Major Project Progress Report

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: 1

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

The majority of 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 actually 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 process of analyzing a binary without actually running it. It is the most straightforward method and allows you to extract the metadata associated with the suspect binary. Static analysis may not reveal all of the required formations, but it can occasionally provide interesting information that can help you decide where to focus your subsequent analysis efforts.
- **Dynamic analysis** (Behavioral Analysis): This is the process of running the suspect binary in a controlled environment and observing its behavior. This analysis technique is simple to use and provides useful information about the binary's activity during execution. This analysis technique is useful, but it does not reveal all of the hostile program's functionalities.
- **Code analysis** focuses on analyzing the code in order to comprehend the inner workings of the binary. Static code analysis and dynamic code analysis are two types of code analysis. Static code analysis entails disassembling the suspect binary and inspecting the code to understand the program's behavior, whereas dynamic code analysis entails a controlled debugging of the suspect binary to understand its functionality.
- **Memory analysis** (also known as memory forensics) is the process of searching the computer's RAM for forensic evidence. It is normally a forensic method, but including it in our malware research may help you comprehend how the malware behaves once it has been installed.

The work carried out so far, started off with the static analysis of malware, where the first step includes lab environment setup. We need a few things before you can set up a lab: a physical system with a virtualization software installed and running Linux, Windows, or macOS as its primary operating system (such as VMware or VirtualBox). We have taken Ubuntu 20.04 LTS operating system and a virtualization software as VirtualBox. Other specifications of the system at least 4 GB RAM and a Quad-Core CPU.

Starting off with static analysis, the steps to be involved will be as follows:

A) File Type Determination

The identification of the malware file type is the first stage in the static analysis process employed for our study. This helps with the analysis of the malware samples' target operating system. Portable Executable (PE)[1] files are frequently used with the Windows operating system, so when we discover them, we can infer that the malware sample is intended to infect a computer running the Windows operating system.

The process of determining file type can be done in varying ways, some of which are listed below.

- **Manual Method:** The suspected malicious file is opened in a hex editor as part of the manual method. A specialized editor called a "hex editor" can open any kind of file and show its contents byte by byte. The first two bytes of Windows PE files carry a file signature of MZ or the hexadecimal digits 4D 5A.
- **Python-based Identification** is an automated approach to determine file type, where the python-magic module is used.

We have successfully completed this phase and the results are depicted in Figure 1:

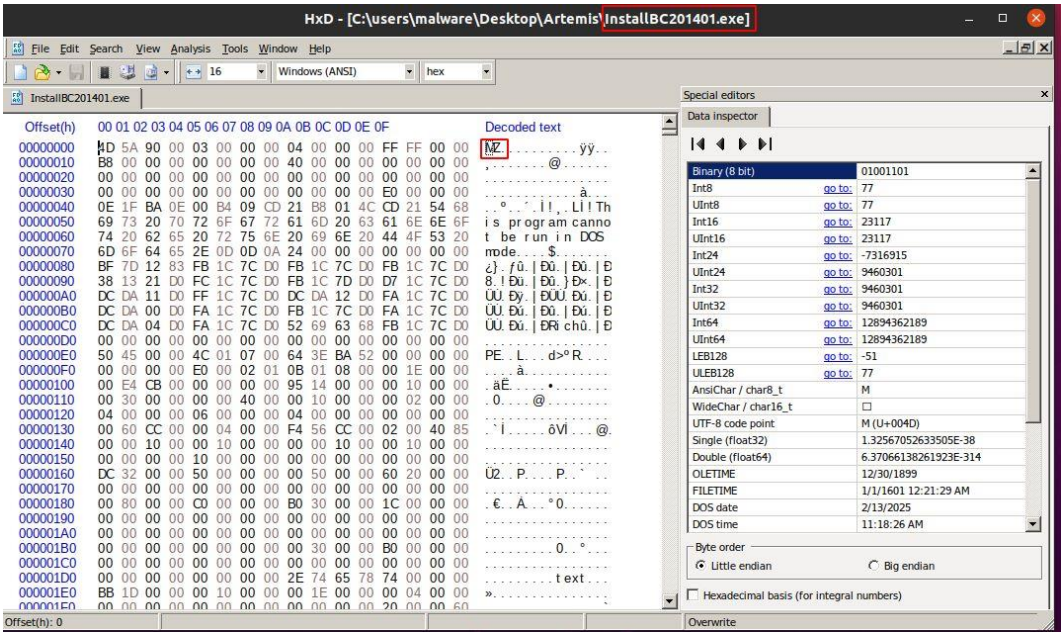

Figure 1: Determining the File Type

B) Malware Fingerprinting

The act of creating hash values for a suspected dangerous file based on its content is known as fingerprinting the malware. SHA1, MD5, and other cryptographic hashing techniques are used to generate these hash values. The suspicious file can be uniquely identified using a cryptographic hash. Available ways of generating cryptographic hash are listed as follows.

