Software Requirement Specification Document for CyberEye
INF02

Mohamed Khaled, Abdelrahman Alaa
Mariam Maged, Shady Amr
Supervised by: Dr. Hossam Abdelrahman, Eng. Tarek Talaat

April 9, 2024

Table 1: Document version history

Version	Date	Reason for Change
1.0	14-Jan-2024	SRS First version's specifications are defined.
2.0	9-Apr-2024	SRS Sec version's specifications are defined.

GitHub: https://github.com/mohamedk314/GradINF02

Contents

Abstract

The Android market looms with threats from malicious software that tend to misbehave and steal data without consent. This security predicament was presented ever since Google's Play Store launched on March 6, 2012 and since then with each passing day, more and more malicious software are being released for the sole purpose of stealing data, and it only seems to be getting worse and worse as attacks are getting more sophisticated [1]. Which raises the question, how can we ensure that our data remains private with our consent and is not being lost, spread, or leaked to other places? This is where our project comes into play. Our project focuses on detecting malicious applications on the Google Play Store by tracking its network activity and analyzing its behaviour and conducting tests which give accurate results on the maliciousness of the app.

1 Introduction

1.1 Purpose of this document

The purpose of this document is to shed light on the importance of the system, the different ways it benefits the user, its market/business relevance, diagramming and usage, and functionalities of the system. Alongside that, we also aim to illustrate how the database is created and how it relates to user data.

1.2 Scope of this document

The specific scope is to let the user in on the functionalities that are both relevant and not relevant to the them (functional and non-functional requirements). Alongside illustrating the use case diagrams, context diagram, UML diagram, and methods of implementation so that they are more and more familiar while having better understanding of the system in a structured manner.

1.3 Business Context

As we all know by now, money isn't the most valuable asset, not gold, not even the most expensive jewelries out there. As a matter of fact, data is officially recognized to be the most valuable asset. [2] It is key to business success and outcome. It plays the biggest role in its development to present targeted marketing and ads to users. With that kept in mind, it's inarguable that securing that data is important for the continuation of any business. Importance of data privacy and security includes [3]:

- **Protecting Users and Building Trust**
- **Preserving Reputation**
- **Reducing Financial Losses**
- **Competitive Advantage**

2 Similar Systems

2.1 Academic

2.1.1 A Lightweight Network-based Android Malware Detection System [4]

Due to the widespread use of Android applications, a large number of malicious apps have entered the market. Through network behavior analysis, a lightweight malware detection system is proposed in this system that watches the network activity of questionable apps. The suggested lightweight network-based malware detection system is seen in the Fig. 5, below. It uses machine learning to differentiate between malware and goodware APKs based on how they behave on the network over time.

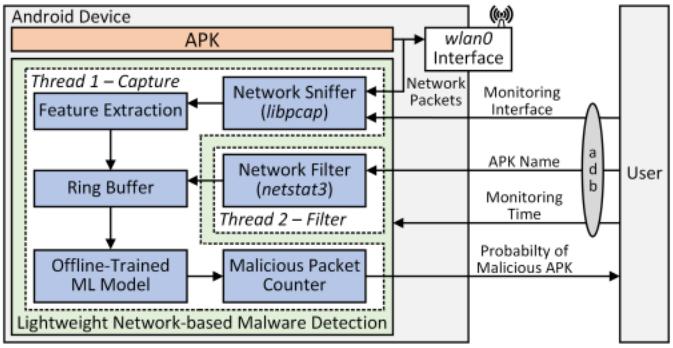


Figure 1: System Overview

Their proposal creates a feature vector for every network packet and uses a flow-based method to extract numerical characteristics from the TCP/IP packet header. For the dataset, they created a script for the good ware group to get 500 most downloaded APKs from the Google Play Store using the gplayclitool. On the other hand, samples from the Android Malware Dataset are broken down into 71 malware families, totaling 24,553 malware samples. A prototype was built of the system using the Random Forest and AdaBoost machine learning algorithms in a Samsung Galaxy S9+. The system results demonstrate that, in terms of classification performance, it is possible to detect malware variants using network features. Even though the classification performance per packet only reaches about 80% accuracy, the Anomaly Score allows them to distinguish between malware and goodware in APK classifications with almost 90% accuracy. In order to accurately categorize an application and expand processing assessment to a large number of applications, the minimum traffic volume required should be optimized.

2.1.2 NTPDroid: A Hybrid Android Malware Detector using Network Traffic and System Permissions [5]

Thousands of malware programs started to target mobile devices. Since then, a number of studies have suggested and assessed extremely precise machine-learning malware detection techniques. Two techniques for identifying Android malware have been proposed: Static Analysis and Dynamic Analysis. The two most often utilized detection attributes are permissions and network

traffic. Malicious programs download malware during runtime to avoid detection based on static permissions. Malware that does not need to be connected to a network tends to avoid network traffic-based detection which can be detected by permissions analysis. Using the FP-Growth algorithm, they train and test the suggested model to produce frequent patterns made up of permissions and traffic data. NTPDroid is their proposed hybrid Android malware detection solution. It is a prototype tool they used to extract traffic features and permissions from the applications through two phases. In the Analysis phase, using a combination of traffic attributes and permissions, they create regular malware patterns and normal datasets. This regular pattern creation can assist us in analyzing the patterns that are considerably present in both malware and healthy samples. These patterns comprise both traffic attributes and permissions. Throughout the detection phase, they identify the malicious applications by utilizing these recurring patterns. The Genome Malware dataset is used in their system. The results which show a detection accuracy of 94.25% demonstrated that, in comparison to employing either the traffic features or permissions, combining the two improved the detection rate.

