memory management, security, cryptography, and inter-process communication. In the WannaCry.mem a suspicious process with an unusual name, such as ed01ebfbc9eb5b, was identified, indicating a malware infection.

The svchost.exe processes in all memory images load various essential DLLs, indicating that they handle core Windows functions, including ntdll.dll, KERNEL32.DLL, advapi32.dll, and RPCRT4.dll. The presence of DLLs such as audiosrv.dll and MMDevAPI.DLL in Jigsaw.mem show audio services, while modules like umpo.dll and umpnpgmgr.dll in Locky.mem indicate power management and Plug and Play.

Memory analysis has proven to be an effective tool for identifying and understanding system activities and detecting ransomware infections. By identifying the processes and loaded DLL modules, more precise mitigation steps can be taken to protect the system from further threats. Future research could explore a broader range of ransomware samples and utilize alternative plugins available in Volatility. This would provide a more comprehensive understanding of different ransomware behaviors and enhance the effectiveness of memory analysis in detecting and mitigating such threats.

REFERENCES

- [1] D. Y. Huang et al., "Tracking Ransomware End-to-end," in 2018 IEEE Symposium on Security and Privacy (SP), IEEE, May 2018, pp. 618–631. doi: 10.1109/SP.2018.00047.
- [2] B. A. Khalaf et al., "An Adaptive Protection of Flooding Attacks Model for Complex Network Environments," Security and Communication Networks, vol. 2021, pp. 1–17, Apr. 2021, doi: 10.1155/2021/5542919.
- [3] Z. K. Maseer, R. Yusof, S. A. Mostafa, N. Bahaman, O. Musa, and B. Ali Saleh Al-rimy, "DeepIoT.IDS: Hybrid Deep Learning for Enhancing IoT Network Intrusion Detection," Computers, Materials & Continua, vol. 69, no. 3, pp. 3945–3966, 2021, doi: 10.32604/cmc.2021.016074.
- [4] A. Tandon and A. Nayyar, "A Comprehensive Survey on Ransomware Attack: A Growing Havoc Cyberthreat," in Advances in Intelligent Systems and Computing, vol. 839, Springer Verlag, 2019, pp. 403–420. doi: 10.1007/978-981-13-1274-8_31.
- [5] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques," Wirel Pers Commun, vol. 112, no. 4, pp. 2597–2609, Jun. 2020, doi: 10.1007/s11277-020-07166-9.
- [6] A. Zimba and M. Chishimba, "On the Economic Impact of Crypto-ransomware Attacks: The State of the Art on Enterprise Systems," European Journal for Security Research, vol. 4, no. 1, pp. 3–31, Apr. 2019, doi: 10.1007/s41125-019-00039-8.
- [7] I. A. Chesti, M. Humayun, N. U. Sama, and N. Z. Jhanjhi, "Evolution, Mitigation, and Prevention of Ransomware," in 2020 2nd International Conference on Computer and Information Sciences, ICCIS 2020, Institute of Electrical and Electronics Engineers Inc., Oct. 2020. doi: 10.1109/ICCIS49240.2020.9257708.
- [8] R. Sihwail, K. Omar, and K. A. Z. Ariffin, "An Effective Memory Analysis for Malware Detection and Classification," Computers, Materials and Continua, vol. 67, no. 2, pp. 2301–2320, 2021, doi: 10.32604/cmc.2021.014510.
- [9] M. Alam, S. Sinha, S. Bhattacharya, S. Dutta, D. Mukhopadhyay, and A. Chattopadhyay, "RAPPER: Ransomware Prevention via Performance Counters," Apr. 2020, [Online]. Available: <http://arxiv.org/abs/2004.01712>
- [10] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," Journal of Information Security and Applications, vol. 40, pp. 44–51, Jun. 2018, doi: 10.1016/j.jisa.2018.02.008.
- [11] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-Defined Networking-based Crypto Ransomware Detection Using HTTP Traffic Characteristics,"
- [12] T. R. Reshmi, "Information security breaches due to ransomware attacks - a systematic literature review," International Journal of Information Management Data Insights, vol. 1, no. 2, Elsevier Ltd, Nov. 01, 2021. doi: 10.1016/j.jjimei.2021.100013.
- [13] M. Weninger, P. Grünbacher, E. Gander, and A. Schörgenhuber, "Evaluating an Interactive Memory Analysis Tool: Findings from a Cognitive Walkthrough and a User Study," Proc ACM Hum Comput Interact, vol. 4, no. EICS, Jun. 2020, doi: 10.1145/3394977.
- [14] V. R. Sali and H. K. Khanuja, RAM Forensics: The Analysis and Extraction of Malicious processes from memory Image using GUI based Memory Forensic Toolkit. 2018.
- [15] R. Sihwail, K. Omar, K. A. Z. Ariffin, and S. Al Afghani, "Malware detection approach based on artifacts in memory image and dynamic analysis," Applied Sciences (Switzerland), vol. 9, no. 18, Sep. 2019, doi: 10.3390/app9183680.

- [16] S.Poudyal, K. P. Subedi, and D. Dasgupta, A Framework for Analyzing Ransomware using Machine Learning. 2018.
- [17] I. Kara, "A Basic Malware Analysis Method," Computer Fraud & Security, 2019.
- [18] J. Kävrestad, M. Birath, and N. Clarke, "Memory Analysis Tools," 2024, pp. 211–219. doi: 10.1007/978-3-031-53649-6_19.
- [19] A. Singh, S. Taterh, and U. Mitra, "An Efficient Tactic for Analysis and Evaluation of Malware Dump File Using the Volatility Tool," 2023, pp. 457. doi: 10.1007/s42979-023-01844-8.
- [20] F. Freiling, T. Grob, T. Muller, and R. Palutke, "Advances in Forensic Data Acquisition," 2018, pp. 63-74. doi: 10.1109/MDAT.2018.2862366.