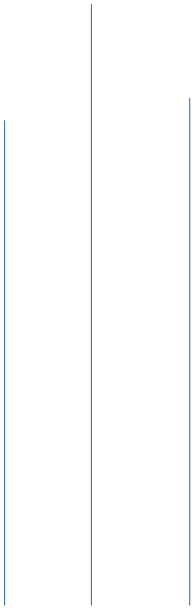
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Deemed to be University)
Patiala, Punjab

Software Engineering(UCS503)

Project Report
on
Predictive Malware Detection System(PMDS) using system logs

Under the Guidance of:
Dr. Himika Sharma



BY:
Ashutosh Kumar Swarnakar(102217263)
Manish Joshi(Regmi)(102217262)
Aadarsha Mahato(102217264)
Nirnaya Ghimire(102217259)
Nitesh Yadav(102217260)

TABLE OF CONTENTS

1. Project selection phase

1.1 Software Bid

Software Bid/ Project Teams

UCS 503- Software Engineering Lab

Group : 2CS9 Dated: 22-JAN-2024

Team Name:Computer Crew
Team ID (will be assigned by Instructor):

Please enter the names of your Preferred Team Members. :

- You are required to form **a three to four person** teams
- Choose your team members wisely. You will not be allowed to change teams.

Name	Roll No	Project Experience	Programming Language used	Signature
Aadarsha Mahato	102217264	Hand Gesture(Computer Vision) using ML Arduino project	c++, SQL,HTML, , Python	
Manish Joshi	102217262	Arduino project App Development	Python, C/C++, SQL	
Nirnaya Ghimire	102217259	Arduino project App Development	C/C++, HTML, CSS, Python	
Nitesh Yadav	102217260	Medical imaging ML based Gold price Prediction	C++, Python, Flask	
Ashutosh kumar swarnakar	102217263	Pneumonia Detection Medical Imaging	Python,HTML,CSS	

Programming Language / Environment Experience

List the languages you are most comfortable developing in, **as a team**, in your order of preference. Many of the projects involve Java or C/C++ programming.

1.C/C++
2.HTML,CSS,JAVASCRIPT
3.PYTHON,SQL

Choices of Projects:

Please select **4 projects** your team would like to work on, by order of preference: *[Write at-least one paragraph for each choice (motivation, reason for choice, feasibility analysis, etc.)]*

	Project Name	Unique Selling Point
First Choice	Predictive Malware Detection System	-Analyses System Logs -Detects Malware -Predicts future malware based on signatures and minimises the threat

Second Choice	Judiciary Management System	-Chatbot to assist cases -Cost effective -Reduces pending cases
Third Choice	Phishing website Detection by Machine learning Technique	-Detects fake website -Minimizes Fraud -Data Safety
Fourth Choice	Vaccine Management System	-vaccine recommendation -Consider Medical Allergy

Additional Remarks/ Inputs

Please tell us about any other factors that we should take into consideration (e.g., if you really would like to work on a project for some particularly convincing reason).

.....

.....

.....

1.2 Brief Project Write up

Abstract:

The escalating threat of malware presents a critical challenge to organizations, leading to data breaches and operational disruptions. Recent incidents, such as Darkside ransomware attacks and the Apache Log4j vulnerability, underscore the urgent need for robust detection and prevention mechanisms. Traditional signature-based antivirus solutions have struggled to keep pace with the evolving threat landscape, particularly on mobile devices, necessitating more adaptive approaches.

As internet usage continues to soar, the risk of cyberattacks grows, exacerbating the prevalence of malware within systems. Despite efforts to fortify software against vulnerabilities, the multifaceted nature of attacks extends beyond technical flaws to encompass social engineering tactics like phishing. Consequently, real-time monitoring systems are indispensable for preemptively identifying and thwarting potential threats.

This project proposes a novel approach to malware detection by leveraging word embedding techniques and Long Short-Term Memory (LSTM) networks. By analyzing system logs for API calls made by malware, this method aims to predict and identify malicious activity. Word embedding preprocessing enables the representation of system log data in a continuous vector space, facilitating more effective analysis by LSTM networks.

Unlike traditional detection methods, which may struggle with the complexity and variability of malware behavior, LSTM networks offer the ability to capture temporal dependencies in sequential data, making them well-suited to detecting patterns indicative of malware activity. Additionally, this

approach mitigates the challenges associated with handling large datasets, as LSTM networks can efficiently process sequential data.

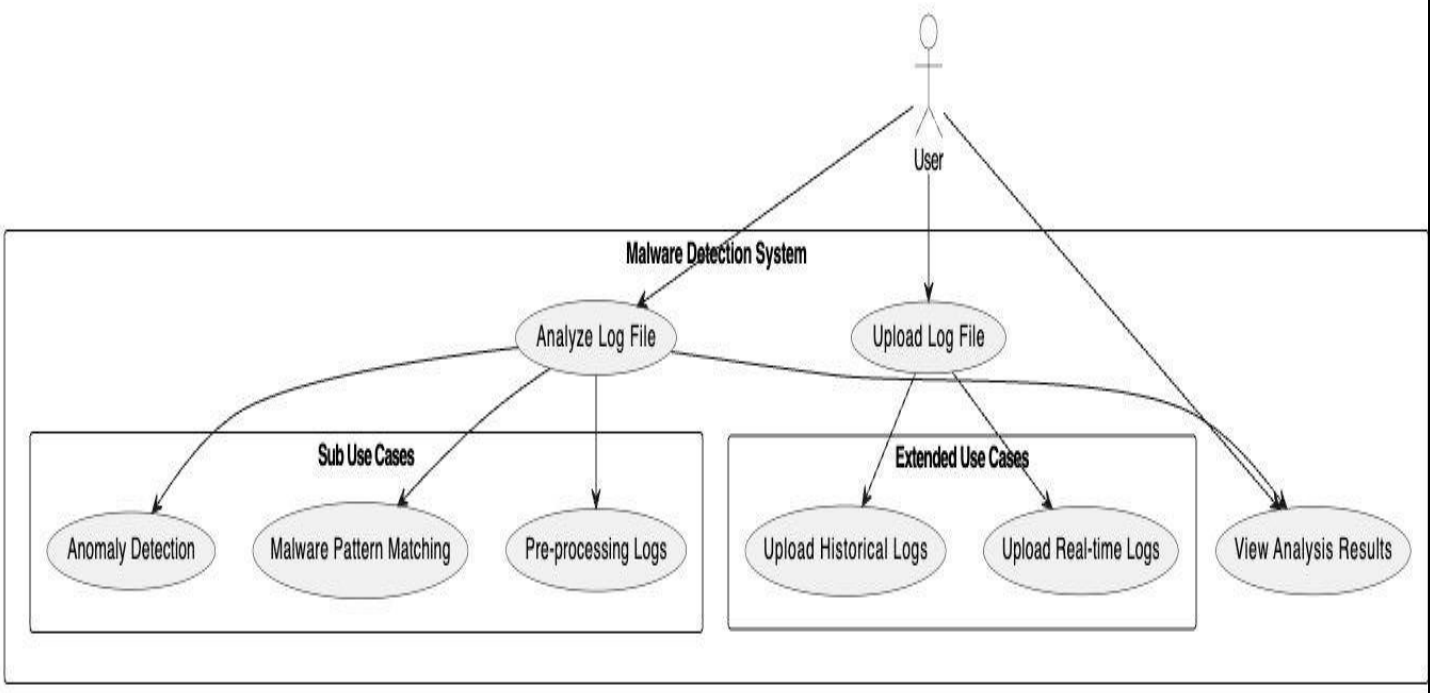
By integrating word embedding and LSTM techniques, this approach aims to enhance the efficiency and accuracy of malware detection, ultimately enabling organizations to better protect their systems and data from evolving threats.

2. Analysis Phase

2.1 Use Cases

-Identifying the various interactions between users and the malware detection system.
-Defining primary and alternative use cases.
-Specifying preconditions, postconditions, and basic flow of events.

2.1.1Use-Case Diagrams



2.1.2 Use Case Templates

Use Case Name: Analyze Malware Sample

Actor: Security Analyst

Description: This use case allows a security analyst to analyze a malware sample to determine its behavior and potential threat level.

Preconditions:
- The security analyst must have access to the malware analysis tool.

Postconditions:
- The analysis results are recorded in the system.

Main Flow:
1. The security analyst uploads the malware sample to the analysis tool.
2. The analysis tool extracts metadata from the malware sample.
3. The analysis tool executes the malware sample in a controlled environment.
4. The analysis tool monitors the behavior of the malware sample.
5. The analysis tool generates a report summarizing the behavior and potential threat level of the malware sample.

Alternate Flows:
- If the malware sample cannot be uploaded due to size limitations, the security analyst must use an alternative method to transfer the sample. (Not described in this template)

Exception Flows:
- If the analysis tool encounters an error during execution, the security analyst is notified and may need to re-run the analysis.
- If the malware sample is found to be highly destructive, the security analyst may need to take immediate action to contain the threat.

Extensions:
- The security analyst may choose to share the analysis results with other security teams for further investigation.
- The security analyst may initiate additional actions based on the analysis results, such as updating firewall rules or deploying new antivirus signatures.