2.1.3 Extensible Android Malware Detection and Family Classification Using Network Flows and API-Calls [6]

Due to their numerous features and ease of use, Android OS-based mobile devices have drawn a large number of end users. Consequently, Android has emerged as a crucial target for malevolent actors to initiate their destructive intentions. Their proposal is about presenting a two-layer Android malware analyzer based on static and dynamic features and enhancing their previous network-flow analyses with appending extracted network-gram sequential relations of API calls. The dataset was divided into two parts. First is putting out a comparison of the Android malware datasets that were previously accessible, taking into account 15 crucial factors. The second part used the CICAndMal2017 dataset, which consists of API calls as dynamic features and Permission and Intent as static features. The analysis consists of two layers Static Binary Classification (SBC) and Dynamic Malware Classification (DMC). An internal perspective of the analysis framework's SBC component is shown in the Fig. 2 below.

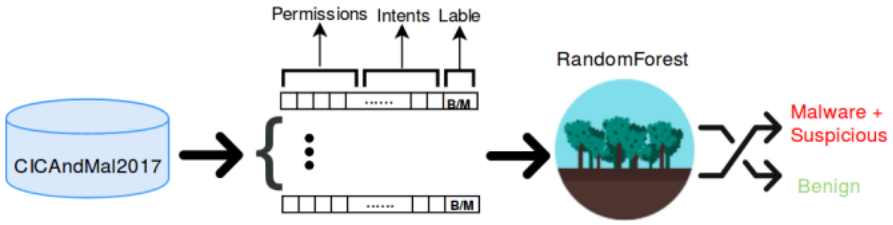


Figure 2: The SBC Analysis Framework

For DMC, it attempts to classify the input malware samples into four malware categories (Adware, ransomware, SMS malware, and scareware) and 39 malware families. Their results show that they were successful in attaining 83.3% precision in dynamic-based malware category classification, 59.7% precision in dynamic-based malware family classification, and 95.3% precision

in static-based malware binary classification at the first layer. Despite maintaining relatively low False Positive rates, they were able to improve results to 95.3% recall in malware binary classification, 81.0% recall in malware category classification, and 61.2% recall in malware family classification.

2.2 Business Applications

2.2.1 PCAPdroid[7]

PCAPdroid is a privacy-friendly open source app that allows you to monitor, analyse, and block connections made by other apps on your device. It also lets you export a PCAP dump of the traffic, inspect HTTP, decrypt TLS traffic.

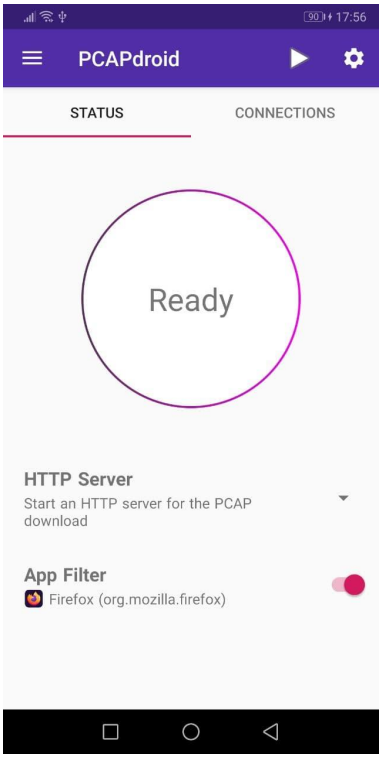


Figure 3: MTD: Malwarebytes for Android

2.2.2 Mobile Threat Defense (MTD) Solutions

MTD solutions are many that provide real-time protection against mobile threats, including those coming from applications on the Google Play Store such as Malwarebytes[8], Symantec Mobile Security[9].

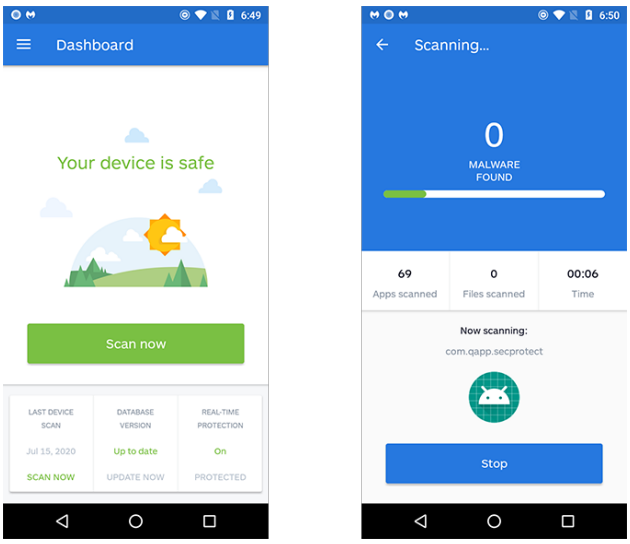


Figure 4: MTD: Malwarebytes for Android

3 System Description

3.1 Problem Statement

Available solutions out there tend to rely on real-time protection only over the Internet [10]. While this has its own advantage, it also poses a significant threat to the user's data and information because whenever the user is dissociated from the Internet it becomes vulnerable to all sorts of malicious activity that were hidden due to the application detecting that it's being monitored. Our project aims to solve this by implementing a profound approach to security by integrating security of the user's device locally without the need to connect to either Wi-Fi or mobile data. We also provide the user with notification once any malicious activity is detected, afterwards we wipe all his data (credit card into/transactions/etc..) and notify the corporate of the incident, if the problem is resolved, the user then has to re-enter his data.

