

ANALYSING ANDROID APP PRIVACY USING CLASSIFICATION ALGORITHM

at

**Sathyabama Institute of Science and Technology
(DEEMED TO BE UNIVERSITY)**

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering Degree in Computer Science and Engineering

By

**DHANEKULA NAGA GANESH
(Reg. No. 370110187)**



**DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING SCHOOL OF COMPITING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
JEPPIAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

March - 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that the project Report is the Bonafide work of **DHANEKULA NAGA GANESH (Reg. No. 37110187)** who carried out the project entitled **“ANALYSING ANDROID APP PRIVACY USING CLASSIFICATION ALGORITHM”** under our supervision from November 2020 to March 2021.

Internal Guide

Dr. N. SRINIVASAN, M.E., Ph.D.,

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **DHANEKULA NAGA GANESH (Reg. No. 37110187)**, hereby declare that the Professional Training Report on "**ANALYSING ANDROID APP PRIVACY USING CLASSIFICATION ALGORITHM**" done under the guidance of **Dr. N. SRINIVASAN, M.E., Ph.D.**, at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean, School of Computing** and **Dr. S. VIGNESHWARI, M.E., Ph.D., Head of the Department, Department of Computer Science and Engineering** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. N. SRINIVASAN, M.E., Ph.D.,** for his valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways For the Completion of the project.

ABSTRACT

The interface allows the user to search for an arbitrary application on the Play Store; the permissions list and the privacy policy are then automatically retrieved, whenever possible. The user has then the ability of selecting a specific permission, and a list of relevant sentences are extracted by the privacy policy and presented to them, along with an accurate description of the permission itself. Such an interface allows the user to quickly evaluate the privacy-related risks of an Android application, by highlighting the relevant sections of the privacy policy and by providing useful information about sensible permissions. We presented a novel approach to the analysis of privacy policies in the context of Android applications. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. The tool is developed with expandability in mind, and further developments in the approach can easily be integrated in order to increase the reliability and effectiveness. In addition, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws. We proposed, A user who wishes to install and use any third-party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scenes.

TABLE OF CONTENTS

Chapter No.	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1	INTRODUCTION	1
	1.1 OBJECTIVE	2
	1.2 DOMAIN INTRODUCTION	2
	1.3 OVERVIEW OF THE PROJECT	3
2	LITERATURE SURVEY	4
3	AIM AND SCOPE OF PRESENT INVESTIGATION	10
	3.1 EXISTING SYSTEM	10
	3.1.1 DISADVANTAGES	10
	3.2 PROPOSED SYSTEM	10
	3.2.1 ADVANTAGES	10
	3.3 SYSTEM ARCHITECTURE	11
	3.4 SYSTEM REQUIREMENTS	11
	3.4.1 HARDWARE REQUIREMENTS	12
	3.4.2 SOFTWARE REQUIREMENTS	12
	3.5 DATA FLOW DIAGRAMS	12
	3.6 UML DIAGRAMS	14
	3.4.1 USE CASE DIAGRAM	15
	3.4.2 CLASS DIAGRAM	15
	3.4.3 ACTIVITY DIAGRAM	16
	3.4.4 SEQUENCE DIAGRAM	17
	3.4.5 COLLABORATION DIAGRAM	18
	3.4.6 COMPONENT DIAGRAM	19
	3.7 MODULES	20
	3.7.1 LOGIN MODULE	20
	3.7.2 ANDROID APPLICATION SCAN	20
	3.7.3 MALWARE DETECTION	21

	3.7.4 ANDROID PERMISSION CHECK	22
	3.7.5 PRIVACY POLICY DATABASE	22
4	EXPERIMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED	24
	4.1 ALGORITHMS USED	24
	4.1.1 DECISION TREE ALGORITHM	24
	4.1.2 NAIVE BAYES ALGORITHM	24
	4.2 FEASIBILITY STUDY	25
	4.2.1 ECONOMIC FEASIBILITY	25
	4.2.2 TECHNICAL FEASIBILITY	26
	4.2.3 SOCIAL FEASIBILITY	26
	4.3 COLLECTION FRAMEWORK	26
	4.4 SYSTEM DESIGN AND TESTING PLAN	27
	4.4.1 INPUT DESIGN	27
	4.4.2 OUTPUT DESIGN	28
	4.4.3 SYSTEM TESTING	28
	4.4.4 BASICS OF SOFTWARE TESTING	29
5	RESULTS AND DICUSSION	30
6	CONCLUSION AND FUTURE WORK	34
	6.1 CONCLUSION	34
	6.2 FUTURE WORK	34
	REFERENCES	36
	APPENDIX	38
	A. SOURCE CODE	38
	B. PAPERWORK	60

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
3.3	SYSTEM ARCHITECTURE	11
3.5	DATA FLOW DIAGRAM	13
3.6.1	USE CASE DIAGRAM	15
3.6.2	CLASS DIAGRAM	16
3.6.3	ACTIVITY DIAGRAM	17
3.6.4	SEQUENCE DIAGRAM	18
3.6.5	COLLABORATION DIAGRAM	19
3.6.6	COMPONENT DIAGRAM	19
4.3	COLLECTION FRAMEWORK	27
5.1	SCANNING PHASE	30
5.2	RISK PERCENTAGE	31
5.3	REPLACEMENT SUGGESTION	32
5.4	EXISTING VS PROPOSED ACCURACY	33

LIST OF ABBREVIATIONS

UML	UNIFIED MODELING LANGUAGE
XML	EXTENSIBLE MARKUP LANGUAGE
GP-PP	GENERIC PERMISSIONS-PRIVACY INVASIVE PERMISSIONS
API	APPLICATION PROGRAMMING INTERFACE
PCFG	PROBABILISTIC CONTEXT FREE GRAMMAR
ASEF	ANDROID SECURITY EVALUATION FRAMEWORK
SAAF	STATIC ANDROID ANALYSIS FRAMEWORK
KNN	K-NEAREST NEIGHBORS
J2SE	JAVA 2 PLATFORM, STANDARD EDITION

CHAPTER - 1

INTRODUCTION

Mobile devices of various operating systems have exhibited a steep increase in the past decade, thus leading to an increase in the number and variety of applications that run on mobile devices. Smartphones are recently used either in a complementary manner to a desktop computer or even to replace it. A closer look at the purpose of using mobile phones reveal that they are mostly used for web browsing, social networking, and online banking. In addition, they are used for mobile-specific functions such as SMS messaging, real-time broadcasting of location, and ubiquitous access. As the capabilities in mobile phone functionality increase (e.g., applications for personal health), mobile phones become more and more attractive for a wider population. Market data surveys reveal that the number of smartphone sales worldwide reached 208 million in 2012.

This was a 38.3% increase compared to the sales in 2011. The development of smartphones and mobile applications has resulted in a major change in people's way of doing tasks in various aspects of daily life, such as making business and conducting social communication. Mobile phone applications exhibit a rich variety not only in common daily life activities but also for users with more specific needs. From games to multimedia applications, navigation systems, and health-related applications, recently available mobile application markets, such as Google's Play Market, Apple's App Store, or Microsoft's Windows Store offer a wide variety of applications to users with different needs. The major application markets have been growing steadily both in terms of the applications offered to the users and the downloads performed by the users. The fast increase in the popularity of smartphones has led to an increase in their potential as a target for malicious activities. This is mainly because users provide access for various types of private information by means of mobile applications.

As a result, some applications in the market have been identified as performing malicious activities. The spread of malicious software is also influenced by the policy of the market providers. For instance, Apple's App Store recently applies a policy that is subject to strict registration and company-issued digital certification before the release of any application, thus providing a security check for the applications listed in their application platform regularly. Others, such as

Google's Play Market, introduce more freedom to developers in uploading their own software to the market. The applications are then removed from the market in case of reported malicious activity. In particular, the Android operating system (in its recent form) allows removal of the malicious application from the device remotely. A similar mechanism of removal is also employed by Apple's App Store. The policy of post-detection removal of malicious software brings the need for early detection of malware applications.

1.1 OBJECTIVE

The main objective of this system, A user who wishes to install and use any third-party app does not understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scenes.

1.2 DOMAIN INTRODUCTION

Android is a mobile operating system developed by Google. It is based on a modified version of the Linux kernel and other open source software, and is designed primarily for touchscreen mobile devices such as smartphones and tablets. In addition, Google has further developed Android TV for televisions, Android Auto for cars, and Wear OS for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The operating system has since gone through multiple major releases, with the current version being 9 "Pie", released in August 2018. Google released the first Android Q beta on all Pixel phones on March 13, 2019. The core Android source code is known as Android Open-Source Project (AOSP), and is primarily licensed under the Apache License.

At Google I/O 2014, the company revealed that there were over one billion active monthly Android users, up from 538 million in June 2013. Android's source code is released by Google under open-source licenses, although most Android

devices ultimately ship with a combination of open source and proprietary software, including proprietary software developed and licensed by Google.

Initially developed by Android, Inc., which Google bought in 2005, Android was unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices.

1.3 OVERVIEW OF THE PROJECT

The GP-PP (Generic Permissions-Privacy invasive permissions) model is useful to classify the permissions into Generic and Privacy invasive permissions. The model proposes a simplistic way for users to decide which apps are dangerous to install. Based on the permission set that a particular app requests, the GP-PP model classifies an app as privacy invasive if majority of the permissions requested are privacy invasive. So, users can decide which set of permissions can be harmful. Therefore, by looking at the set of permissions requested, a user can determine whether an app is privacy invasive or not. In other words, a user can determine whether it is safe to install the said app or not. Although the proposed GP- PP model seems valuable, its efficacy has not been fully assessed.

To address this gap, we test the effectiveness of the GP-PP model using Naïve Bayes Classifier in this paper. In particular, we validate the GP-PP model in order to verify whether the model classifies an app on the basis of permission sets that the app requests.

Our study confirms that greater the number of Privacy-invasive permissions that an app requests, more dangerous it is to install the app. In summary, the major contribution of the paper is the validation of the proposed GP-PP model using Naïve Bayes Classifier and decision tree algorithm for popularity.

CHAPTER - 2

LITERATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system, the above consideration is taken into account for developing the proposed system. The major part of the project development sector considers and fully survey all the required needs for developing the project. For every project Literature survey is the most important sector in software development process. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

Analysis of Bayesian classification-based approaches for Android malware detection:

Mobile malware has been growing in scale and complexity spurred by the unabated uptake of smartphones worldwide. Android is fast becoming the most popular mobile platform resulting in sharp increase in malware targeting the platform. Additionally, Android malware is evolving rapidly to evade detection by traditional signature-based scanning. Despite current detection measures in place, timely discovery of new malware is still a critical issue. This calls for novel approaches to mitigate the growing threat of zero-day Android malware. Hence, the authors develop and analyse proactive machine-learning approaches based on Bayesian classification aimed at uncovering unknown Android malware via static analysis. The study, which is based on a large malware sample set of

majorities of the existing families, demonstrates detection capabilities with high accuracy. Empirical results and comparative analysis are presented offering useful insight towards development of effective static-analytic Bayesian classification-based solutions for detecting unknown Android malware.