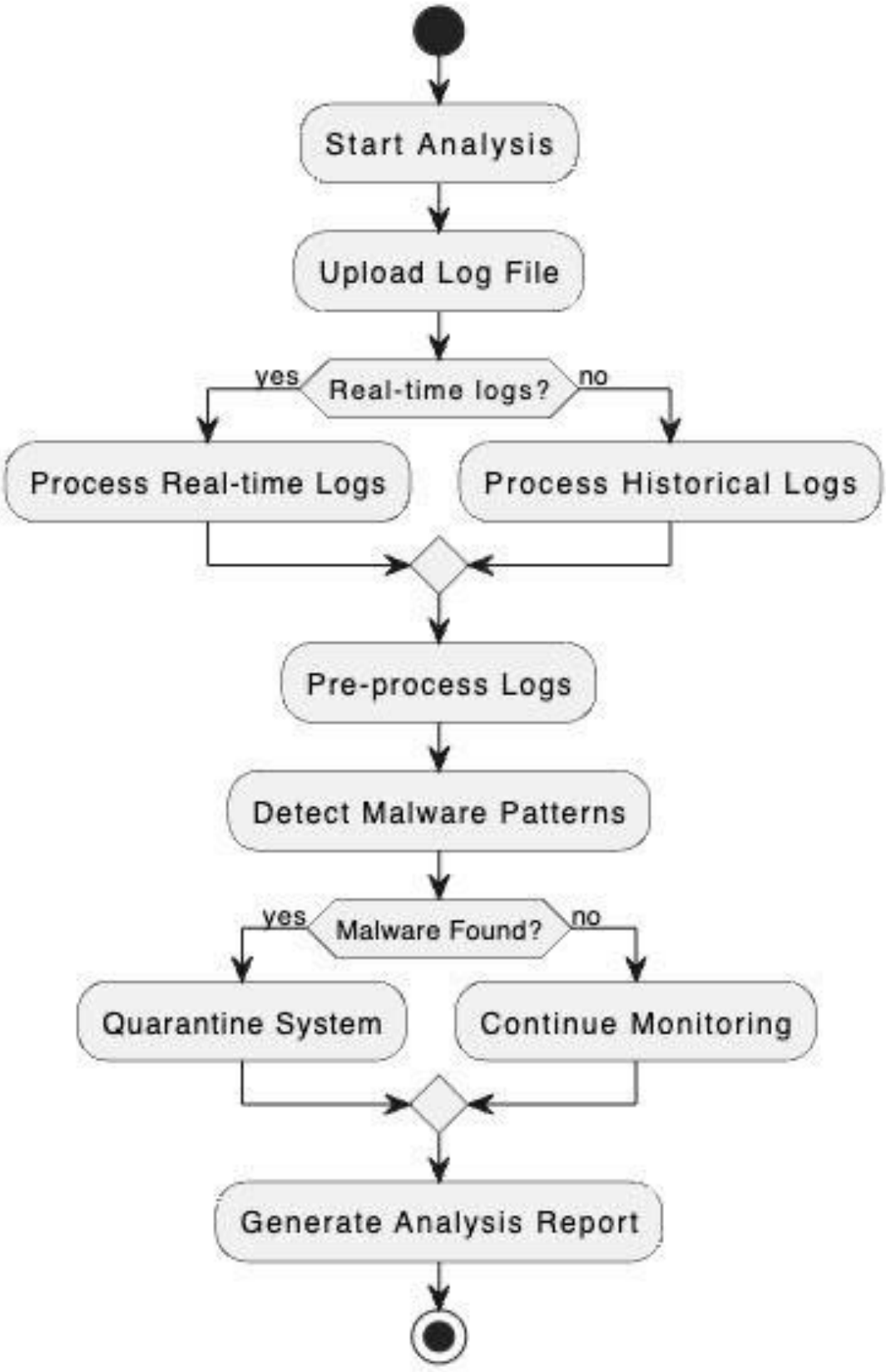
2. 2 Activity Diagram and Swimlane Diagrams

-Model the flow of activities within the malware detection system.

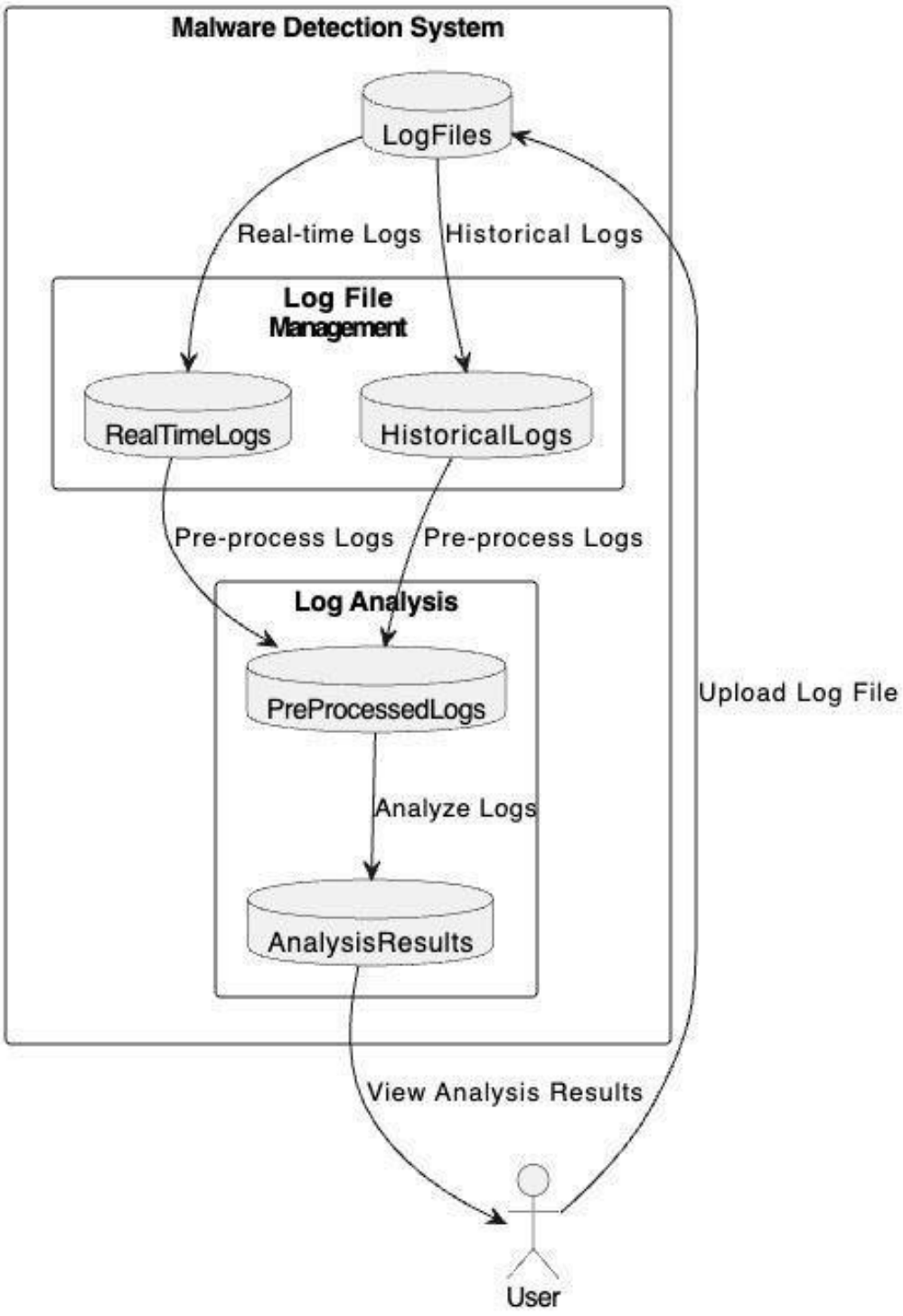
-Illustrate the sequence of actions between different entities (actors or system components).

2.3 Data Flow Diagrams (DFDs)

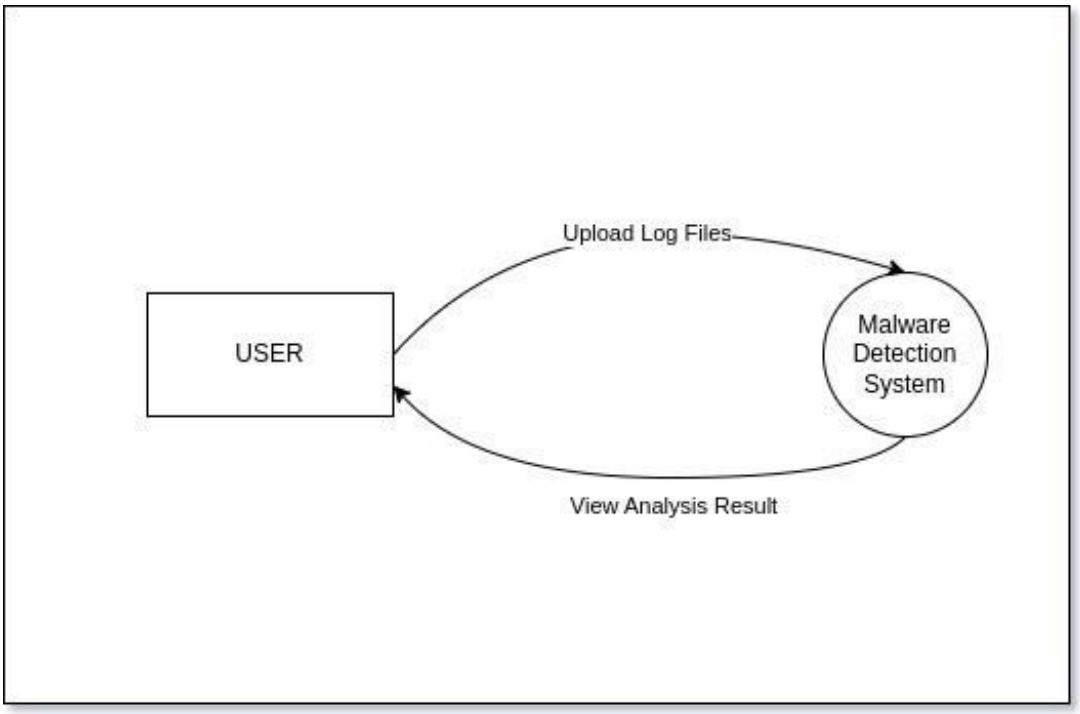
-Illustrates the flow of data within the system.