3.2 System Overview

- Traffic Capture: The system has a live capturing model to capture real-time network traffic. This data is processed in real-time into a Wireshark PCAP file, which is analyzed by an extraction model to retrieve packet information, including IP addresses, port numbers, packet size, etc...

- Analysis: Extracted information undergoes preprocessing before being fed into a pre-trained deep-learning model designed to identify malicious packets. The model processes the extracted features to classify packets as normal benign or malicious.

- AI Model: A deep learning model, trained on data, is made to detect malicious network activities. Taking packets as input, the model classifies them as benign or malicious based on its training.

- Notification: When the model detects malicious activity, the deep learning model triggers an automatic notification to the user. The identified IP address is then added to a blacklist stored in the database.
- Database: A SQLite database is created to store a list of known benign and malicious IP addresses, categorizing them as white-listed or black-listed. The system retrieves IP addresses from the database as needed to determine the nature of incoming packets.
- User Interface: The system provides a user-friendly Android interface displaying historical data. Users can easily start or stop the program through this interface, facilitating efficient monitoring and comprehension of the system's performance.

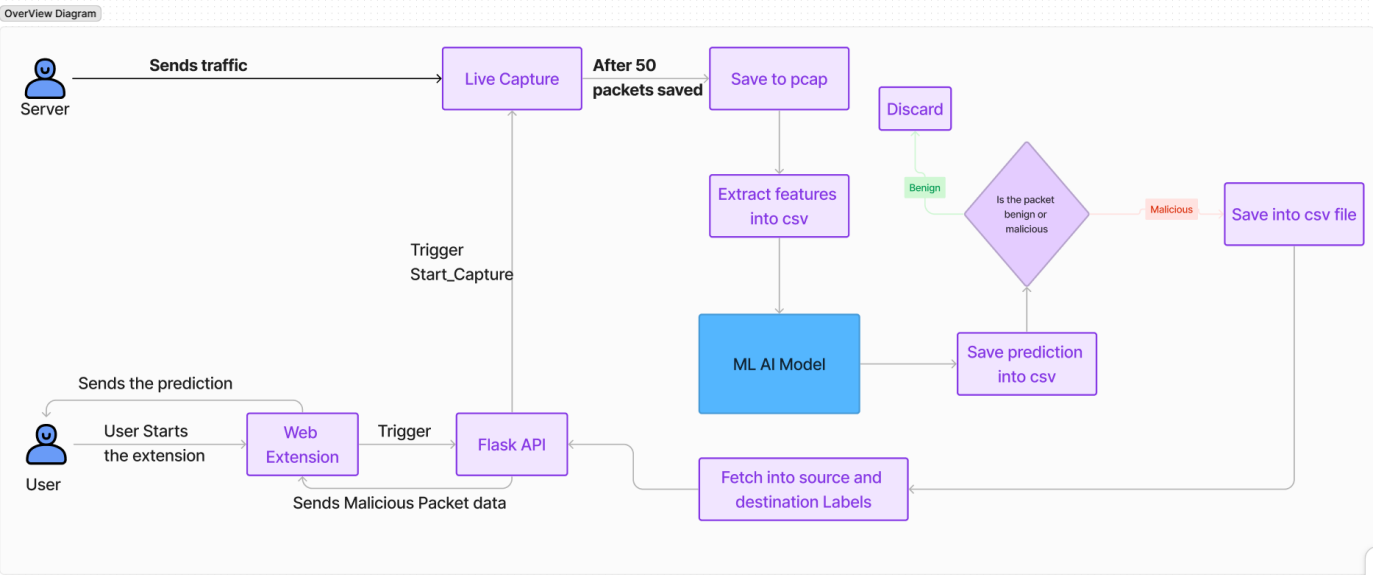


Figure 5: System Overview

3.3 System Scope

The system will include the following:

- Software: develop security software designed to notify Android phones of potential cyber-attacks.
- Real-time Traffic Capture Module: Implement a module to our model to capture network traffic in real-time.
- Traffic extraction Model: Develop a model to extract essential features from captured traffic data, including Timestamps of requests, Header Length, Protocol, Packet Size, and Source app bytes.
- AI Classification Model: Develop a trained deep learning model to classify the traffic, distinguish between benign and malicious activities.

- SQLite Database: Create a SQLite database for the storage of traffic data and the outcome of the AI analysis.
- Whitelisting: Implement a functionality where if the traffic is identified as benign for a couple of times, the IP is stored in a whitelist.
- User Interface: Develop a user interface to display system logs, providing users with insights into the system operations.