Android permissions: a perspective combining risks and benefits

The phenomenal growth of the Android platform in the past few years has made it a lucrative target of malicious application (app) developers. There are numerous instances of malware apps that send premium rate SMS messages, track users' private data, or apps that, even if not characterized as malware, conduct questionable actions affecting the user's privacy or costing them money. In this paper, we investigate the feasibility of using both the permissions an app requests, the category of the app, and what permissions are requested by other apps in the same category to better inform users whether the risks of installing an app is commensurate with its expected benefit. Existing approaches consider only the risks of the permissions requested by an app and ignore both the benefits and what permissions are requested by other apps, thus having a limited effect. We propose several risk signals that and evaluate them using two datasets, one consists of 158,062 Android apps from the Android Market, and another consists of 121 malicious apps. We demonstrate the effectiveness of our proposal through extensive data analysis.

Detecting repackaged smartphone applications in third-party android marketplaces.

Recent years have witnessed incredible popularity and adoption of smartphones and mobile devices, which is accompanied by large amount and wide variety of feature-rich smartphone applications. These smartphone applications (or apps), typically organized in different application marketplaces, can be conveniently browsed by mobile users and then simply clicked to install on a variety of mobile devices. In practice, besides the official marketplaces from platform vendors (e.g., Google and Apple), a number of third-party alternative marketplaces have also been created to host thousands of apps (e.g., to meet regional or localization needs). To maintain and foster a hygienic smartphone app ecosystem, there is a need for each third-party marketplace to offer quality apps to mobile users. In this

paper, we perform a systematic study on six popular Android-based third-party marketplaces. Among them, we find a common “in-the-wild” practice of repackaging legitimate apps (from the official Android Market) and distributing repackaged ones via third-party marketplaces. To better understand the extent of such practice, we implement an app similarity measurement system called Droid MOSS that applies a fuzzy hashing technique to effectively localize and detect the changes from app-repackaging behaviour. The experiments with Droid MOSS show a worrisome fact that 5% to 13% of apps hosted on these studied marketplaces are repackaged. Further manual investigation indicates that these repackaged apps are mainly used to replace existing in-app advertisements or embed new ones to “steal” or re-route ad revenues. We also identify a few cases with planted backdoors or malicious payloads among repackaged apps. The results call for the need of a rigorous vetting process for better regulation of third-party smartphone application marketplaces.

Is this app safe?: a large-scale study on application permissions and risk signals.

Third-party applications (apps) drive the attractiveness of web and mobile application platforms. Many of these platforms adopt a decentralized control strategy, relying on explicit user consent for granting permissions that the apps request. Users have to rely primarily on community ratings as the signals to identify the potentially harmful and inappropriate apps even though community ratings typically reflect opinions about perceived functionality or performance rather than about risks. With the arrival of HTML5 web apps, such user-consent permission systems will become more widespread. We study the effectiveness of user-consent permission systems through a large-scale data collection of Facebook apps, Chrome extensions and Android apps. Our analysis confirms that the current forms of community ratings used in app markets today are not reliable indicators of privacy risks of an app. We find some evidence indicating attempts to mislead or entice users into granting permissions: free applications and applications with mature content request more permissions than is typical; 'look-alike' applications which have names similar to popular applications also request more permissions than is typical. We also find that across all three platforms popular applications request more permissions than average.

Analysing sensitive data transmission in android for privacy leakage detection

Android phones often carry personal information, attracting malicious developers to embed code in Android applications to steal sensitive data. With known techniques in the literature, one may easily determine if sensitive data is being transmitted out of an Android phone. However, transmission of sensitive data in itself does not necessarily indicate privacy leakage; a better indicator may be whether the transmission is by user intention or not. When transmission is not intended by the user, it is more likely a privacy leakage. The problem is how to determine if transmission is user intended. As a first solution in this space, we present a new analysis framework called AppIntent. For each data transmission, AppIntent can efficiently provide a sequence of GUI manipulations corresponding to the sequence of events that lead to the data transmission, thus helping an analyst to determine if the data transmission is user intended or not. The basic idea is to use symbolic execution to generate the aforementioned event sequence, but straightforward symbolic execution proves to be too time-consuming to be practical. A major innovation in AppIntent is to leverage the unique Android execution model to reduce the search space without sacrificing code coverage. We also present an evaluation of AppIntent with a set of 750 malicious apps, as well as 1,000 top free apps from Google Play. The results show that AppIntent can effectively help separate the apps that truly leak user privacy from those that do not.

Android permissions demystified:

Android provides third-party applications with an extensive API that includes access to phone hardware, settings, and user data. Access to privacy- and security-relevant parts of the API is controlled with an install-time application permission system. We study Android applications to determine whether Android developers follow least privilege with their permission requests. We built Stowaway, a tool that detects over privilege in compiled Android applications. Stowaway determines the set of API calls that an application uses and then maps those API calls to permissions. We used automated testing tools on the Android API in order to build the permission map that is necessary for detecting over privilege. We apply Stowaway to a set of 940 applications and find that about one-

third are overprivileged. We investigate the causes of over privilege and find evidence that developers are trying to follow least privilege but sometimes fail due to insufficient API documentation.

Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing

Smartphone security research has produced many useful tools to analyse the privacy-related behaviours of mobile apps. However, these automated tools cannot assess people's perceptions of whether a given action is legitimate, or how that action makes them feel with respect to privacy. For example, automated tools might detect that a blackjack game and a map app both use one's location information, but people would likely view the map's use of that data as more legitimate than the game. Our work introduces a new model for privacy, namely privacy as expectations. We report on the results of using crowdsourcing to capture users' expectations of what sensitive resources mobile apps use. We also report on a new privacy summary interface that prioritizes and highlights places where mobile apps break people's expectations. We conclude with a discussion of implications for employing crowdsourcing as a privacy evaluation technique.

Android platform-based individual privacy information protection system

With the popularity of mobile phones with Android platform, Android platform-based individual privacy information protection has been paid more attention to. In consideration of individual privacy information problem after mobile phones are lost, this paper tried to use SMS for remote control of mobile phones and providing comprehensive individual information protection method for users and completed a mobile terminal system with self-protection characteristics. This system is free from the support of the server and it can provide individual information protection for users by the most basic SMS function, which is an innovation of the system. Moreover, the protection mechanism of the redundancy process, trusted number mechanism and SIM card detection mechanism are the innovations of this system. Through functional tests and performance tests, the system could satisfy user functional and non-functional requirements, with stable operation and high task execution efficiency.

A framework for static detection of privacy leaks in android applications

We report on applying techniques for static information flow analysis to identify privacy leaks in Android applications. We have crafted a framework which checks with the help of a security type system whether the Dalvik bytecode implementation of an Android app conforms to a given privacy policy. We have carefully analysed the Android API for possible sources and sinks of private data and identified exemplary privacy policies based on this. We demonstrate the applicability of our framework on two case studies showing detection of privacy leaks.

CHAPTER – 3

AIM AND SCOPE OF PRESENT INVESTIGATION

3.1 EXISTING SYSTEM

- The GP-PP (Generic Permissions-Privacy invasive permissions) model is useful to classify the permissions into Generic and Privacy invasive permissions. The model proposes a simplistic way for users to decide which apps are dangerous to install.
- Based on the permission set that a particular app requests, the GP-PP model classifies an app as privacy invasive if majority of the permissions requested are privacy invasive. So, users can decide which set of permissions can be harmful.
- We validate the GP-PP model in order to verify whether the model classifies an app on the basis of permission sets that the app requests.

3.1.1 DISADVANTAGES OF EXISTING SYSTEM

- Security is low.
- Low Accuracy.

3.2 PROPOSED SYSTEM

- Identify the list of third party installed applications.
- Extract the complete list of permissions of each application.
- Identify android: protection level of each permission, i.e., Normal or Dangerous of each app.
- Take android app permissions dataset. To identify the harmful applications, apply classification algorithms.
- Note the accuracy of spam classification given by it and time required for execution.
- Results as accuracy among different harmful and normal apps Classifiers are analysed.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- Security is High.
- High Accuracy.

3.3 SYSTEM ARCHITECTURE

In the mobile phones, the applications will be listed. There will be two process steps they are, the applications will be extracted using XML code and the applications will be decompiled using Java code. In XML after data analyzing there will be many permissions will be generated and then in java after data analyzing there will be API function calls. The android apps will be scanned using decision tree algorithm, after scanning process the apps will be classified into malicious and non-malicious. Using decision tree algorithm if it is yes then app is malicious and if it is no then the app is not malicious. By this architecture we are representing our project model

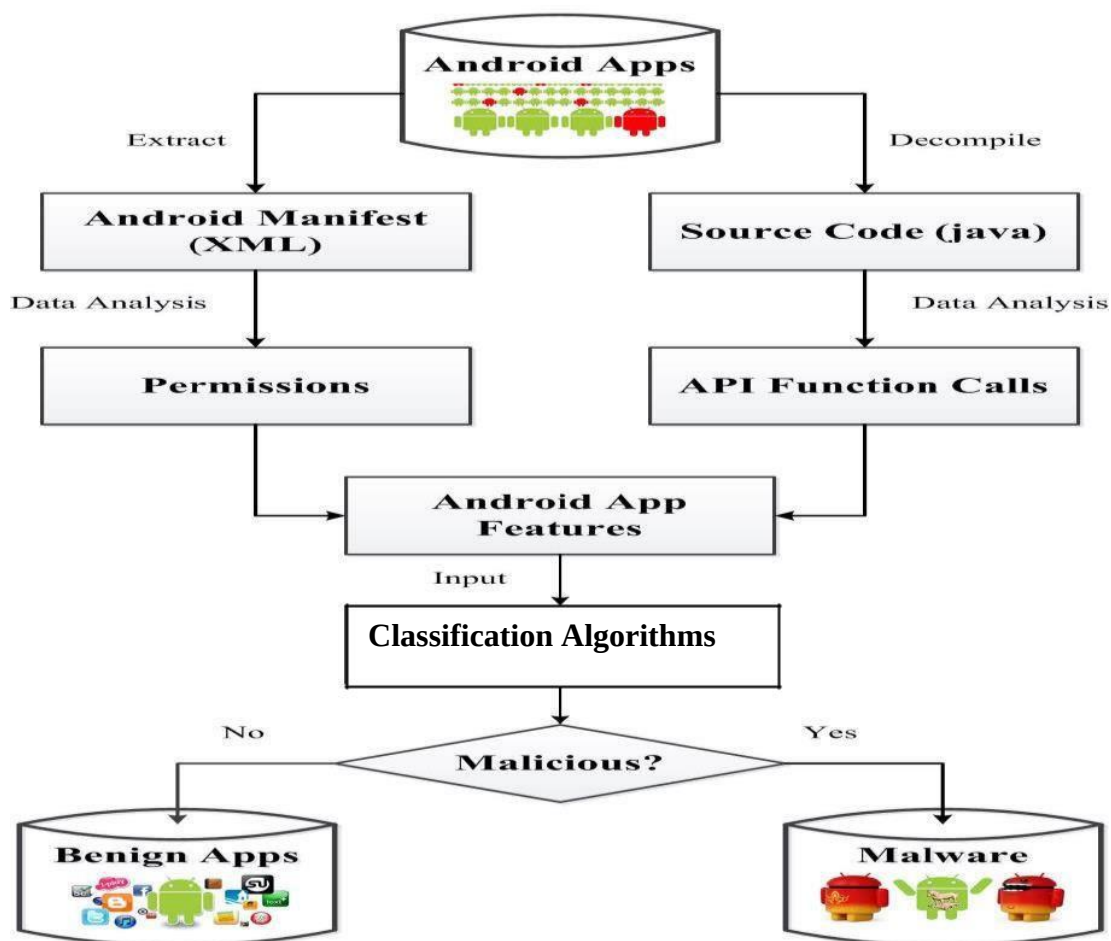


Fig 3.3: System Architecture

3.4 SYSTEM REQUIREMENTS

A Software Requirements Specification (SRS) is a description of a software system to be developed. It lays out functional and non- functional requirements

and may include a set of use cases that describe user interactions that the software must provide. The requirements are contemplated and segregated into user requirements, system requirements and functional requirements. The process to gather the software requirements from client, analyses and document them is known as requirement engineering. The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification document.