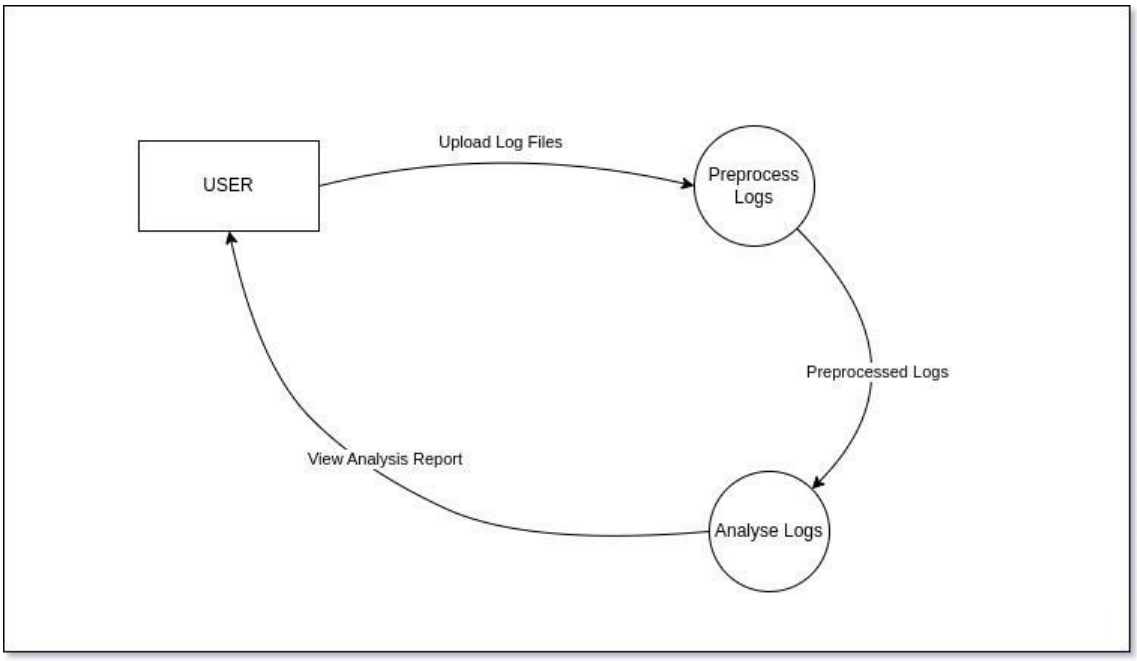
-Identify sources, destinations, storage, and processes involved in data flow.



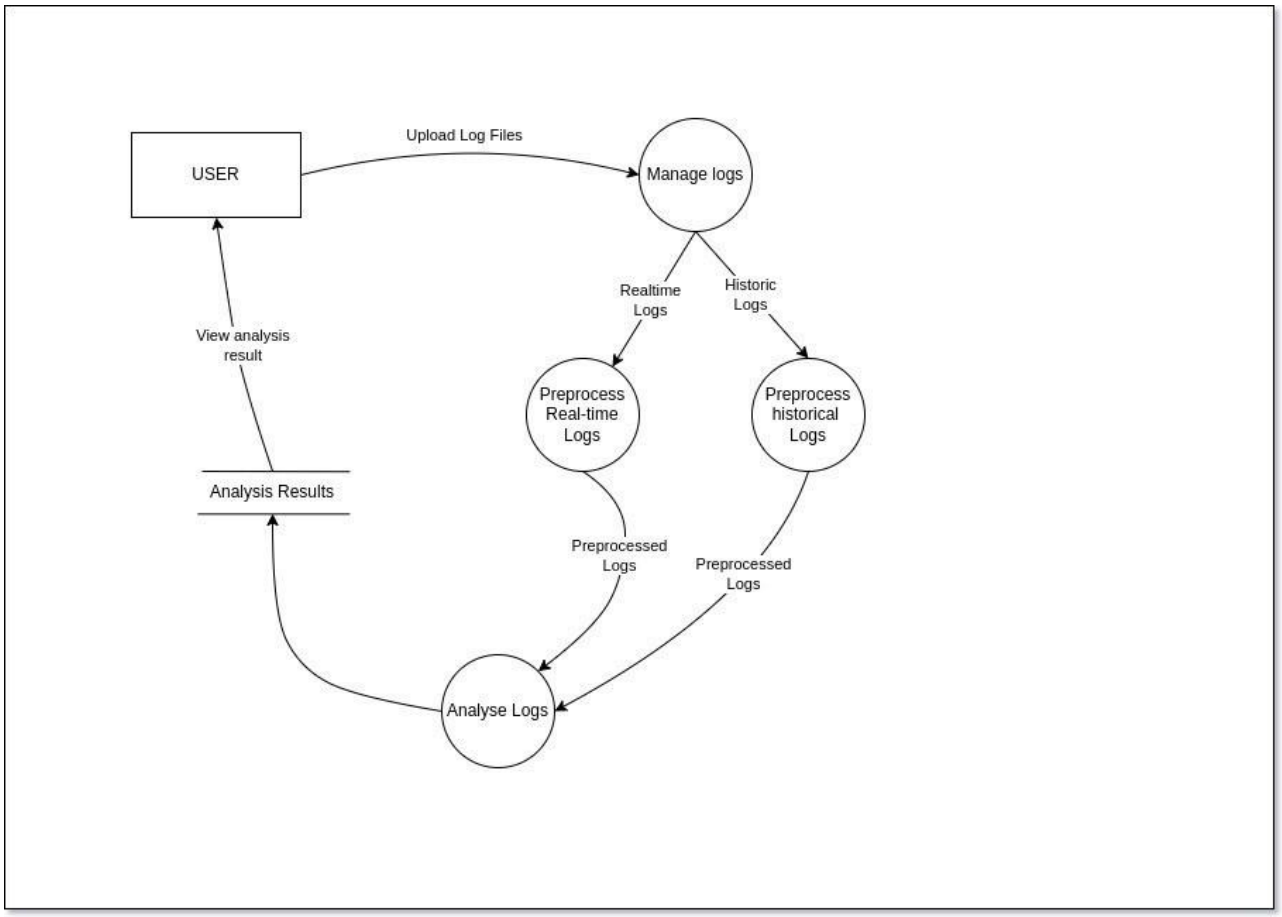
2.3.1 DFD Level 0



2.3.2 DFD Level 1



2.3.2 DFD Level 2



2.4 Software Requirement Specification in IEEE Format

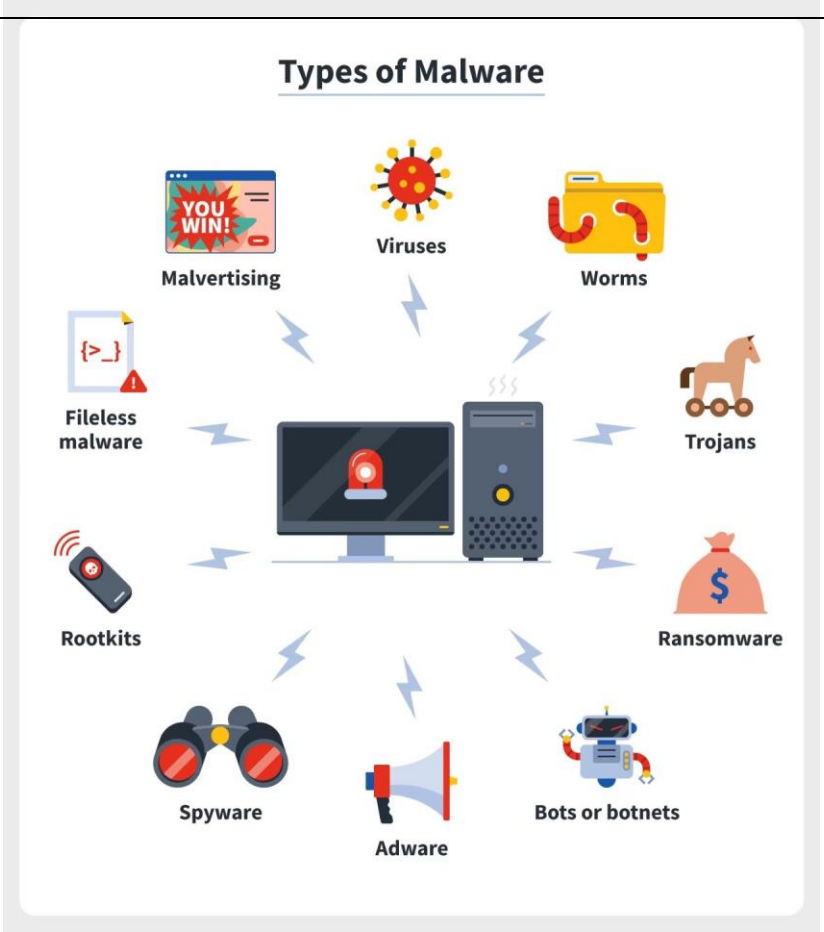
Software Requirements Specification Document(SRS)

Table of Contents

1. Introduction

Malware, short for malicious software, refers to software specifically crafted to disrupt and compromise the functionality of computer or mobile applications. It is designed to clandestinely collect sensitive information and carry out nefarious activities. These activities encompass unauthorized access to private data, surreptitious theft of valuable information, displaying intrusive advertisements, and covertly monitoring user activities.

Types of Malware:



In today's digital landscape, ensuring computer security has always been a critical concern. However, with the proliferation of mobile devices, mobile security has become equally paramount. Mobile phones are rapidly being replaced by smartphones, most of which run on the Android operating system. These smartphones serve dual purposes for users. Firstly, they are used for personal tasks such as accessing photos, contacts, emails, and other personal data. Secondly, these devices are often utilized to access an organization's IT infrastructure, especially with the rise of policies like Bring Your Own Device (BYOD). Unfortunately, this increased integration of smartphones into organizational networks has also led to a surge in malware attacks targeting these devices.

Researchers have identified two primary methods for detecting malware on mobile devices. The first method, known as static analysis, involves studying malicious applications without actually executing them. Techniques used in static analysis include de-compilation, pattern matching, and decryption. However, since unknown applications can employ various forms of obfuscation and encryption, static analysis methods may struggle to identify unknown malware effectively.

