Project Progress Report on

OPEN-SOURCE MALWARE ANALYSIS TOOLKIT: A REVIEW AND COMPARISON

Submitted in partial fulfillment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

Submitted by:

Gaurangi Tripathi **2014657**

Under the Guidance of
Ms. Meenakshi Maindola
Assistant Professor

Project Team ID: MP22CSE08

Project Progress Report No: 2

Department of Computer Science and Engineering
Graphic Era (Deemed to be University)
Dehradun, Uttarakhand
November - 2022

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the Synopsis entitled **"OPEN-SOURCE MALWARE ANALYSIS TOOLKIT: A REVIEW AND COMPARISON"** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering in the Department of Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun shall be carried out by the undersigned under the supervision of **Ms. Meenakshi Maindola, Assistant Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Gaurangi Tripathi 2014657

The above-mentioned students shall be working under the supervision of the undersigned on the **"OPEN-SOURCE MALWARE ANALYSIS TOOLKIT: A REVIEW AND COMPARISON"**

Supervisor **Head of the Department**

Internal Evaluation (By DPRC Committee)

Status of the Synopsis: Accepted / Rejected

Any Comments:

Work Satisfactory: Yes / No

Table of Contents

Chapter 1

Introduction and Problem Statement

1.1 Introduction

Cyber dangers and attacks are possibly developing quickly and are getting worse every day. Malware, often known as malicious software, is a term for a program or piece of software that poses a risk to a system. It could be binary or executable in nature. Malware analysis is the process of identifying and tracking a malware infection's movements within a system. The knowledge gained from malware analysis aids in creating robust but efficient defense software against the malware species chosen for analysis. In terms of gross value, the market for malware analysis is growing quickly every day. Security experts use a variety of tools and techniques to analyze malware to help develop malware detection systems. In this article, we'll review some of the best malware analysis tools on the market and see exactly how they work. Malware analysis tools are not all the same. Some tools necessarily require a higher level of expertise, whereas others can provide an elevated analysis automatically. However, there are a few crucial elements to look out for. Most of the malware analysis methods used today are labor-intensive manual processes that take a lot of time. Better practices are therefore being tried with the use of various technologies in order to create effective procedures.

1.2 Problem Statement

Most of the malware analysis methods used today are labor-intensive manual processes that take a lot of time and human effort. Better practices are therefore being tried with the use of various technologies and automated tools in order to create effective procedures. In this project, we'll examine in-depth the malware forensics tools that are now available and will draw different kinds of inferences from them. In this project, we will be dealing with tools like Cuckoo sandbox, Flare VM, *YARA* Rules, and Python libraries to perform some basic analysis of malware. The number of simple yet complex tools is growing and with it the sophistication, persistence, and unawareness of the new generation of cyber threats and attacks. Unlike conventional malware, which was broad, known, open, and one-time, advanced malware is targeted, unknown, stealthy, personalized, and zero days. Once inside, they disarm host defenses and conceal, replicate, and multiply.

Chapter 2

Objectives

- **An overview of techniques for performing malware analysis:** The project provides you with a detailed yet condensed method for performing malware analysis, along with the necessary tools. There are approaches, but none of them give an analyst a step-by-step manual.
- **To conclude a list of tools for different phases:** In this project, you can find a list of manual and automated tools for malware analysis and steps involved while investigating malware such as detection, packet analysis, website analysis, and other tasks in the project.
- **A detailed dig into malware analysis:** The analysis of malware, its types, detection, etc., has been the subject of numerous studies and research projects. However, none of them offer step-by-step instructions for carrying out malware analysis. So, here we are aiming to get an idea of what one can start with.
- **To conclude an appropriate methodology and tools:** The project conclusions can be used for various motives involved in the related field. Methodologies and tools may differ with the aim or investigation of malware, whether it heads for malware detection, analysis, combat measures, etc.
- **Other key objectives** regarding tool analysis that are to be considered are to find a hybrid solution, that continues protection when the device is disconnected, provides user and entity behavior analytics, anomaly detection, zero-day virus blocking, etc.

Chapter 3

Project Work Carried Out

Users frequently use various analysis techniques to understand the operation and characteristics of malware, as well as to assess its impact on the system. The classification of these analysis techniques is as follows:

- **Static analysis** is the procedure of looking over a binary without running it. The suspect binary's information can be extracted using this method, which is the easiest to use. Even though static analysis may not always show all of the necessary forms, it occasionally offers fascinating data that can guide your choice of where to concentrate your following analysis efforts.
- **Dynamic analysis (Behavioral Analysis):** Running the suspect binary in a supervised environment and watching how it behaves is what this procedure entails. This analysis method is easy to use and offers helpful details on the activities of the binary during execution. Although this analysis method is helpful, it does not expose all of the hostile program's capabilities.
- **Code analysis** focuses on examining the code in order to understand how the binary operates. There are two forms of code analysis: static code analysis and dynamic code analysis. While dynamic code analysis involves a controlled debugging of the suspect binary to understand its operation, static code analysis requires disassembling the suspect binary and analyzing the code to understand the behavior of the program.
- **Memory analysis** (also known as memory forensics) is the procedure of looking through the RAM of the computer for forensic evidence. It is often a forensic technique, but integrating it into our analysis of malware may help you better understand how the threat acts once it has been set up.

Before heading towards our future study, it is important to get along with phase 1 of the project, i.e., word done till now. In our initial study, we started off with the static analysis of the malware, whose steps included the following steps, File Type Determination, Malware Fingerprinting, Anti-virus Scanning, String Extraction, Detecting File Obfuscation, PE Header Inspection, and Malware Classification. Out of these, above-listed steps, we have successfully performed six steps and are left with the last step – Malware Classification. Taking our study forward, we will be starting with carrying out Malware Classification using *YARA* Rules.