- Studying the existing or obsolete system and software
- Conducting interviews of users and developers
- Referring to the database or
- Collecting answers from the questionnaires.

3.4.1 Hardware Requirements:

System : Pentium Dual Core.
Hard Disk : 120 GB.
Monitor : 15" LED
Input Devices : Keyboard, Mouse
Ram : 1GB.

3.4.2 Software Requirements:

Operating system : Windows 7.
Coding Language : Android, Java
Toolkit : Android 2.3 Above
IDE : Eclipse/Android Studio

3.5 DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is a two-dimensional diagram that describes how data is processed and transmitted in a system. The graphical depiction recognizes each source of data and how it interacts with other data sources to reach a mutual output. In order to draft a data flow diagram, one must.

- Identify external inputs and outputs.
- Determine how the inputs and outputs relate to each other.

- Explain with graphics how these connections relate and what they result in.

Role of DFD:

- It is a documentation support which is understood by both programmers and nonprogrammers. As DFD postulates only what processes are accomplished not how they are performed.
- A physical DFD postulates where the data flows and who processes the data.
- It permits analyst to isolate areas of interest in the organization and study them by examining the data that enter the process and viewing how they are altered when they leave.

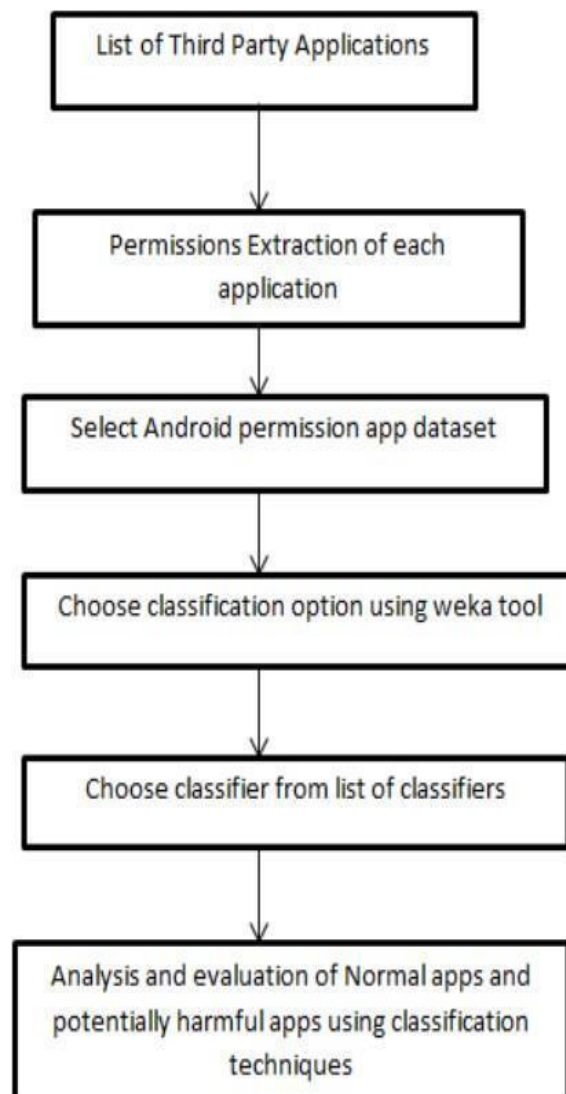


Fig 3.5: Data Flow Diagram

In the above flowchart, the first will be list of third-party application, the second step will be permission extraction from each application in the phones, the third step will be select android permission app dataset, the fourth step will be chosen classification option using weak tool, the fifth step will be chosen classifier from list of classifiers, the sixth step will be analyzing and evaluation of normal apps and potentially harmful apps using classification techniques.

3.6 UML DIAGRAMS

UML is simply another graphical representation of a common semantic model. UML provides a comprehensive notation for the full lifecycle of object-oriented development.

Advantages:

- To represent complete systems (instead of only the software portion) using object-oriented concepts
- To establish an explicit coupling between concepts and executable code
- To consider the scaling factors that are inherent to complex and critical systems.
- To creating a modelling language usable by both humans and machines.
- UML defines several models for representing systems.
 - The class model captures the static structure.
 - The state model expresses the dynamic behavior of objects.
 - The use case model describes the requirements of the user.
 - The interaction model represents the scenarios and messages flows
 - The implementation model shows the work units.
 - The deployment model provides details that pertain to process allocation.

3.6.1 Use Case Diagram:

Use case diagrams overview the usage requirement for system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe “the meant” of the actual requirements. A use case describes a sequence of action that provides something of measurable value to an action and is drawn as a horizontal ellipse.

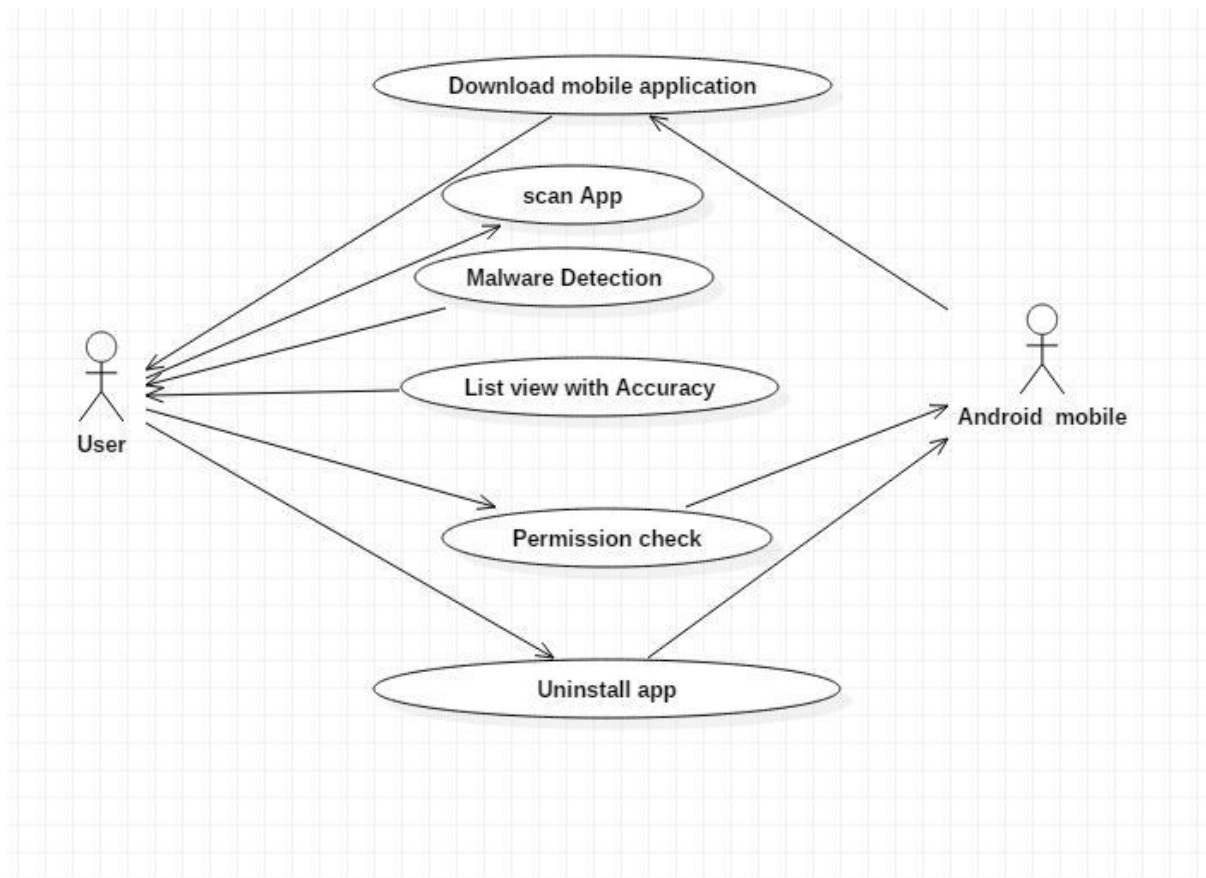


Fig 3.6.1: Use Case Diagram

3.6.2 Class Diagram:

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature. Class diagram represents the object orientation of a system.