- **Generating cryptographic hash using Tools**

Cryptographic hashes of a file can be created on a Linux system using the tools *md5*, *sha256*, and *sha1*. On the other hand, there are a number of online programs that may be used to create cryptographic hashes for the Windows operating system. *HashMyFiles* is one such program that can generate hash values for a single or a group of files while highlighting similar hashes using the same color schemes.

- **Generating cryptographic hash using Python**

In the current generation, python, via its varied libraries and pre-defined functions and hence, serves our aim, i.e., for determining file type, *hashlib* module can be used.

We have successfully completed this phase and the results are depicted in Figure 2:

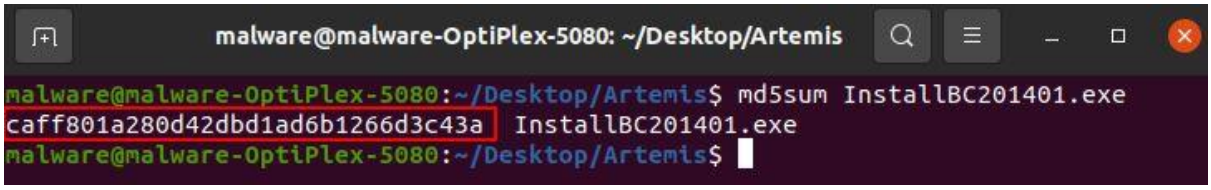

Figure 2: Malware Fingerprinting

C) Anti-virus Scanning

It is advised to run numerous anti-virus scans using different scanners when performing static analysis for malware. Since, no AV scanner can fully trace the behavior of a malicious file, performing AV scans aids in identifying different behavior. *VirusTotal* is a well-liked internet utility for AV scanning. By submitting the file to its server, scanning it with several tools, and showing the results on a live web page, *VirusTotal*[2] is a web-based program that may be used to scan malware files. *VirusTotal* also offers a UI for database searches using hashes, URLs, domains, or IP addresses. The other alternative is to use PE analysis tools such as *pestudio*[2] or PPEE. We have successfully completed this phase and the results are depicted in Figure 3:

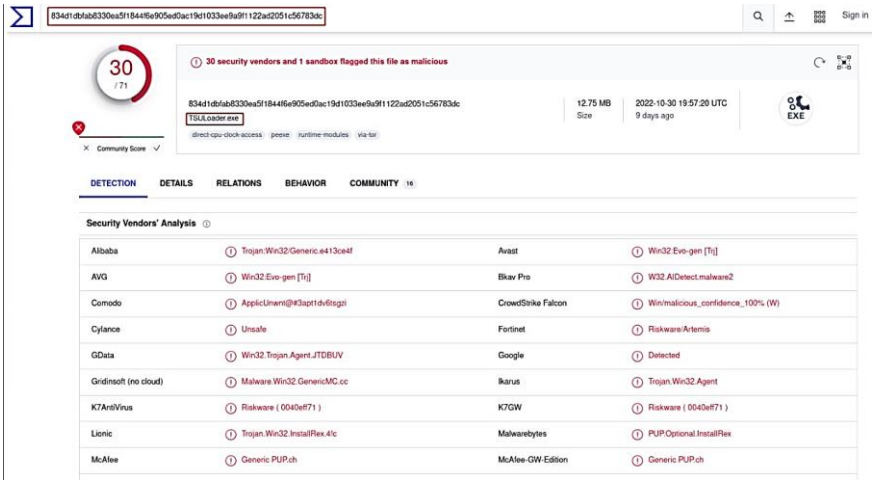

Figure 3: AV Scanning

D) String Extractions

Strings are Unicode or ASCII values of characters embedded within a file. Extracting strings can give pointers about the program/file functionality and the indicators associated with the suspect file/sample. The process of string extraction involves string extraction using various tools, followed by decrypting the string.

- **Extracting Strings using Tools:** Linux operating systems provide a wide range of utilities to meet most of its user requirements, out of which one can be used to extract strings from the suspect malicious files. For a Windows operating system, a popular tool called, *pestudio*, is an excellent, handy, PE analysis tool that gives both ASCII and Unicode strings from an executable file. It is used for the initial assessment of the suspect file. One other tool that can be used for serving the same purpose is PPEE.

- **Obfuscated String Decoding**

String obfuscation is a method used to protect intellectual property, often used to hide malware samples in a system or application. String obfuscation helps in preventing malware detection and tracking. To meet this purpose, one such decoder/tool is *FLOSS*, which is used to extract human-readable strings from the suspected file. *FLOSS*[3] is an acronym for *FireEye Labs Obfuscated String Solver*.

We have successfully completed this phase and the results are depicted in Figure 4:

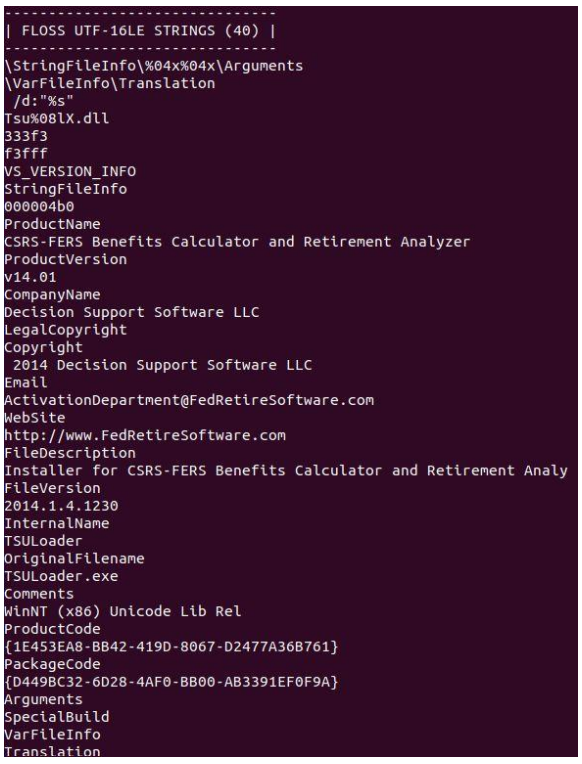

Figure 4: String Extractions

E) Detecting File Obfuscation

Obfuscation is the process of rendering an executable program or source code unreadable and difficult to understand by a human while maintaining its functioning. It is also used to protect the underlying workings of the malware from security researchers, malware analysts, and reverse engineers. The two such programs used to evade detection from security products such as antivirus and to thwart analysis are *Packers* and *Crytpors*[1].

A *Packer* software that unpacks itself in memory when the packed file is executed. It was invented to make the file smaller so that users wouldn't have to unpack them manually before executing.

Cryptors use encryption to obfuscate the executable content and this content is stored in a new file. It is also used to evade the antivirus and it can be classified as undetectable (UD) and fully undetectable (FUD).

An important difference between *Packers* and *Crypters* is that packers use compression techniques while *crypters* use encryption techniques.

We have successfully completed this phase and the results are depicted in Figure 5:

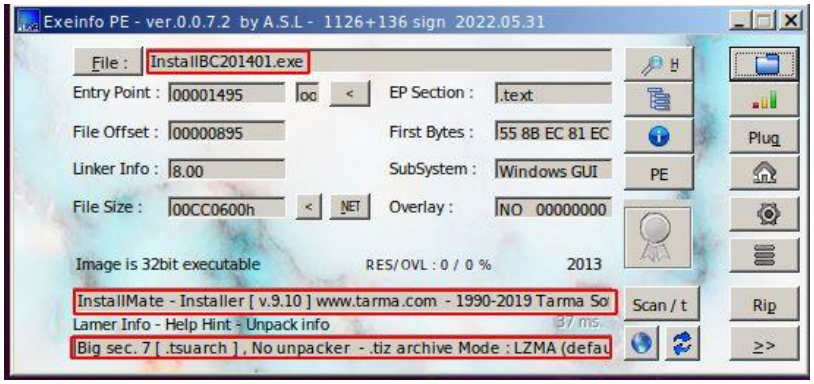

Figure 5: Detecting File Obfuscation

F) PE Header Inspection

The next step in static analysis can be an inspection of PE Headers, which are capable of giving an edge to our analysis now. Windows executable files must obey the PE/COFF, which is the Portable Executable/Common Object File Format. The PE file is a series of structures and sub-components that contain the information required by the operating system to load it into memory. An executable's header (PE header), explains its structure and is included during compilation. There are a number of tools available that are capable of examining and modifying PE Headers and their sub-components, such as *CFF Explorer*[4], *PE Internals* [4], *PPEE (puppy)* [4], *PEBrowse Professional* [4].

We have successfully completed this phase and the results are depicted in Figure 6:

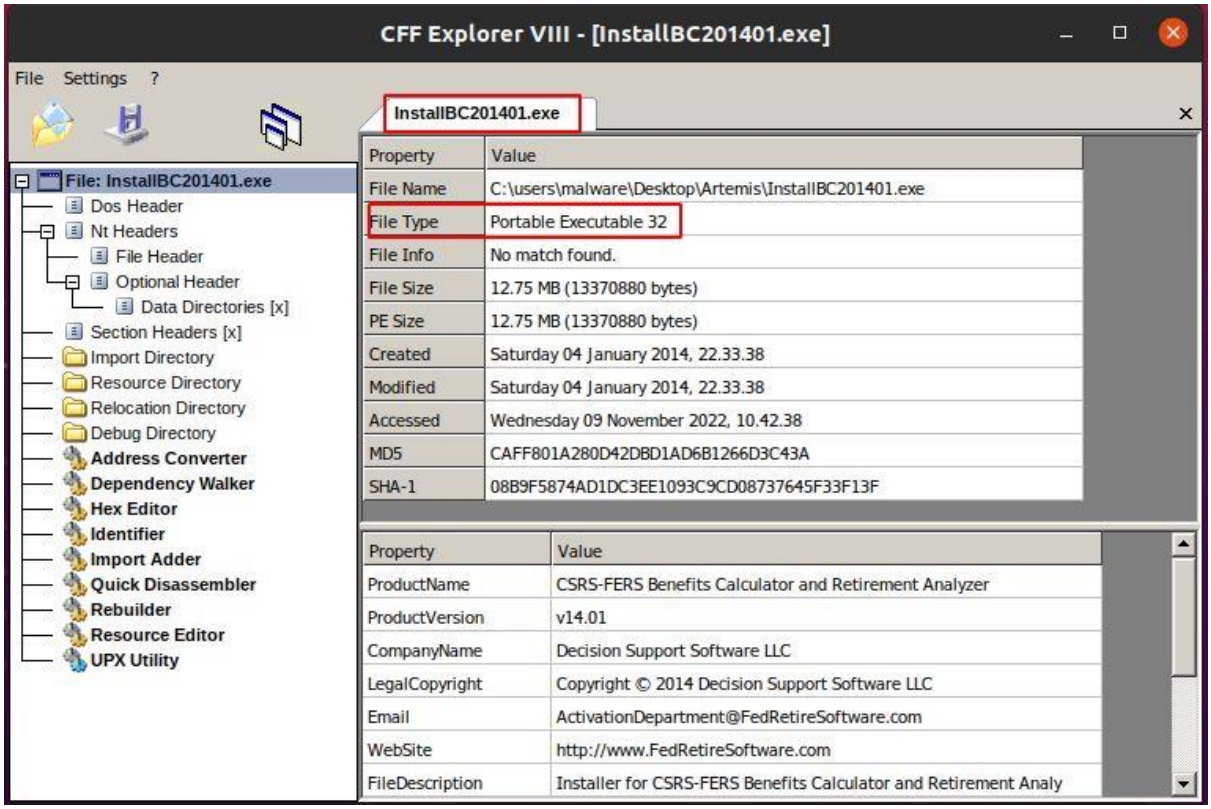

Figure 6: PE Header Examination

G) Malware Classification

When conducting a malware investigation, analysts might be interested in learning whether a given malware sample belongs to a specific malware family or if it shares traits with other previously analyzed samples. Putting the suspected file with previously analyzed samples or the samples stored in a public or private repository can give an understanding of the malware family, its characteristics, and its similarity with the previously analyzed samples. Some of the comparison and classification methods for the suspect binary are described as follows:

- **Categorizing Malware Employing Fuzzy Hashing**

A useful way to compare files for similarity is through fuzzy hashing. *Ssdeep* is a helpful tool for creating fuzzy hashes for samples and for calculating the similarities between samples on a percentage basis. This approach is effective in comparing a suspicious binary with the samples in a repository to determine the samples that are similar; this may aid in identifying the samples that belong to the same malware family or the same actor group. We can use *ssdeep* to calculate and compare fuzzy hashes. In Python, the fuzzy hash can be computed using *python-ssdeep*.

- **Categorizing Malware Employing Import Hash**

Using Import Hashing is another method for locating similar samples and samples utilized by the same threat actor groups. Import hashing, also known as *imphashing*, is a method of calculating hash values based on the names of the libraries and imported functions (APIs) as well as their specific ordering inside the executable. The *imphash* value of the files would typically be the same if they were all created using the same source and procedure. If you come across instances of malware throughout your research that has the same *imphash* values, it indicates that they share an import address table and are likely connected. The *pefile* module in Python may be used to create *imphash*.

- **Categorizing Malware Employing Section Hash**

The malicious software is run through a hashing program that produces a unique hash that identifies that malware. It is just like import hashing and also helps in identifying related samples. When an executable file is loaded in *pestudio*, it helps in calculating the MD5 for .text, .data, .rdata, and so on. The Message-Digest Algorithm 5 (MD5) hash function is the one most commonly used for malware analysis, though the Secure Hash Algorithm 1 (SHA-1) is also popular.

- **Categorizing Malware Employing YARA Rules**

A strong classification and detection tool are YARA. It detects malware using a rule-based methodology. Yara rules are used for classifying and identifying malware samples by creating descriptions of malware families based on textual or binary patterns. It is a very popular open-source and multi-platform tool that provides a mechanism to exploit.

Yara rule is a set of strings and a boolean expression, that determines its logic. Yara rules are very generic and specific and these can be created using any text editor. The sections that are found in the YARA rules are metadata, identifiers, identifier strings, and conditions. An analyst may develop a YARA rule while analyzing a malware sample for a new sample.

The above-mentioned steps of static analysis can be summarized with the help of Figure 7:

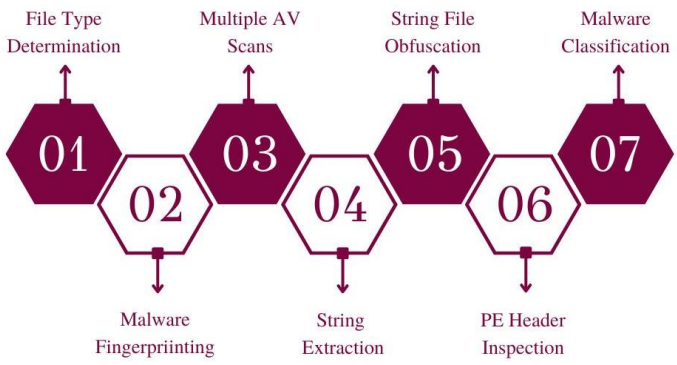

Figure 7: Static Analysis Steps

Chapter 4

Future Work Plan

The future work plan for our project is as follows:

Sl. No.	Work Description	Duration in Days
1.	To perform the last few steps of static malware analysis and provide a conclusive report.	10 - 15 days
2.	To gather information about dynamic analysis tools and techniques and come up with a suitable one for our analysis.	5 - 7 days
3.	To implement a dynamic analysis procedure and provide a conclusive report.	10 – 15 days (or more)
4.	To look forward to, code analysis and in-memory analysis 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	Oct 01, 2022 – Oct 09, 2022	Setting up of lab environment and installation of requirements			
2	Oct 10, 2022 – Oct 17, 2022	Getting information collected for phases in static analysis techniques and appropriate tools.			
3	Oct 17, 2022 – Oct 27, 2022	Performing initial phases of static analysis.			
4	Oct 15, 2022 – Nov 22, 2022	Performing and concluding reports as per the outputs obtained from initial and final phases of static malware analysis.			

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] U. Baldangombo, N. Jambaljav, and S.-J. Horng, “A STATIC MALWARE DETECTION SYSTEM USING DATA MINING METHODS.”

[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*.

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