

**SENTINAL:
THE MALWARE ANALYZER**



**YENEPOYA INSTITUTE OF ARTS, SCIENCE, COMMERCE ANDMANAGEME
A CONSTITUENT UNIT OF YENEPOYA (DEEMED TO BE UNIVERSITY)
BALMATTA, MANGALORE**

**A PROJECT REPORT ON
“SENTINAL: THE MALWARE ANALYZER”**

**SUBMITTED BY
SHAIN MATHEW
IV MSC (CYBERSECURITY, DATA ANALYTICS, AND CLOUD COMPUTING)
22MSCCCD13**

**UNDER THE GUIDANCE
MS. CHAITHRA NAGESH BHANDARY
DEPARTMENT OF COMPUTER SCIENCE**

**IN PARTIAL FULLFILLMENT OF THE REQUIREMENT FOR
THE AWARD OF THEDEGREE OF
MASTER OF COMPUTER SCIENCE**

AUGUST 2024



DECLARATION

I hereby declare that this project report entitled "Sentinal: The Malware Analzer" had been prepared by me towards the partial fulfilment of the requirement for the award of the Master of Computer Science at Yenepoya (Deemed to be University) for the academic year 2023-2024.

I also declare that this project is the result of my own effort and has not been submitted to any other University or an Institution

Place: Mangalore
Date:

SHAIN MATHEW
22MSCCCD13

ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without mentioning the people who made it possible, whose constant guidance and encouragement crown all of the efforts with success. It is a privilege to express my gratitude and respect to all those who inspired and helped me in the completion of this project.

I would like to express our sincere gratitude to our To Principal and Dean of Science **Prof. (Dr.). Arun Bhagawath** for giving us an opportunity to study in this trending field.

I express my deep gratitude to **Vice Principal Dr. Shareena P, Dr. Jeevan Raj, and Mr. Narayan Sukumar** for their guidance and constant encouragement during my Project.

I express my deep gratitude to **HOD Dr. Rathnakar Shetty, Department of Computer Science** for his guidance and constant encouragement during my Project.

I would like to express my sincere gratitude to my internal guide, **Ms. Chaitra Nagesh Bhandary** and my class advisor **Ms. Ayshath Safwana** for guiding supporting for completion of a Project.

And also, I would like to thank our teaching and non-teaching staff of Yenepoya Institute of Arts, Science, Commerce and Management and also to my parents and friends who were directly or indirectly involved in the completion of our project.

SHAIN MATHEW
22MSCCCD13

ABSTRACT

Malware attacks are becoming more complex and difficult to identify in the ever-changing cybersecurity landscape. Despite their value, traditional malware analysis methods frequently fail to recognize sophisticated or unusual malware kinds, such as fileless viruses. By creating a thorough malware analysis platform that combines static, dynamic, and hybrid analytic techniques to offer a multifaceted approach to malware detection and comprehension, the Sentinel project tackles this pressing problem.

Sentinel uses sophisticated static analysis to detect known malware signatures and decompile programs without running the code itself. To enhance the observation of malware activity in real time, including system interactions and network activities, dynamic analysis is carried out in a supervised sandbox environment. Sentinel improves detection accuracy and identifies complex threats that may elude single-method approaches by fusing these methods with hybrid analytic techniques.

Sentinel's sophisticated visualization capabilities are a fundamental component that convert complicated analytical results into understandable and engaging visual representations. Security analysts can swiftly see patterns, anomalies, and trends with the help of these tools, which include heat maps, network graphs, and timeline views. This leads to better decision-making.

Sentinel's primary goal is to provide a safe and efficient environment for analyzing malware by offering comprehensive insights from its analysis procedures. Results from static, dynamic, and hybrid assessments are combined into concise, useful summaries by the platform's extensive reporting features. Security experts can react to new threats more confidently and precisely thanks to these thorough reports, which help them immediately comprehend the behavior of malware.

The financial, healthcare, and critical infrastructure sectors are among those that face increasing malware threats, making this initiative essential. In the current digital era, Sentinel establishes a new benchmark for malware identification and analysis by eliminating the drawbacks of conventional analytic techniques and offering a robust, comprehensive solution. This initiative opens up new possibilities for cybersecurity research in the future and also progresses the field of malware analysis.

CONTENTS

Table No:	Particular	PAGE NO
CHAPTER 1	Introduction	Page no: 1
CHAPTER 2	LITERATUREREVIEW 2.1 Introduction 2.1.1Technology 2.1.2Hardware 2.1.3Operatingsystem 2.1.4Applications 2.1.5Tools 2.2LITERATUREREVIEW	Page no: 4
CHAPTER 3	SOFTWARE REQUIREMENTS 3.1 Introduction 3.2 Requirements Specification	Page no:13
CHAPTER 4	IMPLEMENTATION 4.1 Introduction 4.2 Implementation Steps 4.3 Analysis workflow 4.4 Explained workflow diagram	Page no: 16
CHAPTER 5	CODING 5.1 Introduction 5.2 Code	Page no:36

CHAPTER 6	Results and Discussions 6.1 Project objective 6.2 Scope 6.3 Summary of the findings 6.4 Results	Page no: 49
CHAPTER 7	SCOPE FOR FUTURE WORK	Page no: 55
CHAPTER 8	CONCLUSION	Page no: 57
CHAPTER 9	REFERENCES	Page no: 59

CHAPTER - 1

INTRODUCTION

1.1. OVERVIEW

In the ever-evolving field of cybersecurity, it is important to understand, identify, and mitigate malware risks. Malware puts people, businesses, and even national security at risk. To tackle these issues, "Sentinal: The Malware Analyzer" offers an extremely sophisticated sandbox environment made for in-depth malware investigation. This state-of-the-art platform gives security analysts a potent toolkit that combines sophisticated visualization capabilities with cutting-edge forensic techniques, providing a thorough foundation for researching and battling dangerous malware. Sentinal offers an effective response to modern cybersecurity requirements using the latest technology and rigorous analytical methodology.

Through a systematic and creative approach, the platform's architecture tackles the intricacies of contemporary cybersecurity concerns. Sentinal guarantees that security analysts are adequately equipped to tackle the dynamic nature of cyber threats by merging its sophisticated functionalities with meticulous analytical techniques. Sentinal, then, provides a crucial and innovative solution that satisfies the demands of modern cybersecurity and gives digital environments a crucial advantage in defending against the malware threats that are always evolving.

1.2. OBJECTIVES

"Sentinal: The Malware Analyzer" aims to address three fundamental approaches that are essential to malware analysis: hybrid, dynamic, and static analysis. Static analysis is the process of looking at malware code without running it, with the goal of spotting possible dangers by looking at trends, known signatures, and code structure. In contrast, dynamic analysis involves running malware in a sandbox or controlled environment in order to watch its activities in real time. This technique offers information on the network activity, file changes, and possible payload delivery that malware engages in while interacting with the system.

A thorough knowledge of malware is provided via hybrid analysis, which combines static and dynamic methods. Hybrid analysis improves analysis depth and detection accuracy by fusing the best features of each technique. Finding complex malware that could avoid detection with a single method is much easier using this strategy.

"Sentinal" emphasizes the importance of creating intuitive visualization tools in addition to these analytical approaches. To make it easier to spot patterns, trends, and abnormalities in malware activity, these strategies are made to display analysis results in an understandable and accessible way. Security analysts can more rapidly understand complicated information when it is presented visually, which helps them make decisions and take appropriate action.

1.3 SCOPE

The project's scope encompasses establishing virtual environments for malware research and setting them up using technologies such as VirtualBox. To enable thorough malware research entails installing and setting up several analytic tools, including Flare VM and REMnux. The project involves doing in-depth research utilizing static, dynamic, and hybrid approaches. The results are then presented through sophisticated visualizations made with Flask and Dash, two Python-based frameworks. IT workers, analysts, and security researchers are all included in the target audience. Sentinel wants to improve its capacity to recognize and successfully counteract malware attacks by providing a strong and complete platform.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

In-depth summaries of published research in a particular field of study constitute a literature review. It may be the primary subject of the entire work or it may be a section of a longer paper or article. Contextualizing one's study within the larger academic environment, identifying research directions, gaps, and inconsistencies, and fully familiarizing oneself with current knowledge are the goals of the literature review. In order to support their research questions, hypotheses, and techniques, researchers must show that they have a thorough understanding of their topic by reading and analyzing pertinent literature. Furthermore, literature reviews offer a clear and cohesive synopsis of the state of the subject today, making them an indispensable tool for other scholars. This enables people to comprehend the study's breadth, major results, and areas that require more research fast. In other words, a well-executed literature review adds to the body of academic knowledge by outlining the direction of scientific research and charting the intellectual terrain in addition to verifying the validity and applicability of the suggested study.

A thorough literature assessment serves as the project's theoretical cornerstone. The foundation for this study was established by recent developments in malware detection and analysis methods, particularly in the areas of dynamic and static analysis, machine learning, and hybrid approaches. A full understanding of the current approaches and difficulties in malware detection can be gained from studies like "Similarity-based Intelligent Malware Type Detection through Multiple Sources of Dynamic Characteristics" by Denzer and "Exploration of Tools and Techniques for Malware Detection and Analysis" by Kumar, Sharma, Dhaundiyal, and Jain. These groundwork studies emphasize how crucial it is to combine static and dynamic analytic methods in order to improve malware detection systems' precision and effectiveness.

Large-scale research in cybersecurity, machine learning, and malware behavior analysis provides the foundation for the development of reliable malware detection methods. With regard to machine learning techniques for malware categorization and dynamic and static malware analysis, in particular, the project builds upon existing ideas and technology. Real-time threat identification and mitigation approaches that combine static code analysis and dynamic behavior monitoring have been developed in response to the growing requirement for efficient malware detection.

This overview of the literature looks at the fundamental ideas, developments, difficulties, and potential paths in malware analysis and detection. It emphasizes the value of integrating static and dynamic analytic methods, how machine learning may increase the precision of categorization, and the continuous efforts being made to solve the problems brought on by malicious actors' ever-evolving strategies. By proving that strategies and tactics utilized in malware analysis are backed by empirical evidence and have been successfully applied in comparable circumstances, the study validates the viability of employing a hybrid approach for malware detection. This builds a solid framework for the project's future development and possible scientific contribution, while also enhancing its credibility.

To ensure that the platform is founded on knowledge of the most recent cybersecurity methods and methodologies, a literature study is generally necessary for malware detection and analysis initiatives. By employing tried-and-true techniques and empirical data from earlier studies, the project may gain significant credibility and pave the path for the next advancements and scholarly contributions in the field of cybersecurity and malware detection.

2.1.1 TECHNOLOGIES

1. Static Analysis

- **Signature-based Detection:** Uses predetermined signatures to identify known malware.
- **Code disassembly:** reveals secret instructions by converting executable files into legible code.
- **Pattern Matching:** Looks for unusual and suspicious patterns in the code structure. Heuristic analysis assesses code, even if it has never been seen before, based on its behavior and its danger level.

2. Dynamic Analysis

- **Behavioral Monitoring:** Keeps tabs on how the virus interacts with the operating system, including changes to files, the formation of new processes, and network connections.
- **System Calls Interception:** To comprehend the malware's functioning activity, log the system calls it makes.
- **Network traffic analysis:** Looks for contact with command-and-control servers by tracking and examining both inbound and outgoing network traffic.
- **Resource Utilization Tracking:** Monitors how the virus affects system resources such as memory and CPU consumption.

3. Hybrid Analysis

- **Correlation Analysis:** To improve detection accuracy, cross-reference the findings of static and dynamic analysis.
- **Pattern Recognition:** Finds intricate patterns in virus activity that static or dynamic analysis alone could overlook.

4. Visualization Framework

- **Interactive dashboards** are real-time, interactive displays that let analysts go deeper into particular data points.
- **Heat maps:** Illustrative depictions of threat levels and activity levels among various system elements.

- Network Graphs: Data flows and network connections shown, emphasizing dubious interactions.
- Timeline Views: A chronological depiction of events that helps comprehend the order and timing of malicious activity.

2.1.2 HARDWARE

- Processor: Intel Core i5 or above
- RAM: 8GB minimum (15GB recommended for optimal performance)
- Storage: 100GB free space (SSD preferred for faster read/write operations)
- Graphics: GeForce GTX 1050 Ti (minimum)
- Input Devices: Keyboard, Mouse, USB

2.1.3 OPERATING SYSTEM

This project leverages the capabilities of several systems to provide thorough malware analysis by adopting a multi-layered approach to its operating system environment. Sentinel's default operating system, Windows 11, was selected because to its broad hardware and software compatibility, allowing for seamless interaction with a variety of analytic tools and technologies.

Sentinel integrates FLARE VM, a dedicated virtual machine for sophisticated malware research, into the Windows 10 environment. A comprehensive set of forensic and reverse engineering tools designed specifically for managing intricate malware samples is offered by FLARE VM. Sentinel's capacity to carry out intricate static and dynamic analysis as well as complex reverse engineering activities is improved by this configuration.

Sentinel also incorporates REMnux Linux, a potent distro designed specifically for malware investigation. By offering a wide range of Linux-based tools and utilities, REMnux enhances the Windows and FLARE VM configuration, extending Sentinel's analytical capabilities and facilitating more thorough investigations into malware behavior and features.

Sentinel's versatility and effectiveness in combating a broad range of malware threats are ensured by the combination of Windows, FLARE VM, and REMnux. Additionally, Sentinel offers developers and administrators a user-friendly interface and strong analysis tools.

2.1.4 APPLICATIONS

- VirtualBox : a potent virtualization platform that facilitates the deployment and separation of several operating systems and analysis environments. It is used to construct and manage virtual machines.
- FLARE VM: A dedicated virtual machine environment included into Windows 10 that is intended for sophisticated malware investigation using a variety of forensic and reverse engineering tools.

- REMnux: An extensive collection of tools for analyzing and comprehending harmful software, specifically designed for use with Linux distributions.
- VirusTotal: An internet service that provides threat detection and preliminary analysis by scanning files and URLs with many antivirus engines.

2.1.5 TOOLS

- Static Analysis Tools: IDA Pro, Ghidra, Binary Ninja
- Dynamic Analysis Tools: Cuckoo Sandbox, OllyDbg, x54dbg
- Hybrid Analysis Tools: VMRay, Hybrid Analysis
- Forensic Tools: Volatility, Autopsy, Sleuth Kit
- Visualization Tools: Python-based visualization with Flask and Dash

2.2. Literature Review

The literature review offers a thorough study of the tools and approaches currently used in malware analysis, stressing both their advantages and disadvantages in dealing with the always-changing threat landscape. It explores both static and dynamic analysis methodologies, offering a thorough examination of how each approach aids in the detection and comprehension of malware. The capacity of static analysis, which examines the code without running it, to uncover new or obfuscated malware and its limits in detecting known threats using signature-based detection is evaluated. However, dynamic analysis is analyzed for its shortcomings in handling complex evasion techniques used by modern malware, as well as its effectiveness in revealing behavioral patterns and network activities that static analysis might miss. Dynamic analysis, on the other hand, runs the malware in a sandbox environment to observe its behavior.