Static Analysis - Classification of Malware using *YARA*

YARA is a powerful tool used to identify and classify malware by searching for patterns in files, network traffic, and other data sources. It uses rules written in simple language to define specific characteristics of malware, which can include file names, sizes, strings of code, and network traffic patterns. Once the rules have been written, they can be used to scan for malware in real time and trigger alerts or take other actions to protect the system. *YARA* can be integrated with other security systems, such as intrusion detection systems or firewalls, to provide comprehensive protection against cyberattacks. Some key features contributing to the selection of *YARA* Rules as a tool for our study are:

- *YARA* rules are highly customizable, allowing security professionals to create rules specific to their organization's needs.
- *YARA* is fast and efficient, allowing for real-time detection and classification of malware.
- *YARA* is an open-source tool, meaning that it is free to use and can be modified by anyone to fit their specific needs.

```
rule suspicious_strings
{
strings:
$a = "2(3]3b3h3r3z3"
$b = "Portscanner"
$c = "Keylogger"
condition:
($a or $b or $c)
}
```

Figure 1: Contents of sus.yara file

From the Figure 1, it is clearly seen that we have written YARA Rules, mentioning some strings. These strings are part of those, which were extracted from String Extraction step using FLOSS. After executing this rule, we will be able to see number of files in our system which show similar behavior as that of our suspected binary, as shown in Figure 2.

```
[redacted]:~/Desktop/malware$ yara -r sus.yara sam
suspicious_strings samples//854137.exe
suspicious_strings samples//0.exe
```

Figure 2: Output of YARA file upon execution

Now, that we have already performed static analysis, which involved different steps of analysis, and hence after having analyzed the sample through its various attributes, we will proceed to study the sample by running the sample in a controlled environment. For the same purpose,

i.e., dynamic analysis of suspected binary, we will be setting up our lab with following specifications:

- Host machine: Ubuntu 20.04 LTS
- Virtualization Software: VirtualBox 7.0
- Windows Virtual Machine (Windows 10): VM in which the sample is executed.
- Linux Virtual Machine (Ubuntu 16.04 LTS): VM used for monitoring anomalies in system and network logs.
- FTK Imager – software to be used for in-memory analysis.

PART A: Dynamic Analysis

Dynamic analysis (behavioral analysis) involves running a sample in a controlled environment and monitoring its interactions, activities, and effects on the system to understand the characteristics, functioning, and use of the questionable binary. Malware can interact with a system in many ways, such as launching child processes, dropping files, creating registry keys, downloading components, and accepting commands from a control server. Monitoring these interactions during execution helps in understanding the nature and function of the malware. Dynamic analysis involves several monitoring tasks to achieve these objectives.

The goal is to acquire real-time information on malware activity and how it affects the system. Several forms of monitoring used during dynamic analysis are shown in the following list:

1. **Process monitoring** involves keeping track of each process's activities and looking at the properties of the finished malware process.
2. **File system monitoring,** includes keeping an eye on the behavior of the file system in real-time as malware is executed.
3. **Registry monitoring** involves keeping an eye on the registry keys that the malicious code accesses, modifies, and reads/writes.
4. **Monitoring the live traffic** to and from the system while malware is being executed on the network.

Various kinds of open-source monitoring tools have been used in order to thoroughly analyze the binary sample when it is executed.

1) Process Inspection with Process Hacker

Process Hacker is a free and open-source process viewer and manager for Windows-based operating systems. It allows users to inspect and manage processes running on their system,

as well as monitor system performance and resource usage. For our course of study, we will follow the following steps for process inspection with *Process Hacker*:

- **Launch Process Hacker:** Launching Process Hacker can be done by downloading the tool from the official website and running the executable file.
- **Inspect processes:** Once Process Hacker is launched, it will display a list of all the processes running on the system. We can inspect each process by clicking on its name in the list, which will display detailed information about the process in a separate window.
- **Monitor system performance:** In addition to process inspection, users can use Process Hacker to monitor system performance and resource usage. The tool provides real-time data on CPU usage, memory usage, network activity, and more.
- **Manage processes:** We can also use Process Hacker to manage processes, including terminating, suspending, and resuming them. This can be useful in situations where a particular process is causing system instability or resource usage issues.
- **Customize output:** We can customize the output of Process Hacker by choosing which columns to display and how to sort the data. This can be useful in situations where users want to focus on specific aspects of system performance or process activity.