3.4 System Context

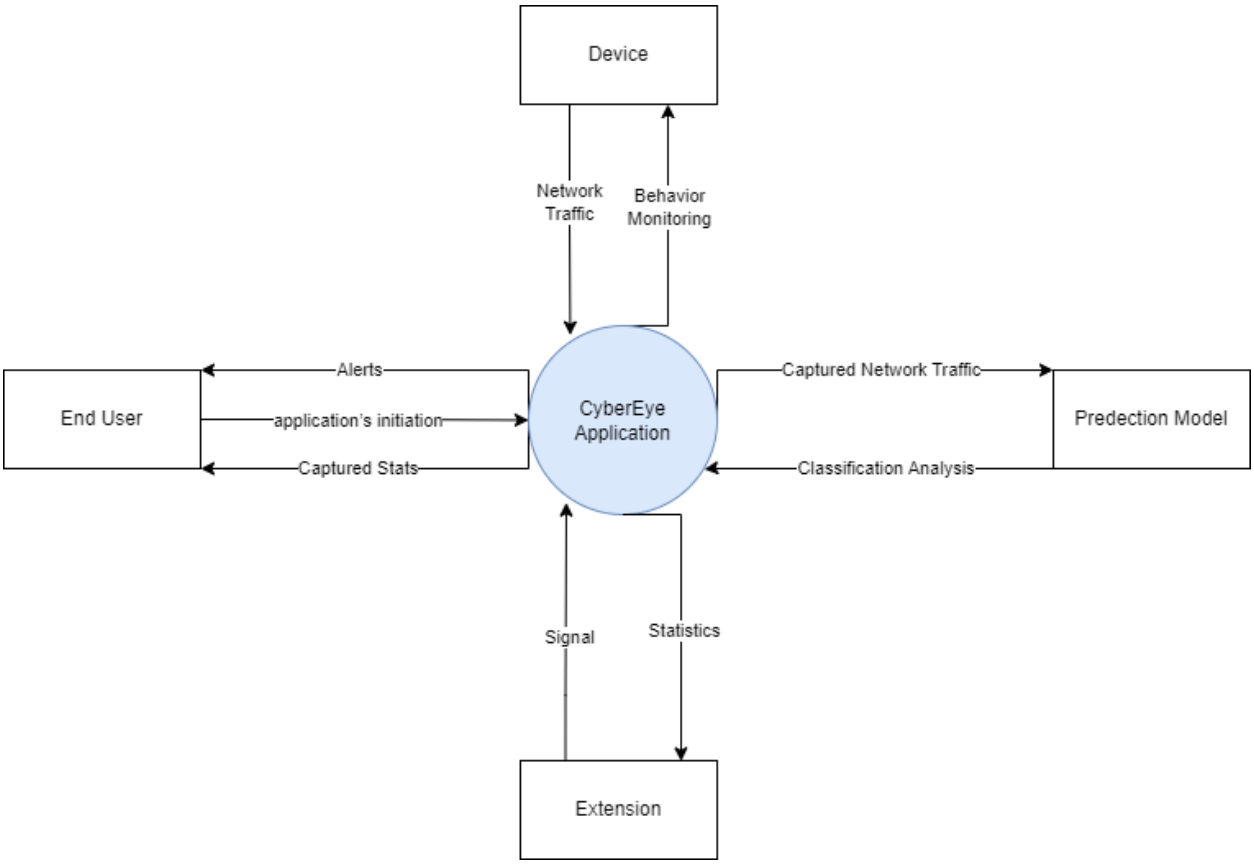


Figure 6: System context

3.5 Objectives

- Malicious Activity Detection: developing AI model for quick identification and classification of any malicious activities in the network.
- Continuous Traffic monitoring: real-time monitoring of network traffic to address potential security threats.
- ML Model: Develop an AI model to predict if incoming network traffic is safe, mainly during interactive application use.

- User-Friendly Interface: Design an intuitive user interface for administrators to easily monitor and manage system operations.
- Efficient Database for Quick Alerts: Set up a SQLite database to store and manage lists of known benign and malicious IP addresses. Enable notifications without unnecessary delays in queries.

3.6 User Characteristics

- Intended Users: The system is meant for network administrators, security analysts, and individuals or organizations responsible for the security infrastructure of their networks. This includes banks, hospitals, etc..
- Skill Requirements: Users are expected to possess an understanding of network security principles. It's also recommended to have knowledge in using applications that come with plugins.

4 Functional Requirements

4.1 System Functions

- ID:01 Start Program • The system initiates and start to monitor the network traffic.
- ID:02 Stop Program • The system should allow users to terminate the program easily.
- ID:03 View Malicious Activity • Users should be able to access a feature enabling the viewing of detected malicious activities.
- ID:04 Alert • The user should receive alerts when a malicious activity is discovered. • The system should be capable of real-time packet capture to monitor network activities.
- ID:05 Send New Data to Model • Users should be able to update the system's detection model by providing new data.
- ID:06 Check IP Status • The system should include a functionality to verify whether an IP address is listed in either the blacklisted or whitelisted.