Hybrid approaches, which mix static and dynamic analytic methodologies to maximize the benefits of both approaches, are further explored in the paper. The benefits of hybrid analysis in thwarting sophisticated threats are covered in this section, along with how it may improve detection accuracy and offer a more comprehensive knowledge of malware activity. Additionally discussed is the incorporation of artificial intelligence and machine learning into hybrid analysis, emphasizing how these technologies may enhance the efficacy and efficiency of malware analysis and detection.

1) Exploration of Tools and Techniques for Malware Detection and Analysis

Author: S. Talukder

The primary goal of this literature review is to provide a comprehensive survey of various tools and techniques employed in the detection and analysis of malware. Malware poses a significant threat to computer systems and networks globally. Effective detection and analysis are essential for preventing cyber attacks and protecting sensitive data.

This review highlights the importance of advanced tools and techniques in combating malware, offering insights into current research trends and innovations in the field. Researchers have extensively studied static and dynamic analysis approaches for detecting malware.

Static analysis involves examining malware code without execution, while dynamic analysis entails running malware in a controlled environment to observe its behavior. Integrating both approaches aims to enhance malware detection capabilities by bridging the gap between static and dynamic methods. Machine learning techniques are increasingly used to classify malware based on behavioral patterns.

By analyzing the behaviors of different malware types, machine learning algorithms can categorize unknown malware into known families, thereby improving threat mitigation strategies. Recent studies have introduced novel detection strategies, such as intelligent malware detection using file relation graphs. By analyzing relationships between files, researchers have developed advanced mechanisms to detect and mitigate malware threats more efficiently. Various specialized tools assist in malware analysis.

For instance, Malzilla facilitates the analysis of websites containing malicious code through features like code deobfuscation and proxy capabilities. These tools enable researchers to understand complex malware behaviors comprehensively. The field of malware detection and analysis is continually evolving. Current research explores areas such as analyzing system call sequences and employing LSTM for detecting Android malware. Efforts in scalable storage of malware metadata and automatic reverse engineering of binaries highlight advancements aimed at enhancing malware analysis capabilities. In conclusion, this literature review underscores the importance of tools and techniques in detecting and analyzing malware. Through advanced analysis methods, machine learning algorithms, and specialized tools, cybersecurity professionals can effectively combat evolving malware threats and safeguard critical systems. The review offers valuable insights into current research trends in malware detection and analysis, emphasizing the need for ongoing innovation and collaboration in cybersecurity. [1]

2) Investigation of Malware and Forensic Tools on Internet

Authors: Tarun Kumar, Sanjeev Sharma, Ravi Dhaundiyal, Parag Jain

The primary goal of this literature review is to classify the approaches, strategies, and tools used in malware analysis into categories for static and dynamic analysis, discussing current issues and problems in the area. In static analysis, the malware's code is examined without being executed, using methods like signature-based detection, decompilation, and disassembly. Tools such as IDA Pro and Ghidra are commonly used for decompilation and disassembly, aiding in understanding the functionality and structure of malware without risking execution. Signature-based detection identifies known malicious code patterns, effective for existing malware but less so against new or polymorphic threats. Dynamic analysis involves executing malware in a controlled setting to observe its behavior. Sandboxing tools like Cuckoo Sandbox

create an isolated environment to run and monitor malware, capturing details like system calls, file changes, and network activity. Behavior-based detection observes malware actions during execution to detect unusual behavior, involving monitoring API calls and profiling system behavior. Hybrid analysis combines static and dynamic techniques, taking advantage of both methods for a comprehensive understanding of malware. Recent research suggests combining static and dynamic features to improve detection accuracy, with tools examining opcode sequences along with runtime behaviors gaining popularity.

Machine learning (ML) has brought significant advancements in malware analysis, providing sophisticated methods for detecting and categorizing malware. ML models like Support Vector Machines (SVM) and neural networks are trained on features extracted from malware samples, identifying patterns and anomalies indicating malicious behavior. Clustering algorithms group similar malware samples, aiding in categorizing and identifying new malware variants based on their behavior and structure. Advanced techniques and future directions include visualization systems that assist analysts in understanding complex malware behaviors and system interactions. Research is focused on creating user-friendly visualization systems for malware analysis. Advanced profiling systems, like Holography, provide detailed insights into malware actions over time, helping to identify sophisticated and stealthy threats. Combining forensic analysis with malware analysis offers a broader perspective on understanding attacks, with techniques like memory forensics and file system analysis being crucial in this integration. The field of malware analysis is continually advancing, with new innovations aimed at enhancing detection and prevention methods. Combining static and dynamic analysis with machine learning provides promising directions for future research. Advanced visualization and profiling techniques improve analysts' ability to understand and effectively counter sophisticated malware threats. [2]

3) Malware Analysis and Reverse Engineering: Unraveling the Landscape of Digital Threats

Authors: A. Singhal and S. Venkataramalingam

The literature review in "Malware Analysis and Reverse Engineering: Unveiling the Landscape of Digital Threats" comprehensively explores significant research contributions in malware analysis and detection. It highlights pivotal studies that offer insights into the dynamic evolution of cyber threats and effective mitigation strategies.

1. Zero-Day Malware Detection

Alhaidari et al. conducted an extensive study on detecting Zero-Day malware, introducing ZeVigilante. This innovative approach integrates machine learning and sandboxing techniques within cybersecurity. By combining static and dynamic analyses, ZeVigilante aims to enhance

the detection of previously unknown malware, addressing challenges posed by rapidly evolving cyber threats.

2. Challenges in Malware Detection

Talukder et al. explored the complexities in malware detection, focusing on obstacles presented by polymorphic and metamorphic features in malicious software. The study highlighted the limitations of traditional signature-based detection methods and advocated for adopting machine learning algorithms to classify and identify unknown malware variants effectively. Additionally, the research provided an overview of various detection and analysis techniques, including memory forensics, packet inspection, sandboxing, and reverse engineering, emphasizing the need for a multi-faceted approach to malware analysis.

These seminal studies underscore the critical need for continuous innovation and advancement in malware analysis and reverse engineering practices. By leveraging cutting-edge technologies and methodologies, researchers and cybersecurity professionals can enhance their capabilities in detecting and mitigating sophisticated malware attacks, fortifying the resilience of digital systems, and ensuring robust data security against evolving cyber threats. [3]

4) Investigation of Stegomalware and Detection Techniques

Tarun Kumar, Sanjeev Sharma, Ravi Dhaundiyal, Parag Jain

Enterprises and consumers alike face a serious danger to their information security from stegomalware, a sort of hidden malware that hides hazardous parts via digital steganography. Stegomalware is a sophisticated kind of malware that evades detection by utilizing digital steganography techniques. To get beyond traditional security measures, attackers insert harmful code into seemingly innocent digital photos. Several detection techniques, such as steganalysis algorithms that examine picture data for hidden information, have been proposed to detect stegomalware. Research efforts are focused on developing powerful detection algorithms that can find viruses concealed in photos.

A comprehensive strategy that includes enhanced network security measures and safe image processing techniques is needed to prevent stegomalware infestations. In order to mitigate such threats, proactive measures like frequent security audits and staff training on stegomalware dangers are essential. Studies conducted on particular stegomalware instances, such as the Hammertoss malware, offer important insights into the strategies employed by malicious actors and the difficulties in identifying such risks. To keep ahead of new threats, cybersecurity research and innovation must continue due to the dynamic nature of stegomalware. We may

fortify cybersecurity defenses and safeguard digital assets from this elusive cyber danger by improving our knowledge of stegomalware and putting into practice efficient remedies. [4]

5) A Review on Fileless Malware Analysis Techniques

AUTHOR :Vala Khushali

Fileless Malware Analysis Techniques

Because cyber threats are always changing and might elude detection through conventional means, it is imperative that fileless malware analysis approaches be studied. The study "A Review on Fileless Malware Analysis Techniques" outlines the distribution, persistence, and execution phases of fileless malware assaults. Fileless malware functions mostly in memory, which makes it challenging for conventional antivirus programs to identify because they frequently depend on file-based signatures. Without leaving any disk footprint, methods like PowerShell and Windows Management Instrumentation (WMI) are commonly employed to carry out and spread these attacks.

A number of sophisticated malware analysis methods that can help identify fileless malware are also covered in the review. Static analysis is disassembling and analyzing code in executable files using tools such as IDA Pro and OllyDbg, without actually running the files. With dynamic analysis, however, questionable files are run in a safe setting so that their behavior can be seen. To improve detection accuracy, hybrid analysis blends static and dynamic approaches. When it comes to fileless malware, memory analysis works especially well since it can detect harmful behaviors like DLL injection and API hooking by looking at memory snapshots. Combining these methods makes it easier to identify and stop fileless malware assaults, which have become more frequent and complex. [5]

CHAPTER 3

SOFTWARE REQUIREMENTS

3.1 INTRODUCTION

Software requirements are in-depth explanations of the functions that a software system must have in order to meet user demands. Domain-specific, non-functional, and functional requirements can be used to categorize these needs.

Functional requirements outline the precise functions that Sentinel: The Malware Analyzer needs to carry out. Using technologies incorporated into FLARE VM, they include automated behavior analysis with Cuckoo Sandbox, supporting sophisticated reverse engineering, and allowing thorough malware research through static and dynamic methodologies. Additionally, the system has to have an intuitive user interface to facilitate smooth operation and allow effective use of REMnux Linux for various analytical activities.

The way the system accomplishes these tasks is described in non-functional requirements. This entails keeping the environment stable and dependable, guaranteeing optimal performance and responsiveness throughout analysis, and sustaining security procedures to safeguard private information. Another important consideration is usability, with an interface that is easy to use for both administrators and developers.

The designs, development, and testing stages of Sentinel's software are based on its requirements. What inputs are accepted, how they should be processed, and what outcomes are anticipated are all specified in the functional requirements, which also define the behavior of the system. Reliability benchmarks, security procedures, performance measures, and usability criteria are examples of operational features that are outlined in non-functional requirements.

Due to the necessity to analyze malware efficiently while preserving strong security and usability, Sentinel's construction entails special and complicated criteria. Ensuring that these criteria are precisely defined, documented, and managed is essential to Sentinel's effective deployment and functionality. This guarantees that user demands are met and that the program successfully tackles the difficulties associated with contemporary virus analysis.

3.2 REQUIREMENTS SPECIFICATION

Features and Functionality

- Capabilities for Malware Analysis: Both static and dynamic methods for thorough analysis of malware samples must be supported by Sentinel. This encompasses the capacity for:
 - Use the IDA Pro and OllyDbg tools, which are included with FLARE VM, to do static analysis.
 - Use Cuckoo Sandbox to automate sandboxing and do dynamic analysis.
 - Use REMnux Linux tools for further tasks related to reverse engineering and forensics.

- Combining Analysis Tool Integration: The system need to easily interface with the following frameworks and analytic tools:
 - Integration for first malware detection with VirusTotal.
 - Application of Yara and Radare2 for in-depth pattern matching and binary analysis.
- User Interface: Sentinel need to have an intuitive user interface that consists of the following:
 - An intuitive dashboard is one that is well-structured, simple to use, and clearly displays data, analytic findings, and system status
 - Effective Data Presentation: Metrics and analytical findings shown for better comprehension and use.
- Information Administration:
 - Safe Data Management: It should be possible for users to handle and examine malware samples in a secure manner, making sure that sample files are isolated and handled correctly.
 - Logs and analytical findings are securely stored.
- Live Malware Execution: Run malware samples in a safe, segregated setting to watch interactions and behavior in real time.
 - Real-time monitoring of process behavior, network activity, and system changes should all be included in this functionality. Behavior is automatically recorded, and reports on malware activity are generated.
 - Integration with sandboxing programs to provide secure operation without endangering the host system.

CHAPTER 4

IMPLEMENTATION

4. INTRODUCTION

Sentinel: The Malware Analyzer is implemented by converting the design specifications into useful software components. Coding the main features, integrating different modules, and thoroughly testing every part to guarantee correct functioning and performance are all included in this step. Creating an easy-to-use user interface, incorporating cutting-edge malware analysis technologies, and setting up a reliable backend system to handle data and analytic processes are some of the phase's primary tasks.

Sentinel aims to offer an all-inclusive and effective malware analysis platform by combining several analytical techniques and frameworks. The capacity to recognize and comprehend malware is essential in today's cybersecurity environment in order to safeguard systems and data. Sentinel combines tools from REMnux Linux and FLARE VM, uses sandboxing solutions like Cuckoo Sandbox for real-time behavior analysis, and combines static and dynamic analysis methodologies to meet this demand.

There are several important elements involved in the implementation process:

- **Integration of Analysis Tools:** Guaranteeing that different tools in the FLARE VM and REMnux environments, such as VirusTotal, Radare2, and Yara, work together seamlessly.
- **Live Malware Execution:** it involves putting malware samples into an environment that is controlled, watching for activity in real time, and recording actions for in-depth study.
- **User Interface Development:** Developing an easy-to-use interface with a dashboard that makes navigation simple and the presentation of analysis results clear.
- **Data management and security:** it involves putting strong procedures in place to handle, store, and protect malware samples and analysis findings while maintaining data integrity and privacy.

Meeting specified performance, scalability, and usability criteria is another major goal of the implementation. Sentinel is built to expand to meet increasing data and user needs while effectively managing massive numbers of malware samples. By integrating these components, Sentinel presents a useful method for malware analysis and provides researchers and cybersecurity experts with a useful tool.

Sentinel turns theoretical designs into a workable, dependable, and efficient malware analysis platform during this implementation phase, ready to take on the complexity of contemporary malware threats.