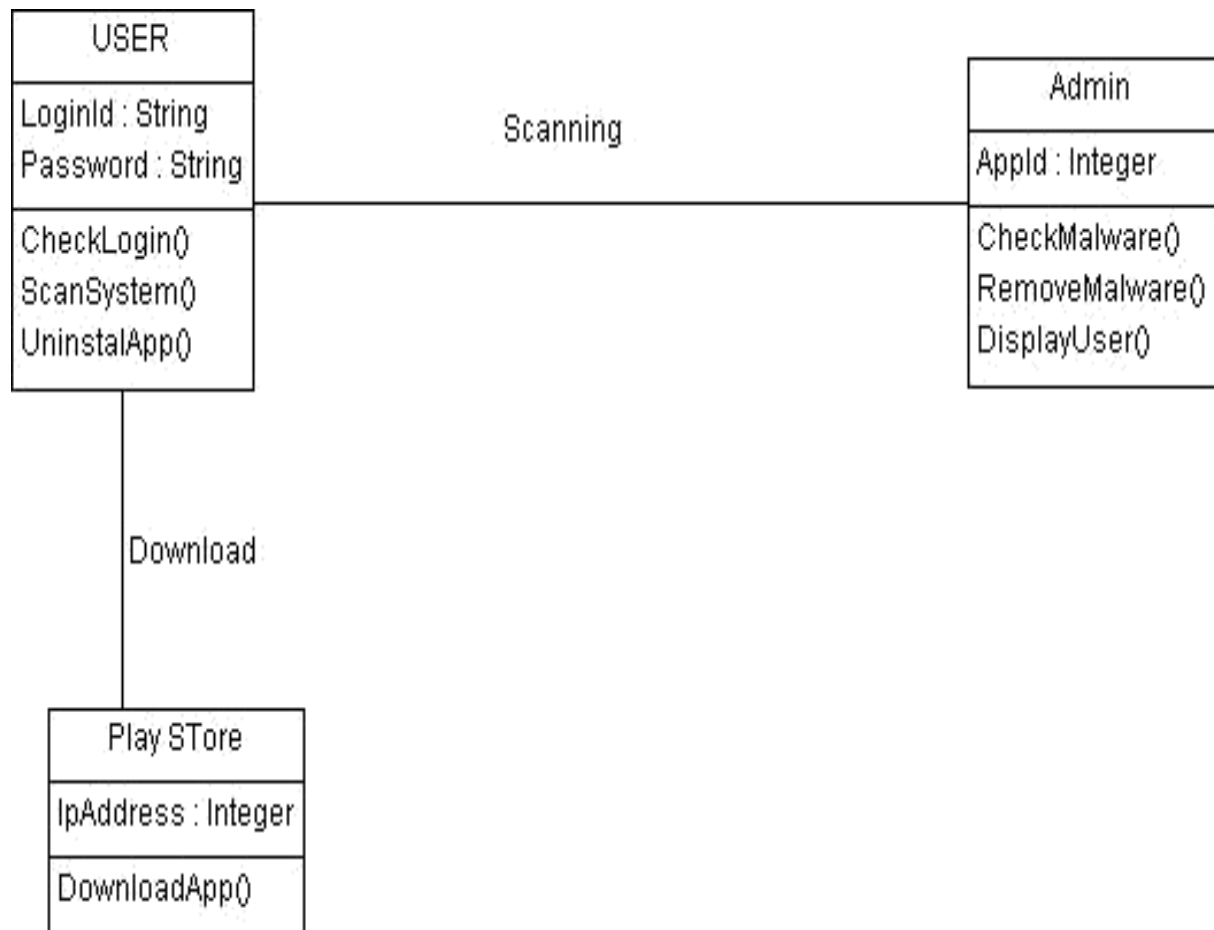


Fig 3.6.2: Class Diagram

3.6.3 Activity Diagram:

Activity diagram are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. Activity diagram consist of Initial node, activity final node and activities in between.

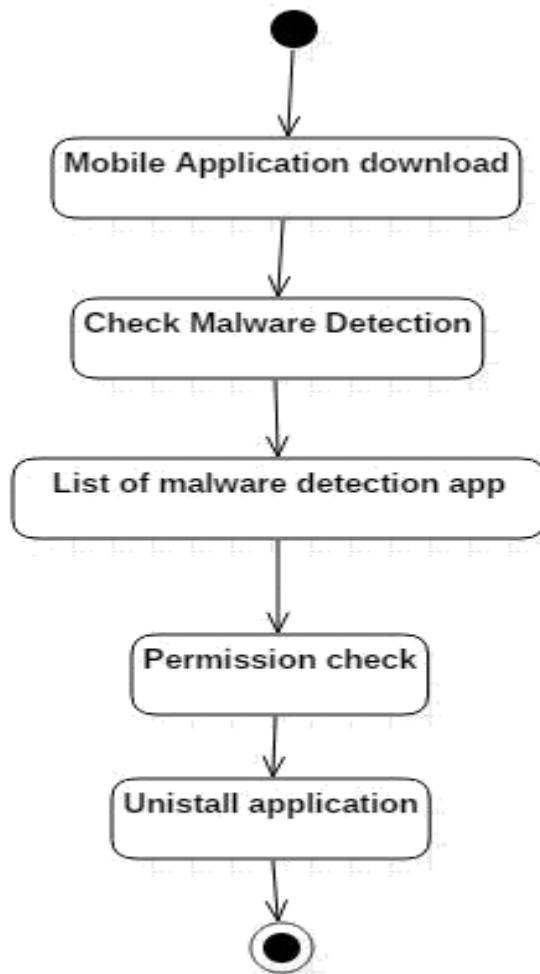


Fig 3.6.3: Activity Diagram

3.6.4 Sequence Diagram

Sequence diagram model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and commonly used for both analysis and design purpose. Sequence diagram are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system.

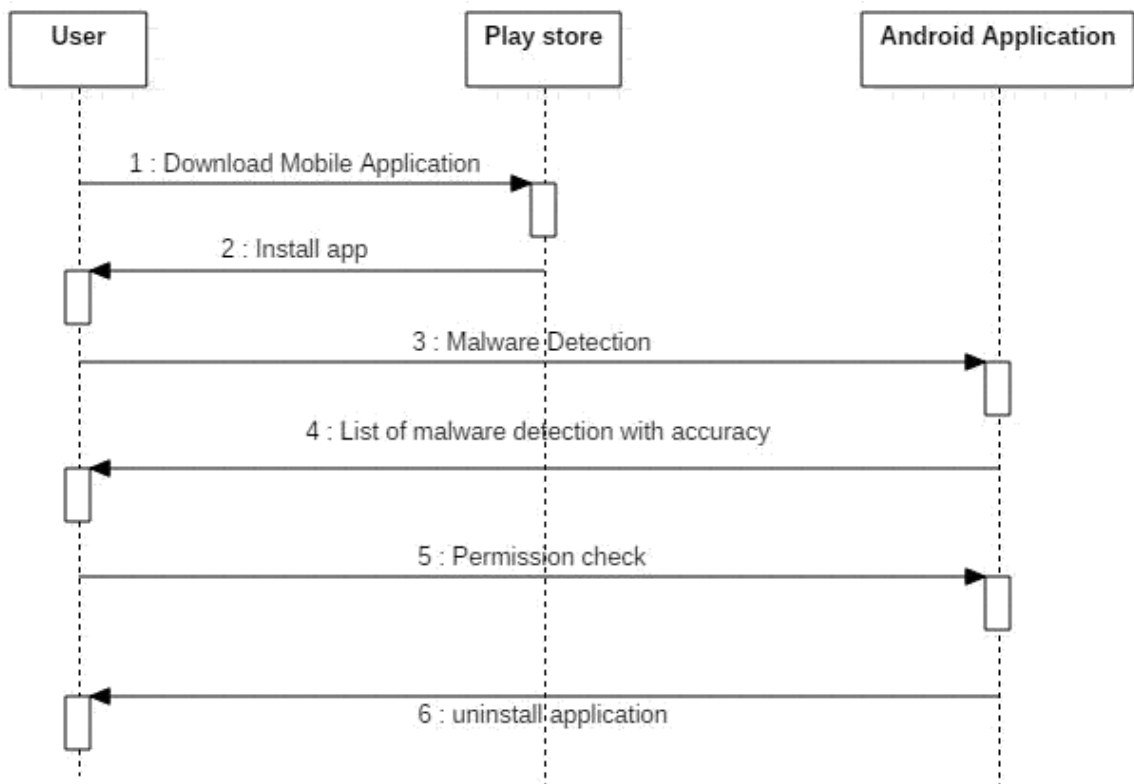


Fig 3.6.4: Sequence Diagram

3.6.5 Collaboration Diagram:

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links. The purpose of collaboration diagram is similar to sequence diagram.

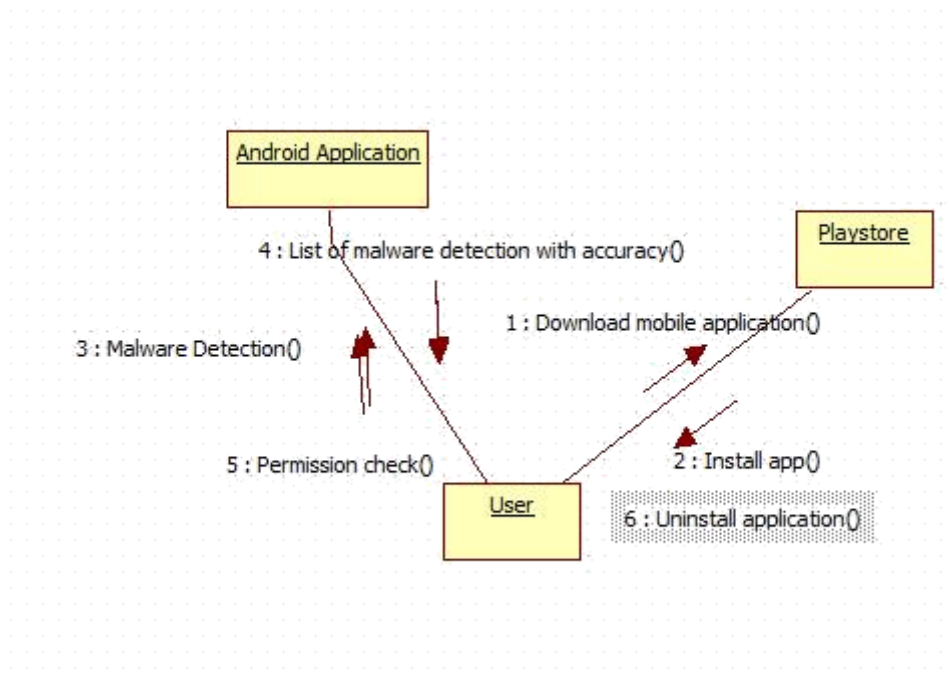


Fig 3.6.5: Collaboration Diagram

3.6.6 Component Diagram:

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.

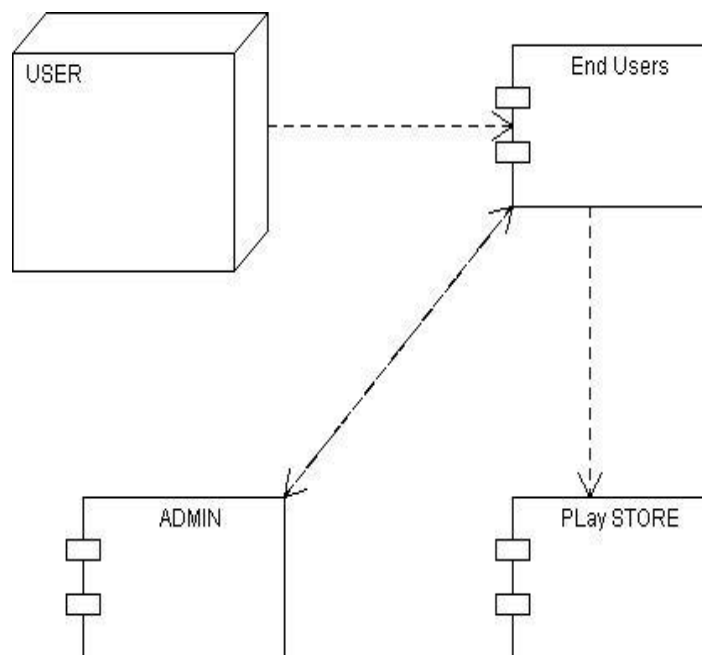


Fig 3.6.6: Component Diagram

3.7 MODULES

- Login and Registration
- Android Application Scan
- Malware Detection
- Android Permission Check
- Privacy Policy Database

3.7.1 LOGIN MODULE

This is the first activity that opens when user open the website. User needs to provide a correct contact number and a password, which user enters while registering, in order to login into the website. If information provided by the user matches with the data in the database table then user successfully login into the website else message of login failed is displayed and user need to reenter correct information. A link to the register activity is also provided for registration of new users.