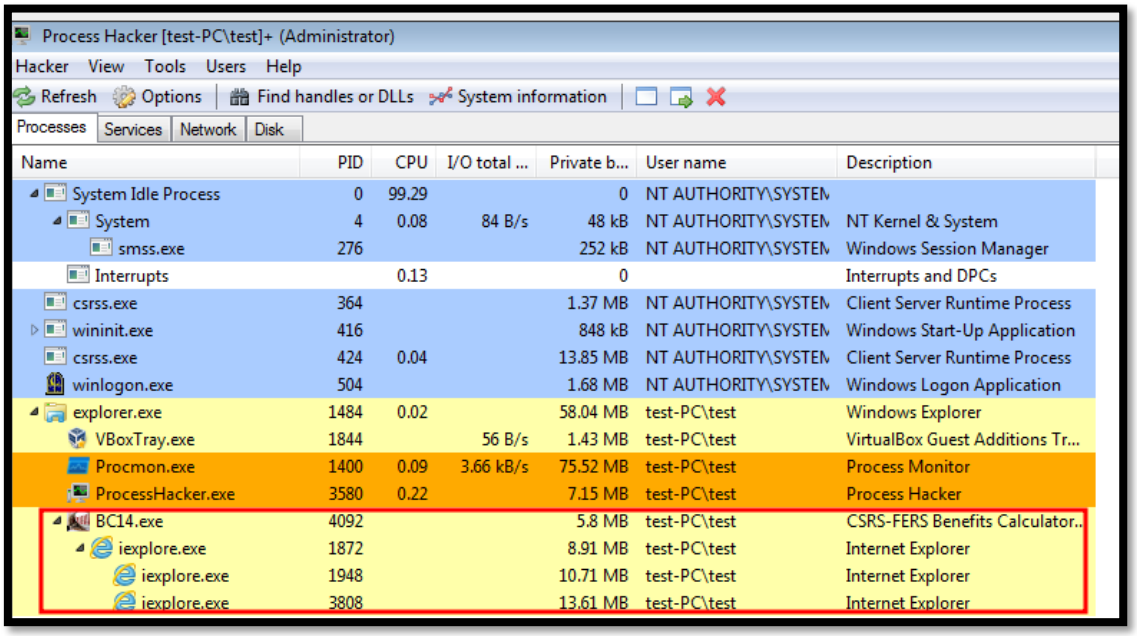

Figure 3: Process Hacker highlighting the processes being run by suspected file

As seen in Figure 3, it is clearly seen that our suspected binary file has caused the highlighted items to instantiate and execute on the machine. Obtaining the list of such applications may help the analyst to study the initial capabilities of the suspected binary.

2) Determining System Interaction with Process Monitor

Process Monitor is a powerful tool for monitoring and analyzing system activity on Windows-based computers. It is capable of capturing events related to file system activity, registry activity, network activity, and process activity, among others. In this article, we will explore how to use Process Monitor to determine system interaction with different processes.

The steps included in determining system interaction with the Process Monitor are:

- **Launch Process Monitor:** The first step is to launch Process Monitor. This can be done by downloading the tool from the Microsoft website and running the executable file.
- **Start capturing data:** Once Process Monitor is launched, it will start capturing data in real time. By default, the tool captures all system activity, but users can customize the capture settings by clicking on the Filter icon in the toolbar.
- **Analyze the captured data:** As the data is captured, it will be displayed in the main window of Process Monitor. Users can analyze the data by scrolling through the events or by using the search and filter capabilities to find specific events.
- **Identify system interactions:** To determine system interaction, users should focus on events related to file and registry access, network activity, and process and thread activity. For example, if a user suspects that a particular application is causing system slowdowns, they can use Process Monitor to monitor the activity of that application and see if there are any interactions with other system components that might be causing the issue.
- **Act:** Once system interactions have been identified, users can take action to resolve any issues or conflicts. For example, if a particular application is causing system slowdowns, the user might choose to uninstall the application or adjust its settings to reduce its impact on system performance.

On running Process Monitor, after executing the suspected binary, as administrator of machine, we will immediately notice that it captures all the system events, as shown in the following screenshot. To stop capturing the events, you can press Ctrl + E, and to clear all the events we can press Ctrl+ X. The Figure 4 shows the activities captured by Process Monitor on a clean system.

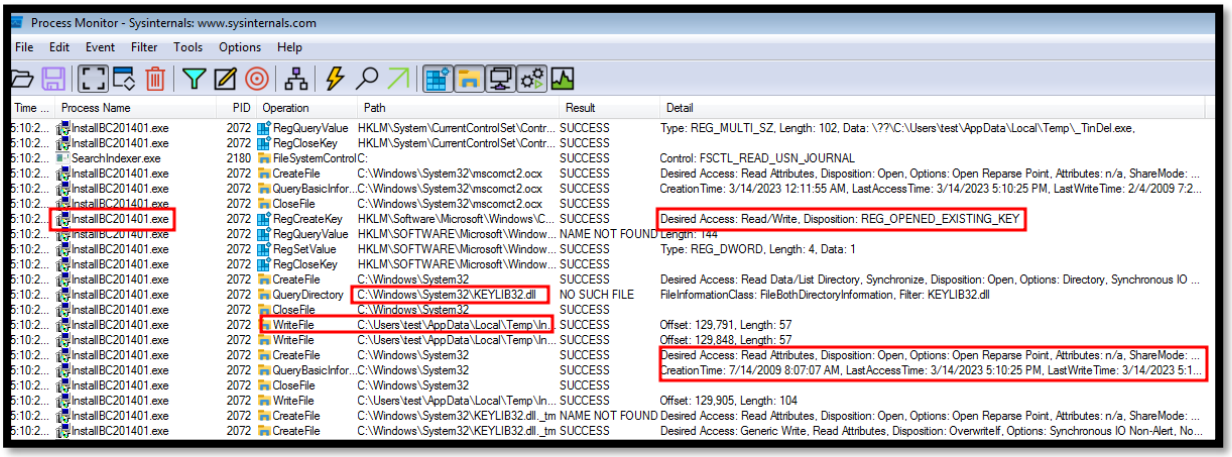

Figure 4: Process Monitor monitoring activities carried out by suspected binary

3) Capturing Network Traffic with Wireshark

Wireshark is a free and open-source network protocol analyzer that allows users to capture and analyze network traffic in real time. It is a powerful tool for network troubleshooting and analysis, as well as network security and monitoring. In this article, we will discuss how to use Wireshark to capture network traffic and some of the key features and functionality of the tool.
Features of Wireshark:

- **Real-time network traffic capture:** Wireshark allows users to capture network traffic in real-time, providing detailed information about each packet sent and received on the network.
- **Protocol analysis:** The tool provides detailed protocol analysis, including decoding each protocol layer, allowing users to identify issues and troubleshoot network problems.
- **Customizable display:** Users can customize the display of captured network traffic, including choosing which columns to display, applying filters, and more.
- **Network security:** Wireshark can also be used for network security and monitoring, allowing users to detect suspicious network activity and potential security threats.

Capturing Network Traffic using Wireshark:

- **Download and install Wireshark:** The first step is to download and install Wireshark from the official website.
- **Launch Wireshark:** Once Wireshark is installed, launch the tool and select the network interface that you want to capture traffic on.