4.2 IMPLEMENTATION STEPS

4.2.1 Creating Windows-10 VM in VirtualBox

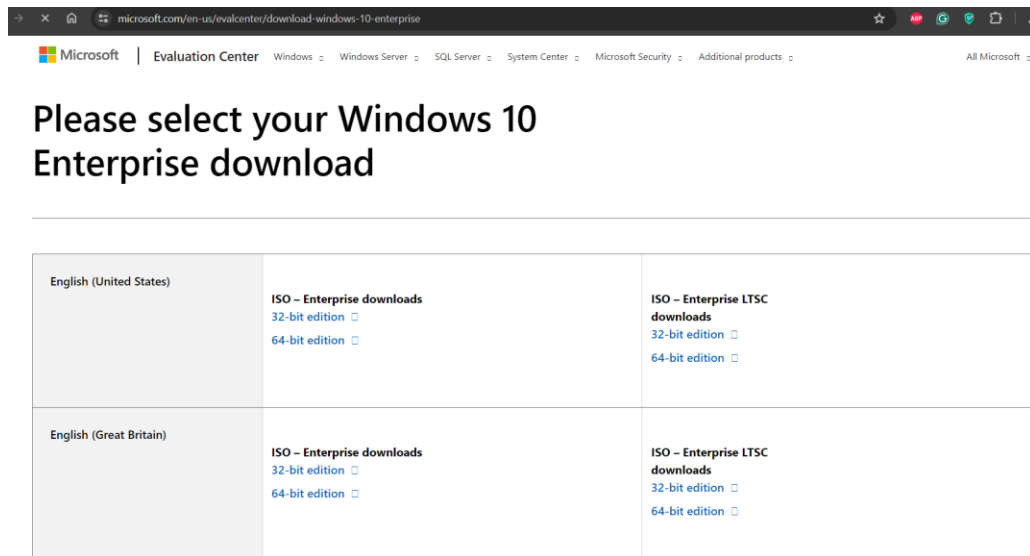
a) Installation of VirtualBox:

- Download windows from downloads page of ‘virtualbox.com’



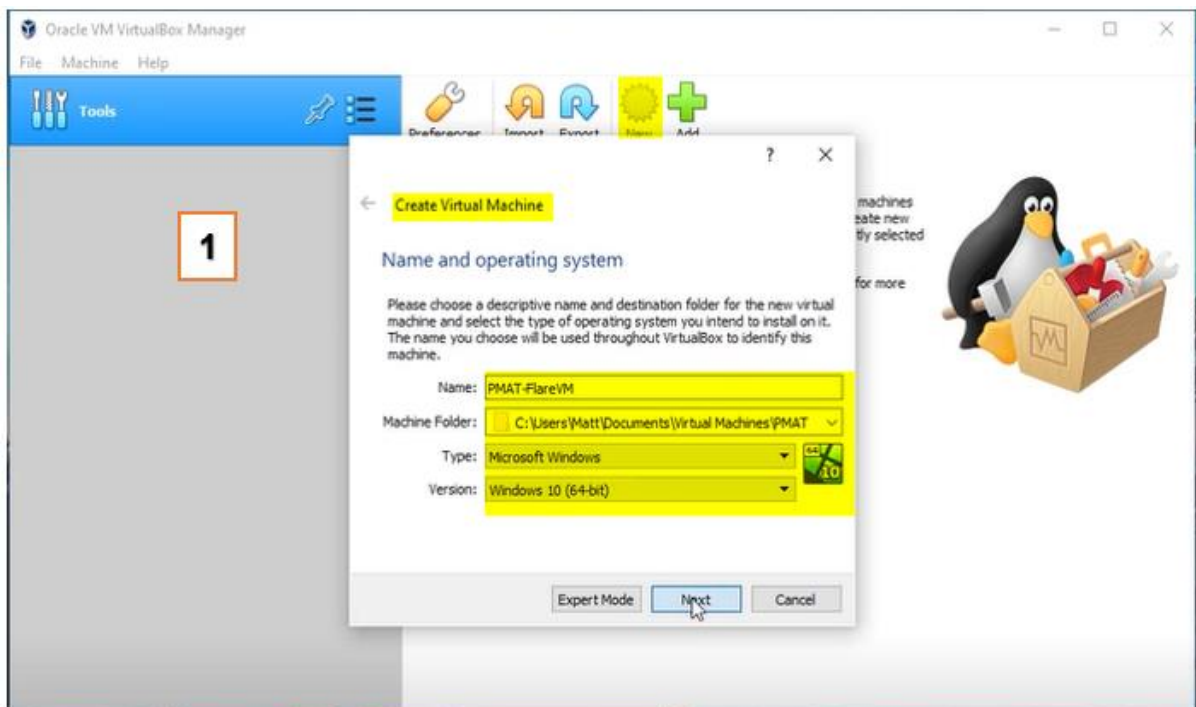
b) Download Windows 10 From Microsoft Official Website:

Windows Enterprise edition from official Microsoft Website

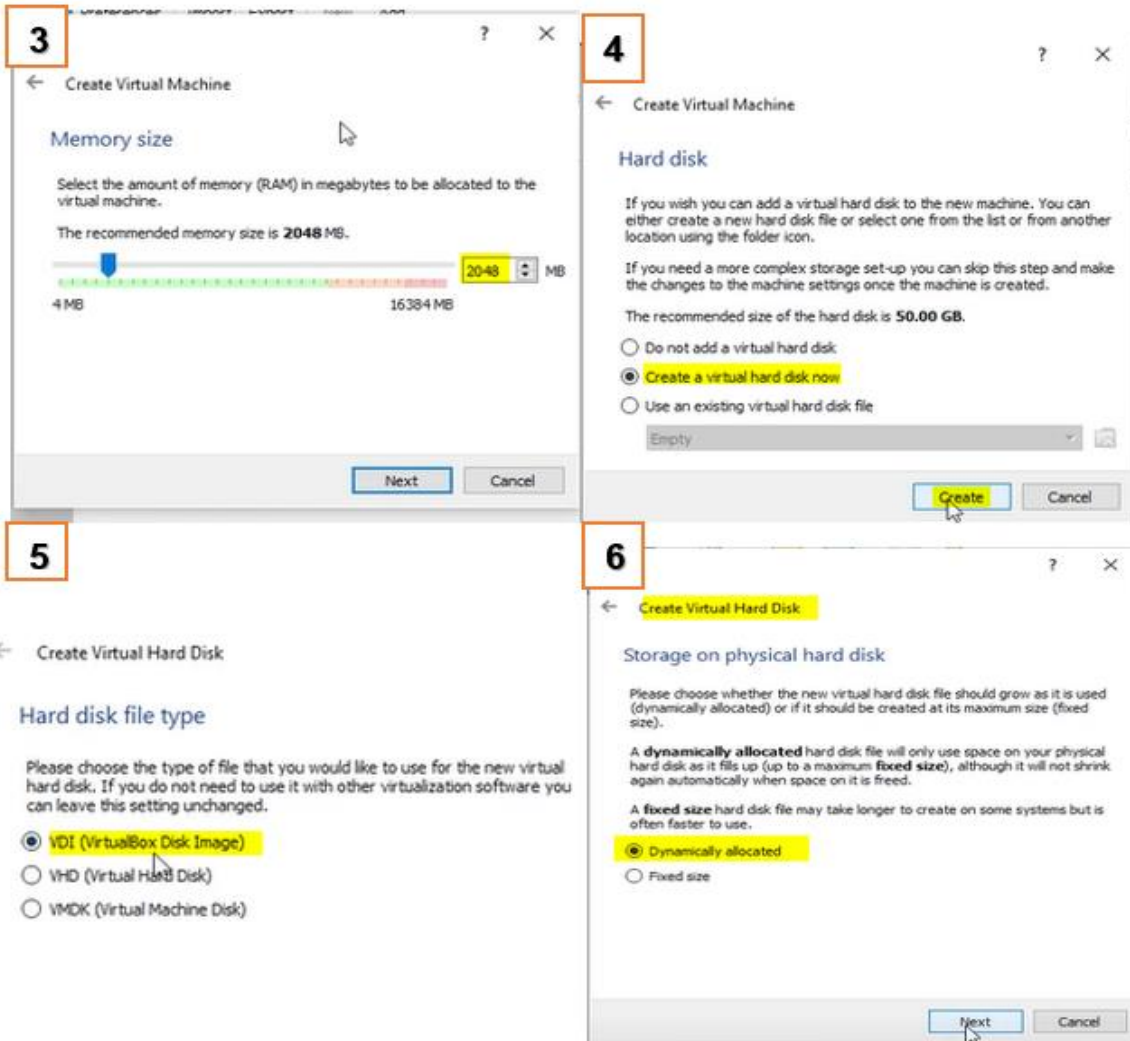


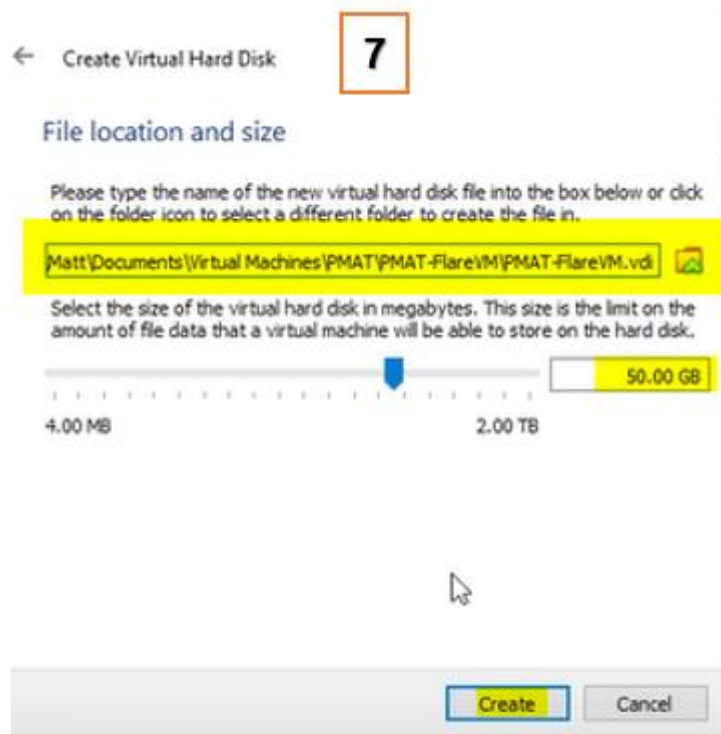
1. Generate a New Virtual Machine:

- Double click & Open VirtualBox.
- Click on the "New" button.
- Name The VM (for me it was “PMAT-VM”).
- Select the directory location for saving VM files.



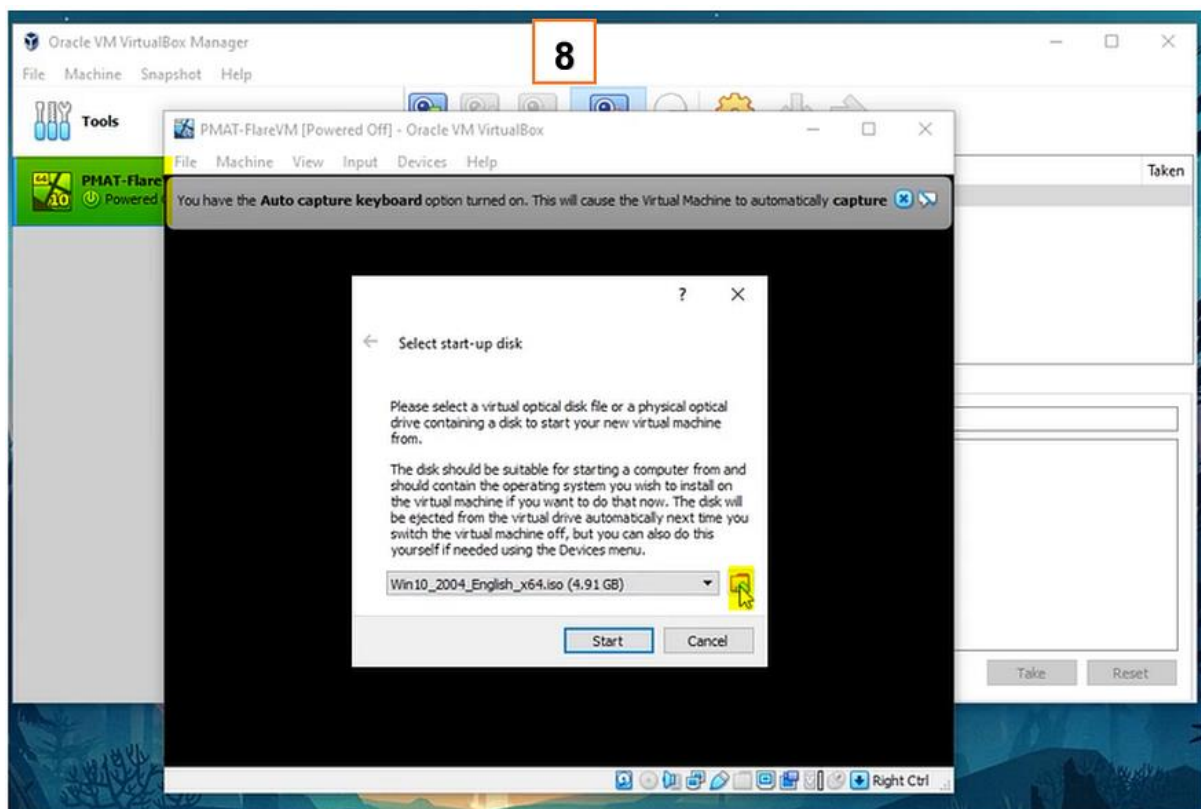
- Allocate minimum 80 GB space for.
- Set the RAM to minimum 4 GB(16GB recommended).



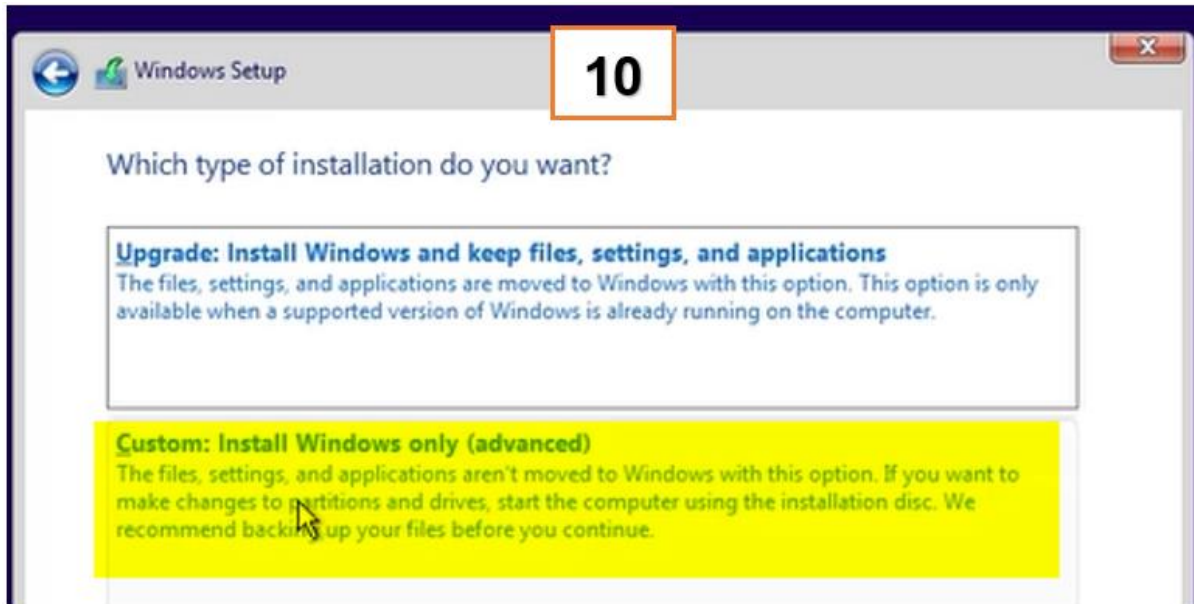


2. Install Windows 10:

- After that run the windows 10.



- Select the 'Windows Enterprise ISO' image file.
- Select custom install windows only(advanced) option and proceed installation.



- Select the necessary unallocated partition.
- Then, Complete the Windows installation process.
- Restart the Main system.