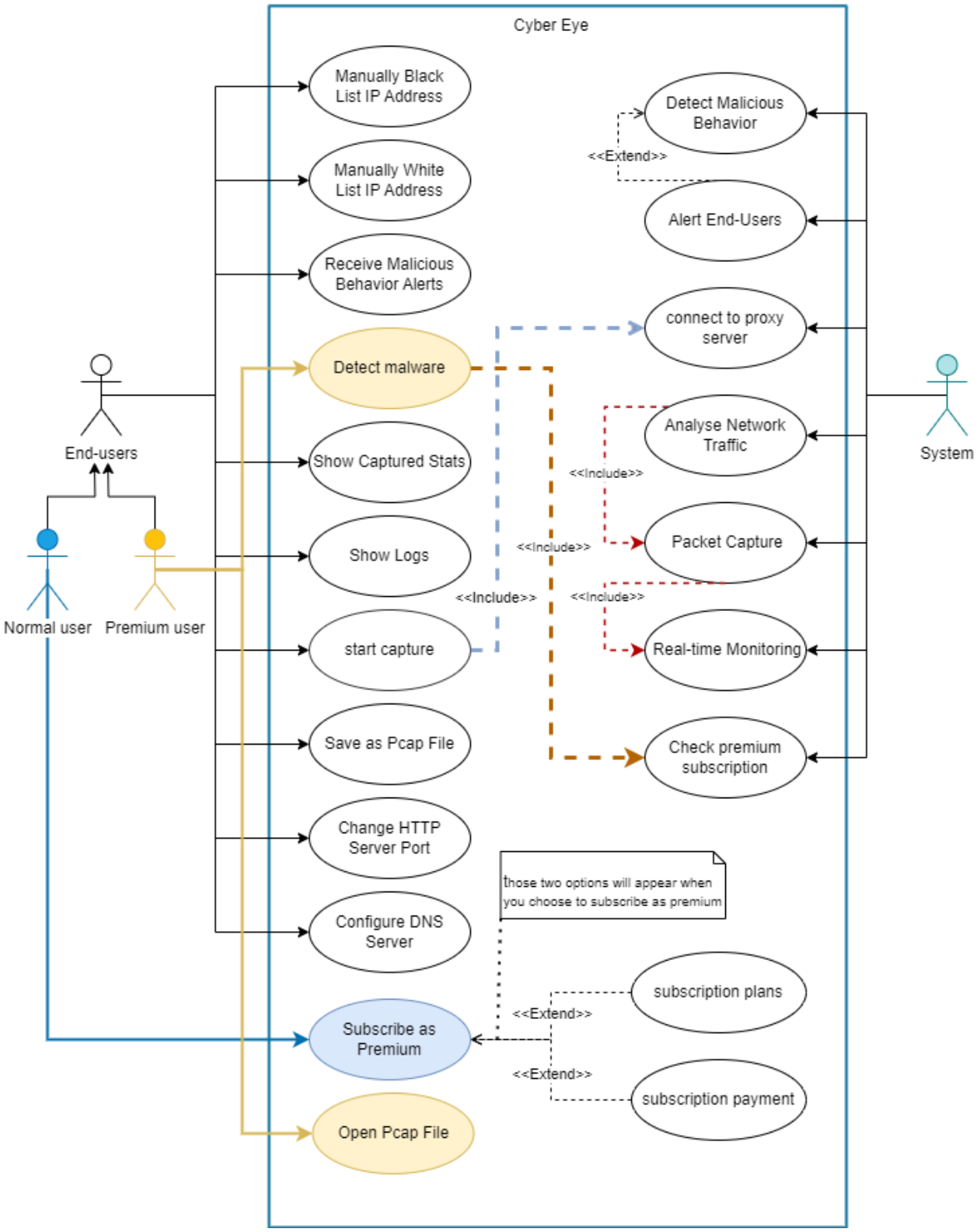


Figure 7: Use case Diagram

4.2 Detailed Functional Specification

Table 2: Live Packet Capture

Name	capture()
Code	ID:06
Priority	Extreme
Critical	The data is captured
Description	Takes the packets and captures them in pcap file
Input	Network Packets
Output	Captured data is stored in a pcap file
Pre-condition	Network connection should be established
Post-condition	Data should be captured
Dependency	None
Risk	Packet loss due to network dissociation could cause some data to be incomplete

Table 3: Alert

Name	Alert()
Code	ID:04
Priority	High
Critical	The malicious activity is found
Description	Alerts the user if any malicious activity is found.
Input	None
Output	Alert message
Pre-condition	Malicious activity is found
Post-condition	The user receives the alert message
Dependency	User Input
Risk	None

Table 4: Send New Data to Model

Name	send()
Code	ID:07
Priority	High
Critical	Data is sent to model
Description	The system gets sent new network data from the user where it's then trained to improve the model
Input	User's network data
Output	Better A.I. detection model
Pre-condition	Pcap file with network activities
Post-condition	The A.I. model is further enhanced
Dependency	The most capable model with the highest score to conduct the testing/training
Risk	Some data can be passed benign simply because they weren't identified before

Table 5: Check IP Status

Name	checkIP()
Code	ID:08
Priority	High
Critical	IP is categorized
Description	The IP is tested and checked whether it's been previously blacklisted or if it's whitelisted
Input	An IP address
Output	Categorization of that IP address
Pre-condition	A trained A.I. model
Post-condition	IP is placed in either white or black list
Dependency	None
Risk	None

Table 6: view Malicious Activity

Name	viewMaliciousActivity()
Code	ID:03
Priority	Medium
Critical	The system won't be able to view malicious activity without capturing them
Description	Users should be able to access a feature enabling the viewing of detected malicious activities
Input	Captured Malicious activity
Output	Malicious activity details
Pre-condition	The system should have captured malicious activity
Post-condition	The system displays the detected malicious activities
Dependency	Detection of malicious activities
Risk	None

5 Design Constraints

- Operating System Compatibility: The system shall comply with API 30 ("R") for Android 11.0 and above.
- Network Connection: A high-speed internet connection is required.