- **Start capturing network traffic:** To start capturing network traffic, click on the "Start" button in the main window of Wireshark. The tool will begin capturing all network traffic on the selected interface.
- **Analyze network traffic:** Once network traffic is captured, users can analyze the captured data by reviewing the packets in the main window of Wireshark. The tool provides detailed information about each packet, including the source and destination IP addresses, protocol information, and more.
- **Customize display:** We can customize the display of captured network traffic by using filters, choosing which columns to display, and more. This can be useful in situations where users want to focus on specific types of network traffic or events.
- **Save and share captured data:** We can save the captured network traffic to a file for later analysis or share it with other team members for collaborative analysis.

When the malware is executed, we want to capture the network traffic generated as a result of running the malware; this will help us to understand the communication channel used by the malware and will also help in determining network-based indicators. Wireshark is a packet sniffer that allows you to capture network traffic. As we had set up Windows VM and Linux VM with the same IP address, we will capture all the communication of the suspected binary through Wireshark on Windows VM. These captured packets will give insights about network traffic volume, protocol usage, source and destination address, packet timing, packet contents, etc., as seen in Figure 5.

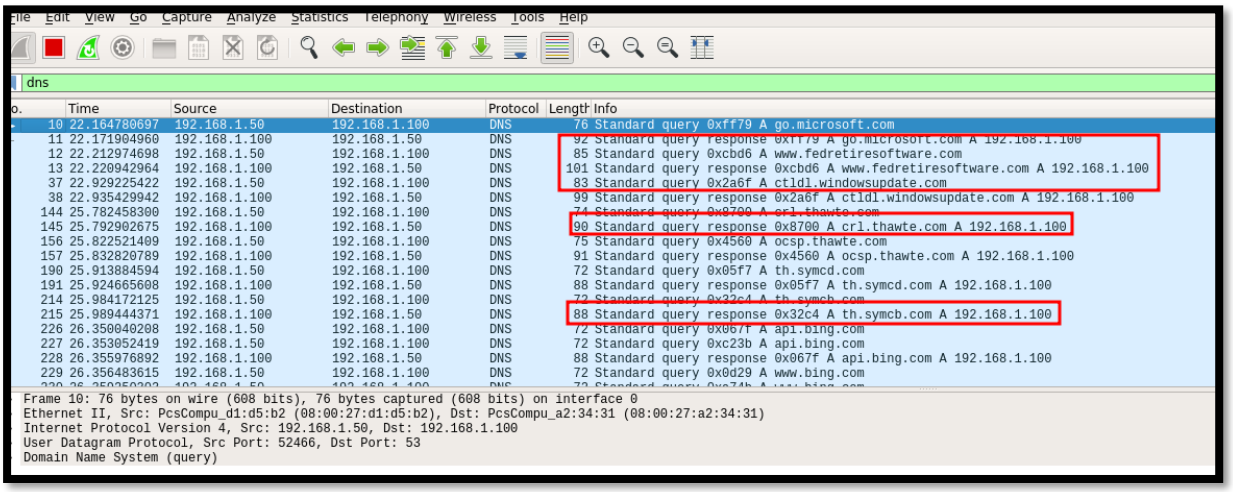

Figure 5: Wireshark capturing network traffic details

4) Simulating Services with INetSim

INetSim is a free and open-source tool for simulating common internet services such as HTTP, DNS, and SMTP. It is designed to provide a safe and controlled environment for testing and

experimenting with various network security tools and techniques. In this article, we will discuss how to use INetSim to simulate services and some of the key features and functionality of the tool.

In our study, we will follow the following steps to make use of INetSim:

- **Download and install INetSim:** The first step is to download and install INetSim from the official website.
- **Configure INetSim:** Once installed, users can configure INetSim by modifying the configuration file, which is located in the INetSim installation directory.
- **Start INetSim:** Once configured, users can start INetSim by running the inetd.py script in the INetSim installation directory.
- **Connect to simulated services:** Users can connect to the simulated services by pointing their client applications to the IP address and port number specified in the INetSim configuration file.
- **Customize service behavior:** Users can customize the behavior of the simulated services by modifying the configuration file, including response times, error messages, and more.
- **Analyze traffic:** Once traffic is generated, users can analyze the traffic using various network security tools and techniques, including packet sniffers, intrusion detection systems, and more.

INetSim can analyze network traffic, analyze malware behavior, and payload, and, generate signatures that can be used to detect the malware on other systems. In the above figure, INetSim is simulating a request from the sample that we executed that would otherwise be displayed on a web browser after connection with a remote server, and be able to extract and download malicious content.

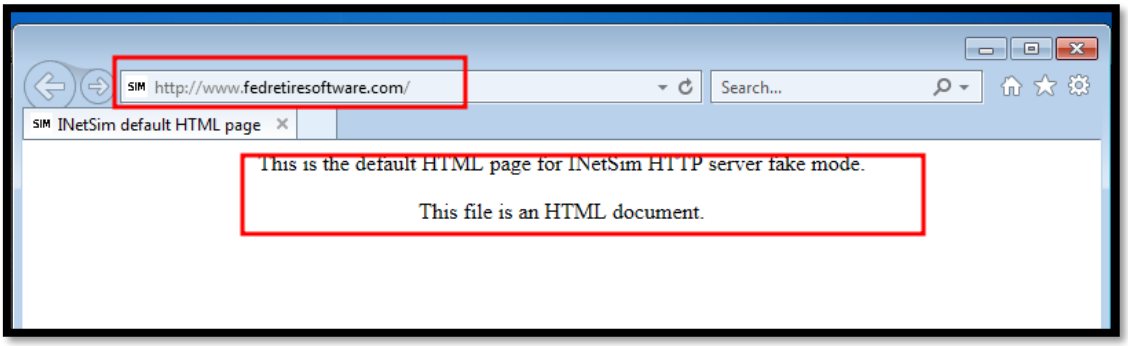

Figure 6: Simulating services using INetSim

As the number of malware attacks increases exponentially, malware analysis has become an essential aspect of cybersecurity. Manual malware analysis can be time-consuming and labor-intensive. However, Automating Malware Analysis(sandbox) can help automate the process.