4.2.2 Downloading and Configuring the Flare VM

FLARE-VM Overview:

- 'FLARE-VM' is a Virtual Machine that comes with pre-installed tools for malware analysis and reverse engineering.
1. **Download and Install Chrome Browser inside the windows 10:**
 - i. [Chrome Download Link](https://www.google.com/chrome/) : 'https://www.google.com/chrome/'
 2. Download Flare-VM from [Flare-VM GitHub Repo](https://github.com/mandiant/flare-vm): 'https://github.com/mandiant/flare-vm'
 3. **Optional: Take a Snapshot of the VM by opening machine and take snapshot(for a backup).**
 4. **Select Powershell and 'Run as Administrator' and Download Packages:**
 - i. Download the VCLibs package:

Powershell cmd:

```
"wget https://aka.ms/Microsoft.VCLibs.x54.14.00.Desktop.appx -
usebasicparsing -o VCLibs.appx"
```

14

```
Administrator: Windows PowerShell
PS C:\Users\Public\Desktop> ls

Directory: C:\Users\Public\Desktop

Mode                LastWriteTime         Length Name
----                -
-a----           8/18/2023   3:50 PM           2282 Google Chrome.lnk

PS C:\Users\Public\Desktop> wget https://aka.ms/Microsoft.VCLibs.x64.14.00.Desktop.appx -usebasicparsing -o VCLibs.appx
PS C:\Users\Public\Desktop>
```

- ii. Download the Windows Terminal MSIX bundle:

Powershell cmd:

“wget

https://github.com/microsoft/terminal/releases/download/v1.15.3455.0/Microsoft.WindowsTerminal_Win10_1.15.3455.0_8wekyb3d8bbwe.msixbundle-UseBasicParsing-o-winterterminal.msixbundle”

15

```
Administrator: Windows PowerShell
PS C:\Users\Public\Desktop> wget https://github.com/microsoft/terminal/releases/download/v1.15.3465.0/Microsoft.WindowsTerminal_Win10_1.15.3465.0_8wekyb3d8bbwe.msixbundle -UseBasicParsing -o winterterminal.msixbundle
PS C:\Users\Public\Desktop>
```

5. Install the Packages:

- i. Install VCLibs:

Powershell cmd:

“Add-AppxPackage [C:\path\to\downloaded\VCLibs.appx]”

- ii. Install Windows Terminal:

Powershell cmd:

“Add-AppxPackage [C:\path\to\downloaded\winterterminal.msixbundle]”

16

```
Administrator: Windows PowerShell
PS C:\Users\Public\Desktop> Add-AppxPackage .\VCLibs.appx
PS C:\Users\Public\Desktop> Add-AppxPackage .\winterterminal.msixbundle
PS C:\Users\Public\Desktop>
```

6. Adjust Windows VM Settings:

- i. Disable the proxy auto-detect Settings.

- Search for “Proxy Settings” in Windows Search bar
- Turn off “Automatically Detect Settings”

- ii. Disable the Tamper Protection.

- Search for “Defender” then open Defender settings and turn off all Defender Settings
- iii. Disable the Windows Defender through Group Policy (GPO).
- Search For “group policy” in windows search bar
 - In GPO, go through path: ‘ *Administrative Templates then Windows Components then Microsoft Defender Antivirus then Enable* ’
 - Turn Off “Microsoft defender”
- iv. Disable the Windows Firewall through GPO.
- After that in GPO, go through path: ‘ Administrative Templates then go to Network then Network Connections then Windows Defender Firewall then Standard profile ’
 - Turn off “Protect All Network Connections”
- v. **Note:** If windows Defender is not turned off correctly, it will cause problems in Flare-VM installation. Use [Defender Control](https://www.sordum.org/9480/defender-control-v2-1/) to fully disable Windows Defender if necessary.

Defender control: ‘<https://www.sordum.org/9480/defender-control-v2-1/>’

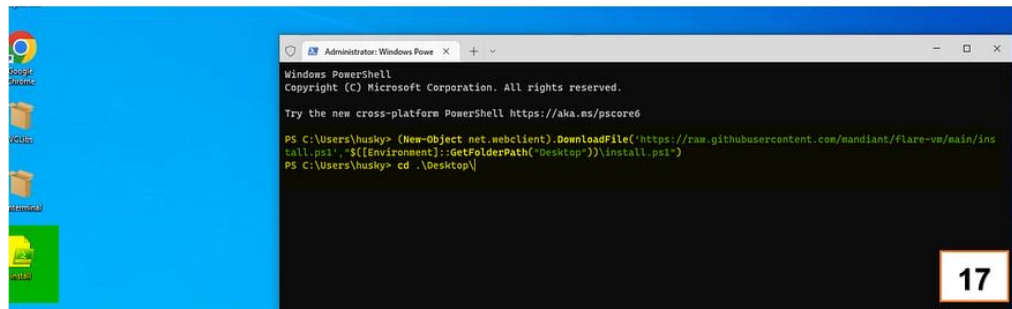


7. FLARE-VM installation:

- i. Download install.ps1 file:

Powershell:

```
“(New-Object
net.webclient).DownloadFile('https://raw.githubusercontent.com/mandiant/flare-vm/main/install.ps1',"$([Environment]::GetFolderPath('Desktop'))\install.ps1"
)”
```

- ii. Using 'cd' Change directory to Desktop and then run the unblock-file:

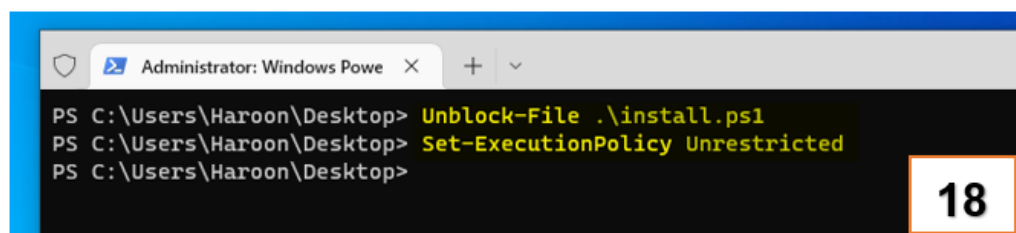
Powershell:

“Unblock-File .\install.ps1”

- iii. Set up execution policy to unrestricted:

Powershell:

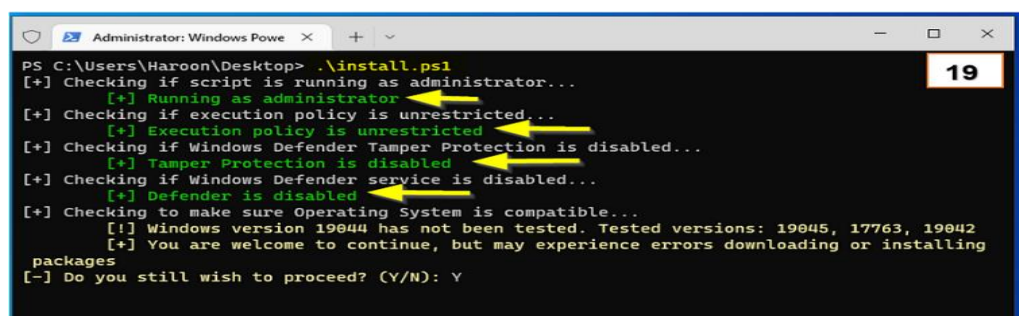
“Set-ExecutionPolicy Unrestricted”

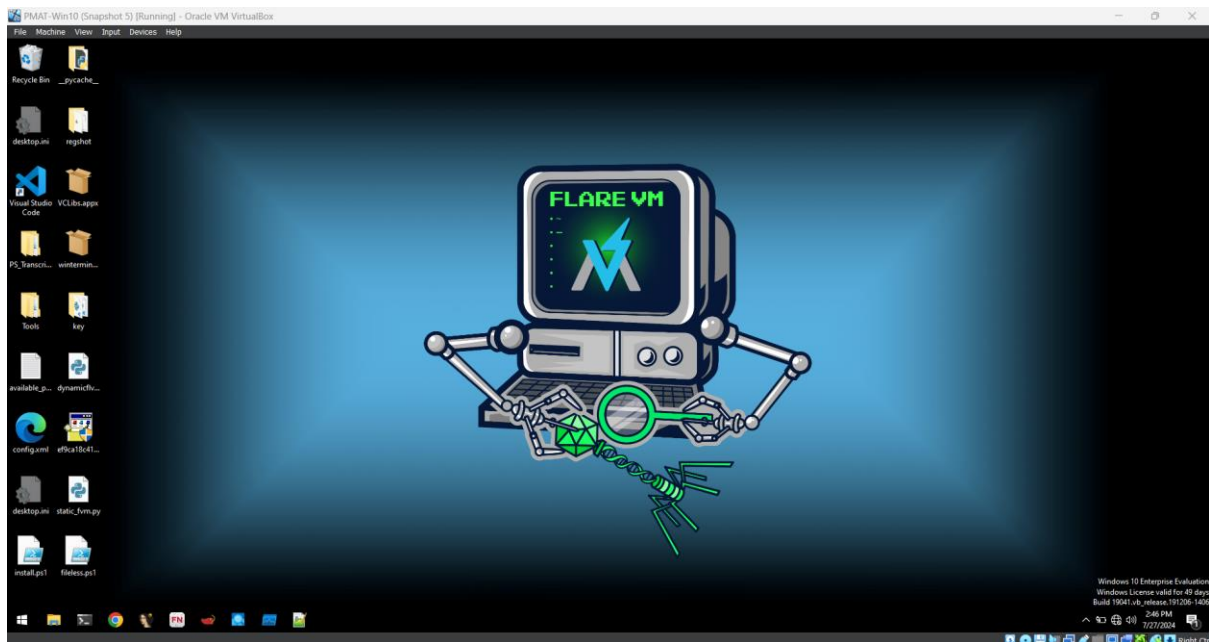


- iv. Run the install.ps1 file with custom configuration:

Powershell:

“.\install.ps1 -customConfig”





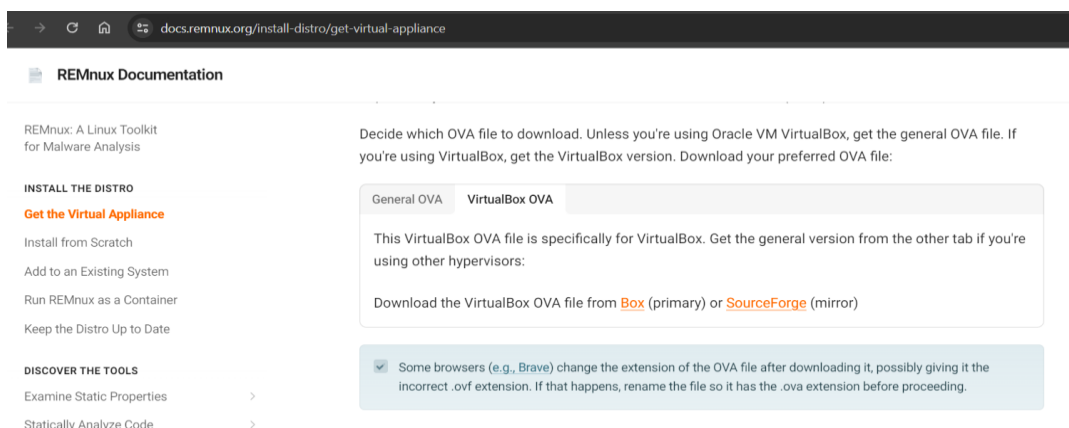
Rename the Windows VM:

- Some malwares can identify sandboxes by viewing username so to avoid that rename the VM to something that doesn't hint at a sandbox environment. Go to 'Settings' then 'System' then 'About', and click the 'Rename this PC' button.

4.2.3 Setting up an Isolated Network for LAB

REMnux Overview:

- Note: REMnux is an Open-source Linux distribution specialized in malware analysis and Reverse Engineering.
- Next step is to Download the [REMnux OVA File for VirtualBox from :](https://docs.remnux.org/install-distro/get-virtual-appliance) <https://docs.remnux.org/install-distro/get-virtual-appliance>



1. Import the REMnux OVA file in VirtualBox:

- Start the REMnux VM and ensure that it is running properly.

← Import Virtual Appliance

Appliance to import

Please choose the source to import appliance from. This can be a local file system to import OVF archive or one of known cloud service providers to import cloud VM from.

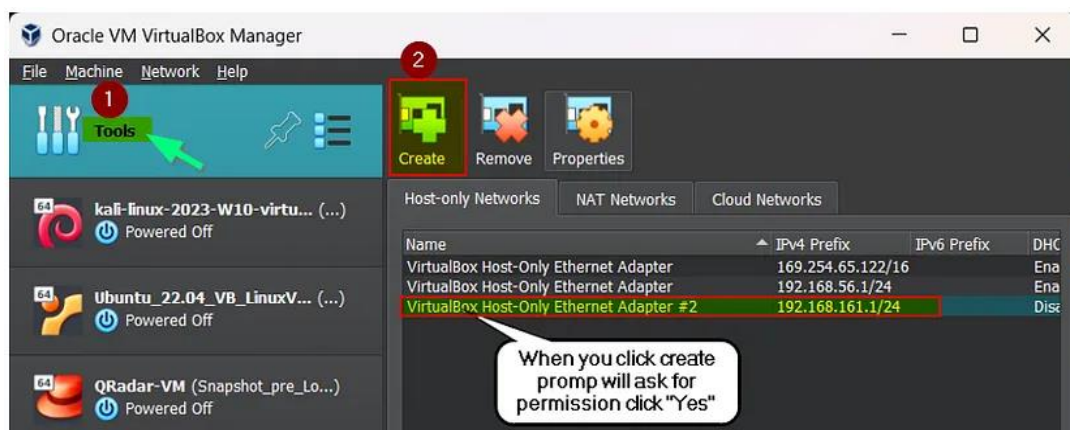
Source: Local File System

Please choose a file to import the virtual appliance from. VirtualBox currently supports importing appliances saved in the Open Virtualization Format (OVF). To continue, select the file to import below.

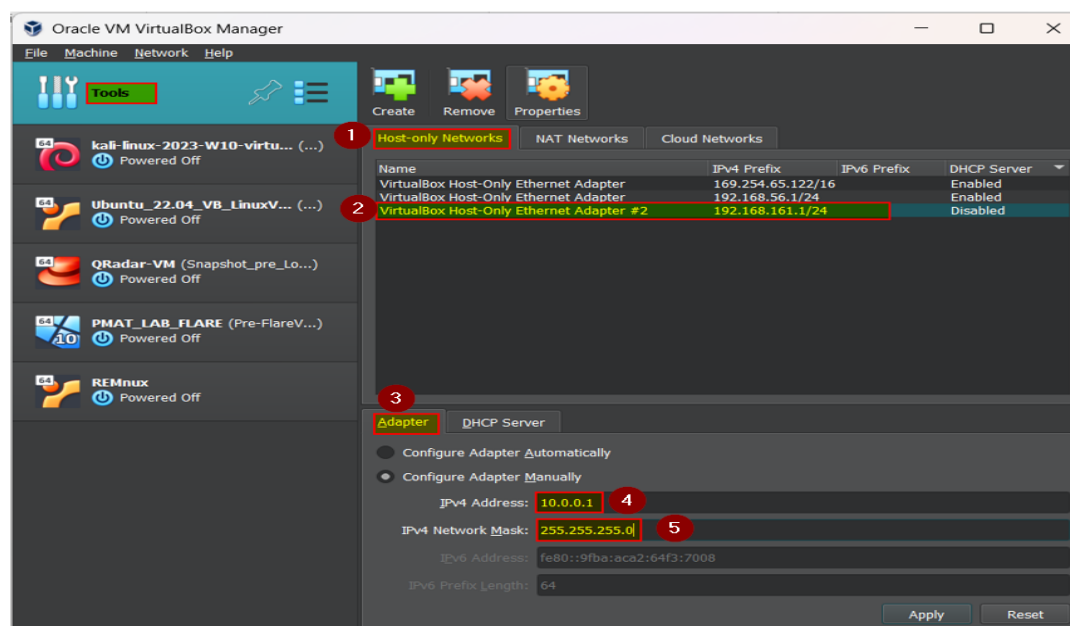
File: ls\remnux-v7-focal-virtualbox.ova

2. Create a custom Host-Only Network Adapter:

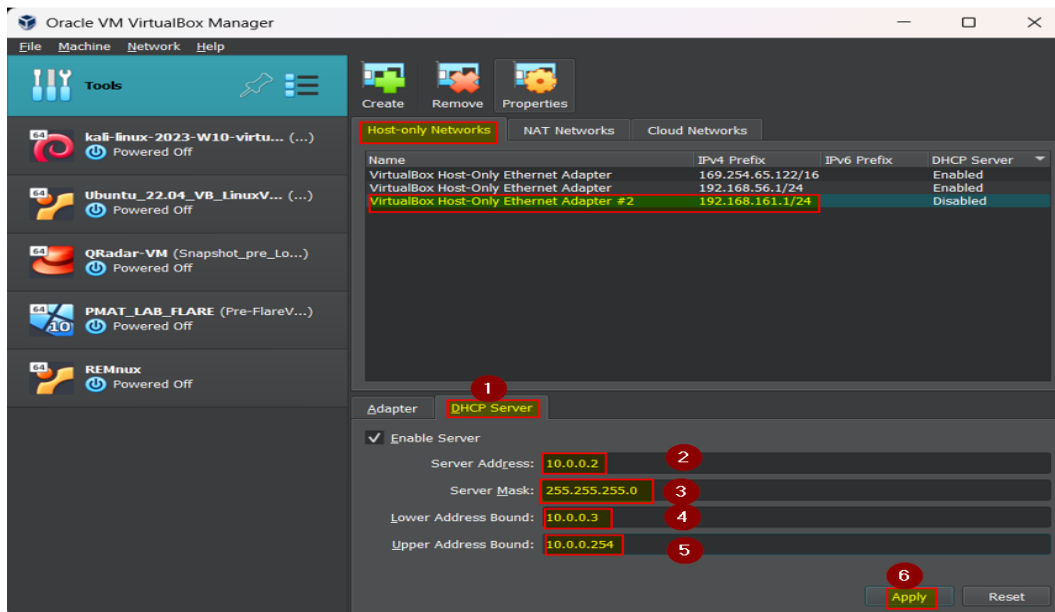
- Open 'VirtualBox then Tools then Create'.



- Configure the adapter address and the DHCP server settings.

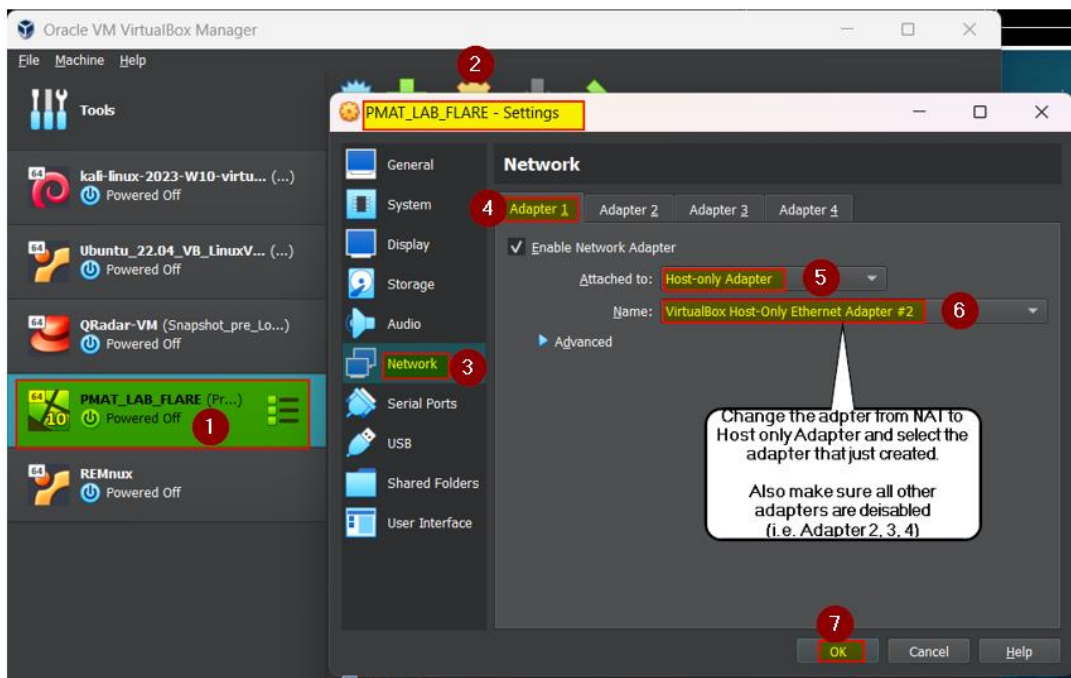


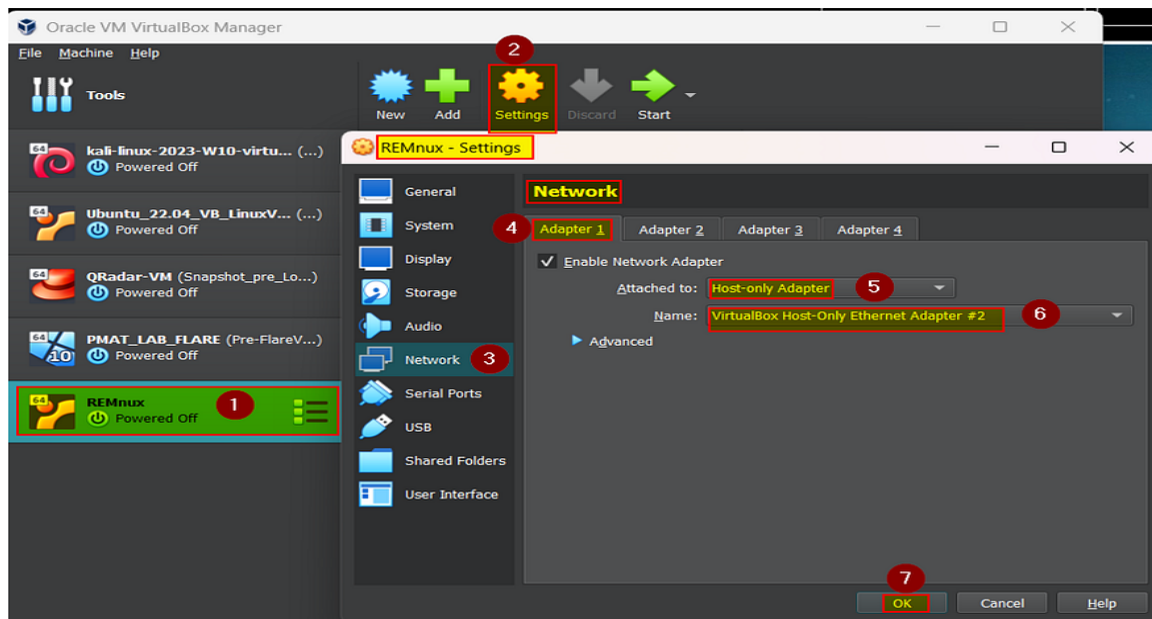
- Set custom Static IP addresses for both OS.



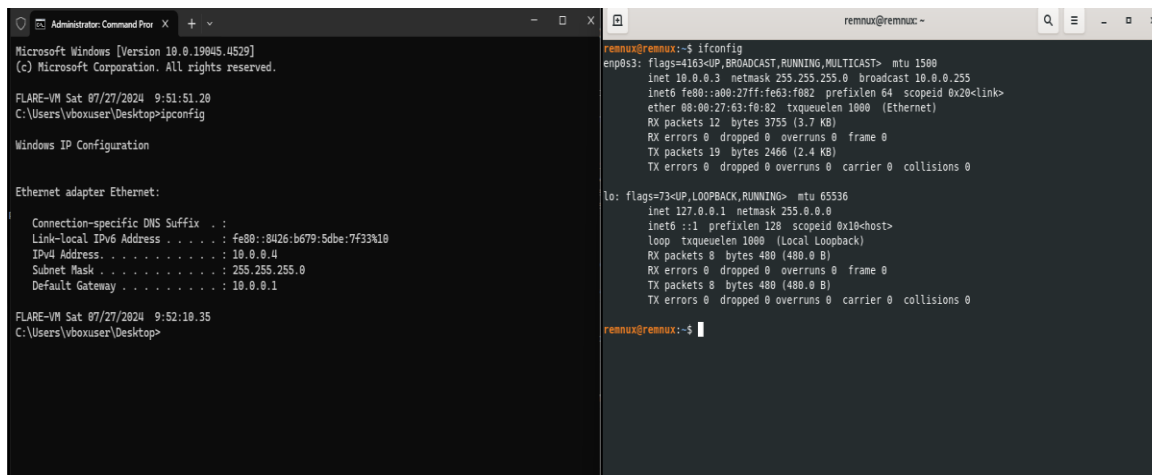
3. Change the Network Settings:

- Put both 'FlareVM' and 'REMnux' to 'Host-Only' Adapter.

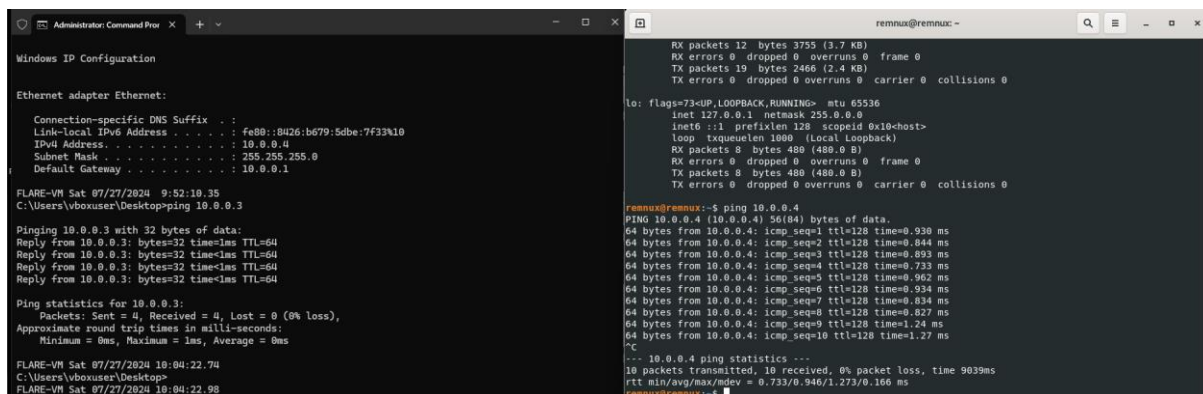




4. Test the Network Connection:



- By using 'ping' command make sure that both VMs can communicate each other internally.



- Also make sure it cannot access the real network.

```

FLARE-VM Sat 07/27/2024 10:04:22.74
C:\Users\vboxuser\Desktop>
FLARE-VM Sat 07/27/2024 10:04:22.98
C:\Users\vboxuser\Desktop>ping 192.168.56.1

Pinging 192.168.56.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

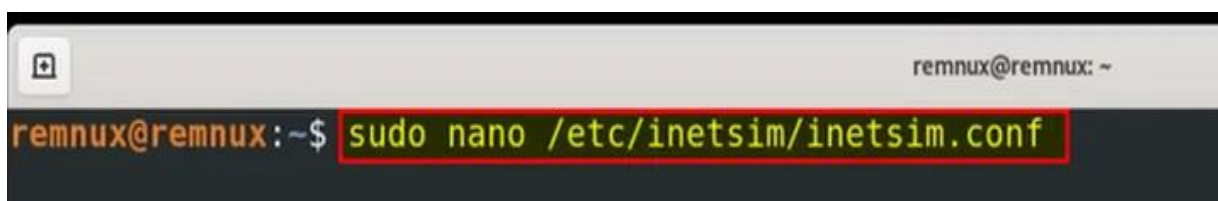
Ping statistics for 192.168.56.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

FLARE-VM Sat 07/27/2024 10:11:34.56
C:\Users\vboxuser\Desktop>
desktop.ini  static_fm.py

```

5. Set up a Fake DNS Server:

- By Enabling DNS service in REMnux by configuring file 'inetsim.conf'.



```

remnux@remnux: ~$ sudo nano /etc/inetsim/inetsim.conf

```

- Remove '#' from '#start_service dns' to make it uncomment

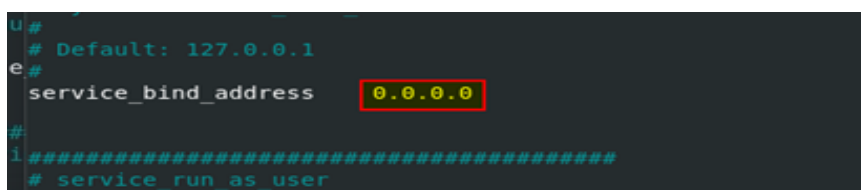


```

GNU nano 4.8 /etc/inetsim/inetsim.conf
# Available service names are:
# dns, http, smtp, pop3, tftp, ftp, ntp, time_tcp,
# time_udp, daytime_tcp, daytime_udp, echo_tcp,
# echo_udp, discard_tcp, discard_udp, quotd_tcp,
# quotd_udp, chargen_tcp, chargen_udp, finger,
# ident, syslog, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, irc, https
#
start_service dns
start_service http
start_service https
start_service smtp
start_service smtps
start_service pop3
start_service pop3s
start_service ftp
start_service ftps
#start_service tftp
#start_service irc

```

- Change 'service_bind_address' to 0.0.0.0



```

U#
# Default: 127.0.0.1
e#
service_bind_address 0.0.0.0
#
1 #####
# service_run_as_user