6 Non-functional Requirements

- Maintainability : The system shall be easy to maintain by continuously providing improvements and updates.
- Usability : The system will provide easy navigation and enhance overall usability by providing familiar and understandable UI and satisfactory UX.
- Security : The system keep users informed about any malicious activities on the phone.
- Performance : The system must demonstrate accurate traffic capture, ensuring high-performance levels in identifying and analyzing network activities.
- Reliability : The system is expected to deliver a high detection rate, effectively distinguishing between benign and malicious IP addresses.
- Availability : The system have the ability to maintain continuous availability, ensuring users can capture and detect malicious activity at any time.

7 Data Design

The CICAndMal2017 dataset, consisting of over 355,000 instances of Android network traffic data, has been employed to construct a predictive model with a focus on enhancing network security. This extensive dataset encompasses 85 features, incorporating labels that distinguish between normal connections and various types of attacks, namely Adware, Scareware, and SMS Malware. more than 67,000 instances, is labeled as Benign, 147443 as Android Adware 117082 Android Scareware and, 67397 as Android SMS Malware which is over 300,000 malicious.

	A	B	C	D	E	F	G	H	I	J	K
1		Flow ID	Source IP	Source Port	Destination IP	Destination Port	Protocol	Timestamp	Flow Duration	Total Fwd Packets	Label
2	0	172.217.6.202-10.42.0.211-443-50004-6	10.42.0.211	50004	172.217.6.202	443	6	13/06/2017 11:52:39	37027	1	Android_Adware
3	1	172.217.6.202-10.42.0.211-443-35455-6	10.42.0.211	35455	172.217.6.202	443	6	13/06/2017 11:52:39	36653	1	Android_Adware
4	2	131.253.61.68-10.42.0.211-443-51775-6	10.42.0.211	51775	131.253.61.68	443	6	13/06/2017 11:52:42	534099	8	Android_Adware
5	3	131.253.61.68-10.42.0.211-443-51775-6	10.42.0.211	51775	131.253.61.68	443	6	13/06/2017 11:52:43	9309	3	Android_Adware
6	4	131.253.61.68-10.42.0.211-443-51776-6	10.42.0.211	51776	131.253.61.68	443	6	13/06/2017 11:52:42	19890496	8	Android_Adware
7	5	10.42.0.211-23.208.43.179-50186-443-6	10.42.0.211	50186	23.208.43.179	443	6	13/06/2017 11:52:43	19196263	5	Android_Adware
8	6	10.42.0.211-23.208.43.179-50186-443-6	10.42.0.211	50186	23.208.43.179	443	6	13/06/2017 11:53:03	4304325	1	Android_Adware
9	7	172.217.3.110-10.42.0.211-443-40087-6	10.42.0.211	40087	172.217.3.110	443	6	13/06/2017 11:53:16	37926	1	Android_Adware
10	8	172.217.3.110-10.42.0.211-443-52564-6	10.42.0.211	52564	172.217.3.110	443	6	13/06/2017 11:53:17	36631	1	Android_Adware
11	9	10.42.0.211-31.13.71.3-46130-443-6	10.42.0.211	46130	31.13.71.3	443	6	13/06/2017 11:53:33	36535	1	Android_Adware
12	10	10.42.0.211-31.13.71.1-47658-443-6	10.42.0.211	47658	31.13.71.1	443	6	13/06/2017 11:53:33	221194	2	Android_Adware
13	11	10.42.0.211-31.13.71.1-47658-443-6	10.42.0.211	47658	31.13.71.1	443	6	13/06/2017 11:53:34	914675	2	Android_Adware
14	12	10.42.0.211-31.13.71.1-47658-443-6	10.42.0.211	47658	31.13.71.1	443	6	13/06/2017 11:53:37	3692864	2	Android_Adware
15	13	10.42.0.211-31.13.71.1-47658-443-6	10.42.0.211	47658	31.13.71.1	443	6	13/06/2017 11:53:48	14770330	2	Android_Adware