As a result, a second method, known as dynamic analysis, has been proposed. Dynamic analysis involves monitoring an application's behaviour, such as network access, phone calls, and message sending, in real-time. This analysis is typically conducted within a sandbox environment to prevent malware from infecting actual production systems. Many such sandboxes are virtual systems that can be easily reverted to a clean state after the analysis is completed.

Overall, both static and dynamic analysis play crucial roles in mobile security by helping to detect and mitigate the risks posed by malware on smartphones and other mobile devices.

1.1 Purpose

The purpose of this document is to outline the comprehensive requirements for the development of a Predictive Malware Detective System (PMDS) that primarily leverages system logs for identifying and mitigating potential security threats within a System/PC. This PMDS aims to enhance the organization's cybersecurity posture by proactively detecting and neutralizing malware threats as well as keeping the signatures of the detected malware and predicting the future malware. Basically, it will keep auto updating itself on upcoming future.

1.2 Scope

This system aims to analyse system logs generated by various network device such as PC, to detect suspicious activities indicative of malware presence. It will provide real-time monitoring, analysis, and alerting capabilities to assist system administrators and security personnel in promptly identifying and responding to potential security breaches. In near future we can use it to detect malware presence even by the logs generated by servers, endpoints and other networking devices as well and can be upgrade upto the point where it can be integrated to and operating system or any antivirus programs.

1.3 Definitions, Acronyms, and Abbreviations

- SRS: Software Requirements Specification
- IDS: Intrusion Detection System
- IPS: Intrusion Prevention System
- API: Application Programming Interface

1.4 References

1. Y. Chang, S. Wang, 'The Concept of Attack Scenarios and its Applications in Android Malware Detection', IEEE 18th International Conference on High Performance Computing and Communications 2016, pp. 1485-1492.
2. S. Saxena and S. Mancoridis, "Malware Detection using Behavioral Whitelisting of Computer Systems," *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, Woburn, MA, USA, 2019, pp. 1-6, doi: 10.1109/HST47167.2019.9032977.

2. Overall Description

2.1 Product Perspective

The PMDS will operate as a standalone software application integrated with existing network infrastructure, including firewalls, IDS/IPS systems, and logging mechanisms. It will interact with these components to collect, parse, and analyse system logs for potential malware activity.

In a Windows environment, dynamic analysis can be performed by observing system logs. Windows logs capture various events, including system calls made by running applications. By

analysing these logs, researchers can identify suspicious behaviour indicative of malware activity. The system call sequences recorded in Windows logs provide valuable information for constructing a dataset used for classifying unknown applications.

2.2 Product Functions

1. **Log Collection:**

   Gather system logs from networking devices like Pc which is running windows operating system.

2. **Log Parsing:**

   Extract relevant information from raw log data, including timestamps, source IP addresses, destination IP addresses, and event descriptions.

3. **Anomaly Detection:**

   Utilize machine learning algorithms and pattern recognition techniques to identify anomalous behaviour within log data that may indicate malware activity.

4. **Alerting:**

   Generate real-time alerts for suspicious events detected during log analysis, providing detailed information for further investigation.

5. **Reporting:**

   Generate periodic reports summarizing malware detection statistics, trends, and potential security risks identified within the network.

   Implements automated response mechanisms to quarantine and mitigates the impact of detected malware infections.

6. **Self Updating:**

   It is not necessary that signatures of various malware are same but there is some chances that malware applications carries something common traits and signatures which can be useful to detect future malware and hence after detecting the newer kind it will update itself with that kind and keep going .

   Basically it have a built-in mechanism to automatically update its detection algorithms and databases with newly identified malware traits and signatures.

   It periodically fetches updates from reputable threat intelligence sources and incorporate them into its detection capabilities.

7. **Adaptive Learning:**

   Employ machine learning techniques to adaptively learn from new malware samples and improve detection accuracy over time.

Continuously refine detection algorithms based on feedback from detected threats and false positives.

8. **Prediction:**

   Based on the self updating feature it will be able to track the activities even if there is small common traits and analyse it until it detects as malware which can be it's unique feature..

2.3 User Classes and Characteristics

- **System Administrators:** Responsible for configuring and maintaining the Malware Detection System.
- **Security Analysts:** Analyse alerts and investigate potential security incidents identified by the system.

2.4 Operating Environment

The system will be compatible with major operating systems including Windows, Linux, and Unix. It will require adequate computational resources for log processing and analysis, as well as network connectivity to access log data from remote sources.

2.5 Design and Implementation Constraints

- Compliance with relevant privacy regulations and data protection laws.
- Integration with existing network infrastructure and security systems.
- Scalability to accommodate large volumes of log data in enterprise environments.

2.6 Assumptions and Dependencies

- Availability of sufficient log data from network devices and endpoints.
- Access to necessary APIs or protocols for log collection and analysis.
- Adequate training data for machine learning algorithms used in anomaly detection.

3. External Interface Requirements

3.1 User Interfaces

The system will provide a web-based user interface for system administrators and security analysts to configure settings, view alerts, and generate reports.

3.2 Hardware Interfaces

The system will interface with standard hardware components including servers, network devices, and storage systems for log collection and processing.

3.3 Software Interfaces

Integration with existing network infrastructure components such as firewalls, IDS/IPS systems, and SIEM platforms through standard APIs or protocols.

3.4 Communication Interfaces

The system will communicate with external systems and services using standard network protocols such as TCP/IP, HTTP, and SNMP.

4. System Features

1. **Machine Learning Algorithms:**

   Utilizes machine learning models to analyse historical data, identify patterns, and predict potential malware behaviours based on learned characteristics.

2. **Behavioural Analysis:** Monitors the behaviour of software and systems to detect deviations from normal patterns that may indicate the presence of malware. This includes analysing process behaviours, network activities, file interactions, and system calls.

3. **Anomaly Detection:** Identifies anomalies in system behaviour or network traffic that may signify the presence of malware. This can include deviations in resource usage, abnormal file access patterns, or unusual network communication.

4. **Sandboxing:** Utilizes sandbox environments to execute suspicious files or programs in isolation, observing their behaviour and interactions with the system to determine if they exhibit malicious characteristics.

5. **Heuristic Analysis:** Applies heuristic rules and algorithms to evaluate files or programs for potential malicious intent based on known malware traits or suspicious attributes.

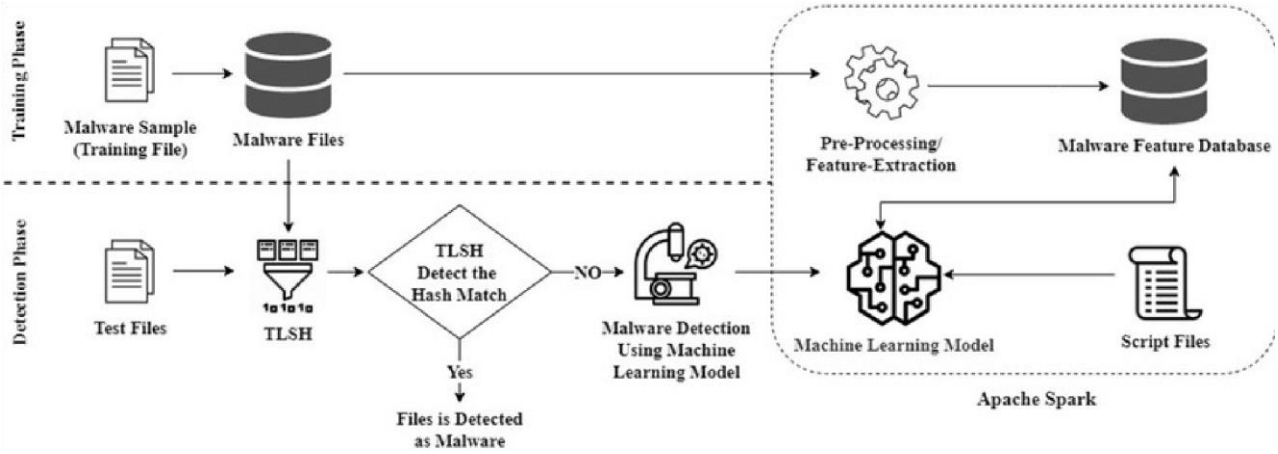
6. **Threat Intelligence Integration:** Integrates with external threat intelligence feeds to enrich analysis with up-to-date information on known malware signatures, indicators of compromise (IOCs), and emerging threats.

7. **Dynamic Analysis:** Conducts dynamic analysis of code execution and runtime behaviour to detect malicious activities such as code injection, process manipulation, or system configuration changes.

8. **Predictive Modelling:** Utilizes predictive modelling techniques to forecast potential malware threats based on historical trends, emerging attack vectors, and evolving malware tactics.

9. **Continuous Learning:** Adapts and improves over time by continuously learning from new malware samples, evolving attack techniques, and feedback from detected threats and false positives.

10. **Integration with Security Ecosystem:** Integrates seamlessly with existing security infrastructure components such as SIEM (Security Information and Event Management) systems, endpoint protection platforms, and network security appliances for coordinated threat detection and response.
11. **Scalability and Performance:** Scales to handle large volumes of data and network traffic while maintaining high performance in terms of detection accuracy and response time.



12. **Real-Time Alerting and Response:** Generates real-time alerts upon detecting potential malware threats, providing actionable intelligence for swift response and remediation.
13. **Privacy and Compliance:** Ensures compliance with relevant data privacy regulations and industry standards while protecting sensitive information and maintaining user privacy during the detection process.

5. Non-functional Requirements

5.1 Performance Requirements

- The system should process log data in real-time with minimal latency.
- It should be capable of handling large volumes of log data efficiently.
- The system should scale horizontally to accommodate growing log data volumes and user loads.

5.2 Security Requirements

- The system should employ encryption for data transmission and storage to ensure confidentiality.
- Access controls should be implemented to restrict system access to authorized users only.
- It should comply with industry-standard security practices and regulations.