```


- Change the 'dns_default_ip' to REMnux VM IP address

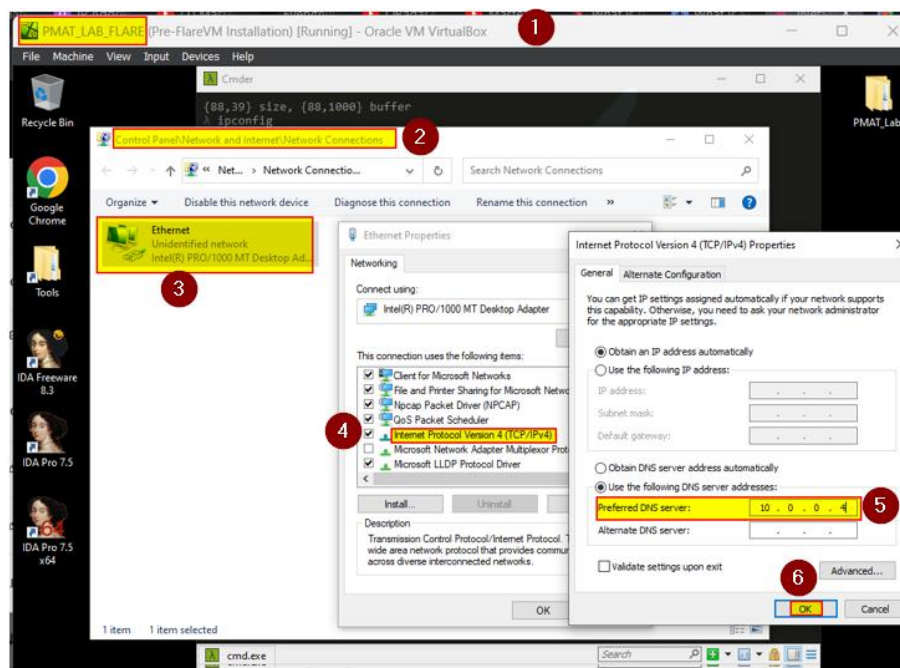
```
# Syntax: dns_default_ip <IP address>
#
# Default: 127.0.0.1
#
dns_default_ip 10.0.0.4
#####
# dns_default_hostname
```

IP Address of Remnux server that was 10.0.0.4

```
remnux@remnux: ~
remnux@remnux:~$ sudo nano /etc/inetsim/inetsim.conf
remnux@remnux:~$ inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 1782) ===
Session ID: 1782
Listening on: 10.0.0.4
Real Date/Time: 2023-08-30 16:00:26
Fake Date/Time: 2023-08-30 16:00:26 (Delta: 0 seconds)
Forking services...
* dns_53_tcp_udp - started (PID 1786)
* http_80_tcp - started (PID 1787)
* smtp_25_tcp - started (PID 1789)
* smtps_465_tcp - started (PID 1790)
* pop3s_995_tcp - started (PID 1792)
* pop3_110_tcp - started (PID 1791)
* ftps_990_tcp - started (PID 1794)
* https_443_tcp - started (PID 1788)
* ftp_21_tcp - started (PID 1793)
done.
Simulation running.
```

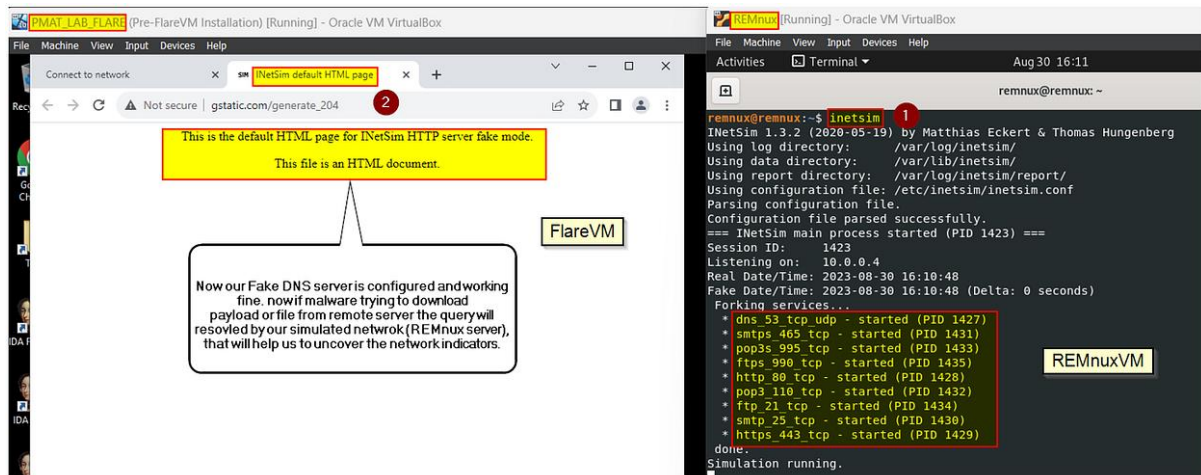
DNS service is started

- In FlareVM, set the DNS server IP to REMnux VM IP.

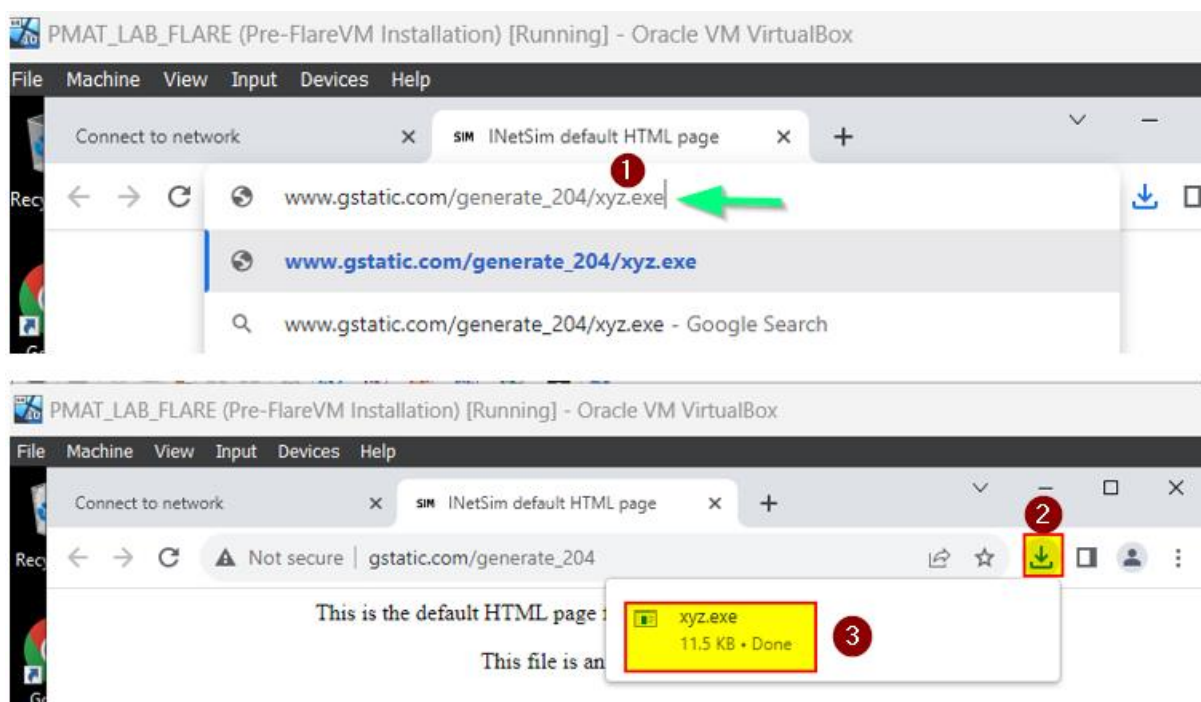


6. Verify the Setup:

- Make sure that REMnux is appropriately answering the DNS queries

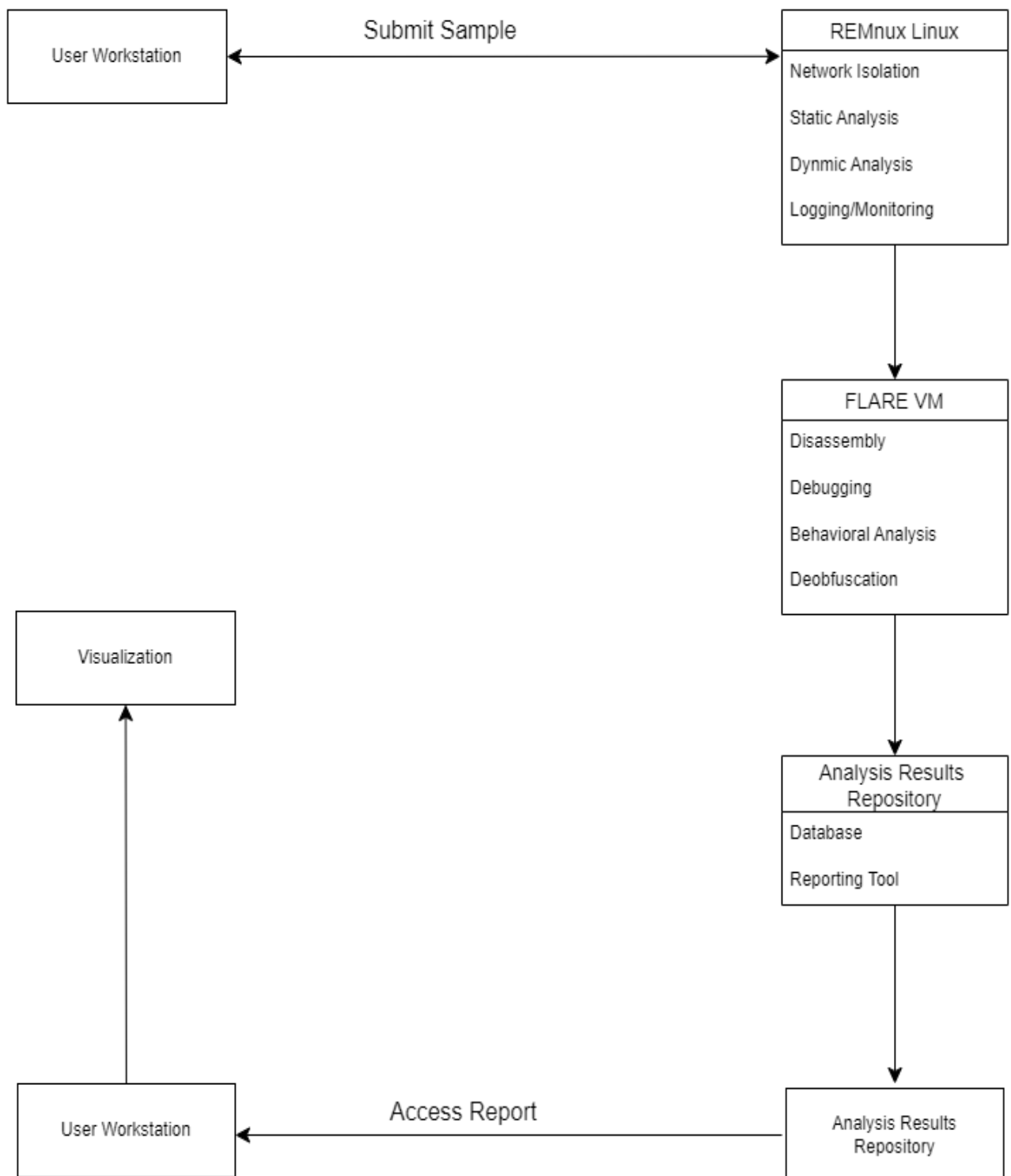


- Modify the current URL by adding '/xyz.exe'

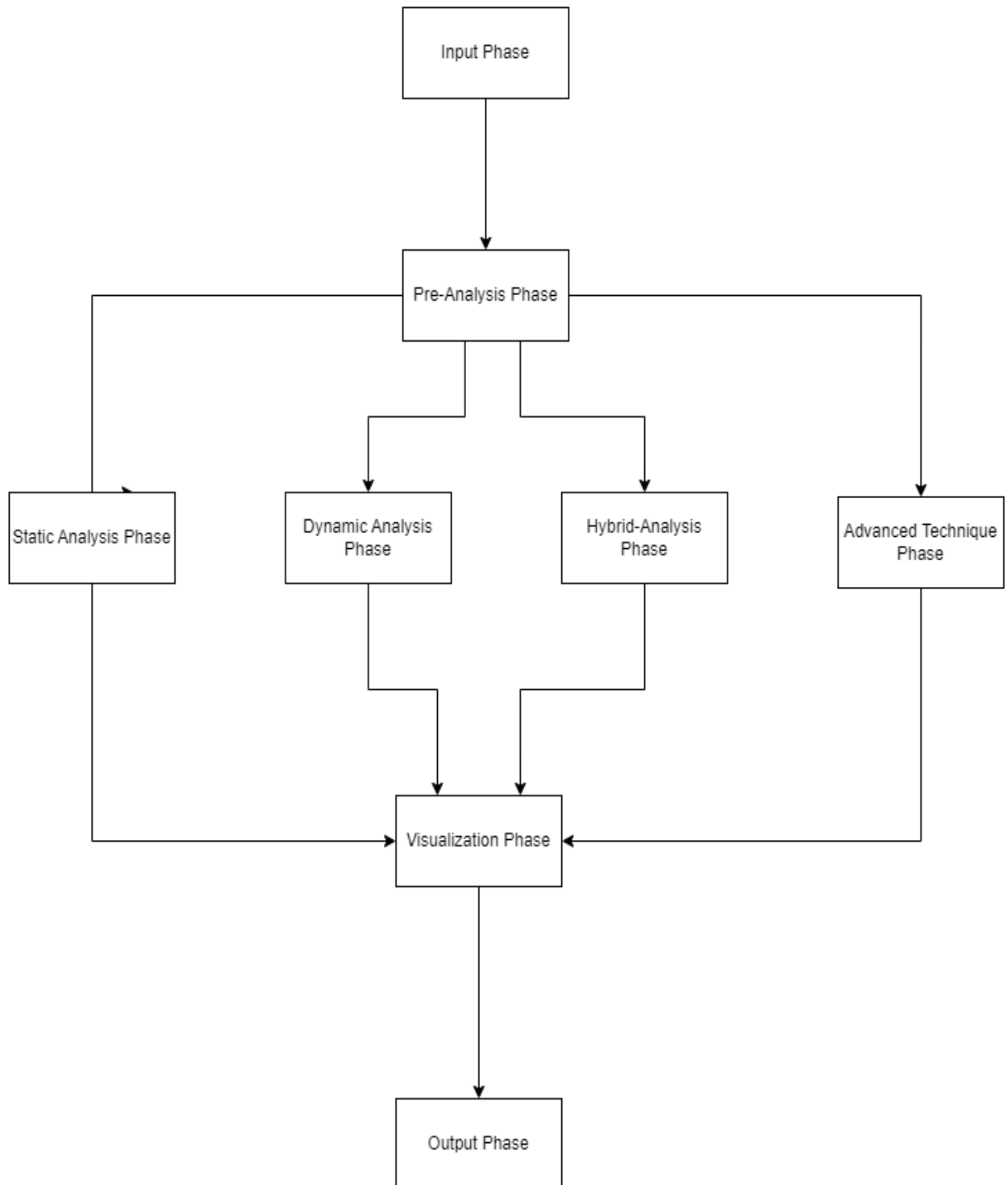


[6]

4.3 Analysis Workflow



4.4 EXPLAINED WORKFLOW DIAGRAM



4.5 Visualization

Tools based on Python are used to display the data gathered from the analytical processes. The analytical findings are displayed using interactive web apps built with Flask and Dash. The activity of the virus, network connections, and system interactions are all shown intelligibly by these visualizations, which facilitate analysts' interpretation of the data and help them reach relevant conclusions.