Figure 8: ML Dataset

8 Preliminary Object-Oriented Domain Analysis

Initial Class Diagram

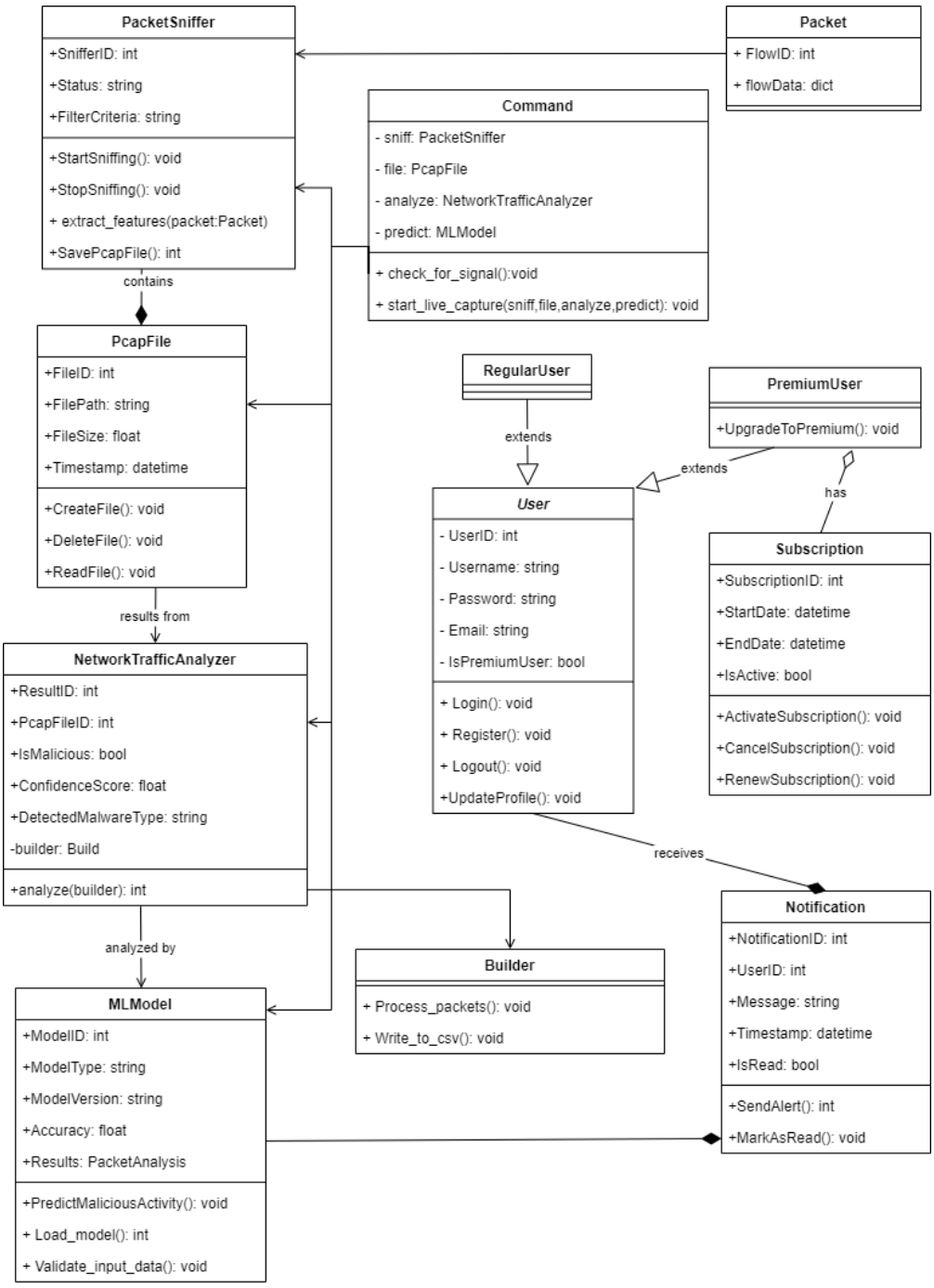


Figure 9: UML Diagram

9 Operational Scenarios

Scenario 1:
Assumption: The user will start the application for live packet capture.
Normal: The application monitors all incoming traffic, checking against a predefined blacklist

and whitelist. Whitelisted IP addresses are allowed, while those on the blacklist are immediately blocked.
What can go wrong: The application might incorrectly classify traffic, leading to legitimate packets being blocked or malicious packets being allowed. There could be performance issues if the application consumes excessive system resources, impacting the functionality of other applications.

Scenario 2:
Assumption: Upon initiation, the application uses a database for initial packet verification and then employs a predictive classification model to sort and categorize incoming packets.
Normal: Packets identified as malicious by the classification model are labeled as such. These associated IP addresses are then blocked and added to a blacklist to prevent future access.
What can go wrong: The predictive classification model might have a high false positive or false negative rate, leading to incorrect blocking or allowing of packets.

Scenario 3:
Assumption: The application begins by checking a database and utilizes a predictive classification model to categorize received packets.
Normal: Packets determined to be benign are allowed to pass without any action, and the system operates as intended without interrupting the user's activities.
What can go wrong: The classification model may incorrectly identify benign packets as malicious. The reliance on a database for initial checks might lead to outdated information being used for packet classification.

10 Project Plan

Table 7: Project Plan

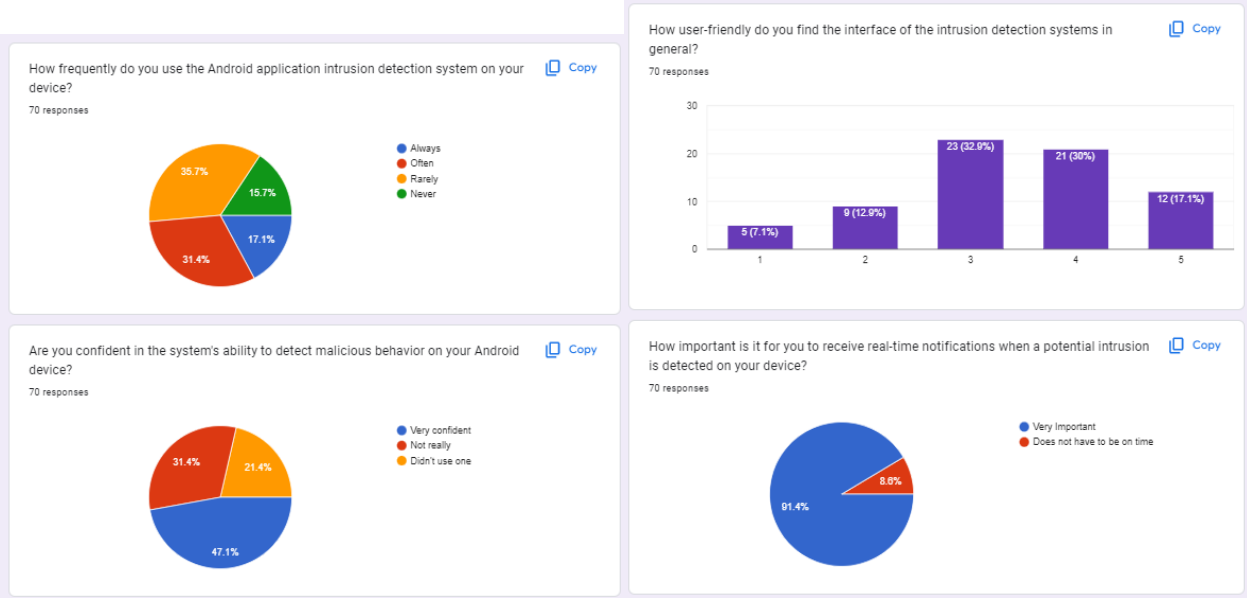
Task Name	Start date	End date	Duration
SRS Document	29/12/2023	13/1/2024	16 days
Backend Development	11/12/2023	13/1/2024	33 days
Frontend Development	20/12/2023	14/1/2024	23 days
Plugin Development	14/12/2023	14/1/2024	31 days

11 Appendices

11.1 Definitions, Acronyms, Abbreviations

- PCAP: Packet Capture
- MTD: Mobile Threat Defense
- IP: internet protocol

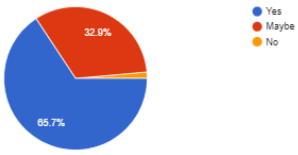
11.2 Supportive Documents



Do you feel that the intrusion detection system has had a positive impact on the security of your Android device?

Copy

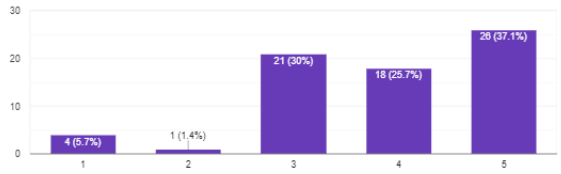
70 responses



How likely are you to recommend this intrusion detection system to other Android device users?

Copy

70 responses



Have you ever encountered any issues or bugs while using the intrusion detection system? If yes, please describe the problem.

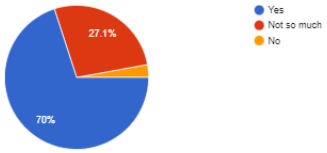
70 responses

- no i have not
- No I don't face any issues
- i did not face any actually
- Never
- No I didn't encounter any
- NA
- no
- No
- didn't use one before

Do you believe that the intrusion detection systems have helped you become more aware of potential security threats on your Android device?

Copy

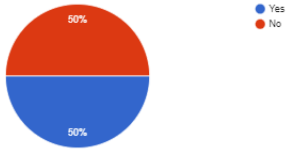
70 responses



Have you experienced any performance issues or slowdowns on your device while the intrusion detection system is active?

Copy

70 responses



Are there any specific features or functionalities you would like to see added to the intrusion detection system in future updates?

30 responses

- Faster and easier to use
- detect hard encoded malicious software
- maybe red flag some apps that other users say they have malware or something?
- Yes
- an alert notification
- Thanks
- No thats good
- No thank you and good luck
- Enhanced network traffic analysis and alarma detection

References

[1] Author(s). *Title of the Article*. Publication Year. URL: https://www.vice.com/en/article/43z93g/hackers-hid-android-malware-in-google-play-store-exodus-esurv.

[2] Vijay Cherukuri. *Data Is the Key to Everything: Why Data Is the Most Valuable Commodity for Businesses*. 2022. URL: https://www.kdnuggets.com/2022/03/data-valuable-commodity-businesses.html.

[3] Author(s) or Organization. *12 Reasons Why Data is Important*. Publication Year. URL: https://www.c-q-l.org/resources/guides/12-reasons-why-data-is-important/.

[4] Igor Jochem Sanz, Martin Andreoni Lopez, Eduardo Kugler Viegas, et al. "A lightweight network-based android malware detection system". In: (2020), pp. 695–703.

[5] Anshul Arora and Sateesh K Peddoju. "NTPDroid: a hybrid android malware detector using network traffic and system permissions". In: (2018), pp. 808–813.

[6] Laya Taheri, Andi Fitriah Abdul Kadir, and Arash Habibi Lashkari. "Extensible android malware detection and family classification using network-flows and API-calls". In: (2019), pp. 1–8.

[7] Author(s) or Organization. *ppcap*. Publication Year. URL: https://play.google.com/store/apps/details?id=com.emanuelef.remote_capture&hl=en&gl=US.

[8] Author(s) or Organization. *Malwarebytes*. Publication Year. URL: https://www.malwarebytes.com/.

[9] Author(s) or Organization. *Symantec Mobile Security*. Publication Year. URL: https://docs.broadcom.com/doc/endpoint-protection-mobile-en.

[10] José Gaviria de la Puerta, Iker Pastor-López, Igone Porto, et al. "Detecting malicious Android applications based on the network packets generated". In: *Neurocomputing* 456 (2021), pp. 629–636. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.08.095. URL: https://www.sciencedirect.com/science/article/pii/S0925231221009504.