3.7.1.1 REGISTRATION MODULE

A new user who wants to access the app needs to register first before login. By clicking on register button in login activity, the register activity gets open. A new user registers by entering full name, password and contact number. A user needs to enter password again in confirm password textbox for confirmation. When user enters the information in all textboxes, on the click of register button, the data is transferred to database and user is directed to login activity again. Registered user then needs to login in order to access the app. Validations are applied on all the textboxes for proper functioning of the app. Like information in each textbox is must that is each textbox, either it is of name, contact, password or confirm password, will not be empty while registering. If any such textbox is empty app will give message of information is must in each textbox. Also, data in password and confirm password fields must match for successful registration. Another validation is contact number must be valid one that is of 10 digits. If any such validation is violated, then registration will be unsuccessful and then user needs to register again. Message that app will display when one of the fields is empty. If all such information is correct user will be directed to login activity for

login into the app.

3.7.1.2 Application Download

In this module, many applications are already download in mobile phone through play store. After installation just find the malware present in various applications.

3.7.2 ANDROID APPLICATION SCAN

In this module, many applications are already download in mobile phone through play store. After installation just find the malware present in various applications. Malware of this kind can't be detected by victimization the quality signatures approach or by applying regular static or dynamic analysis strategies.

The detection is performed on the basis of application's network traffic patterns solely. For each application, a model representing its specific approach pattern is learned regionally (i.e., on the device).

Semi- supervised machine-learning methods are used for learning the normal behavioral patterns and for detecting deviations from the application's expected behavior. These methods were implemented and evaluated on Android devices.

3.7.3 MALWARE DETECTION

In this proposed system to improve the security of the mobile apps one methodology is proposed which will evaluate the mobile applications security based on the cloud computing platform and data mining. The task of identifying malware will be categorized into analysis, classification, detection, and ultimate containment of malware. Several classification techniques are utilized to classify malware in keeping with their instances and this has created it potential to acknowledge the sort and activities of a malware and new variant. For such kinds of threats to mobile devices there should be some security mechanism to be implemented. Analysis of malware has got do with distinctive the instances of malware by completely different classification schemes by exploitation the attributes of well-known malware characteristics. Here also a prototype system named Malware detection system is presented to identify the mobile app's virulence or benignancy. Malware detection has to do with the quick detection and validation of any instance of malware in order to prevent further damage to the system. The last a part of the work is containment of the malware, which involves effort at stopping escalation and preventing further damages to the

system. A commercial antivirus uses signature based mostly technique wherever the information should be often updated to possess the newest virus information detection mechanisms. However, the zero-day malicious exploit malware cannot be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file to detect abnormous file segments.

3.7.4 ANDROID PERMISSION CHECK

A user who wishes to install and use any third-party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scenes. If an app needs to use resources or information outside of its own sandbox, the app has to request the appropriate permission. You declare that your app needs a permission by listing the permission in the app manifest and then requesting that the user approve each permission at runtime. If your app needs a dangerous permission, you must check whether you have that permission every time you perform an operation that requires that permission. The system's behavior after you declare a permission depends on how sensitive the permission is. Some permissions are considered "normal" so the system immediately grants them upon installation. Other permissions are considered "dangerous" so the user must explicitly grant your app access. For more information about the different kinds of permissions, see Protection levels. If a user keeps trying to use functionality that requires a permission, but keeps denying the permission request, that probably means the user does not understand why the app needs the

Permissions to provide that functionality permissions to provide that functionality.

3.7.5 PRIVACY POLICY DATABASE

In the implementation, Android Security Evaluation Framework (ASEF) and Static Android Analysis Framework (SAAF) are adopted, the two representative dynamic and static analysis methods to evaluate the Android apps and estimate Malware detection with Accuracy. Privacy is important to us, and we are committed to protecting personal information that is processed in our database

product and service. This database privacy statement explains our purposes and legal basis for what information we may collect about data subjects, sources of information and how the information is processed. We presented a novel approach to the analysis of privacy policies in the context of Android applications. Compared with traditional method, such as permission pattern-based method, Malware combines the dynamic and static analysis methods to comprehensively evaluate an Android app. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. To find your apps and their permissions on mechanical man. open the Settings and so faucet Apps & notifications, App info, and also the app you are inquisitive about. Select the Permissions entry to see all the privileges the app Uninstall. After the Malware is detected we will get notification for uninstalling the application.

CHAPTER – 4

EXPIREMENTAL OR MATERIALS AND METHODS; ALGORITHMS USED

4.1 ALGORITHMS USED

4.1.1 *Decision Tree Algorithm:*

This algorithm is used to predict class or value of target variables by learning decision rules inferred from prior data (training data).

As it implies from the name, decision trees are data structures that have a structure of the tree. The training dataset is used for the creation of the tree, that is subsequently used for making predictions on the test data. In this algorithm, the goal is to achieve the most accurate result with the least number of the decisions that must be made. Decision trees can be used for both classification and regression problems.

Decision tree method achieved its popularity because of its simplicity. It can deal well with large datasets and can handle the noise in the datasets very well. Another advantage is that unlike other algorithms, such as SVM or KNN, decision trees operate in a “white box”, meaning that we can clearly see how the outcome is obtained and which decisions led to it. These facts made it a popular solution for medical diagnosis, spam filtering, security screening and other fields.

4.1.2 *Naive Bayes Algorithm:*

It is used to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple class.

Naive Bayes is the classification machine learning algorithm that relies on the Bayes Theorem. It can be used for both binary and multi-class classification problems. The main point relies on the idea of treating each feature independently. Naive Bayes method evaluates the probability of each feature independently, regardless of any correlations, and makes the prediction based on the Bayes Theorem. That is why this method is called “naive” – in real-world problems features often have some level of correlation between each other. To understand the algorithm of Naive Bayes, the concepts of class probabilities and conditional

probabilities should be introduced first.

a. Class Probability is a probability of a class in the dataset. In other words, if we select a random item from the dataset, this is the probability of it belonging to a certain class.

b. Conditional Probability is the probability of the feature value given the class.

The advantages of using this method include its simplicity and easiness of understanding. In addition to that, it performs well on the data sets with irrelevant features, since the probabilities of them contributing to the output are low. Therefore, they are not considered when making predictions. Moreover, this algorithm usually results in a good performance in terms of consumed resources, since it only needs to calculate the probabilities of the features and classes, there is no need to find any coefficients like in other algorithms. As already mentioned, its main drawback is that each feature is treated independently, although in most cases this cannot be true. (Bishop 2006).

4.2 FEASIBILITY STUDY

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

The feasibility study investigates the problem, and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost, and benefits of such a solution, and the feasibility of such a solution.

The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

4.2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will

have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must justify. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4.2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

4.2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

4.3 COLLECTION FRAMEWORK

Collection framework was not part of original Java release. Collections was added to J2SE 1.2. Prior to Java 2, Java provided adhoc classes such as Dictionary, Vector, Stack and Properties to store and manipulate groups of objects. Collection framework provides many important classes and interfaces to collect and organize group of alike objects.

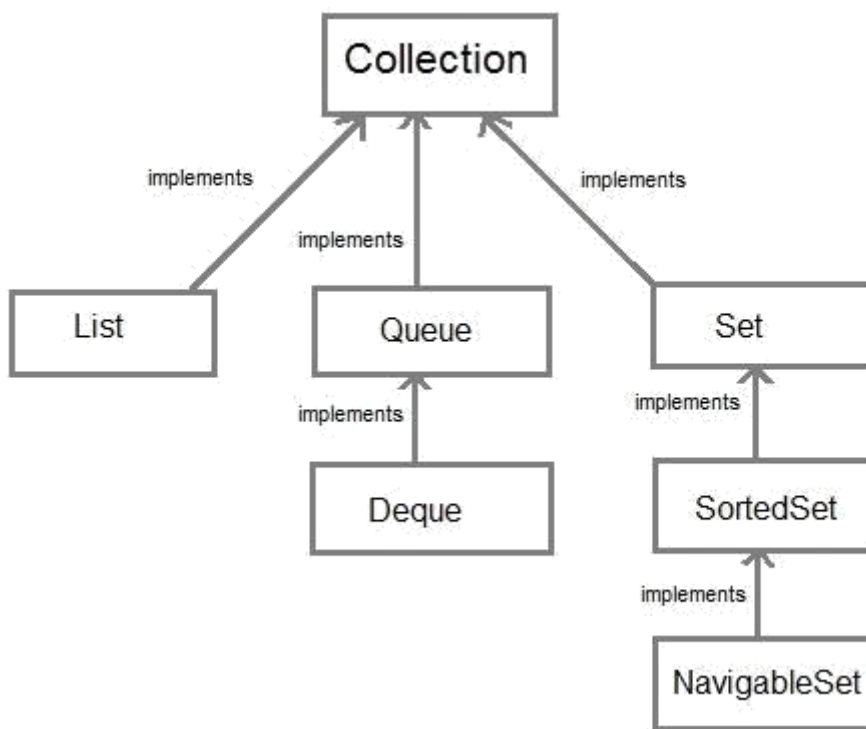


Fig 4.3: Collection Framework

4.4 SYSTEM DESIGN AND TESTING PLAN

4.4.1 Input Design:

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error

occur.

4.4.2 Output Design:

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making. The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

4.4.3 SYSTEM TESTING

4.4.3.1 Test plan

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also, to assess the feature of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words, software testing is a verification and validation process.

4.4.3.2 Verification

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to.

4.4.3.3 Validation

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure

the product is built as per customer requirements.

4.4.4 BASICS OF SOFTWARE TESTING

There are two basics of software testing: black box testing and white box testing.

4.4.4.1 Black box Testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

4.4.4.2 White box Testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing. Black box testing is often used for validation and white box testing is often used for verification.

CHAPTER – 5

RESULTS AND DISCUSSIONS

This project highlights the results of android privacy check and the snapshots for each of the activities are shown along the discussion of each activity describing it's working. Each snapshot describes every single step of analysis of android app privacy. The main activities as well as the working of the project is shown in the snapshots and gives the files that are including the project.

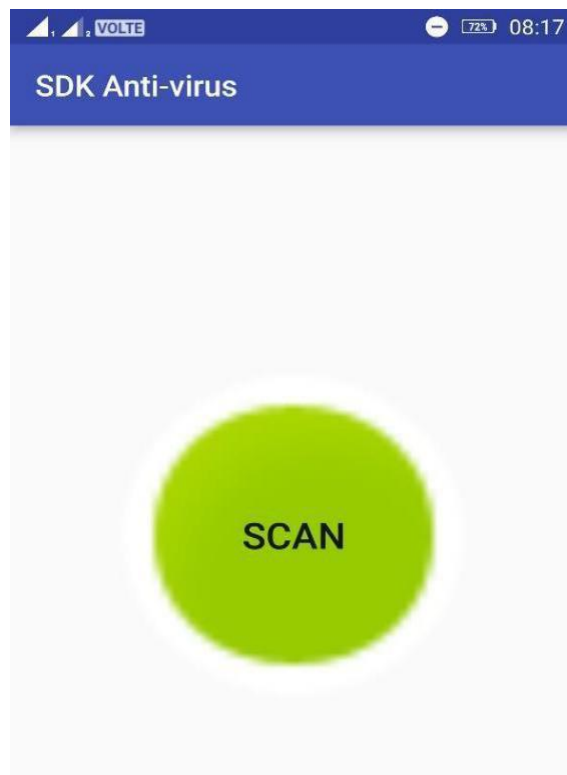


Fig 5.1: Scanning phase

At this scanning phase this application will scan all the apps in our android operating system and calculate the risk percentage.