5.3 Software Quality Attributes

- The system should be reliable, with minimal downtime for maintenance and upgrades.
- It should be easy to deploy, configure, and maintain.

- The system should provide accurate detection of malware while minimizing false positives.

5.4 Integration Requirements:

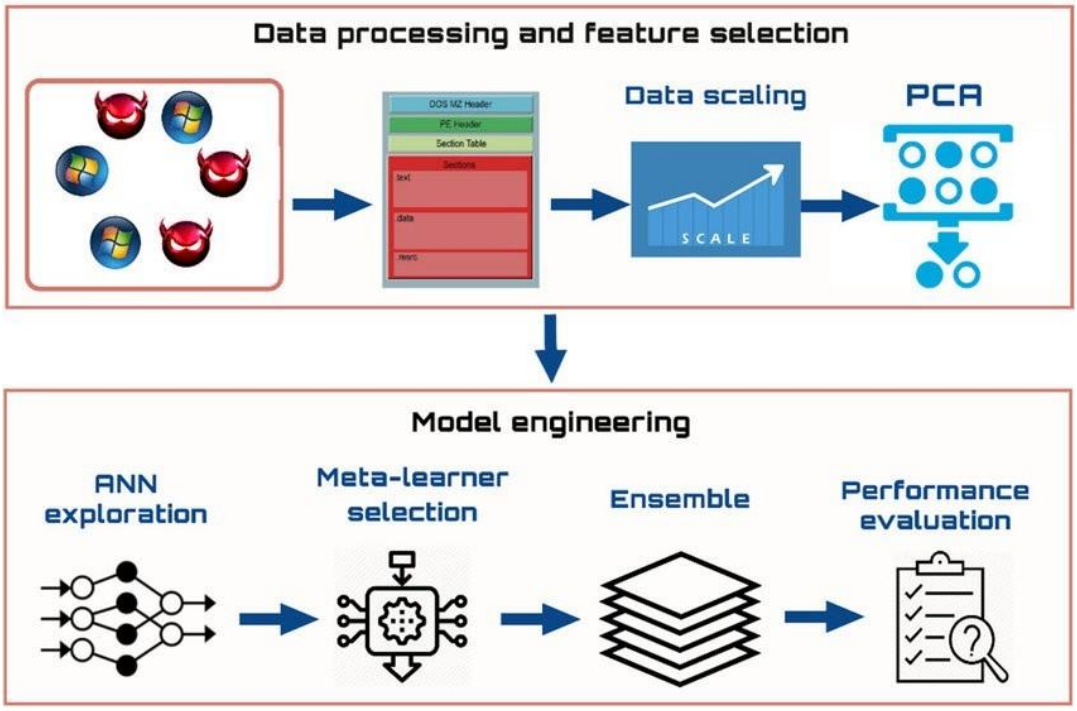
- The PMDS should integrate seamlessly with existing security infrastructure components such as SIEM systems, endpoint protection platforms, and network security appliances.
- Support interoperability with third-party threat intelligence feeds and malware analysis tools to enhance detection capabilities.

5.5 Compliance Requirements:

- Ensure compliance with relevant regulatory standards and industry best practices regarding data privacy and cybersecurity.
- Provide audit trails and logging capabilities to facilitate compliance reporting and forensic investigations.

5.6 Scalability:

]PMDS is designed to scale horizontally to accommodate increasing network traffic and growing volumes of malware samples. It Ensures that the system can handle large-scale deployments across distributed network environments i.e. a collection of nodes(PCs).



5.7 Conclusion:

This document has outlined the requirements for the development of a Predictive Malware Detection System that leverages common traits and signatures shared among malware applications to proactively detect and mitigate future malware threats within an organization's network infrastructure. By adhering to these requirements, the PMDS will enhance the organization's cybersecurity defences and enable timely detection and response to emerging malware threats.

6. Other Requirements

6.1 Legal and Compliance Requirements

- The system should comply with relevant privacy regulations such as GDPR, HIPAA, etc.
- It should adhere to industry standards and best practices for information security.

6.2 Documentation Requirements

- The system should include comprehensive documentation covering installation, configuration, and usage.
- It should provide API documentation for integration with external systems and services.

6.3 Training and Support Requirements

- Training should be provided to system administrators and security analysts on system usage and best practices.
- Ongoing technical support should be available to address user inquiries and resolve issues promptly.

2.5 User Stories and Story Cards

User Stories:

As a system administrator, I want to upload system logs to the malware detection system so that I can identify potential security threats.
As a security analyst, I want the system to analyze system logs in real-time so that I can quickly respond to any detected malware.
As a user, I want to receive alerts when the malware detection system identifies suspicious activity in the system logs so that I can take appropriate action.
As a compliance officer, I want to generate reports based on the analysis of system logs to demonstrate regulatory compliance and identify areas for improvement. As a developer, I want to integrate the malware detection system with our existing logging infrastructure so that we can centrally manage log data and streamline the detection process.

User Cards:

Upload Logs
User Story: As a system administrator, I want to upload system logs to the malware detection system so that I can identify potential security threats.

Real-time Analysis
User Story: As a security analyst, I want the system to analyze system logs in realtime so that I can quickly respond to any detected malware.

Alerting

User Story: As a user, I want to receive alerts when the malware detection system identifies suspicious activity in the system logs so that I can take appropriate action.

Reporting

User Story: As a compliance officer, I want to generate reports based on the analysis of system logs to demonstrate regulatory compliance and identify areas for improvement.

Integration

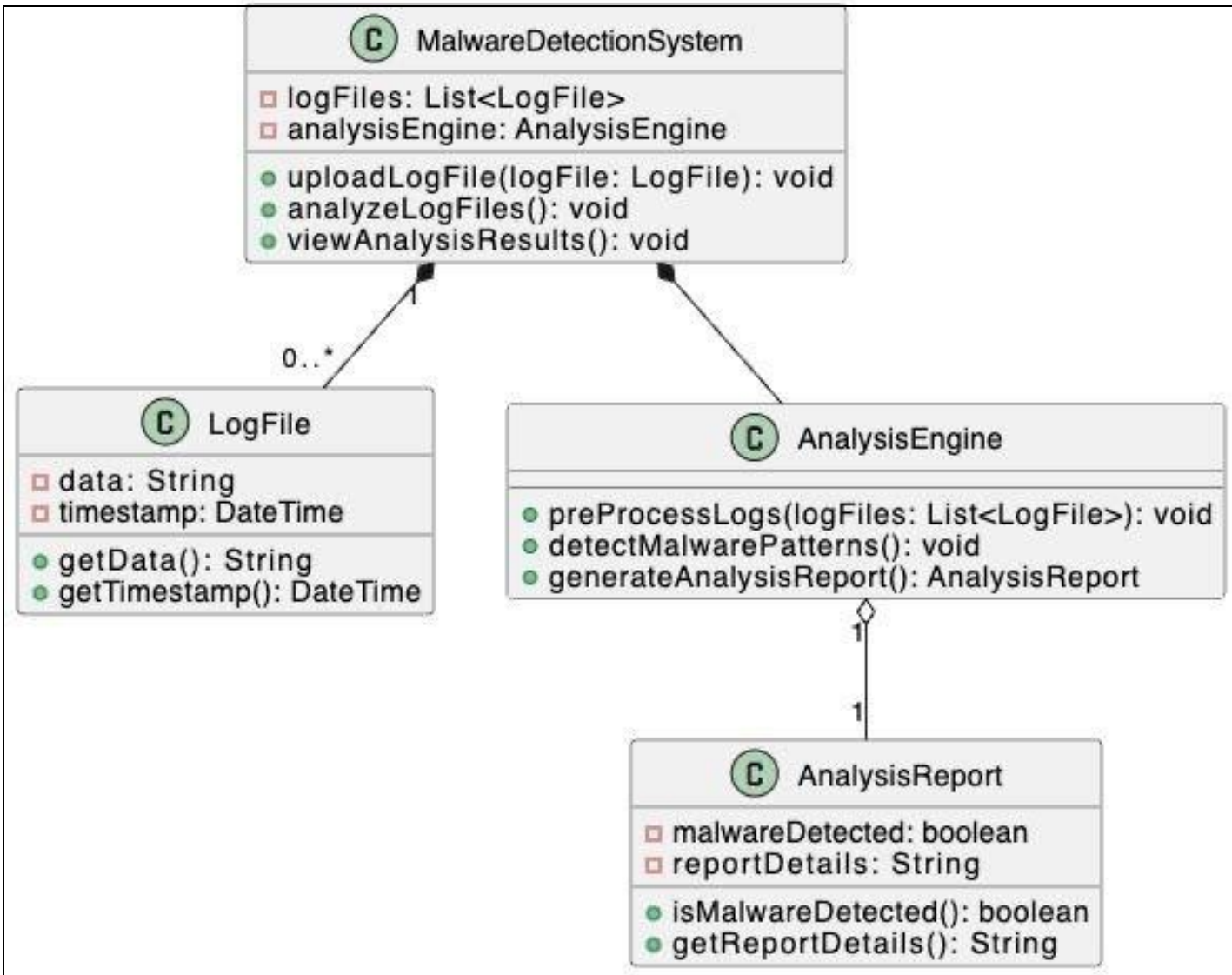
User Story: As a developer, I want to integrate the malware detection system with our existing logging infrastructure so that we can centrally manage log data and streamline the detection process.