- Flask: A tool for developing web applications' backend that controls data flow and integrates with analytic tools.
- Dash: This tool is used to create dynamic and interactive visuals that let analysts work with and study data.

4.6 Visualization Effectiveness

The presentation of findings has been greatly improved by visualization tools, enabling a clearer comprehension of complicated data. It is now simpler for analysts to explain their findings and reach relevant conclusions thanks to the development of dynamic and user-friendly visualizations made possible by the usage of Flask and Dash. Patterns, connections, and abnormalities in the data are facilitated by these representations.

CHAPTER 5

CODING

5 CODING

5.1 INTRODUCTION

Coding is essential to the Sentinel: The Malware Analyzer's development since it allows for the display of analysis findings and the use of custom scripts. This project uses cutting-edge programming approaches to create a comprehensive malware analysis platform with automated analysis features and seamless visualization.

- Custom Python scripts: Designed to handle different parts of malware investigation, these scripts form the core of Sentinel. These scripts are in charge of collecting behavioral data, doing static analysis, and running malware samples in safe sandbox conditions. In addition, they oversee duties including pattern recognition, code decomposition, and system interface monitoring. The scripts provide comprehensive and effective analysis of many malware kinds, including fileless malware, which needs special handling because it is not file-based, by automating these operations
- Visualization Backend: Sentinel's visualization component has a Python-developed backend of its own. Data processing and integration with the analysis findings are handled by this backend. It arranges, aggregates, and formats the data in accordance with frontend requirements in order to get it ready for display. This backend is essential for controlling data flow and guaranteeing that the results of the analyses are appropriately reflected in the visualizations.
- Visualization Frontend: An interactive and user-friendly way to see the data is the goal of the visualization component's front end. The interface generates interactive heat maps, network graphs, timeline views, and dashboards using libraries like Plotly and Dash. By offering understandable, comprehensible depictions of malware activity and patterns, these visualizations assist security professionals in deciphering complicated data. A smooth user experience is ensured by the frontend's interaction with the visualization backend to retrieve and show data.

Sentinel: The Malware Analyzer, in short, combines a powerful visualization component with its own frontend and backend with bespoke scripts for automated malware analysis. Frontend developers create engaging and educational visuals; backend developers handle data integration and processing. The platform's capacity to recognize, evaluate, and comprehend malware threats is improved by this blend of sophisticated visualization methods and customized scripting.

5.2 CODING

5.2.1 Scripts

a. Static analysis in FlareVM:

```
import subprocess

import hashlib

import os

# Define the file to be analyzed

file_path =
r"C:\Users\vboxuser\Desktop\ef9ca18c417cd6cba8249dfdd943f4a993a4536c89
f12adbd05496855c2a2e3c.exe"

analysis_dir = r"C:\analysis"

ida_path = r"C:\Tools\ida_launcher.exe" # Update to ida_launcher.exe

strings_path = r"C:\Tools\sysinternals\strings.exe"

# Create analysis directory if it doesn't exist

if not os.path.exists(analysis_dir):

    os.makedirs(analysis_dir)

# Calculate file hashes

hashes = {}

for algo in ['md5', 'sha1', 'sha256']:

    hash_func = hashlib.new(algo)

    with open(file_path, 'rb') as f:

        while chunk := f.read(8192):

            hash_func.update(chunk)
```

```

hashes[algo] = hash_func.hexdigest()

# Save hashes to a report file
with open(f'{analysis_dir}\\hashes.txt', "w") as f:
    for algo, hash_value in hashes.items():
        f.write(f'{algo.upper()}: {hash_value}\n')

# Extract strings using strings.exe
result = subprocess.run([strings_path, "-a", file_path], capture_output=True,
text=True)
with open(f'{analysis_dir}\\strings.txt', "w") as strings_file:
    strings_file.write(result.stdout)
if result.stderr:
    print("Error output:", result.stderr)

# Check if IDA Pro path exists
if not os.path.exists(ida_path):
    print(f'IDA Pro executable not found at: {ida_path}')
else:
    # Disassemble the file using IDA Pro launcher
    subprocess.run([ida_path, "-B", file_path])

# Generate a report
with open(f'{analysis_dir}\\report.txt', "w") as report:
    report.write("Hashes:\n")
    with open(f'{analysis_dir}\\hashes.txt') as hashes_file:

```

```

report.write(hashes_file.read())

report.write("\nStrings:\n")

with open(f'{analysis_dir}\\strings.txt') as strings_file:
    report.write(strings_file.read())

print("Static analysis completed. Check the C:\\analysis folder for results.")

```

b. Dynamic Analysis in FlareVM:

```

import subprocess
import time
import os

# Define the file to be analyzed and directories
file_path = r"C:\Users\ vboxuser\Downloads\xyz.exe"
analysis_dir = r"C:\analysis"

# Ensure analysis directory exists
if not os.path.exists(analysis_dir):
    os.makedirs(analysis_dir)

# Path to Procmon and Wireshark
procmon_path = r"C:\Tools\sysinternals\Procmon.exe"
wireshark_path = r"C:\Program Files\Wireshark\Wireshark.exe"

# Start Process Monitor to capture system changes
try:
    procmon = subprocess.Popen([procmon_path, "/Quiet", "/Minimized",
f"/Backingfile {os.path.join(analysis_dir, 'procmon.pml')}"])
except Exception as e:
    print(f'Failed to start Procmon: {e}')
    exit(1)

# Start Wireshark to capture network traffic
try:

```

```

        wireshark = subprocess.Popen([wireshark_path, "-k", "-i", "1", f'-w
        {os.path.join(analysis_dir, 'traffic.pcap')}'])
    except Exception as e:
        print(f'Failed to start Wireshark: {e}')
        procmon.terminate()
        exit(1)

# Run the malware sample
try:
    subprocess.run([file_path], check=True)
except Exception as e:
    print(f'Failed to run malware sample: {e}')
    procmon.terminate()
    wireshark.terminate()
    exit(1)

# Wait for the malware to execute (adjust the sleep time as needed)
time.sleep(60)

# Stop Process Monitor and Wireshark
procmon.terminate()
wireshark.terminate()

print("Dynamic analysis completed. Check the C:\\analysis folder for
results.")

```

c. Static Analysis in REMnux linux :

```

import subprocess

import hashlib

# Define the file to be analyzed
file_path = "/path/to/malware/sample.exe"
analysis_dir = "/analysis"

```



```

# Calculate file hashes

hashes = {}

for algo in ['md5', 'sha1', 'sha256']:
    hash_func = hashlib.new(algo)
    with open(file_path, 'rb') as f:
        while chunk := f.read(8192):
            hash_func.update(chunk)
    hashes[algo] = hash_func.hexdigest()

# Save hashes to a report file

with open(f'{analysis_dir}/hashes.txt', "w") as f:
    for algo, hash_value in hashes.items():
        f.write(f'{algo.upper()}: {hash_value}\n')

# Extract strings using strings command

subprocess.run(["strings", file_path, ">", f'{analysis_dir}/strings.txt'],
shell=True)

# Disassemble the file using radare2

subprocess.run(["r2", "-A", "-q", "-c", "aaa;afl", file_path, ">",
f'{analysis_dir}/disassembly.txt"], shell=True)

# Generate a report

with open(f'{analysis_dir}/report.txt', "w") as report:
    report.write("Hashes:\n")

```

```

with open(f'{analysis_dir}/hashes.txt') as hashes_file:
    report.write(hashes_file.read())
report.write("\nStrings:\n")
with open(f'{analysis_dir}/strings.txt') as strings_file:
    report.write(strings_file.read())
report.write("\nDisassembly:\n")
with open(f'{analysis_dir}/disassembly.txt') as disassembly_file:
    report.write(disassembly_file.read())

print("Static analysis completed. Check the /analysis folder for results.")

```

d. Dynamic Analysis in REMnux Linux:

```

import subprocess
import time
import os
import signal

# Define the file to be analyzed
file_path = "/path/to/malware/sample.exe"
analysis_dir = "/analysis"

# Start Wireshark to capture network traffic
wireshark = subprocess.Popen(["wireshark", "-k", "-i", "eth0", f'-w {analysis_dir}/traffic.pcap'])

# Run the malware sample

```

```

malware = subprocess.Popen(["wine", file_path])

# Wait for the malware to execute (adjust the sleep time as needed)
time.sleep(60)

# Stop Wireshark
os.kill(wireshark.pid, signal.SIGTERM)

# Stop the malware process
malware.terminate()

print("Dynamic analysis completed. Check the /analysis folder for results.")

```

5.2.2 Visualization:

a. Backend:

```

import os
import json
from flask import Flask, jsonify
from flask_cors import CORS
import pandas as pd

app = Flask(__name__)
CORS(app)

# Function to read data from a JSON file

```

```

def read_data_from_file(file_path):
    if os.path.exists(file_path):
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    else:
        return []

@app.route('/')
def index():
    return "Flask server is running!"

@app.route('/api/data', methods=['GET'])
def get_data():
    file_path = r"C:\Users\shain\OneDrive\Desktop\output.json" # Corrected file path
    data = read_data_from_file(file_path)
    df = pd.DataFrame(data)
    return jsonify(df.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```

b. Frontend:

```

import dash
from dash import dcc, html
import dash_bootstrap_components as dbc
import requests
import pandas as pd
from dash.dependencies import Input, Output
import plotly.express as px

# IP address of your FLARE VM
FLARE_VM_IP_ADDRESS = '192.168.56.1'
API_URL = f'http://{FLARE_VM_IP_ADDRESS}:5000/api/data'

# Fetch data from the Flask backend
def fetch_data():
    response = requests.get(API_URL)
    data = response.json()
    df = pd.DataFrame(data)
    df['timestamp'] = pd.to_datetime(df['timestamp'])
    return df

# Initialize the Dash app
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])

app.layout = dbc.Container([
    dbc.Row([

```

```

        dbc.Col(html.H1("Sentinel: Malware Analyzer Visualization",
className='text-center mb-4'), width=12)

    ),
    dbc.Row([
        dbc.Col(dcc.Graph(id='timeline-graph'), width=12),
        dbc.Col(dcc.Graph(id='activity-bar-chart'), width=12),
        dbc.Col(dcc.Graph(id='activity-pie-chart'), width=12)
    ])
])

```

```

@app.callback(
    Output('timeline-graph', 'figure'),
    [Input('timeline-graph', 'id')]
)
def update_timeline_graph(_):
    df = fetch_data()

    print("DataFrame Columns:", df.columns) # Debug line
    print("DataFrame Sample:", df.head()) # Debug line

    fig = px.scatter(df, x='timestamp', y='activity', color='severity',
hover_data=['details', 'process_name', 'network_ip'])

    fig.update_layout(title='Malware Activity Timeline', xaxis_title='Time',
yaxis_title='Activity')

    return fig

```

```

@app.callback(
    Output('activity-bar-chart', 'figure'),

```

```

    [Input('activity-bar-chart', 'id')]
)
def update_bar_chart(_):
    df = fetch_data()
    activity_count = df['activity'].value_counts().reset_index()
    activity_count.columns = ['activity', 'count']
    fig = px.bar(activity_count, x='activity', y='count', title='Activity Count')
    return fig

@app.callback(
    Output('activity-pie-chart', 'figure'),
    [Input('activity-pie-chart', 'id')]
)
def update_pie_chart(_):
    df = fetch_data()
    activity_count = df['activity'].value_counts().reset_index()
    activity_count.columns = ['activity', 'count']
    fig = px.pie(activity_count, names='activity', values='count', title='Activity
Distribution')
    return fig
if __name__ == '__main__':
    app.run_server(debug=True, host='127.0.0.1')

```

CHAPTER 6

RESULTS AND DISCUSSION

6.1 PROJECT OBJECTIVE

The Malware Analysis Sandbox project aims to create a comprehensive, reliable, and easy-to-use platform for malware analysis and detection. The goal of this project is to meet the urgent demand for sophisticated malware analysis tools in the constantly developing cybersecurity industry. The initiative guarantees that security experts can efficiently recognize, evaluate, and reduce malware risks by combining various analytical approaches with cutting-edge technology.

The primary aim is to develop and execute a malware analyzer that can do static, dynamic, and hybrid analysis, bolstered by user-friendly visualization strategies. This guarantees that the analysis's findings are presented in an understandable manner, making it easier to spot trends and abnormalities. The following are the principal goals of this project:

- Examine and use analytic methods that are static, dynamic, and hybrid and that are appropriate for thorough malware analysis.
- When performing dynamic analysis, provide a stable sandbox environment in which malware may be executed and seen without risk.
- Incorporate sophisticated visual aids into your study to deliver findings in a clear and understandable way.
- Achieve great precision in identifying malware and analyzing its activity to make a substantial impact in the field of cybersecurity.
- Assist in the analytical process by offering a safe and secure platform that preserves sensitive data and data integrity.

6.2 SCOPE

This is a sophisticated project that aims to create a comprehensive platform for malware analysis and detection. It is designed and implemented with great care. This project's primary goals are as follows:

- Recognize malware analysis methodologies and their applications.
 - Develop a thorough understanding of the methodologies used in hybrid, dynamic, and static analysis.
 - Investigate how malware behavior is presented and interpreted via visualization.
- Examine the frameworks and technologies currently in use for malware analysis.
 - Look at industry-standard malware analysis environments such as FLARE VM, REMnux, and others.
 - Look at the data representation tools and libraries that are already available.
- Recognize the needs of platform, such as the kinds of analyses it can handle, its speed constraints, and security hazards.

- Describe the kinds of malware that the platform will look for and the particular methods that will be needed for analysis.
- Ascertain performance benchmarks and make sure the platform is capable of effectively managing thorough analysis.
- Make that, in order to safeguard sensitive data during analysis, the platform complies with stringent security protocols.
- Examine several malware analysis techniques and select the ones that most closely match your use case.
 - Examine the benefits and drawbacks of using static, dynamic, and hybrid analytic methods.
 - Choose the most efficient mix of analysis techniques to deliver thorough malware findings.
 - Incorporate cutting-edge visualization techniques to improve the analysis results' accessibility and clarity.

6.3 SUMMARY OF THE FINDINGS

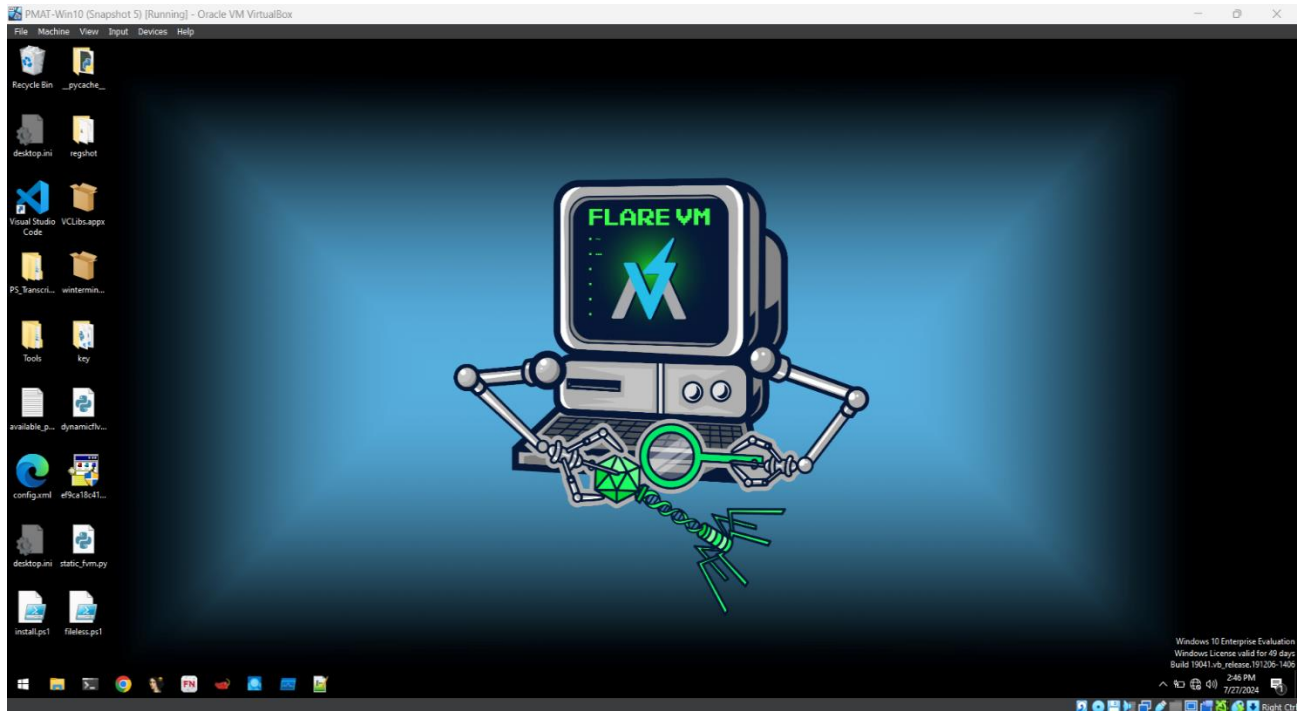
‘Sentinel: The Malware Analyzer’, an extensive and cutting-edge platform for accurate and comprehensive malware analysis and detection, is the main goal of this project. To offer a multifaceted approach to comprehending malware activity, the system combines static analysis, dynamic analysis, and hybrid analysis methodologies.

- **Static Analysis:** To find known malware signatures without running the code, the system disassembles the code and matches patterns. This approach has drawbacks, especially when dealing with fileless malware, even if it works well for rapid detection. Due to its non-permanent nature and absence of file-based signatures, fileless malware, which runs directly in memory and bypasses conventional file systems, cannot be efficiently identified using static analysis alone.
- **Dynamic Analysis:** Sentinel runs malware in a sandbox environment to track its movements in real time. This method reveals behavioral patterns that static analysis can overlook and offers insights into system relationships, network activity, and resource use. For example, fileless malware can be seen by its memory-based activities, including the execution of scripts or the behaviors of malicious processes, which would otherwise elude static detection techniques. By operating in a controlled environment, the sandbox environment assures that the virus cannot really cause any harm.

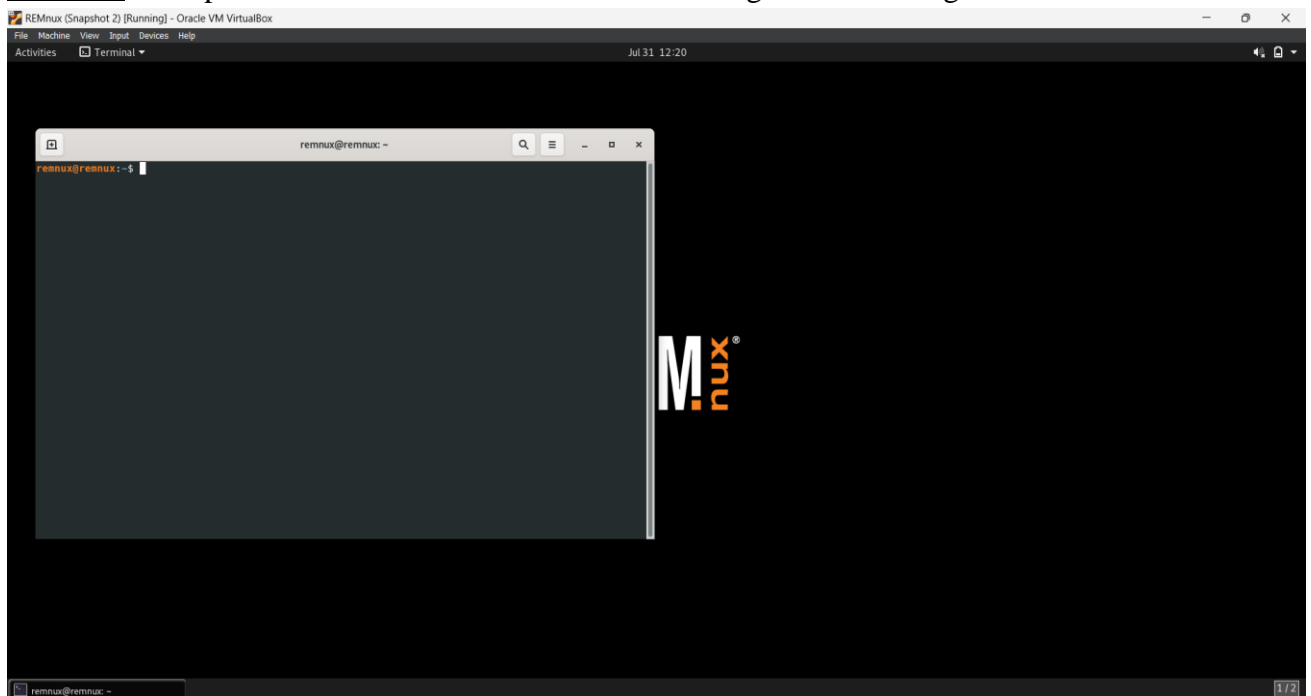
- **Hybrid Analysis:** Sentinel improves detection accuracy and offers a thorough insight of malware activity by fusing static and dynamic analysis techniques. Cross-referencing results using pattern recognition and correlation analysis reveals intricate dangers, such as fileless malware, which may avoid detection with just one technique.
- **Visualization Tools:** To display analytic findings in an understandable and clear manner, sophisticated visualization techniques are used. Security analysts may make educated decisions more quickly by identifying patterns, anomalies, and trends with the use of interactive dashboards, heat maps, network graphs, and timeline views. This is especially helpful for comprehending the behavior of advanced threats, such as fileless malware, which can display complex and erratic patterns of activity.
- **Security and Compliance:** Throughout the analytical process, Sentinel guards the privacy and accuracy of the data. To safeguard sensitive data, role-based access rules are in place and all data transmissions and storage are encrypted. Robust data privacy and protection is ensured by the system's compliance with industry standards and laws, including the CCPA and GDPR.
- **Thorough Reporting:** Results from static, dynamic, and hybrid analysis are combined to provide comprehensive reports. Security experts may gain a comprehensive picture of malware behaviors, including fileless malware, from these reports, which offer actionable insights. The reports' thoroughness improves the capacity to efficiently address attacks and adjust to changing malware strategies.

6.4 RESULTS

FlareVM: an open-source, publicly accessible security distribution for Windows that is intended for use by forensicators, penetration testers, incident responders, malware analysts, and reverse engineers.

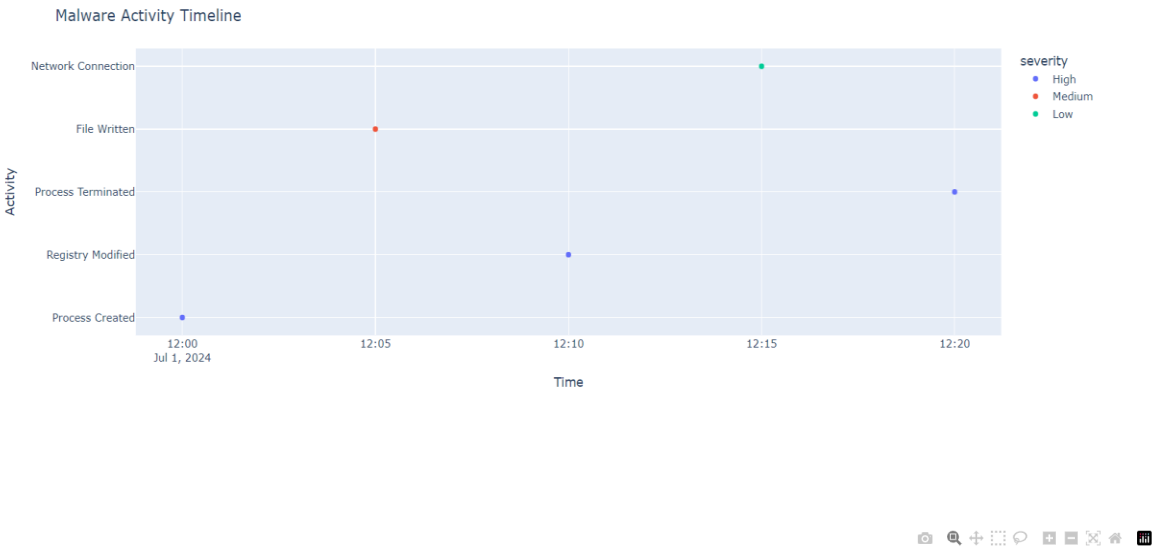


REMnux: An open-source Linux toolkit for deconstructing and examining malware



Visualization:

Sentinel: Malware Analyzer Visualization



Activity Distribution



CHAPTER 7

SCOPE FOR FUTURE WORK

7. SCOPE FOR FUTURE WORK

- **Increased Efficiency:**
 - Algorithm Improvements: To cut down on overhead and boost the speed and precision of threat identification, keep refining the algorithms used for static and dynamic malware analysis.
 - Real-time analytics: To enable prompt identification and reaction to new threats, improve the system's real-time analytic capabilities.
- **Enhanced Scalability:**
 - Cloud Integration: Modify Sentinel to run on cloud platforms, making use of cloud resources to scale up and down quickly and manage higher workloads as necessary.
 - Distributed Computing: Investigate the application of distributed computing methods to enhance the system's capacity to effectively handle massive amounts of data.
- **Enhanced Visualization :**
 - Mobile Compatibility: Create apps for smartphones and tablets that offer safe and convenient malware scanning features.
 - GUI that is intuitive: Create a more complex graphical user interface that improves usability and user interaction. Possible features include the ability to drag and drop files, real-time analysis findings, and thorough history monitoring.
 - Interactive Reporting: Put in place sophisticated reporting tools that let users alter and engage with analytical results, offering more comprehensive understanding and enhanced assistance for making decisions.

CHAPTER 8
CONCLUSION

8. CONCLUSION

Sentinel: The Malware Analyzer is a complete solution that combines several analytic approaches and cutting-edge visualization capabilities. It marks a significant leap in the field of malware identification and analysis. The platform guarantees comprehensive and effective detection of a wide range of malware types, including advanced threats like fileless malware, thanks to its multifaceted methodology that combines static, dynamic, and hybrid analysis.

The foundation of the project's success is its advanced visualization component, which has its own frontend and backend, and its strong backend, which uses Python scripts to automate difficult malware research operations. Security analysts can now engage with data in a straightforward manner and obtain insightful knowledge about malware activity and trends thanks to this integration, which also makes the user experience smooth.

In conclusion, a malware analysis sandbox shows how cutting-edge malware analysis methods and contemporary software development approaches may be used in real-world situations. By offering a strong architecture that strikes a balance between security, effectiveness, and scalability, the project raises the bar for malware research and detection. Its strong analytical skills combined with its capacity to protect data secrecy underscore its potential uses in a variety of domains, including cybersecurity, banking, and healthcare.

To summarize, Sentinel: The Malware Analyzer seeks to solve significant cybersecurity issues by utilizing cutting-edge malware analysis technology. The project intends to develop a workable and secure solution for malware detection and analysis by carefully reviewing previous research and utilizing cutting-edge analysis techniques. This will make a significant contribution to the field of cybersecurity and establish new benchmarks for safeguarding digital infrastructures in the digital age.

CHAPTER 9
REFERENCES

REFERENCES

1. S. Talukder, "Exploration of Tools and Techniques for Malware Detection and Analysis," 2020.
2. T. Kumar, S. Sharma, R. Dhaundiyal, and P. Jain, "Investigation of Malware and Forensic Tools on Internet," 2018.
3. R. Yadav and D. Singh, "Malware Analysis and Reverse Engineering: Unraveling the Digital Threat Landscape," 2024.
4. K. Choudhary, "Implementation, Detection and Prevention of Stegomalware," 2020.
5. V. Khushali, "A Review on Fileless Malware Analysis Techniques," 2020.
6. M. Haroon, "Flare-VM Sandbox Guide: Creating an Isolated Lab Environment for Malware Analysis & Reverse Engineering," Medium, Sep. 22, 2023. [Online]. Available: <https://medium.com/@haroon00525/flare-vm-lab-setup-isolated-lab-environment-for-malware-analysis-6e7c23af875>.
7. S. Yusirwan, Y. Prayudi, I. Riadi, and A. Dahlan, "Implementation of Malware Analysis using Static and Dynamic Analysis Method," 2015.
8. A. Singhal and S. Venkataramalingam, "Malware Analysis and Reverse Engineering: Unraveling the Digital Threat Landscape," 2023.
9. M. N. Alenezi, H. Alabdulrazzaq, A. A. Alshaher, and M. M. Alkharang, "Evolution of Malware Threats and Techniques: A Review," 2020.
10. M. Issakhani, P. Victor, A. Tekeoglu, and A. H. Lashkari, "PDF Malware Detection based on Stacking Learning," 2022.
11. [FLARE VM Documentation](<https://github.com/mandiant/flare-vm>)
12. [REMnux Documentation](<https://docs.remnux.org/>)

PAPER NAME

SENTINAL: THE MALWARE ANALYZER

AUTHOR

SHAIN MATHEW

WORD COUNT

8320 Words

CHARACTER COUNT

54236 Characters

PAGE COUNT

63 Pages

FILE SIZE

5.1MB

SUBMISSION DATE

Aug 1, 2024 12:50 PM GMT+5:30

REPORT DATE

Aug 1, 2024 12:51 PM GMT+5:30

● 3% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 3% Internet database
- 1% Publications database
- Crossref database
- Crossref Posted Content database
- 0% Submitted Works database

● Excluded from Similarity Report

- Bibliographic material
- Quoted material
- Cited material
- Small Matches (Less than 14 words)

● 3% Overall Similarity

Top sources found in the following databases:

- 3% Internet database
- 1% Publications database
- Crossref database
- Crossref Posted Content database
- 0% Submitted Works database

TOP SOURCES

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	digital.lib.usu.edu Internet	1%
2	medium.com Internet	<1%
3	vegibit.com Internet	<1%
4	dash.plotly.com Internet	<1%
5	huggingface.co Internet	<1%
6	qiita.com Internet	<1%
7	coursehero.com Internet	<1%
8	adilmoujahid.com Internet	<1%



gitea.fhgr.ch
Internet

<1%

[Sources overview](#)