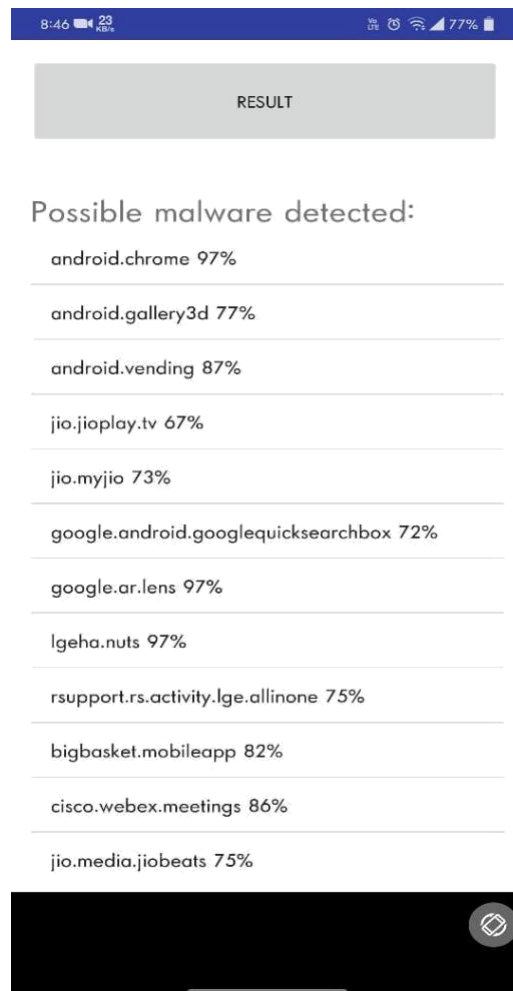


Fig 5.2: Risk percentage

After scanning all the applications in our android operating system, it will calculate and display the amount of risk in percentage (%).

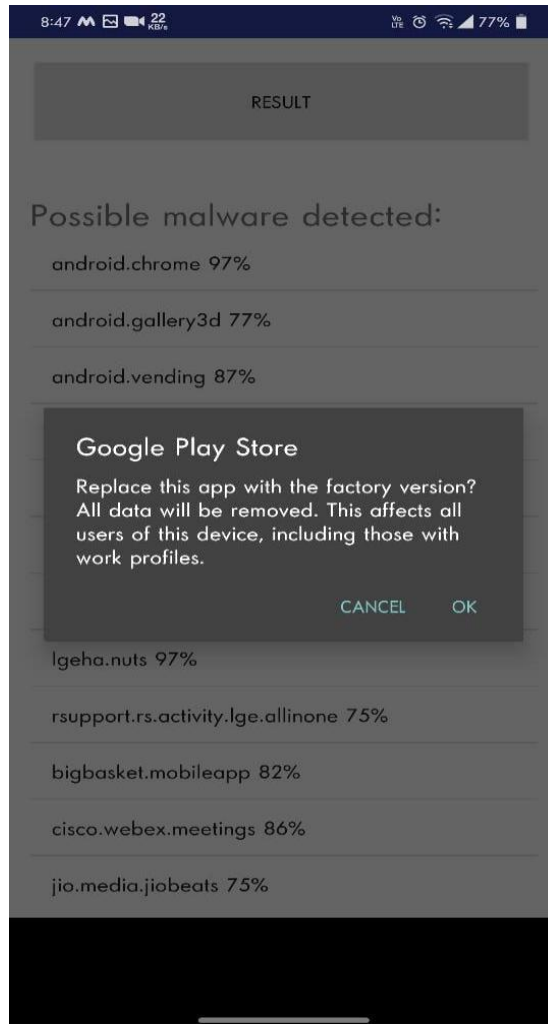


Fig 5.3: Replacement suggestion

When we click on the application risk percentage, it will suggest us to replace the application with factory version. Then the current version will be deleted, and it will replace with factory version by removing all the permissions which are initially given by us.

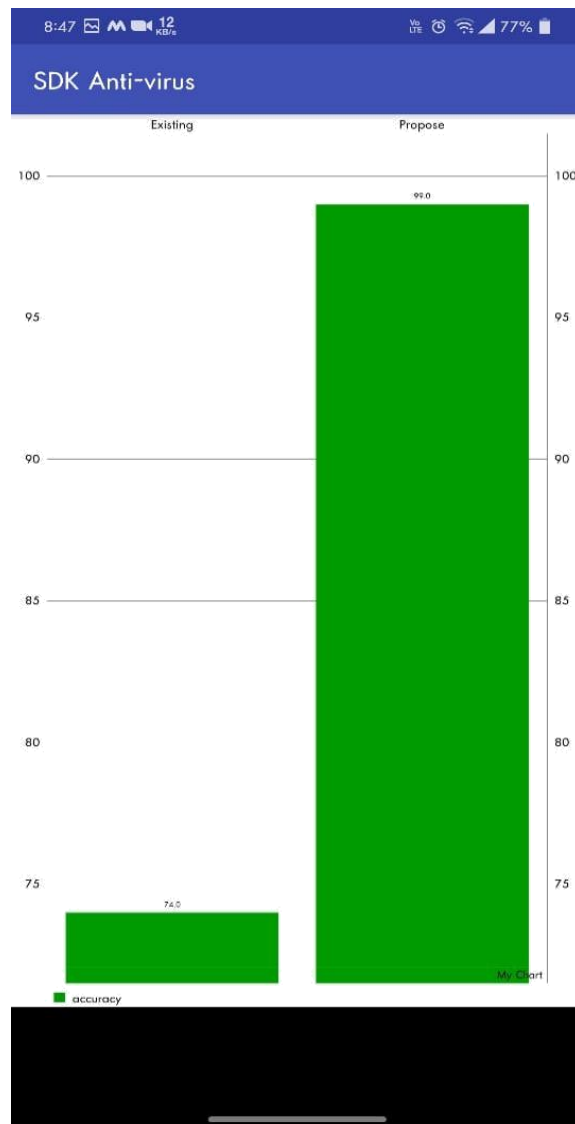


Fig 5.4: Existing vs Proposed accuracy

Finally, it will compare the amount of accuracy in the existing system and the proposed system in a graphical representation.

CHAPTER - 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this system, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws. We proposed, A user who wishes to install and use any third-party app doesn't understand the significance and meaning of the permissions requested by an application, and thereby simply grants all the permissions as a result of which harmful apps also get installed and perform their malicious activity behind the scenes.

6.2 FUTURE WORK

The study performed in this project was a proof-of-concept. Therefore, several future improvements related to the practical implementation of this project can be identified:

Implement feature extraction in the inline mode:

Currently, the feature extraction is performed after the files were run in the sandbox and the reports were generated. This approach will result in delays in the file analysis when implemented. Instead, it is advised to extract the features as they are processed by the sandbox, so that there will be no need to go through the reports again.

Use a wider dataset:

Although the dataset that was used in this study is broad, covering most of the malware types that are relevant to the modern world, it does not cover all possible types. Collecting a malware dataset is a tedious task that requires a lot of time and effort. For more accurate evaluation of the predictors, it is advised to test the models on all the possible types of malware: spyware, adware, rootkits, backdoor, banking malware, etc. In addition to that, it is important to understand that the model will only be able to predict the samples of the families that it has seen earlier. In other words, in a real-world application, the maximum number of possible families should be used before the launch of the project for real-world

environments.

Use pre-selected APIs:

In this work, the big overhead in the data processing was created by the need of selecting the relevant API calls and removing the redundant ones. For further implementation, only the APIs that were identified as relevant in this study can be used. This will decrease the amount of time required for data preprocessing, reduce the performance requirements of the machine on which the analysis is being done and decrease the level of feature selection to be made. However, it should be noted, that for more accurate description, the relevant APIs should be extracted from the biggest possible dataset. Also, it is advised to select the relevant APIs per malware family, as this will result in another level of flexibility and accuracy.

REFERENCES

- [1] Yerima, S. Y., Sezer, S., & McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security*, 8(1), 25-36.
- [2] Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012, June). Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies* (pp. 13-22). ACM.
- [3] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012, February). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy* (pp. 317-326). ACM.
- [4] Chia, P. H., Yamamoto, Y., & Asokan, N. (2012, April). Is this app safe?: a large-scale study on application permissions and risk signals. In *Proceedings of the 21st international conference on World Wide Web* (pp. 311-320). ACM.
- [5] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., & Wang, X. S. (2013, November). Appintents: Analyzing sensitive data transmission in android for privacy leakage detection. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 1043-1054). ACM.
- [6] Zhang, W., Li, X., Xiong, N., & Vasilakos, A. V. (2016). Android platform-based individual privacy information protection system. *Personal and Ubiquitous Computing*, 20(6), 875-884.
- [7] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011, October). Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 627-638). ACM.
- [8] Lin, J., Amini, S., Hong, J. I., Sadeh, N., Lindqvist, J., & Zhang, J. (2012, September). Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (pp. 501-510). ACM.
- [9] Mann, C., & Starostin, A. (2012, March). A framework for static detection of privacy leaks in android applications. In *Proceedings of the 27th annual ACM symposium on applied computing* (pp. 1457-1462). ACM.

[10] Stevens, R., Gibler, C., Crussell, J., Erickson, J., & Chen, H. (2012, May). Investigating user privacy in android ad libraries. In Workshop on Mobile Security Technologies (MoST) (p. 10).