3. Design Phase

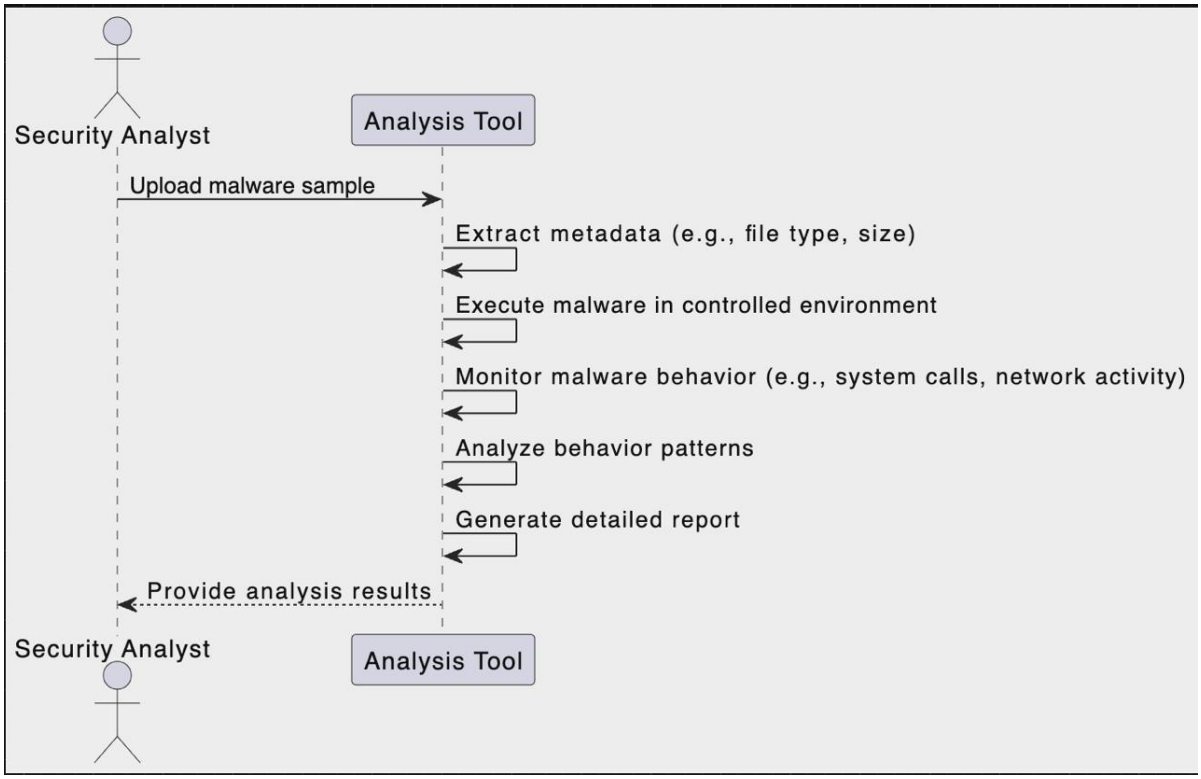
3.1 Class Diagram

-Model the static structure of the malware detection system.

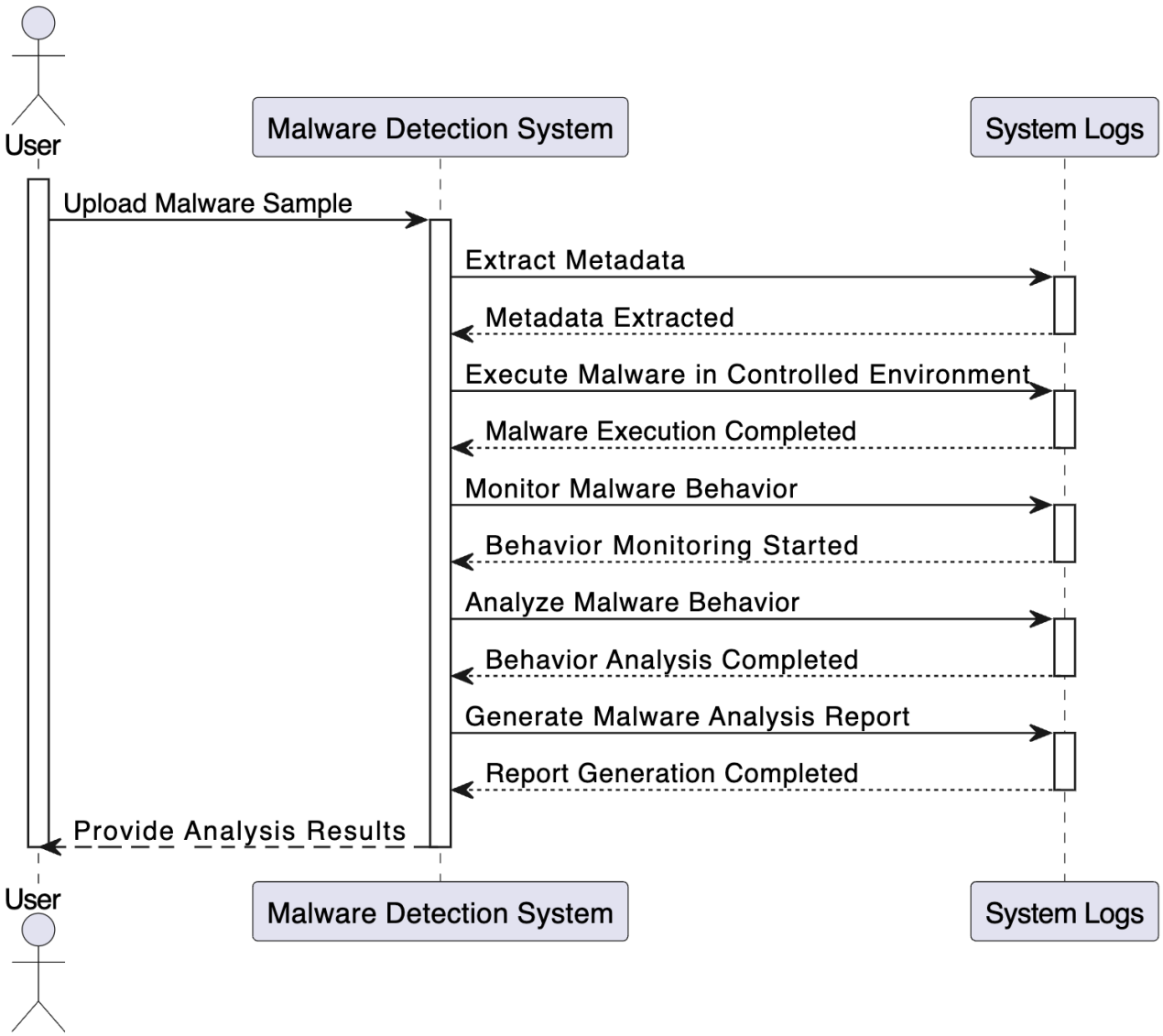
-Identify classes, attributes, methods, and their relationships.



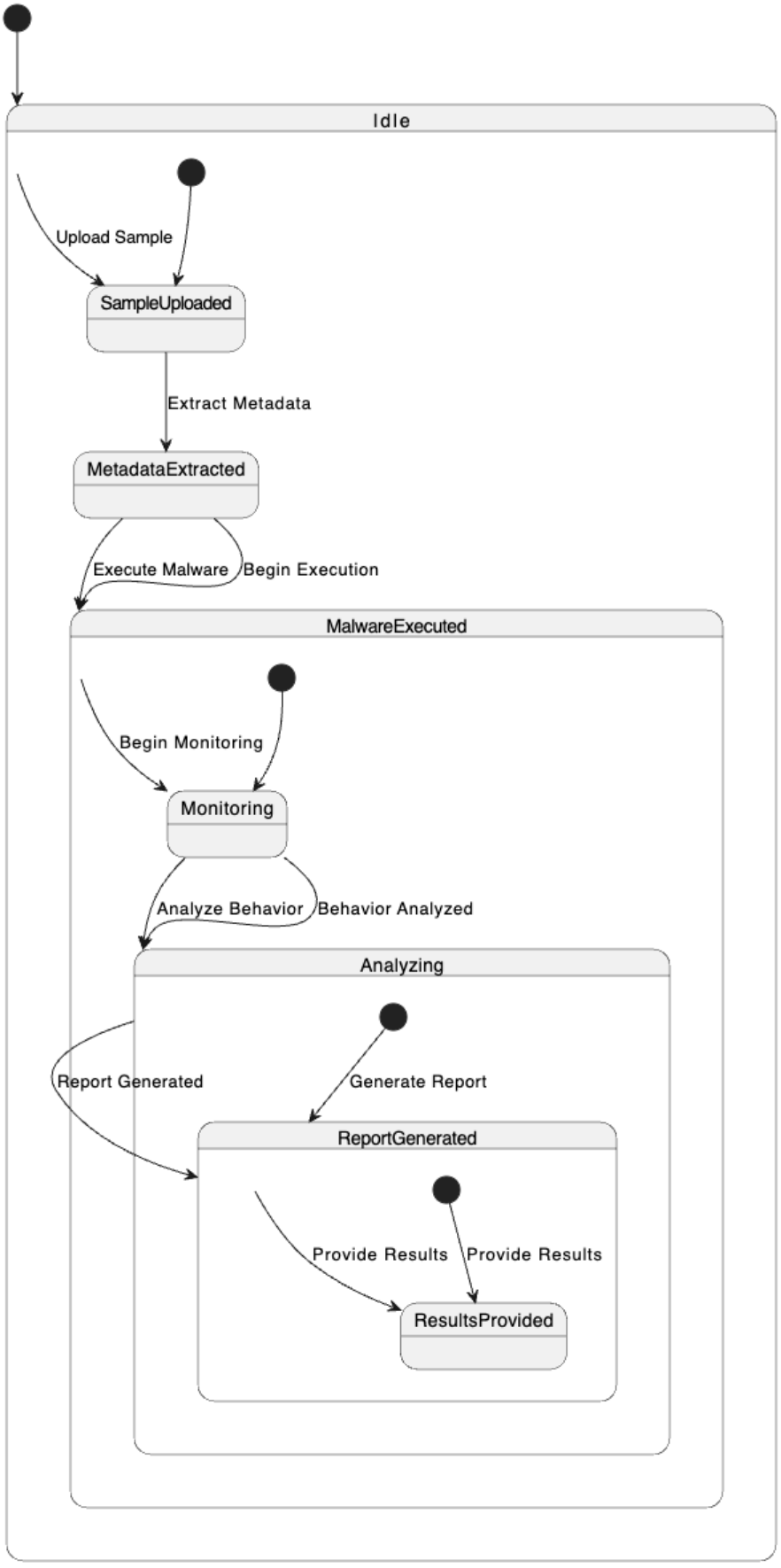
3.2 Collaboration Diagram



3.3 Sequence diagram



3.4 State chart Diagram



4. Testing Phase

Unit Testing:

Test each individual module or component of the PMDS software application to ensure that they function correctly in isolation.