PART B: In-Memory Analysis

Digital investigation and malware analysis frequently rely on memory analysis, which involves examining a dumped memory image from a targeted machine following the execution of malware. Through this process, numerous artifacts can be obtained, such as the process list and its corresponding threads, networking information and interfaces (TCP/UDP), kernel modules (including those that are concealed), Bash and command history, system calls, kernel hooks, and more. This phase is critical because gaining a better understanding of the malware's capabilities is always beneficial. To perform memory analysis, usually we go through 2 major steps:

- Memory Acquisition
- Memory Analysis

1) **Memory Acquisition using FTK Imager:** Memory acquisition refers to the process of collecting a memory image from a device, such as a computer or mobile device. This process involves creating a snapshot of the device's volatile memory, which contains the running processes, network connections, open files, and other system information. The goal of memory acquisition is to obtain a complete and accurate snapshot of the device's memory, without altering or damaging any of the data.

   FTK Imager is a forensic imaging tool that can be used to acquire various types of digital evidence, including memory images. Memory acquisition using FTK Imager involves using the FTK Imager software to create a snapshot of the device's volatile memory. To acquire a memory image using FTK Imager, we have installed the software on a separate device from the one being acquired. Once installed, the software can be used to create a bootable USB drive or CD that can be used to boot the device being acquired. After booting the device with the FTK Imager bootable drive, the software can be used to acquire a memory image. This is done by selecting the "*Capture Memory*" option in the FTK Imager interface and specifying the location where the memory image should be saved, as shown in Figure 7.

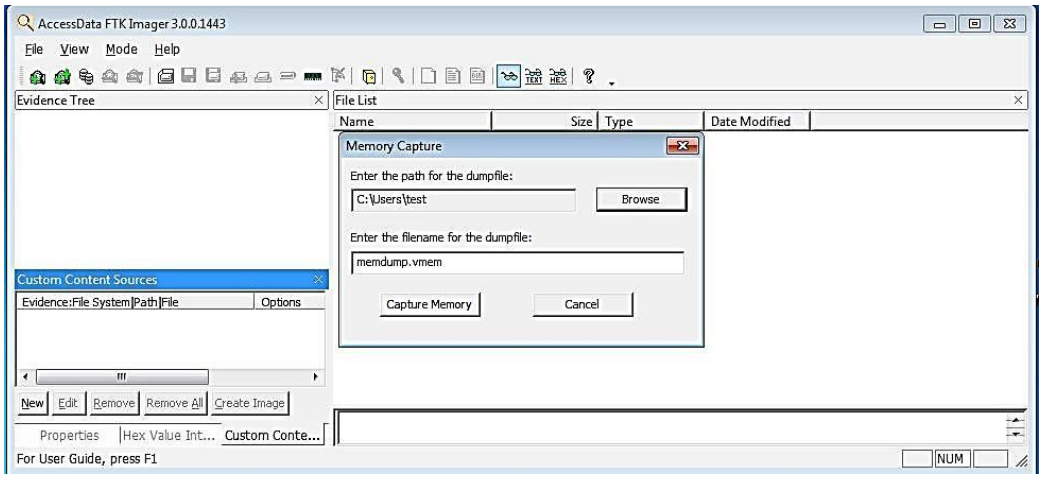

Figure 7: Setting file path and file name of captured memory dump file

FTK Imager can acquire memory images from both live systems and hibernation files. The acquired memory image can then be analyzed using forensic analysis tools to extract valuable information for digital investigation and malware analysis.

2) Memory analysis:

Memory analysis is the process of examining a memory image of a device, such as a computer or mobile device, to extract valuable information for digital investigation and malware analysis. Memory analysis involves analyzing the contents of the device's volatile memory, which includes the running processes, network connections, open files, and other system information. Memory analysis can be used to identify malware infections, determine the extent of an attack, and gather evidence for forensic investigations. The analysis process involves examining the memory image for suspicious or malicious activity, such as the presence of malware code, network connections to known malicious servers, or evidence of data exfiltration. Various memory analysis tools are available to aid in the analysis process, including open-source tools like Volatility and Rekall, and commercial tools like FTK Imager and Encase. These tools can be used to extract information from the memory image, such as process lists, network connections, registry keys, and other artifacts that can provide insight into the device's activity and the presence of any malicious activity.

The command "*vol.py -f memdump.vmem imageinfo*" is first run on our Linux VM and is used with the Volatility memory forensics framework, which is used to analyze the memory dump file and provides details such as the operating system version, the architecture (32-bit or 64-bit), and the service pack level. In Figure 8, it can be clearly seen volatility gives initial information about the operating system's kernel, such as the kernel version, build

number, memory layout, physical address extension, number of processors, etc. based on KDBG, which provides a starting point for analyzing the kernel memory.

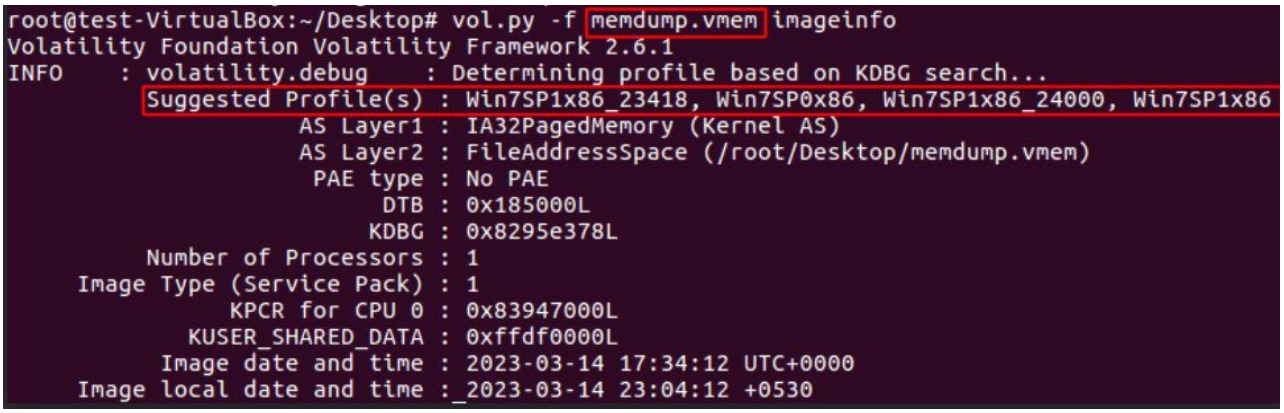

Figure 8: Taking image information of memory dump

As seen in Figure 8, some profiles are suggested, which are a list of memory analysis profiles that are recommended for analyzing a given memory dump file. For our further analysis, we will be studying '*Win7SP0x86'* profile using manual Linux commands, as shown in Figure 9. Starting with the command "*vol.py -f memdump.vmem --profile=Win7SP0x86 pslist*", which is used in the Volatility memory forensics framework to list the running processes in a memory dump file

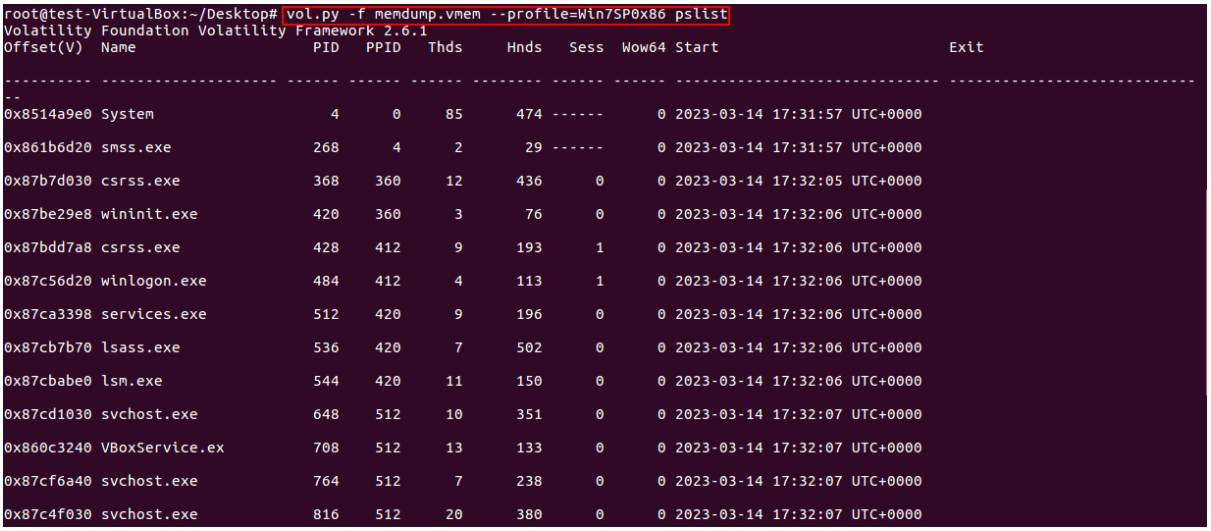

Figure 9: Taking running process list and their related information at the time of memory dump

The output will display the process ID (PID), parent process ID (PPID), process name, and other details for each running process in the memory dump file as seen in Figure 9.

Further, the command "*vol.py -f memdump.vmem --profile=Win7SP0x86 pstree*" can used in the Volatility memory forensics framework to display a tree-like structure of the running processes in a memory dump file.

One important aspect that can contribute to our investigation is DLL(Dynamic Link Libraries), which are capable of providing information about the behavior and capabilities of malware. Moreover, DLL can help us in detecting code injection, identify malicious DLLs, understand malware behavior etc. We will use DLLs for our further analysis, using manual method, as shown in Figure 10.

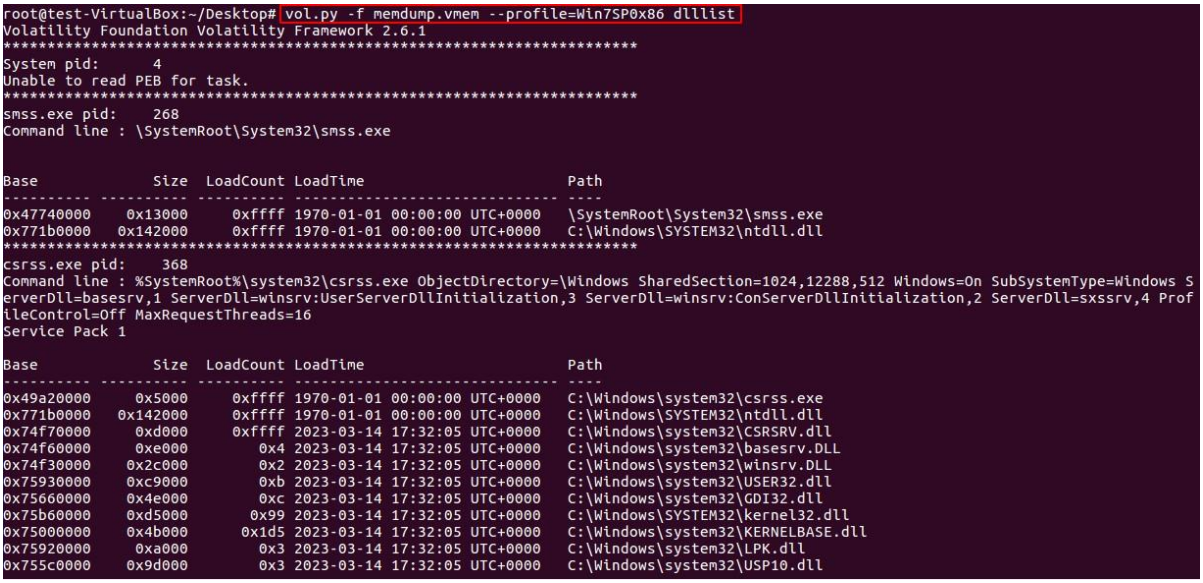

Figure 10: Obtaining list of loaded DLLs in memory dump

The command "*vol.py -f memdump.vmem --profile=Win7SP0x86 dlllist*" is used in the Volatility memory forensics framework to list the loaded DLLs (Dynamic Link Libraries) in a memory dump file. The above-used commands only provide information about all the running processes themselves and not about the command line processes. For this purpose, '*cmdline'* plugin is used, as shown in Figure 11.

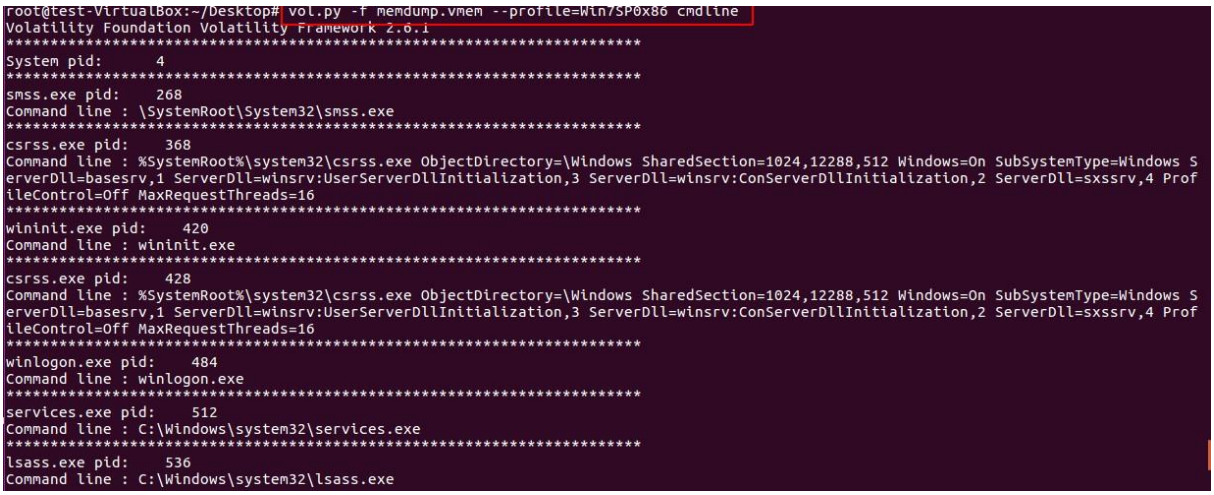

Figure 11: Fetching information of running command line processes

The command "*vol.py -f memdump.vmem --profile=Win7SP0x86 cmdline*" is used to extract information about the commands that were executed on the system at the time the memory dump was created.

We can have a closer view of the memory dump file by extracting kernel-loaded files using "*vol.py -f memdump.vmem --profile=Win7SP0x86 modules*" command in the Linux terminal.

```
root@test-VirtualBox:~/Desktop# vol.py -f memdump.vmem --profile=Win7SP0x86 modules
Volatility Foundation Volatility Framework 2.6.1
Offset(V)  Name                 Base             Size File
---------- -------------------- ---------- ---------- ----
0x85147c98 ntoskrnl.exe         0x82835000   0x40f000 \SystemRoot\system32\ntoskrnl.exe
0x85147c20 hal.dll              0x8280d000    0x28000 \SystemRoot\system32\halacpi.dll
0x85147ba0 kdcom.dll            0x80b95000     0x8000 \SystemRoot\system32\kdcom.dll
0x85147b20 mcupdate.dll         0x8c43e000    0x85000 \SystemRoot\system32\mcupdate_GenuineIntel.dll
0x85147aa0 PSHED.dll            0x8c4c3000    0x11000 \SystemRoot\system32\PSHED.dll
0x85147a20 BOOTVID.dll          0x8c4d4000     0x8000 \SystemRoot\system32\BOOTVID.dll
0x851479a8 CLFS.SYS             0x8c4dc000    0x42000 \SystemRoot\system32\CLFS.SYS
0x85147930 CI.dll               0x8c51e000    0x68000 \SystemRoot\system32\CI.dll
0x851478b0 Wdf01000.sys         0x8c586000    0x81000 \SystemRoot\system32\drivers\Wdf01000.sys
0x85147830 WDFLDR.SYS           0x8c607000     0xe000 \SystemRoot\system32\drivers\WDFLDR.SYS
0x851477b8 ACPI.sys             0x8c615000    0x48000 \SystemRoot\system32\drivers\ACPI.sys
0x85147738 WMILIB.SYS           0x8c65d000     0x9000 \SystemRoot\system32\drivers\WMILIB.SYS
0x851476b8 msisadrv.sys         0x8c666000     0x8000 \SystemRoot\system32\drivers\msisadrv.sys
0x85148008 pci.sys              0x8c66e000    0x2a000 \SystemRoot\system32\drivers\pci.sys
0x85148f88 vdrvroot.sys         0x8c698000     0xb000 \SystemRoot\system32\drivers\vdrvroot.sys
0x85148f08 partmgr.sys          0x8c6a3000    0x11000 \SystemRoot\System32\drivers\partmgr.sys
0x85148e88 compbatt.sys         0x8c6b4000     0x8000 \SystemRoot\system32\DRIVERS\compbatt.sys
0x85148e08 BATTC.SYS            0x8c6bc000     0xb000 \SystemRoot\system32\DRIVERS\BATTC.SYS
0x85148d88 volmgr.sys           0x8c6c7000    0x10000 \SystemRoot\system32\drivers\volmgr.sys
0x85148d08 volmgrx.sys          0x8c6d7000    0x4b000 \SystemRoot\System32\drivers\volmgrx.sys
0x85148c88 mountmgr.sys         0x8c722000    0x16000 \SystemRoot\System32\drivers\mountmgr.sys
0x85148c08 atapi.sys            0x8c738000     0x9000 \SystemRoot\system32\drivers\atapi.sys
0x85148b88 ataport.SYS          0x8c741000    0x23000 \SystemRoot\system32\drivers\ataport.SYS
```