APPENDIX

A. SOURCE CODE

SecondActivity.java: -

```
package com.example.kevinwu.project;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageInfo; import
android.content.pm.PackageManager; import
android.content.pm.ResolveInfo; import
android.os.Environment; import
android.os.Handler;
import android.os.Looper;
import
android.support.v7.app.AppCompatActivity;
import android.os.Bundle; import android.util.Log;
import android.view.View;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class SecondActivity extends Activity {
```

```

private TextView Scan;
private TextView txtProgress;
private ProgressBar progressBar;
private int pStatus = 0;
private int malwareCount = 0;
private Handler handler = new Handler();

//Number of innocent data
private static int NUM_INNO_POINTS = 0;
//Number of malicious data
private static int NUM_MALI_POINTS = 0;
//dimension
private static int DIMENSION = 135;

@Override
protected void onCreate(Bundle savedInstanceState)
{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_second);
    Scan = (TextView) findViewById(R.id.textView1); txtProgress
    = (TextView) findViewById(R.id.txtProgress); progressBar =
    (ProgressBar) findViewById(R.id.progressBar);

    new Thread( new Runnable() {
        @Override
        public void run()
        {
            //retrieve all applications and their
            permissions Bundle b=new Bundle();
            final Intent mainIntent = new Intent(Intent.ACTION_MAIN, null);
            mainIntent.addCategory(Intent.CATEGORY_LAUNCHER);
            final List pkgAppsList =
getPackageManager().queryIntentActivities(mainIntent, 0);

            String [] possiList = new String [pkgAppsList.size()];

```

```

//array if the permissions flag
int [] perFlag = new int[135];

BufferedReader reader = null;

try {
    reader = new BufferedReader(new
InputStreamReader(getAssets().open("malware.txt"),"UTF-8"));
    String line;
    while ((line = reader.readLine()) != null) {
        if(line.length() > 0) {
            StringBuilder text = new StringBuilder();
            text.append(line);
            Log.d("Test:", "" + text);
            text.append('\n');
        }
    }
}
catch (IOException e) {
    e.printStackTrace();
}
finally{
    if(reader != null){
        try {
            reader.close();
        } catch (IOException e) {

        }
    }
}

//read the innocent permission file
ArrayList<Point> innocent = new ArrayList<Point>();
try
{

```

```

        reader = new BufferedReader(
            new
InputStreamReader(getAssets().open("Innocentapp.txt")));
        // do reading, usually loop until end of file reading
        String mLine;
        //build observations
        while ((mLine = reader.readLine()) != null)
        {
            //read one line of permission
            ArrayList<Double> oneLine = new ArrayList<Double>();
            for(int a = 0; a < mLine.length(); a++)
            {

oneLine.add((double)Character.getNumericValue(mLine.charAt(a)));
                }
                Point oneData = new Point(oneLine);
                //add point to observations
                innocent.add(oneData);
                NUM_INNO_POINTS++;
            }
        } catch (IOException e)
        {
            e.printStackTrace();
        }
//warning: hard code remove the first element in the first element of
the observation
        innocent.get(0).data.remove(0);
        innocent.get(0).dimension -=1;
        //build centroid
        ArrayList<Point> centroid = new
        ArrayList<Point>(); for(int a =0; a <DIMENSION;
        a++) {
            ArrayList<Double> centroid_location =
new ArrayList<Double>(DIMENSION);

```

```

        //initialize
        for(int x = 0; x < DIMENSION; x++)
        {
            centroid_location.add((double)0);
        }
        centroid_location.set(a, (double)1);
        Point pt = new Point(centroid_location);
        centroid.add(pt);
    }
    KMeans innocent_kmeans = new KMeans(innocent, centroid);
    System.out.println(innocent_kmeans.calculate(NUM_INNO_POINTS)
    //read the malicious permission file
    ArrayList<Point> malicious = new ArrayList<Point>();
    try
    {
        reader = new BufferedReader(
            new InputStreamReader(getAssets().open("malware.txt")));
        // do reading, usually loop until end of file reading
        String mLine;
        //build observations
        while ((mLine = reader.readLine()) != null)
        {
            //read one line of permission
            ArrayList<Double> oneLine = new ArrayList<Double>();
            for(int a = 0; a < mLine.length(); a++)
            {
oneLine.add((double)Character.getNumericValue(mLine.charAt(a)));
            }
            Point oneData = new Point(oneLine);
            //add point to observations
            malicious.add(oneData);
            NUM_MALI_POINTS++;
        }
    }

```

```

    } catch (IOException e)
    {
        e.printStackTrace();
    }

    //warning: hard code remove the first element in the first element
of the observation
    KMeans malicious_kmeans = new KMeans(malicious, centroid);

    System.out.println(malicious_kmeans.calculate(NUM_MALI_POINTS));

    //Make prediction
    NaiveBayes prediction = new NaiveBayes(innocent_kmeans.clusters,
    malicious_kmeans.clusters, NUM_INNO_POINTS, NUM_MALI_POINTS);
    //runs through each app
    for(int i = 0 ;i < pkgAppsList.size();i++){
        //reset the array
        for(int x = 0; x < 134; x++){
            perFlag[x] = 0;
        }

        Object obj = pkgAppsList.get(i);
        ResolveInfo resolveInfo = (ResolveInfo) obj;
        PackageInfo packageInfo = null;
        try {
            packageInfo =
getPackageManager().getPackageInfo(resolveInfo.activityInfo.packageName,
PackageManager.GET_PERMISSIONS);
        } catch (PackageManager.NameNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        Log.d("", "");
    }

```

```

        Log.d("app: ", pkgAppsList.get(i).toString());
        String[] permission = packageInfo.requestedPermissions;
        if(permission == null){
            Log.d("NO permission", "-----");
        }else {
            for(int j = 0; j < permission.length; j++) {

                if(permission[j].compareTo("android.permission.ACCESS_CHECKIN_PROPERTIES") == 0){
                    perFlag[0] = 1;
                }else
                if(permission[j].compareTo("android.permission.ACCESS_COARSE_LOCATION") == 0){
                    perFlag[1] = 1;
                }else
                if(permission[j].compareTo("android.permission.ACCESS_FINE_LOCATION") == 0){
                    perFlag[2] = 1;
                }else
                if(permission[j].compareTo("android.permission.ACCESS_LOCATION_EXTRA_COMMANDS") == 0){
                    perFlag[3] = 1;
                }else
                if(permission[j].compareTo("android.permission.ACCESS_NETWORK_STATE") == 0){
                    perFlag[4] = 1;
                }else
                if(permission[j].compareTo("android.permission.ACCESS_NOTIFICATION_POLICY") == 0){
                    perFlag[5] = 1;
                }else
                if(permission[j].compareTo("android.permission.ACCESS_WIFI_STATE") == 0){
                    perFlag[6] = 1;
                }else

```

```

if(permission[j].compareTo("android.permission.ACCOUNT_MANAGER") ==
    0){ perFlag[7] = 1;
    }else
if(permission[j].compareTo("android.permission.ADD_VOICEMAIL") == 0){
    perFlag[8] = 1;
    }else
if(permission[j].compareTo("android.permission.BATTERY_STATS") == 0){
    perFlag[9] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_ACCESSIBILITY_SERVICE
") == 0){
    perFlag[10] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_APPWIDGET") ==
    0){ perFlag[11] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_CARRIER_MESSAGING_S
ERVICE") == 0){
    perFlag[12] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_CARRIER_SERVICES")
== 0){
    perFlag[13] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_CHOOSER_TARGET_SER
VICE") == 0){
    perFlag[14] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_DEVICE_ADMIN") ==
    0){ perFlag[15] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_DREAM_SERVICE") == 0){
    perFlag[16] = 1;
    }else

```

```

if(permission[jj].compareTo("android.permission.BIND_INCALL_SERVICE") ==
    0){ perFlag[17] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_INPUT_METHOD") == 0){
    perFlag[18] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_MIDI_DEVICE_SERVICE") ==
0){
    perFlag[19] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_NFC_SERVICE") ==
    0){ perFlag[20] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_NOTIFICATION_LISTENE
R_SERVICE") == 0){
    perFlag[21] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_PRINT_SERVICE") ==
    0){ perFlag[22] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_REMOTEVIEWS") == 0){
    perFlag[23] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_TELECOM_CONNECTIO
N_SERVICE") == 0){
    perFlag[24] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_TEXT_SERVICE") ==
    0){ perFlag[25] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_TV_INPUT") == 0){
    perFlag[26] = 1;
    }else
if(permission[jj].compareTo("android.permission.BIND_VOICE_INTERACTION")

```

```

== 0){
    perFlag[27] = 1;
}
else
if(permission[j].compareTo("android.permission.BIND_VPN_SERVICE") ==
    0){ perFlag[28] = 1;
    }
else
if(permission[j].compareTo("android.permission.BIND_WALLPAPER") == 0){
    perFlag[29] = 1;
    }
else
if(permission[j].compareTo("android.permission.BLUETOOTH") ==
    0){ perFlag[30] = 1;
    }
else
if(permission[j].compareTo("android.permission.BLUETOOTH_ADMIN") == 0){
    perFlag[31] = 1;
    }
else
if(permission[j].compareTo("android.permission.BLUETOOTH_PRIVILEGED") ==
    0){
    perFlag[32] = 1;
    }
else
if(permission[j].compareTo("android.permission.BODY_SENSORS") ==
    0){ perFlag[33] = 1;
    }
else
if(permission[j].compareTo("android.permission.BROADCAST_PACKAGE_REMO
VED") == 0){
    perFlag[34] = 1;
    }
else
if(permission[j].compareTo("android.permission.BROADCAST_SMS") ==
    0){ perFlag[35] = 1;
    }
else
if(permission[j].compareTo("android.permission.BROADCAST_STICKY") == 0){
    perFlag[36] = 1;
    }
else
if(permission[j].compareTo("android.permission.BROADCAST_WAP_PUSH") ==
    0){

```

```

        perFlag[37] = 1;
    }else
if(permission[j].compareTo("android.permission.CALL_PHONE") ==
    0){ perFlag[38] = 1;
    }else
if(permission[j].compareTo("android.permission.CALL_PRIVILEGED") == 0){
    perFlag[39] = 1;
    }else
if(permission[j].compareTo("android.permission.CAMERA") ==
    0){ perFlag[40] = 1;
    }else
if(permission[j].compareTo("android.permission.CAPTURE_AUDIO_OUTPUT") ==
0){
    perFlag[41] = 1;
    }else
if(permission[j].compareTo("android.permission.CAPTURE_SECURE_VIDEO_OUT
PUT") == 0){
    perFlag[42] = 1;
    }else
if(permission[j].compareTo("android.permission.CAPTURE_VIDEO_OUTPUT") ==
0){
    perFlag[43] = 1;
    }else
if(permission[j].compareTo("android.permission.CHANGE_COMPONENT_ENABL
ED_STATE") == 0){
    perFlag[44] = 1;
    }else
if(permission[j].compareTo("android.permission.CHANGE_CONFIGURATION") ==
0){
    perFlag[45] = 1;
    }else
if(permission[j].compareTo("android.permission.CHANGE_NETWORK_STATE")
== 0){
    perFlag[46] = 1;

```

```

        }else
if(permission[j].compareTo("android.permission.CHANGE_WIFI_MULTICAST_ST
ATE") == 0){
        perFlag[47] = 1;
        }else
if(permission[j].compareTo("android.permission.CHANGE_WIFI_STATE") ==
0){ perFlag[48] = 1;
        }else
if(permission[j].compareTo("android.permission.CLEAR_APP_CACHE") == 0){
        perFlag[49] = 1;
        }else
if(permission[j].compareTo("android.permission.CONTROL_LOCATION_UPDATE
S") == 0){
        perFlag[50] = 1;
        }else
if(permission[j].compareTo("android.permission.DELETE_CACHE_FILES") ==
0){ perFlag[51] = 1;
        }else
if(permission[j].compareTo("android.permission.DELETE_PACKAGES") == 0){
        perFlag[52] = 1;
        }else
if(permission[j].compareTo("android.permission.DIAGNOSTIC") ==
0){ perFlag[53] = 1;
        }else
if(permission[j].compareTo("android.permission.DISABLE_KEYGUARD") == 0){
        perFlag[54] = 1;
        }else if(permission[j].compareTo("android.permission.DUMP")
== 0){
        perFlag[55] = 1;
        }else
if(permission[j].compareTo("android.permission.EXPAND_STATUS_BAR") ==
0){ perFlag[56] = 1;
        }else
if(permission[j].compareTo("android.permission.FACTORY_TEST") == 0){

```

```

        perFlag[57] = 1;
    }else
    if(permission[j].compareTo("android.permission.FLASHLIGHT") == 0){
        perFlag[58] = 1;
    }else
    if(permission[j].compareTo("android.permission.GET_ACCOUNTS") == 0){
        perFlag[59] = 1;
    }else
    if(permission[j].compareTo("android.permission.GET_ACCOUNTS_PRIVILEGED" )
    == 0){
        perFlag[60] = 1;
    }else
    if(permission[j].compareTo("android.permission.GET_PACKAGE_SIZE") ==
    0){ perFlag[61] = 1;
    }else
    if(permission[j].compareTo("android.permission.GET_TASKS") == 0){
        perFlag[62] = 1;
    }else
    if(permission[j].compareTo("android.permission.GLOBAL_SEARCH") == 0){
        perFlag[63] = 1;
    }else
    if(permission[j].compareTo("android.permission.INSTALL_LOCATION_PROVIDE
    R") == 0){
        perFlag[64] = 1;
    }else
    if(permission[j].compareTo("android.permission.INSTALL_PACKAGES") ==
    0){ perFlag[65] = 1;
    }else
    if(permission[j].compareTo("android.permission.INSTALL_SHORTCUT") == 0){
        perFlag[66] = 1;
    }else
    if(permission[j].compareTo("android.permission.INTERNET") == 0){
        perFlag[67] = 1;
    }else

```

```

if(permission[j].compareTo("android.permission.KILL_BACKGROUND_PROCESS
ES") == 0){
    perFlag[68] = 1;
}else
if(permission[j].compareTo("android.permission.LOCATION_HARDWARE") == 0){
    perFlag[69] = 1;
}else
if(permission[j].compareTo("android.permission.MANAGE_DOCUMENTS") == 0){
    perFlag[70] = 1;
}else
if(permission[j].compareTo("android.permission.MASTER_CLEAR") == 0){
    perFlag[71] = 1;
}else
if(permission[j].compareTo("android.permission.MEDIA_CONTENT_CONTROL")
== 0){
    perFlag[72] = 1;
}else
if(permission[j].compareTo("android.permission.MODIFY_AUDIO_SETTINGS") ==
0){
    perFlag[73] = 1;
}else
if(permission[j].compareTo("android.permission.MODIFY_PHONE_STATE") ==
0){ perFlag[74] = 1;
}else
if(permission[j].compareTo("android.permission.MOUNT_FORMAT_FILESYSTEM
S") == 0){
    perFlag[75] = 1;
}else
if(permission[j].compareTo("android.permission.MOUNT_UNMOUNT_FILESYSTE
MS") == 0){
    perFlag[76] = 1;
}else if(permission[j].compareTo("android.permission.NFC")
== 0){
    perFlag[77] = 1;

```

```

        }else
if(permission[j].compareTo("android.permission.PACKAGE_USAGE_STATS") ==
0){
        perFlag[78] = 1;
        }else
if(permission[j].compareTo("android.permission.PERSISTENT_ACTIVITY") == 0){
        perFlag[79] = 1;
        }else
if(permission[j].compareTo("android.permission.PROCESS_OUTGOING_CALLS")
== 0){
        perFlag[80] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_CALENDAR") ==
0){ perFlag[81] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_CALL_LOG") == 0){
        perFlag[82] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_CONTACTS") == 0){
        perFlag[83] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_EXTERNAL_STORAGE")
== 0){
        perFlag[84] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_FRAME_BUFFER") ==
0){ perFlag[85] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_INPUT_STATE") == 0){
        perFlag[86] = 1;
        }else
if(permission[j].compareTo("android.permission.READ_LOGS") ==
0){ perFlag[87] = 1;
        }else

```

```

if(permission[j].compareTo("android.permission.READ_PHONE_STATE") == 0){
    perFlag[88] = 1;
}
else
if(permission[j].compareTo("android.permission.READ_SMS") == 0){
    perFlag[89] = 1;
}
else
if(permission[j].compareTo("android.permission.READ_SYNC_SETTINGS") == 0){
    perFlag[90] = 1;
}
else
if(permission[j].compareTo("android.permission.READ_SYNC_STATS") == 0){
    perFlag[91] = 1;
}
else
if(permission[j].compareTo("android.permission.READ_VOICEMAIL") ==
0){ perFlag[92] = 1;
}
else
if(permission[j].compareTo("android.permission.REBOOT") == 0){
    perFlag[93] = 1;
}
else
if(permission[j].compareTo("android.permission.RECEIVE_BOOT_COMPLETED")
== 0){
    perFlag[94] = 1;
}
else
if(permission[j].compareTo("android.permission.RECEIVE_MMS") ==
0){ perFlag[95] = 1;
}
else
if(permission[j].compareTo("android.permission.RECEIVE_SMS") == 0){
    perFlag[96] = 1;
}
else
if(permission[j].compareTo("android.permission.RECEIVE_WAP_PUSH") ==
0){ perFlag[97] = 1;
}
else
if(permission[j].compareTo("android.permission.RECORD_AUDIO") == 0){
    perFlag[98] = 1;
}
else

```

```

if(permission[j].compareTo("android.permission.REORDER_TASKS") == 0){
    perFlag[99] = 1;
}
else
if(permission[j].compareTo("android.permission.REQUEST_IGNORE_BATTERY_
OPTIMIZATIONS") == 0){
    perFlag[100] = 1;
}
else
if(permission[j].compareTo("android.permission.REQUEST_INSTALL_PACKAGE
S") == 0){
    perFlag[101] = 1;
}
else
if(permission[j].compareTo("android.permission.RESTART_PACKAGES") == 0){
    perFlag[102] = 1;
}
else
if(permission[j].compareTo("android.permission.SEND_RESPOND_VIA_MESSAG
E") == 0){
    perFlag[103] = 1;
}
else
if(permission[j].compareTo("android.permission.SEND_SMS") == 0){
    perFlag[104] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_ALARM") == 0){
    perFlag[105] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_ALWAYS_FINISH") == 0){
    perFlag[106] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_ANIMATION_SCALE") ==
0){
    perFlag[107] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_DEBUG_APP") ==
0){ perFlag[108] = 1;
}
else

```

```

if(permission[j].compareTo("android.permission.SET_PREFERRED_APPLICATIONS") == 0){
    perFlag[109] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_PROCESS_LIMIT") == 0){
    perFlag[110] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_TIME") == 0){
    perFlag[111] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_TIME_ZONE") == 0){
    perFlag[112] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_WALLPAPER") == 0){
    perFlag[113] = 1;
}
else
if(permission[j].compareTo("android.permission.SET_WALLPAPER_HINTS") == 0){
    perFlag[114] = 1;
}
else
if(permission[j].compareTo("android.permission.SIGNAL_PERSISTENT_PROCESSES") == 0){
    perFlag[115] = 1;
}
else
if(permission[j].compareTo("android.permission.STATUS_BAR") == 0){
    perFlag[116] = 1;
}
else
if(permission[j].compareTo("android.permission.SYSTEM_ALERT_WINDOW") == 0){
    perFlag[117] = 1;
}
else
if(permission[j].compareTo("android.permission.TRANSMIT_IR") == 0){
    perFlag[118] = 1;
}
else

```

```

if(permission[j].compareTo("android.permission.UNINSTALL_SHORTCUT") == 0){
    perFlag[119] = 1;
}
else
if(permission[j].compareTo("android.permission.UPDATE_DEVICE_STATS") ==
0){
    perFlag[120] = 1;
}
else
if(permission[j].compareTo("android.permission.USE_FINGERPRINT") ==
0){ perFlag[121] = 1;
}
else
if(permission[j].compareTo("android.permission.USE_SIP") == 0){
    perFlag[122] = 1;
}
else
if(permission[j].compareTo("android.permission.VIBRATE") ==
0){ perFlag[123] = 1;
}
else
if(permission[j].compareTo("android.permission.WAKE_LOCK") == 0){
    perFlag[124] = 1;
}
else
if(permission[j].compareTo("android.permission.WRITE_APN_SETTINGS") ==
0){ perFlag[125] = 1;
}
else
if(permission[j].compareTo("android.permission.WRITE_CALENDAR") == 0){
    perFlag[126] = 1;
}
else
if(permission[j].compareTo("android.permission.WRITE_CALL_LOG") ==
0){ perFlag[127] = 1;
}
else
if(permission[j].compareTo("android.permission.WRITE_CONTACTS") == 0){
    perFlag[128] = 1;
}
else
if(permission[j].compareTo("android.permission.WRITE_EXTERNAL_STORAGE") ==
0){
    perFlag[129] = 1;
}

```

```

        }else
if(permission[j].compareTo("android.permission.WRITE_GSERVICES") == 0){
    perFlag[130] = 1;
    }else
if(permission[j].compareTo("android.permission.WRITE_SECURE_SETTINGS")
== 0){
    perFlag[131] = 1;
    }else
if(permission[j].compareTo("android.permission.WRITE_SETTINGS") ==
    0){ perFlag[132] = 1;
    }else
if(permission[j].compareTo("android.permission.WRITE_SYNC_SETTINGS") ==
0){
    perFlag[133] = 1;
    }else
if(permission[j].compareTo("android.permission.WRITE_VOICEMAIL") ==
    0){ perFlag[134] = 1;
    }

    //Log.d("Permission: ",permission[j]);
    }
}
//convert perFlag array to arraylist
ArrayList<Double> dbPerFlag = new ArrayList<Double>();
for(int a = 0; a < DIMENSION; a++) {

    dbPerFlag.add((double)perFlag[i]);
}
//this method returns a boolean
if(prediction.isMalicious(dbPerFlag))
{
    malwareCount++;
    String string = pkgAppsList.get(i).toString();
    if(string.contains("com."))

```

```

    {
        String[] parts = string.split("(?<= com.)");
        int ln = parts.length;
        Log.d("Permission: ", String.valueOf(ln));
        if(ln!=0 && ln!=1) {
            String s = parts[1];
            String[] name = s.split("(?=/)");
            possiList[malwareCount - 1] = name[0];
        }
    }
}
//-----
-----

//remove after
try {
    Thread.sleep(50);
} catch (InterruptedException e) {
    e.printStackTrace();
}
String flag = "";

for(int k = 0; k < perFlag.length; k++) {
    flag += String.valueOf(perFlag[k]);
}
Log.d("flag: ", flag);

//method to change the status bar percent
if((pStatus + Math.ceil(100.0/pkgAppsList.size()))<=100 ) {
    pStatus += Math.ceil(100.0 / pkgAppsList.size());
}else{pStatus = 100;}
handler.post(new Runnable() {
    @Override
    public void run() {
        progressBar.setProgress(pStatus);
    }
}

```

```

        txtProgress.setText(pStatus + " %");
    }
});

// goto the next activity
if(i == (pkgAppsList.size()-1)){
    //String com = "Complete";
    //Scan.setText(com);
    try {
        Thread.sleep(1000);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
    Intent third = new
Intent("com.example.kevinwu.project.activity_third");
    third.putExtra("mcount",
malwareCount); b.putStringArray("list",
possiList); third.putExtras(b);
    finish();
    startActivity(third);

}
}
}
}).start();
}
}

```