Verify that the software can effectively interact with existing network infrastructure components such as firewalls, IDS/IPS systems, and logging mechanisms. Validate the parsing and analysis of system logs to accurately detect potential malware activity.

Integration Testing:

Test the integration of the PMDS software application with the existing network infrastructure.
Verify that the PMDS can effectively communicate and exchange data with firewalls, IDS/IPS systems, and logging mechanisms.
Ensure seamless interaction between different components of the PMDS, such as log collection, parsing, and analysis.

System Testing:

Test the PMDS as a whole system in a simulated or controlled environment that closely resembles the production environment.
Validate the overall functionality and performance of the PMDS in detecting potential malware activity.
Verify that the PMDS operates efficiently and effectively within the Windows environment, including dynamic analysis of system logs.

Dynamic Analysis Testing:

Perform dynamic analysis testing to evaluate the PMDS's ability to observe system logs and capture various events, including system calls made by running applications.
Analyze the system call sequences recorded in Windows logs to identify suspicious behavior indicative of malware activity.
Validate the accuracy and effectiveness of the PMDS in constructing datasets used for classifying unknown applications based on system call sequences

Testing Cases:

Test Cases:

- **Test Case 1:**

  On entering the correct api call of the system log the model predicts the possible type of malware that might be causing the calls.

Test Case No.: 1 **Test Case Name** : Api Call with Trojan

System : PMDS

Designed By : Computer Crew **Design Date** : 10/04/2024 **Executed By**: Manish Joshi
Execution Date 18/04/2024

Short Description : Testing of trojan affected system.

Pre-Conditions
• The system must be malware affected.
• The api call must be read properly.

Step	Action	Expected System Response	Pass/Fail	Comments
1.	Select the file with the api call file.			
2.	Submit the file.	The processing window appears.		
3.		The System predicts the api call contains trojan.	Pass	

Therefore, output was as expected. Test case Successful.

- **Test Case 2:**

  On entering the correct api call (other than trojan) of the system log the model predicts the possible type of malware that might be causing the calls.

Test Case No.: 2 **Test Case Name** : Api Call with Downloader

System : PMDS**Designed By** : Computer Crew **Design Date** : 10/04/2024 **Executed By**: Manish Joshi

Execution Date 18/04/2024

Short Description : Testing of Downloader affected system.

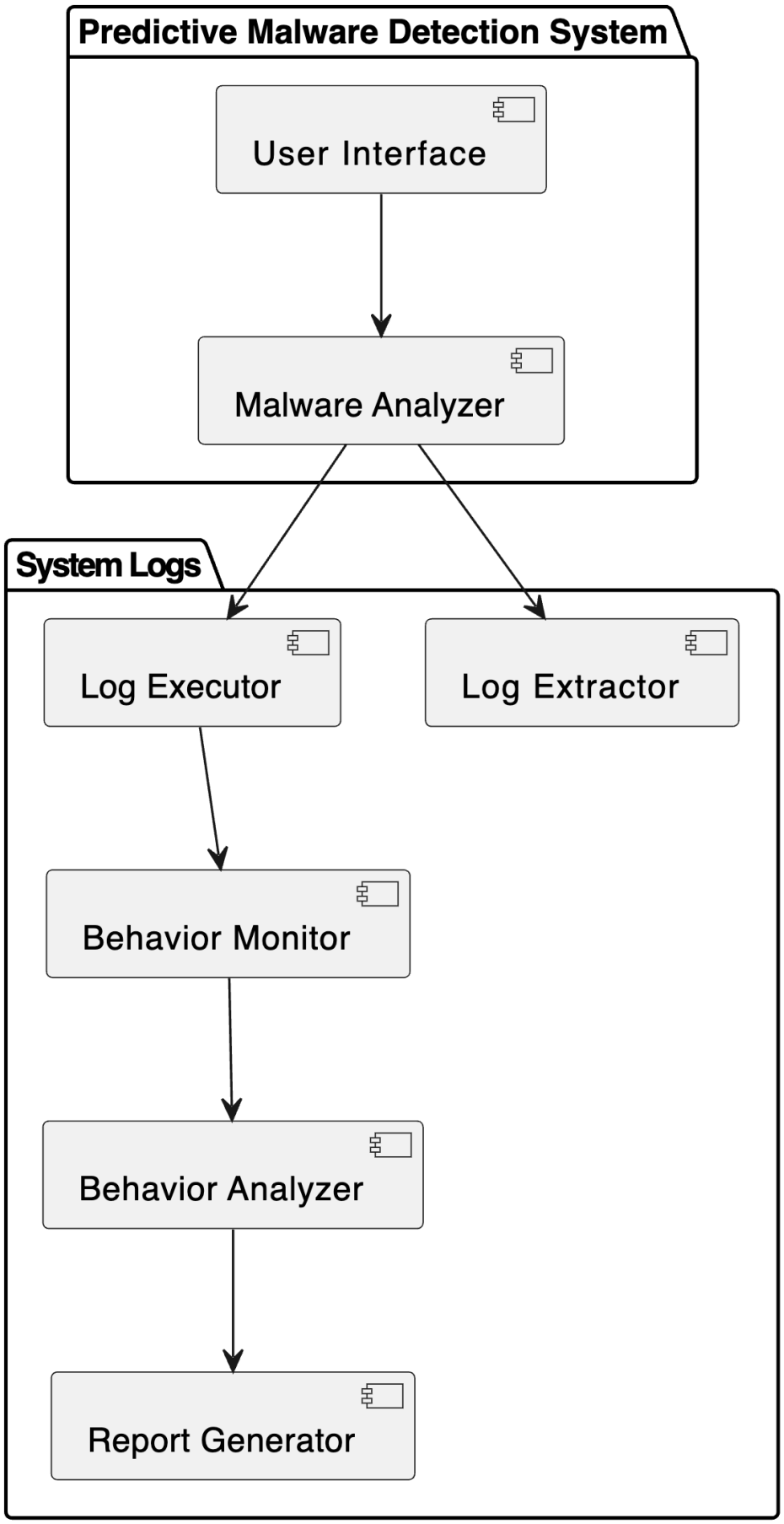
Pre-Conditions
• The system must be malware affected.
• The api call must be read properly.

Step	Action	Expected System Response	Pass/Fail	Comments
1.	Select the file with the api call file.			
2.	Submit the file.	The processing window appears.		
3.		The System predicts the api call contains Downloader.	Pass	

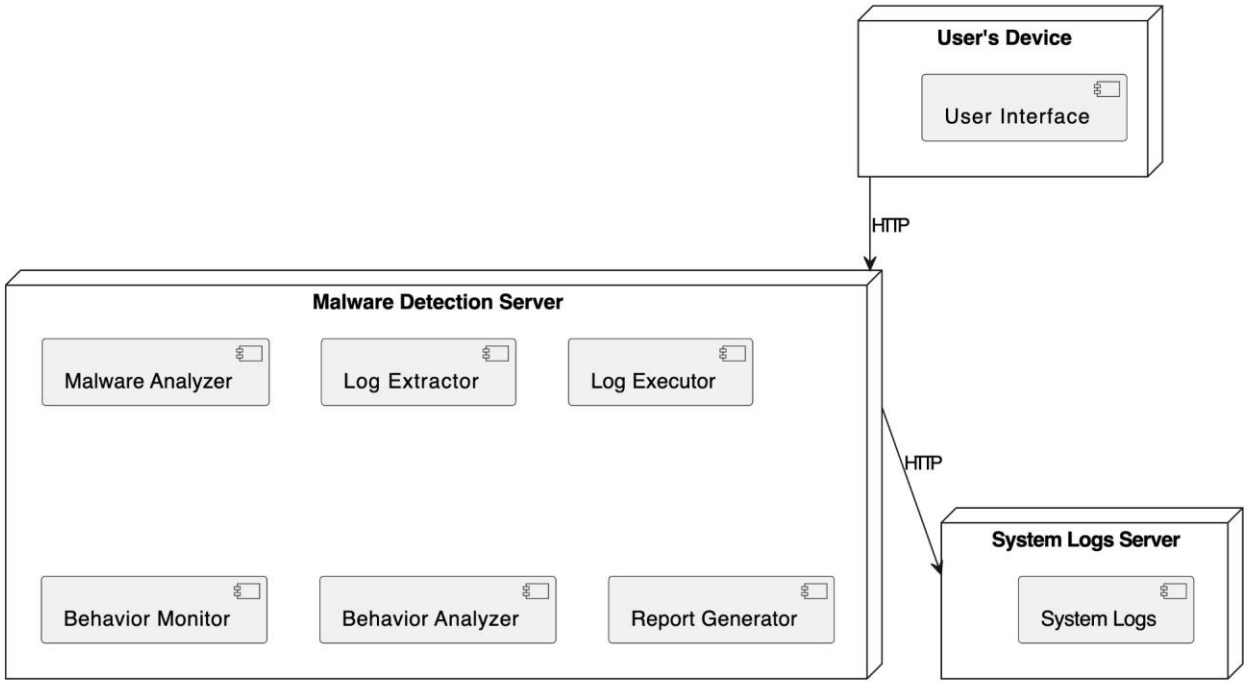
Therefore, output was as expected. Test case Successful.