Figure 12: Getting application list that are running on kernel-level

From the Figure 12, it is clearly seen that there is a list of applications and other files that are loaded in the system's kernel. An analyst can make out interrelationships between them to study system behavior caused by the suspect binary file.

Chapter 4

Future Work Plan

The future work plan for our project is as follows:

Sl. No.	Work Description	Duration in Days
1.	To gather information about code analysis tools and techniques and come up with a suitable one for our analysis.	5 - 7 days
2.	To implement a code analysis procedure and provide a conclusive report.	10 – 15 days (or more)
3.	To look forward to, other automated methodologies.	10 - 12 days

Chapter 5

Weekly Task

The report of project work allocated by the supervisor is as follows:

Week No.	Date: From-To	Work Allocated	Work Completed (Yes/No)	Remarks	Guide Signature
1.	Dec 01, 2022 – Dec 09, 2022	Setting up of lab environment and installation of software required for dynamic analysis.	Yes		
2.	Dec 10, 2022 – Dec 16, 2022	Getting information collected for phases in dynamic analysis techniques and appropriate tools.	Yes		
3.	Jan 3, 2022 – Jan 14 2022	Performing initial phases of dynamic analysis.	Yes		
4.	Jan 15, 2023 – Jan 30, 2023	Performing and concluding reports as per the outputs obtained from the initial and final phases of static malware analysis.	Yes		
5.	Feb 01, 2023 – Feb 06, 2023	Gathering details of in-memory analysis and preparing the workflow.	Yes		
6.	Feb 07, 2023 – Feb 11, 2023	Setting up of lab environment and installation of software required for in-memory analysis.	Yes		

7.	Feb 12, 2023 – Feb 28, 2023	Performing memory forensics to analyze the behavior of suspected binary with the system memory.	Yes		
8.	Mar 03, 2023 – Mar 10, 2023	Preparing a conclusive report from the obtained outputs	Yes		

References

[1] X. Xiao, S. Zhang, F. Mercaldo, G. Hu, and A. K. Sangaiah, "Android malware detection based on system call sequences and LSTM," Multimed Tools Appl, vol. 78, no. 4, pp. 3979–3999, Feb. 2019, doi: 10.1007/s11042-017-5104-0.

[2] F. Azuaje, "Witten IH, Frank E: Data Mining: Practical Machine Learning Tools and Techniques 2nd edition," Biomed Eng Online, vol. 5, no. 1, Dec. 2006, doi: 10.1186/1475-925x-5-51.

[3] "Basic Static Malware Analysis using open-source tools. B. Jaya Prasad, Haritha Annangi & Krishna Sastry Pendyala," B. Jaya Prasad, Haritha Annangi & Krishna Sastry Pendyala, 2016 .

[4] S. Narayan Tripathy, S. Kumar Kapat, M. Soujanya, and S. Kumar Das, "Static Malware Analysis : A Case Study," 2017.

[5] N. Mohd, A. Singh, and H. S. Bhadauria, "Intrusion Detection System Based on Hybrid Hierarchical Classifiers," Wirel Pers Commun, vol. 121, no. 1, pp. 659–686, Nov. 2021, doi:10.1007/s11277-021-08655-1.

[6] R. Sihwail, K. Omar, and K. Akram Zainol Ariffin, "An Effective Memory Analysis for Malware Detection and Classification," Computers, Materials & Continua, vol. 67, no. 2, pp. 2301–2320, 2021, doi: 10.32604/cmc.2021.014510.

[7] R. Sihwail, K. Omar, and K. A. Zainol Ariffin, "A Survey on Malware Analysis Techniques: Static, Dynamic, Hybrid and Memory Analysis," International Journal on Advanced Science, Engineering and Information Technology, vol. 8, no. 4–2, p. 1662, Sep. 2018, doi: 10.18517/ijaseit.8.4-2.6827.

[8] C. Rathnayaka and A. Jamdagni, "An Efficient Approach for Advanced Malware Analysis Using Memory Forensic Technique," 2017 IEEE Trustcom/BigDataSE/ICESS, Aug. 2017, Published, doi: 10.1109/trustcom/bigdatase/icess.2017.365.

[9] R. Sihwail, K. Omar, K. Zainol Ariffin, and S. Al Afghani, "Malware Detection Approach Based on Artifacts in Memory Image and Dynamic Analysis," Applied Sciences, vol. 9, no. 18, p. 3680, Sep. 2019, doi: 10.3390/app9183680.

[10] M. Ishrat, Dr. M. Alamgeer, and Dr. M. Saxena, "Comparison of Static and Dynamic Analysis for Runtime Monitoring," (PDF) Comparison of Static and Dynamic Analysis for Runtime Monitoring | Dr. Mohammad Alamgeer, Mohammad Ishrat, and Dr. Manish Saxena - Academia.edu.