5. Deployment Phase

5.1 component Diagram 5.2

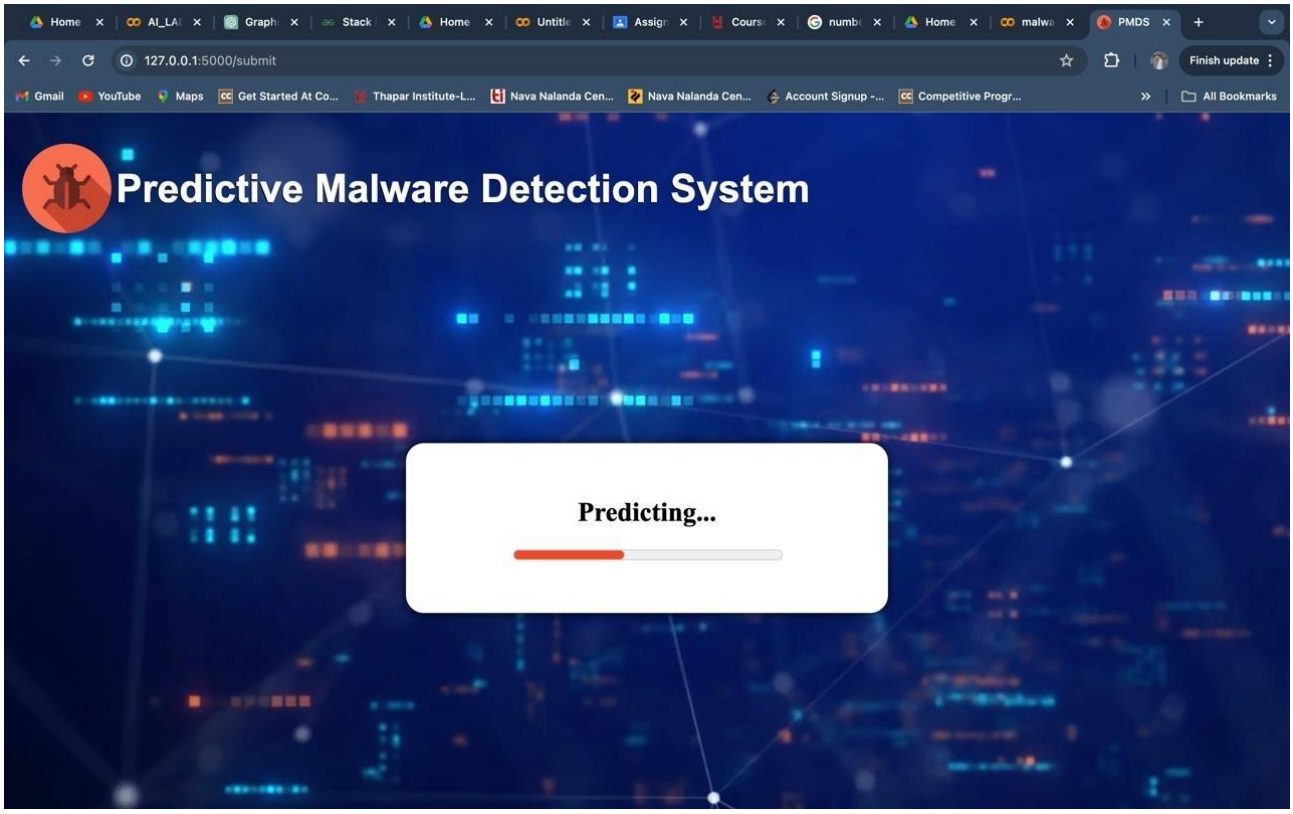


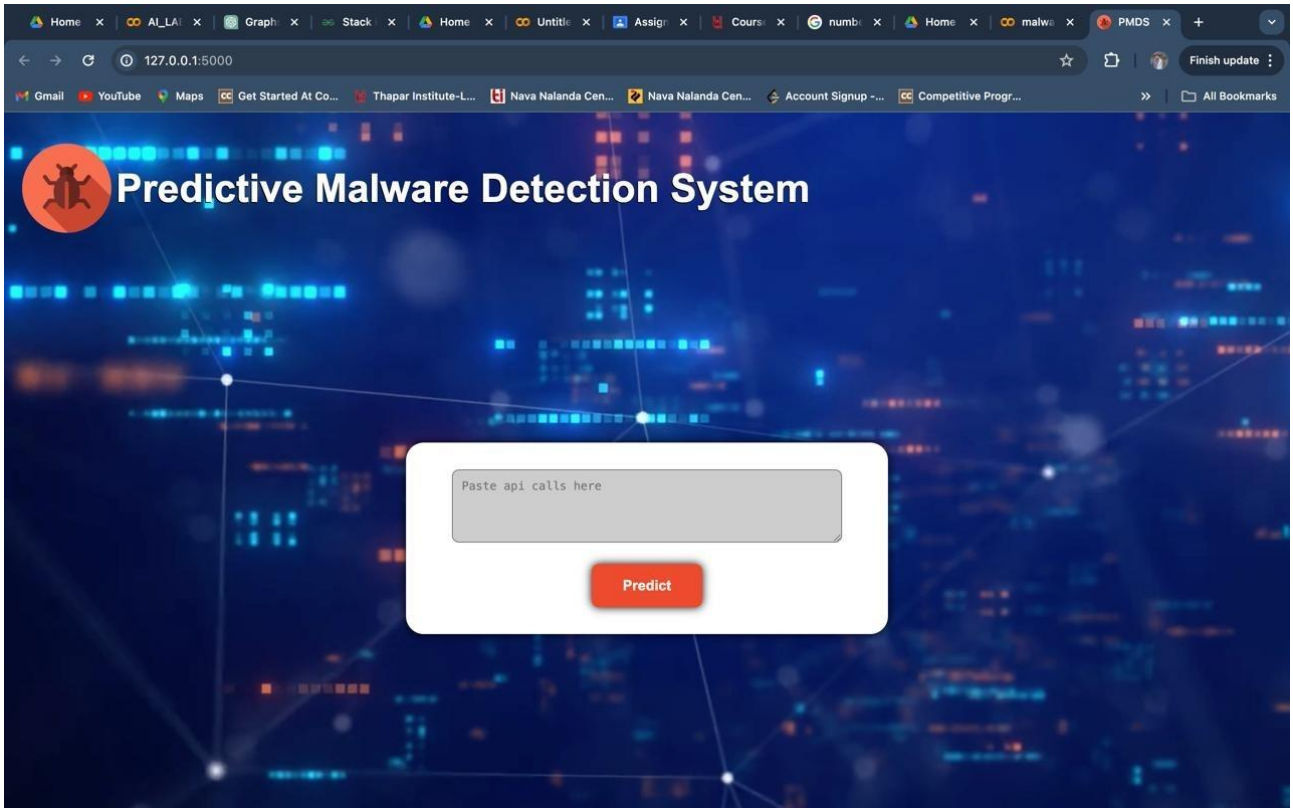
Deployment Diagram

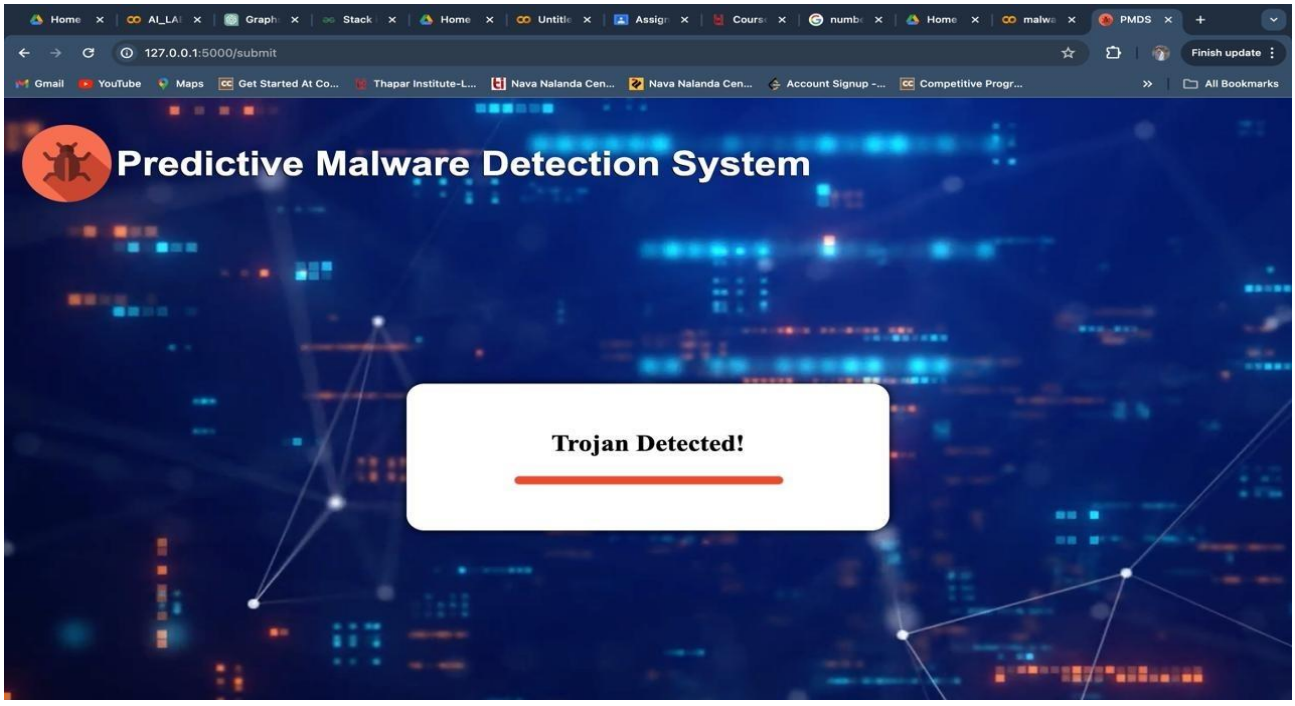


5.3 OUTPUT

SampleScreenshots







Sample Video: