

ANDROID MALWARE DETECTION USING MACHINE LEARNING

A Project Report Phase - II

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Electronics and Communication
Engineering

By

SHAIK MASTHAN (39130418)

SHAIK ISMAIL (39130416)



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING
SCHOOL OF ELECTRICAL AND ELECTRONICS**

SATHYABAMA

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

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119**

APRIL 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project is the bonafide work of **SHAIK MASTHAN** (39130418) and **SHAIK ISMAIL** (39130416) who carried out the project entitled "**ANDROID MALWARE DETECTION USING MACHINE LEARNING**" under our supervision of **Dr. R. Indhu**, from November 2023 to April 2023.

Internal Guide

Dr. R. Indhu, M.E, Ph.D

Assistant Professor

Head of the Department

Dr.T.RAVI M.E., Ph.D.,

Professor

Submitted for Viva Voce Examination held on :

Internal Examiner

External Examiner

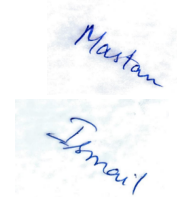
DECLARATION

We, **SHAIK MASTHAN VALLI** (39130418) and **SHAIK ISMAIL** (39130416) hereby declare that the Project Report entitled “**ANDROID MALWARE DETECTION USING MACHINE LEARNING**” done by us under the guidance of **Dr. R. Indhu, M.E, Ph.D**, Professor, Dept. of Electronics and Communication Engineering, SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY, CHENNAI is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Electronics and Communication Engineering.

DATE: 26/04/2023

PLACE: Chennai

SIGNATURE OF CANDIDATES

Two handwritten signatures in blue ink. The top signature reads 'Masthan' and the bottom signature reads 'Ismail'.

ACKNOWLEDGEMENT

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

We convey our thanks to **Dr. N.M. NANDHITHA, M.E., Ph.D.**, Dean, School of Electricals and Electronics, and **Dr. T. RAVI, M.E., Ph.D.**, Communication Engineering for providing necessary support and details at the right time during the progressive reviews.

We would like to express our sincere and deep sense of gratitude to our Project Guide **Dr.R.Indhu, M.E., Ph.D.**, for his valuable guidance, suggestions, and constant encouragement that paved the way for the successful completion of our project work.

We wish to express our thanks to all Teaching and non-teaching staff members of the Department of Electronics and Communication Engineering who were helpful in manyways 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 to select a specific permission, and a list of relevant sentences are 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 extracted by the privacy policy and presented to them, along with an 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 expanding ability in mind, and further developments in the approach can easily be integrated in order to increase thereliability 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 scene.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	iv
	LIST OF FIGURES	vii
1	INTRODUCTION	
	1.1 INTRODUCTION	1
	1.2 DOMAIN INTRODUCTION	2
	1.3 OVERVIEW OF THE PROJECT	3
2	LITERATURE SURVEY	
	2.1 LITERATURE SURVEY	4
3	REQUIREMENT ANALYSIS	
	3.1 EXISTING SYSTEM	10
	3.2 DISADVANTAGES OF EXISTING SYSTEM	10
	3.3 PROPOSED SYSTEM	10
	3.4 ADVANTAGES OF PROPOSED SYSTEM	10
4	DESCRIPTION OF PROPOSED SYSTEM	
	4.1 SYSTEM ARCHITECTURE	11
	4.2 SYSTEM REQUIREMENTS	12
5	IMPLEMENTATION DETAILS	
	5.1 PENTIUM DUAL CORE	13
	5.2 JAVA	14
	5.3 WINDOWS	14
	5.4 ANDRIOD STUDIO	15
6	RESULTS AND DISCUSSION	
	6.1 MODULES	17
	6.2 ANDRIOD PERMISSION CHECK	19
	6.3 APPLICATION OF JAVA	20
	6.4 FEATURES OF JAVA	20
	6.5 COLLECTION OF FRAMEWORK	22
	6.6 SYSTEM DESIGN AND TESTING PLAN	23

6.7	USECASE DIAGRAM	24
6.8	ACTIVITY DIAGRAM	24
6.9	SEQUENCE DIAGRAM	25
6.10	COLLABORATION DIAGRAM	25
6.11	MODULES	26
6.12	SYSTEM STUDY	29
6.13	FUNCTIONAL REQUIREMENTS	30
7	CONCLUSION	
7.1	<i>CONCLUSION</i>	32
7.2	<i>REAL TIME IMPLEMENTATION</i>	33
7.3	<i>FUTURE WORK</i>	36
	REFERENCES	37
	APPENDIX	
	A .SOURCE CODE	39
	B.SCREEN SHOTS	67
	C.RESEARCH PAPER	70

LIST OF FIGURES

FIGURE NO	FIGURE TITLE	PAGE NO
4.1	SYSTEM ARCHITECTURE	11
5.1	PENTIUM DUAL CORE	13
5.2	JAVA	14
5.3	WINDOWS	14
6.1	PLATFORM INDEPENDENT	21
6.2	COLLECTION OF FRAMEWORK	22
6.3	USE CASE DIAGRAM	24
6.4	ACTIVITY DIAGRAM	24
6.5	SEQUENCE DIAGRAM	25
6.6	COLLABORATION DIAGRAM	25
6.7	SCANNING OF ANTI VIRUS	27
6.8	ANDROID PERMISSION CHECK	28
6.9	MALWARE DETECTION PROCESS	29

CHAPTER 1

INTRODUCTION

1.1 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.

1.2 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

2.1 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 are 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 majority 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 DroidMOSS that applies a fuzzy hashing technique to effectively localize and detect the changes from app-repackaging behavior. The experiments with DroidMOSS 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.

Analyzing 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 overprivilege 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 overprivilege. We apply

Stowaway to a set of 940 applications and find that about one-third are overprivileged. We investigate the causes of overprivilege 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 analyze the privacy-related behaviors 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 analyzed 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 .

CHAPTER 3

EXISTING AND PROPOSED SYSTEM

REQUIREMENT ANALYSIS

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.

3.2 DISADVANTAGES OF EXISTING SYSTEM

- Security is low
- Slow to detect

3.3 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.

3.4 ADVANTAGES OF PROPOSED SYSTEM

- Security is high.
- It can find all kinds of data-set here and more proficiency there to use.
- Fast and Flexible

CHAPTER 4

PROPOSED SYSTEM

4.1 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 analysing there will be many permissions will be generated and then in java after data analysing 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.

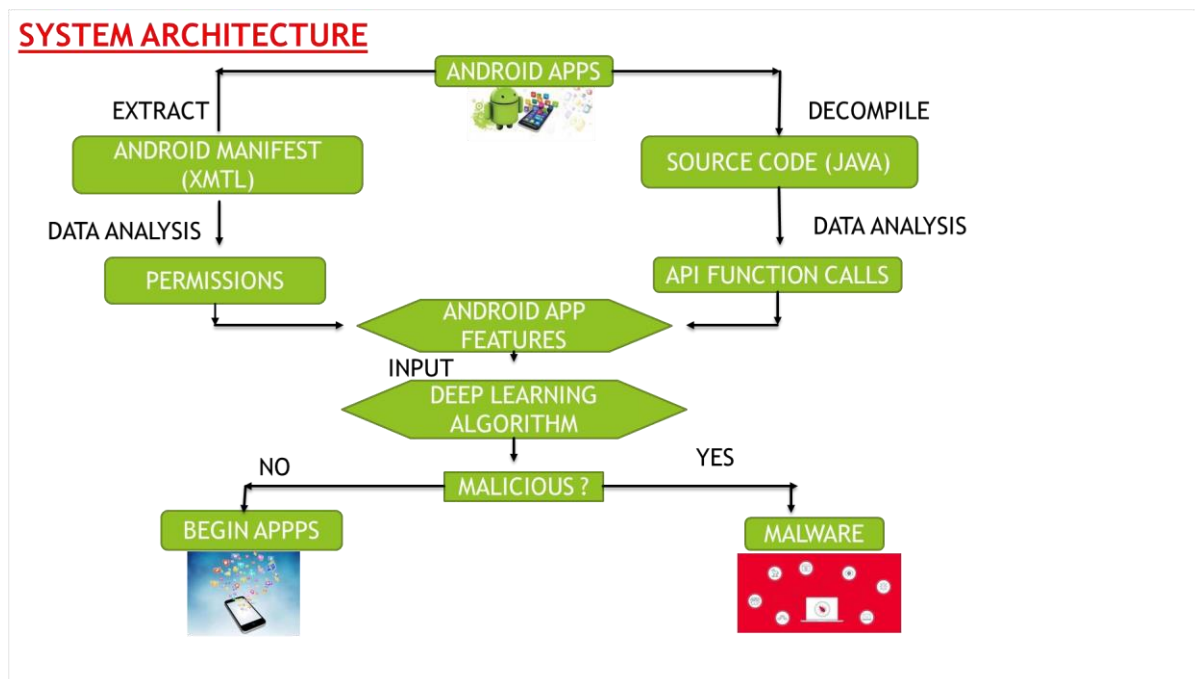


Figure 4.1: System Architecture

4.2 SYSTEM REQUIREMENTS

4.2.1 HARDWARE REQUIREMENTS:

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

4.2.2 SOFTWARE REQUIREMENTS:

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

CHAPTER -5

IMPLEMENTATION DETAILS

5.1 PENTIUM DUAL CORE

The Pentium Dual-Core brand was used for mainstream x86-architecture microprocessors from Intel from 2006 to 2009 when it was renamed to Pentium. The processors are based on either the 32-bit Yonah or (with quite different microarchitectures) 64-bit Merom-2M, Allendale, and Wolfdale-3M core, targeted at mobile or desktop computers.



Figure 5.1: Pentium Dual Core

5.2 JAVA

The Java language has undergone several changes since JDK 1.0 as well as numerous additions of classes and packages to the standard library. Since J2SE 1.4, the evolution of the Java language has been governed by the Java Community Process (JCP), which uses Java Specification Requests (JSRs) to



Figure 5.2:Java

5.3 WINDOWS

Microsoft Windows is a group of several graphical operating system families, all of which are developed, marketed, and sold by Microsoft. Each family caters to a certain sector of the computing industry.

Active Windows families include Windows NT and Windows Embedded; these may encompass subfamilies, e.g. Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x, Windows Mobile and Windows Phone.



Figure 5.3:Windows

5.4 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0. The current stable version is 3.3, which was released in January 2019.

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touch screen mobile devices such as smart phones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear).

The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touch screen input, it has also been used in game consoles, digital cameras, regular PCs, and other electronics.

As of 2015, Android has the largest installed base of all operating systems. As of July 2013, the Google Play store has had over one million Android applications ("apps") published, and over 50 billion applications downloaded. An April–May 2013 survey of mobile application developers found that 71% of them create applications for Android; another 2015 survey found that 40% of full-time professional developers see Android as the "priority" target platform, which is more than iOS (37%) or other platforms.

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.

Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially, released running other operating systems. The operating system's success has made it a target for patent litigation as part of the so-called "smartphone wars" between technology companies.

CHAPTER-6

RESULTS AND DISCUSSION

6.1 MODULES

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

6.1.1 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.

6.1.2 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 field is empty. If all such information is correct user will be directed to login activity for login into the app.

6.1.3 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.

6.1.4 Android Application Scan

In this module, many application's are already download in mobile phone through play store. After installation just find the malware present in various application's. 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 behavioural patterns and for detecting deviations from the application's expected behaviour. These methods were implemented and evaluated on Android devices.

6.1.5 Malware Detection

In this proposed system in order 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 categorised into analysis, classification, detection and ultimate containment of malware. Several classification techniques are utilized in order 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 so as to possess the newest virus information detection mechanisms. However, the zero-day malicious exploit malware can not be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file to detect abnormous file segments.

6.2 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 scene. 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 doesn't understand why the app needs the permission to provide .

6.3 APPLICATION OF JAVA

Java is widely used in every corner of world and of human life. Java is not only used in softwares but is also widely used in designing hardware controlling software components. There are more than 930 million JRE downloads each year and 3 billion mobile phones run java.

Following are some other usage of Java:

1. Developing Desktop Applications
2. Web Applications like Linkedin.com, Snapdeal.com etc
3. Mobile Operating System like Android
4. Embedded Systems
5. Robotics and games etc.

6.4 FEATURES OF JAVA

The prime reason behind creation of Java was to bring portability and security feature into a computer language. Beside these two major features, there were many other features that played an important role in moulding out the final form of this outstanding language. Those features are;

6.4.1.simple

Java is easy to learn and its syntax is quite simple, clean and easy to understand. The confusing and ambiguous concepts of C++ are either left out in Java or they have been re-implemented in a cleaner way.

Eg: Pointers and Operator Overloading are not there in java but were an important part. .

6.4.2 Object Oriented

In java everything is Object which has some data and behaviour. Java can be easily extended as it is based on Object Model.

6.4.3 Robust

Java makes an effort to eliminate error prone codes by emphasizing mainly on compile time error checking and runtime checking. But the main areas which

Java improved were Memory Management and mishandled Exceptions by introducing automatic Garbage Collector and Exception Handling.

6.4.4 Platform Independent

Unlike other programming languages such as C, C++ etc. which are compiled into platform specific machines. Java is guaranteed to be write-once, run-anywhere language.

On compilation Java program is compiled into byte code. This byte code is platform independent and can be run on any machine, plus this byte code format also provide security. Any machine with Java Runtime Environment can run Java Programs.

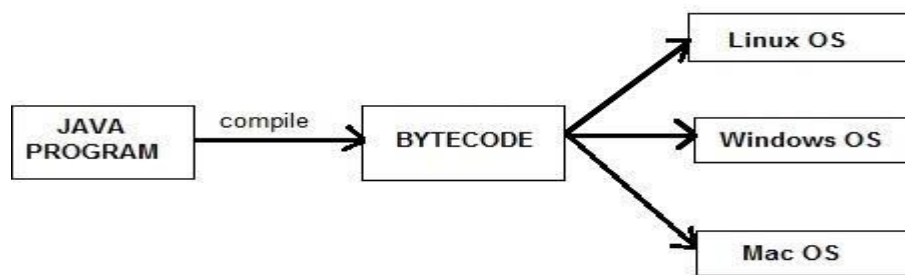


Figure 6.1.platform independent

6.4.5 Secure

When it comes to security, Java is always the first choice. With java secure features it enable us to develop virus free, temper free system. Java program always runs in Java runtime environment with almost null interaction with system OS, hence it is more secure.

6.4.6 Multi-Threading

Java multithreading feature makes it possible to write program that can do many tasks simultaneously. Benefit of multithreading is that it utilizes same memory and other resources to execute multiple threads at the same time, like While typing, grammatical errors are checked along.

6.4.7 Architectural Neutral

Compiler generates byte codes, which have nothing to do with a particular computer architecture, hence a Java program is easy to interpret on any machine.

6.4.8 Portable

Java Byte code can be carried to any platform. No implementation dependent features. Everything related to storage is predefined, example: size of primitive data types

6.4.9 High Performance

Java is an interpreted language, so it will never be as fast as a compiled language like C or C++. But, Java enables high performance with the use of just-in-time compiler.

6.5 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.

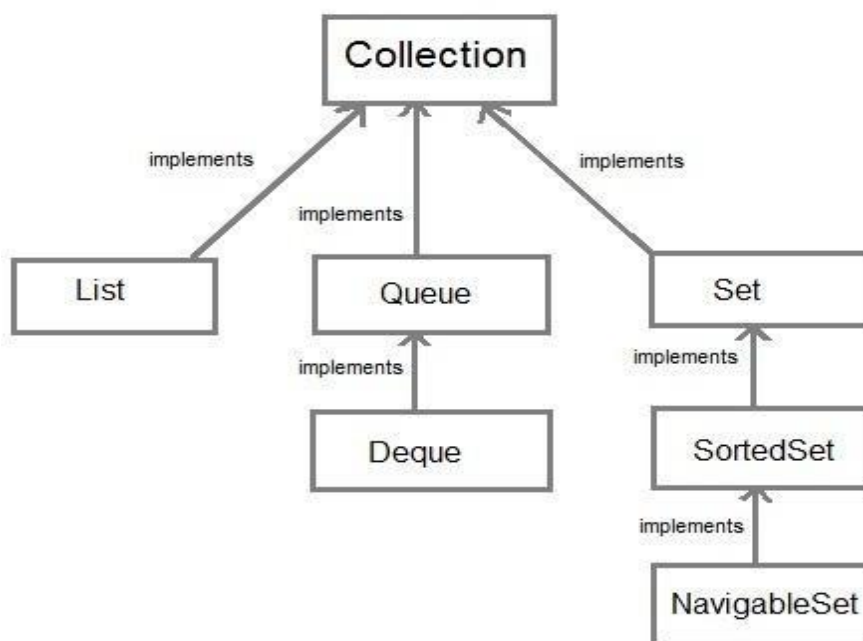


Figure 6.2.Collection Framework

6.6 SYSTEM DESIGN AND TESTING PLAN

6.6.1 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.

6.6.2 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.

6.7 USECASE DIAGRAM

The below diagram describes the usecase function.

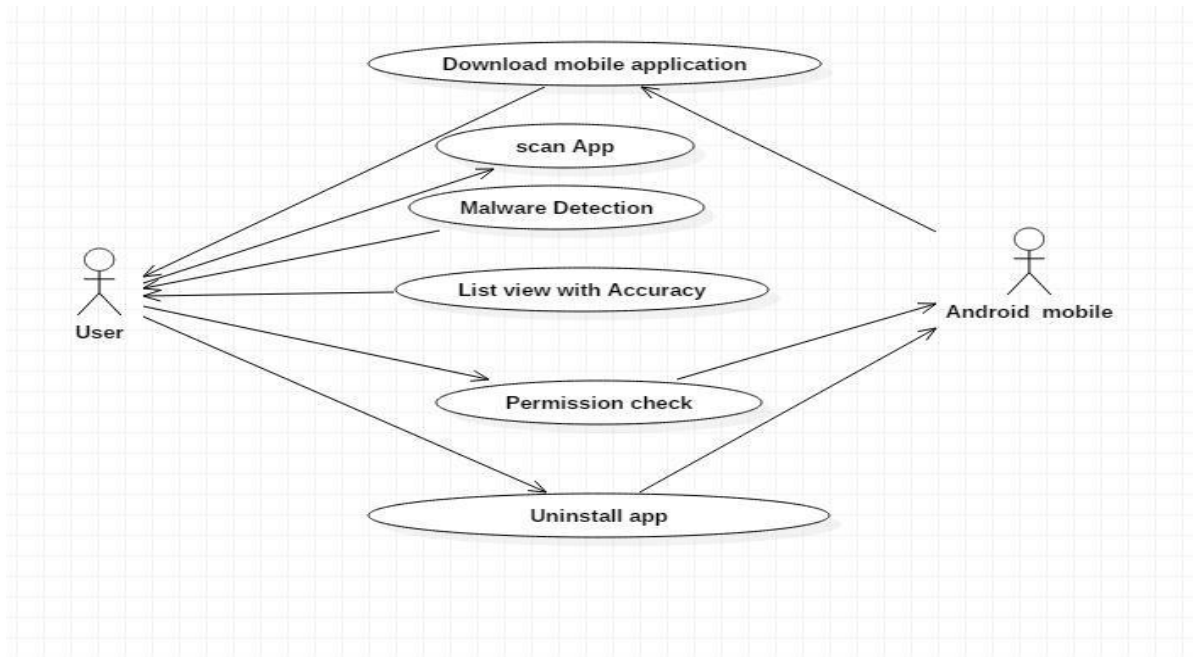


Figure 6.3:Use Case Diagram

6.8 ACITIVITY DIAGRAM

The below diagram describes the activity function.

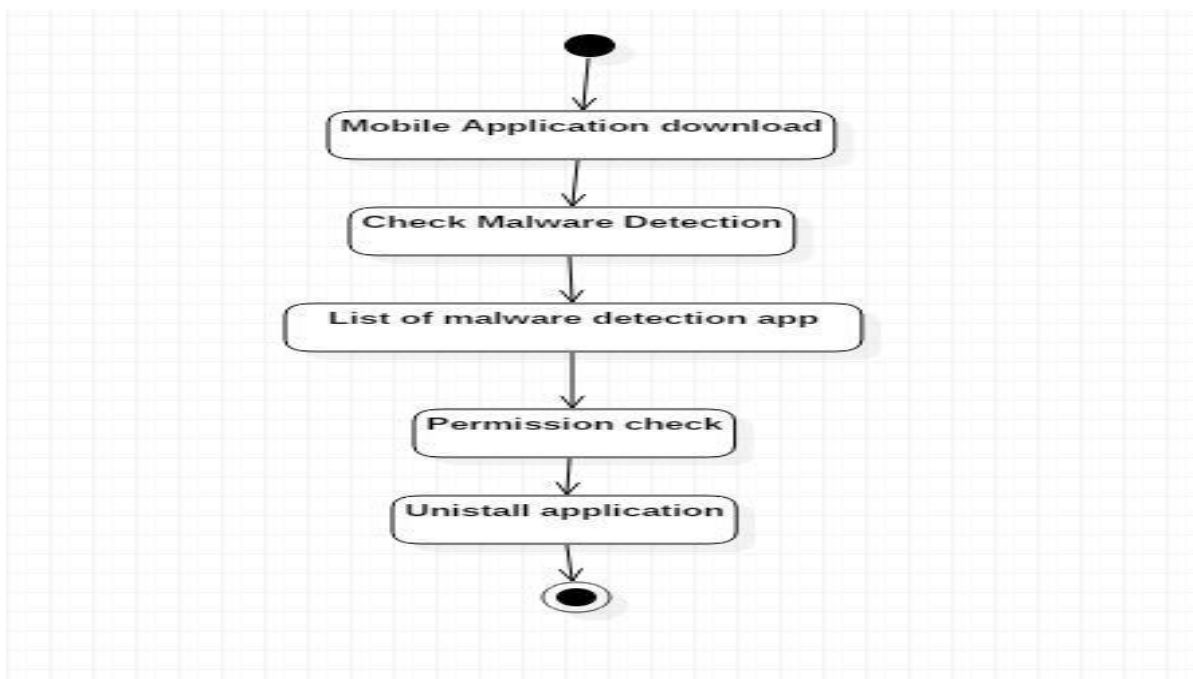


Figure 6.4:Activity Diagram

6.9 SEQUENCE DIAGRAM

The below diagram describes the sequence function.

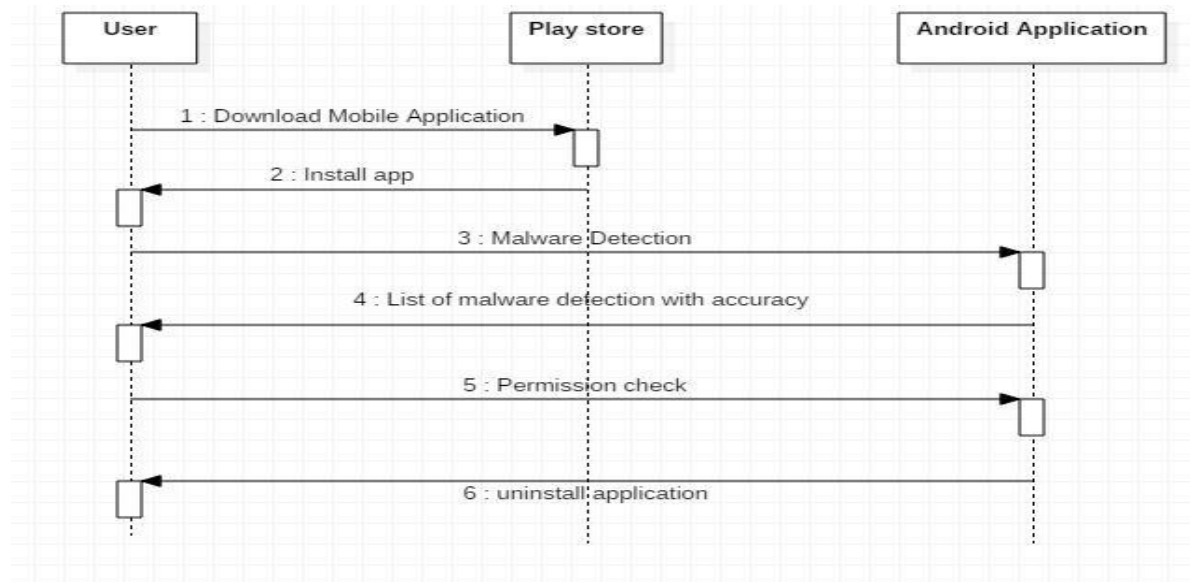


Figure 6.5:Sequence Diagram

6.10 COLLABORATION DIAGRAM

The below diagram describes the collaboration function.

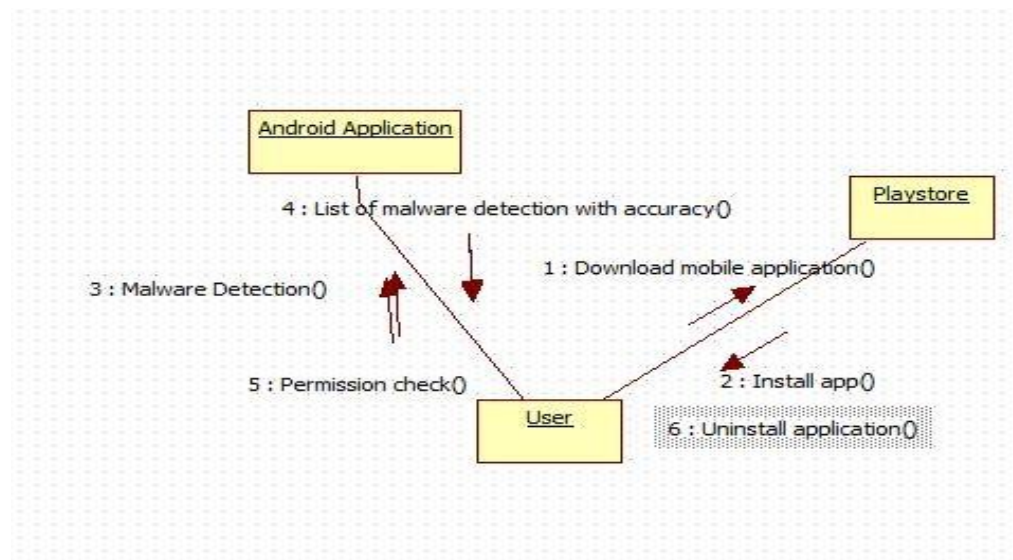


Figure 6.6:Collaboration Diagram

6.11 MODULES

- Application Download Module
- Android Application Scan
- Malware Detection
- Android Permission Check
- Privacy Policy Database

6.11.1 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.

6.11.2 Android Application Scan

In this module, many application's are already download in mobile phone through play store. After installation just find the malware present in various application's. 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 behavioural patterns and for detecting deviations from the application's expected behaviour. These methods were implemented and evaluated on Android devices.

Malware Detection

In this proposed system in order 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 categorised into analysis, classification, detection and ultimate containment of malware. Several classification techniques are utilized in order 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 so as to possess the newest virus information detection mechanisms. However, the zero-day malicious exploit malware can not be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file to detect abnormous file segments.

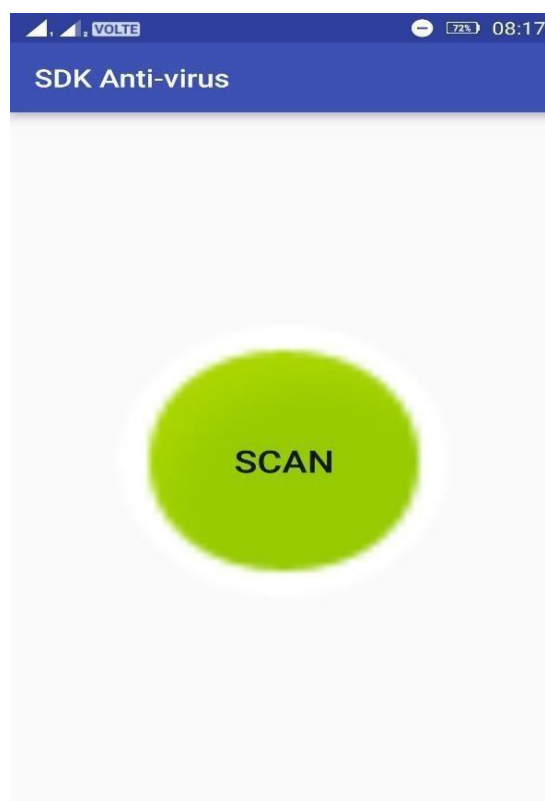


Figure 6.7: Scanning of Anti virus

6.11.3 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 scene. 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 doesn't understand why the app needs the permission to provides.



Figure 6.8:Android Permission Check

6.11.4 MALWARE DETECTION PROCESS

The image mentioned below is the output obtained during malware detection process.



Figure 6.9. malware detection process

6.12 SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed 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.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

6.12.1 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 be justified. 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.

6.12.2 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.

6.12.3 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. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

6.12.4 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

6.13 FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system .

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

CHAPTER-7

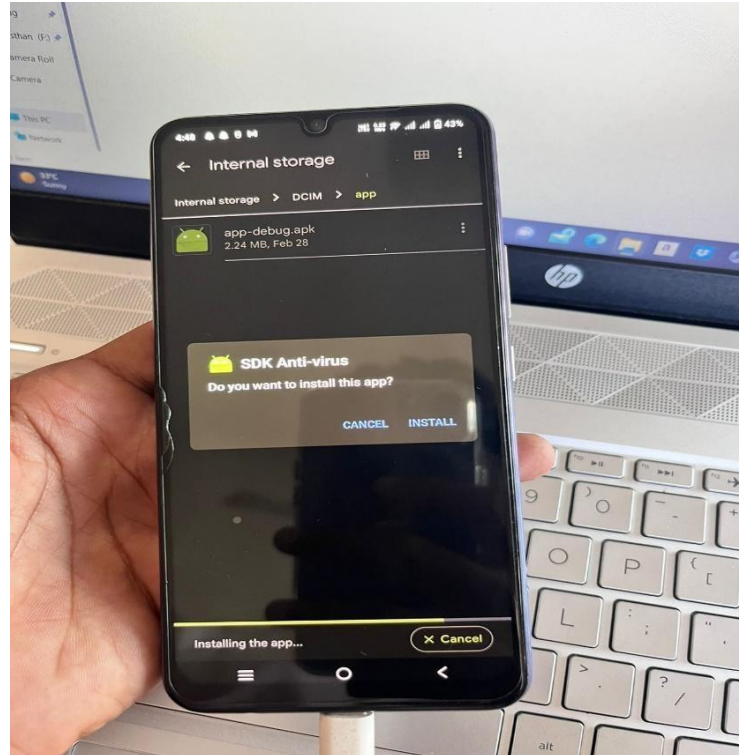
CONCLUSION

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 scene.

7.2 REAL TIME IMPLEMENTATION

7.2.1 Application Download

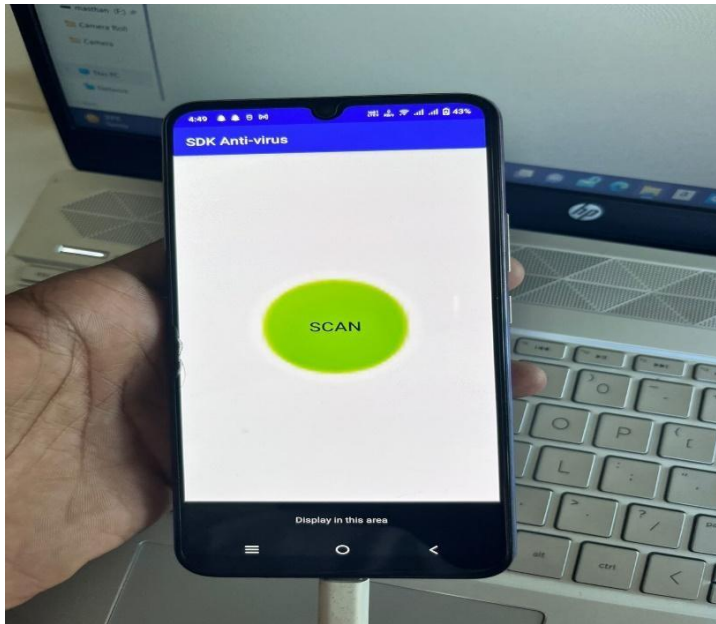
In this module, many applications are already download in mobile phone through play store. After installation just find the malware present in various applications.



7.2.2 ANDROID APPLICATION SCAN

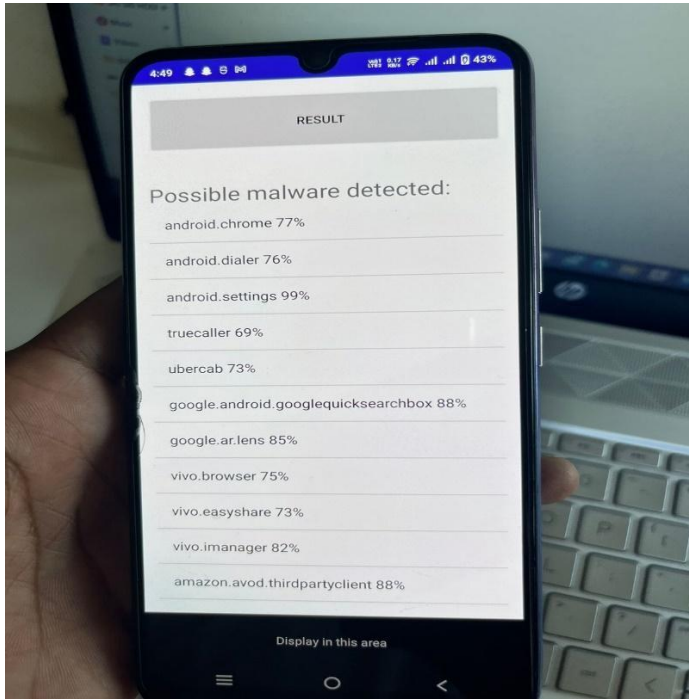
In this module, many application's are already download in mobile phone through play store. After installation just find the malware present in various application's. Malware of this kind can't be detected by victimization the quality signatures approach or by applying regular static or dynamic analysis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



7.2.3 MALWARE DETECTION

In this proposed system in order 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 categorised into analysis, classification, detection and ultimate containment of malware. Several classification techniques are utilized in order 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.



7.3 FUTURE WORK

- The widespread use of Android-enabled mobile devices in the global market is an attractive Android platform for malware development. An active and reliable way to detect malware in the Android environment is a complex problem that, as always, does not find its solution.
- Future work will discuss the development of a welldeveloped and new algorithm for detecting malicious code. Integration techniques, as well as the use of additional ML algorithms in accordance with the sequential receipt, use, explicit provisions, features.
- Many other static data-related functions are real workbooks that can be easily integrated into the file with the phrase method to create a large set. As a result of this research, the next natural step would be to share your Android malware families that are based on permissions.
- Finally, contradictions, Machine training can be seen as a response to enemy attacks, the point is to show state functions to trick ML algorithms.

REFERENCES

- [1] Rishab Agrawal , Vishal Shah , Sonam Chavan . Android Malware Detection Using Machine Learning. International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE).IEEE,(2020) .
- [2] Darus,Fauzi Mohd,Salleh Noor Azurati Ahmad,and Aswami Fadillah Mohd Ariffin."Android Malware Detection Using Machine Learning on Image Patterns"2018 Cyber Resilience Conference(CRC).IEEE,2018.
- [3] Vrama,P.Ravi Kiran,Kotari Prudvi raj,and KV Subba Raju."Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms."2017 InternationalConference on I-SMAC (IoT in Social,Mobile,Analytics and Cloud)(I-SMAC).IEEE2017.
- [4] Chang,Wei-Ling, Hung-Min Sun,and Wei Wu."An Android Behaviou Based Malware Detection Method using Machine Learning." 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC).IEEE, 2016.
- [5] Koli, J. D. "RanDroid: Android malware detection using random machine learning classifiers." 2018 Technologies for Smart-City Energy Security and Power (ICSESP).IEEE,2018.
- [6] 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.
- [7] 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. 33
- [8] 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.
- [9] 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.
- [10] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., & Wang, X. S. (2013, November)Appintent: 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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. 3

APPENDIX

A.SOURCE CODE

K MEANS JAVA

```
package com.example.kevinwu.project;

import java.util.ArrayList;
import java.util.Scanner;
import java.io.*;

public class KMeans
{
    //Number of Clusters. This metric should be related to the number of points
    private int NUM_CLUSTERS = 0;
    //container
    public ArrayList<Point> points;
    public ArrayList<Cluster> clusters;

    public KMeans(ArrayList<Point> observations, ArrayList<Point> centroids)
    {
        this.points = observations;
        NUM_CLUSTERS = centroids.size();
        this.clusters = new ArrayList<Cluster>();
        for(int i =0; i < NUM_CLUSTERS; i++)
        {
            Cluster one = new Cluster(i);
            one.setCentroid(centroids.get(i));
            this.clusters.add(one);
        }
    }

    //The process to calculate the K Means, with iterating method.
    public int calculate(int n_points)
    {
        int iteration = 0;
        boolean finish = false;
        // Add in new data, one at a time, recalculating centroids with each
new one.
        while(!finish)
        {
            //Clear cluster state
            clearClusters();
```

```

        ArrayList<Cluster> lastCentroids = new ArrayList<Cluster>();
        //copy every element
        for(int i = 0; i < this.clusters.size(); i++)
        {
            Cluster temp = new Cluster(this.clusters.get(i));
            lastCentroids.add(temp);
        }
        //Assign points to the closer cluster
        assignCluster(n_points);
        //Calculate new centroids.
        calculateCentroids();

        ArrayList<Cluster> currentCentroids = new
ArrayList<Cluster>(this.clusters);
        iteration++;
        //Calculates total distance between new and old Centroids
        double distance = 0;
        for(int i = 0; i < lastCentroids.size(); i++)
        {
            distance +=
Point.distance(lastCentroids.get(i).centroid,currentCentroids.get(i).centroid);
        }
        //if no centroids changed
        if(distance == 0)
        {
            finish = true;
        }
    }
    return iteration;
}

//clear all points in clusters
private void clearClusters()
{
    for(int i = 0 ; i < NUM_CLUSTERS; i++)
    {
        clusters.get(i).clear();
    }
}

private void assignCluster(int n_points)

```

```

{
    double max = Double.MAX_VALUE;
    double min = max;
    int cluster = 0;
    double distance = 0.0;

    for(int i = 0; i< n_points;i++)
    {
        min = max;
        for(int j = 0; j<NUM_CLUSTERS; j++)
        {
            Cluster c = clusters.get(j);
            distance = Point.distance(points.get(i), c.getCentroid());
            if(distance < min)
            {
                min = distance;
                cluster = j;
            }
        }
        points.get(i).setCluster(cluster);
        clusters.get(cluster).addPoint(points.get(i));
    }
}

//recompute the centroid based on the observations the cluster has
private void calculateCentroids()
{
    for(int i = 0; i< NUM_CLUSTERS; i++) {
        ArrayList<Point> list = clusters.get(i).points;
        int n_points = list.size();
        //only recompute the centroid when the cluster is not empty i.e. it
has > 0 points
        if (n_points > 0)
        {
            ArrayList<Double> sum = new ArrayList<Double>();
            int dimension = list.get(0).dimension;
            //accumulate the data at each dimension
            for (int j = 0; j < dimension; j++) {
                double data = 0;
                for (int k = 0; k < n_points; k++) {
                    data += list.get(k).data.get(j);
                }
                data = data / n_points;
            }
        }
    }
}

```

```

        sum.add(data);
    }
    Point centroid = new Point(sum);
    clusters.get(i).centroid = centroid;
}
}
}
}

```

NAIVE BAYES.JAVA

```

package com.example.kevinwu.project;

import java.util.ArrayList;

public class NaiveBayes
{
    //pre-processed clusters
    private ArrayList<Cluster> innocent = new ArrayList<Cluster>();
    private ArrayList<Cluster> malicious = new ArrayList<Cluster>();
    //build naive bayes probabilistic table
    ArrayList<Double> inno_yes = new ArrayList<Double>();
    ArrayList<Double> inno_no = new ArrayList<Double>();
    ArrayList<Double> mali_yes = new ArrayList<Double>();
    ArrayList<Double> mali_no = new ArrayList<Double>();
    //Number of innocent data
    private static int NUM_INNO_POINTS = 0;
    //Number of malicious data
    private static int NUM_MALI_POINTS = 0;

    public NaiveBayes(ArrayList<Cluster> innocent, ArrayList<Cluster>
malicious, int num_inno, int num_mali)
    {
        this.innocent = innocent;
        this.malicious = malicious;
        this.NUM_INNO_POINTS = num_inno;
        this.NUM_MALI_POINTS = num_mali;
        //go through the innocent clusters
        for(int i = 0; i < innocent.size(); i++)
        {
            double yes_probability = (double)innocent.get(i).points.size() /
NUM_INNO_POINTS;

```

```

        inno_yes.add(yes_probability);
        inno_no.add(1 - yes_probability);
    }
    //go through the malicious clusters
    for(int i = 0; i < malicious.size(); i++)
    {
        double yes_probability = (double)malicious.get(i).points.size() /
NUM_MALI_POINTS;
        mali_yes.add(yes_probability);
        mali_no.add(1 - yes_probability);
    }
}
//make a prediciton based on the permission list
public boolean isMalicious(ArrayList<Double> permission)
{
    //probability of being malicious
    double mali_prob = 1;
    //probability of being innocent
    double inno_prob = 1;
    //look at each permission in the permission array
    for(int i = 0; i < permission.size(); i++)
    {
        //only multiply the probability when both innocent and malicious
probability are not zero
        if(inno_yes.get(i) != 0 && mali_yes.get(i) != 0)
        {
            //does not have the permission
            if(permission.get(i) == 0.0)
            {
                inno_prob*= inno_no.get(i);
                mali_prob*= mali_no.get(i);
            }
            else
            {
                inno_prob*= inno_yes.get(i);
                mali_prob*= mali_yes.get(i);
            }
        }
    }
    //compare
    if(mali_prob >= inno_prob)
        return true;
}

```

```

        else
            return false;
    }
}

```

ANDROID MANIFEST.XML

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.kevinwu.project">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="SDK Anti-virus"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".Main3Activity">
            <intent-filter>
                <action
android:name="com.example.kevinwu.project.Main3Activity" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity
            android:name=".Main2Activity"
            android:label="@string/title_activity_main2"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity android:name=".result" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity">
            <intent-filter>
                <action
android:name="com.example.kevinwu.project.SecondActivity" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

```

        </intent-filter>
    </activity>
    <activity android:name=".activity_third">
        <intent-filter>
            <action
android:name="com.example.kevinwu.project.activity_third" />

            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </activity>
</application>

</manifest>

```

CLUSTER.JAVA

```

package com.example.kevinwu.project;

import java.util.ArrayList;
import java.util.List;

public class Cluster {

    public ArrayList<Point> points;
    public Point centroid;
    public int id;

    //Creates a new Cluster
    public Cluster(int id) {
        this.id = id;
        this.points = new ArrayList();
        this.centroid = null;
    }
    //copy
    public Cluster(Cluster cpy)
    {
        this.points = new ArrayList<Point>();
        //copy each element
        for(int i =0; i<cpy.points.size();i++)
        {
            Point temp = new Point(cpy.points.get(i));
            this.points.add(temp);
        }
    }
}

```

```

        this.centroid = new Point(cpy.centroid);
        this.id = cpy.id;
    }

    public List getPoints() {
        return points;
    }

    public void addPoint(Point point) {
        points.add(point);
    }

    public void setPoints(ArrayList<Point> points) {
        this.points = points;
    }

    public Point getCentroid() {
        return centroid;
    }

    public void setCentroid(Point centroid) {
        this.centroid = centroid;
    }

    public int getId() {
        return id;
    }

    public void clear() {
        points.clear();
    }
}

```

RESULT.JAVA

```

package com.example.kevinwu.project;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.graphics.Color;
import com.github.mikephil.charting.charts.BarChart;
import com.github.mikephil.charting.data.BarData;
import com.github.mikephil.charting.data.BarDataSet;

```

```

import com.github.mikephil.charting.data.BarEntry;
import com.github.mikephil.charting.interfaces.datasets.IBarDataSet;
import com.github.mikephil.charting.utils.ColorTemplate;

import com.github.mikephil.charting.utils.ColorTemplate;
import java.util.ArrayList;
public class result extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_result);
        BarChart chart = (BarChart) findViewById(R.id.barchart);
        // BarData data1=new BarData()
        BarData data = new BarData(getXAxisValues(), (IBarDataSet)
getDataSet());
        chart.setData(data);
        chart.setDescription("My Chart");
        chart.animateXY(2000, 2000);
        chart.invalidate();
    }
    private ArrayList<BarDataSet> getDataSet() {
        ArrayList<BarDataSet> dataSets = null;

        ArrayList<BarEntry> valueSet1 = new ArrayList<>();
        BarEntry v1e1 = new BarEntry(110.000f, 0); // Jan
        valueSet1.add(v1e1);
        BarEntry v1e2 = new BarEntry(40.000f, 1); // Feb
        valueSet1.add(v1e2);
        BarEntry v1e3 = new BarEntry(60.000f, 2); // Mar
        valueSet1.add(v1e3);
        BarEntry v1e4 = new BarEntry(30.000f, 3); // Apr
        valueSet1.add(v1e4);
        BarEntry v1e5 = new BarEntry(90.000f, 4); // May
        valueSet1.add(v1e5);
        BarEntry v1e6 = new BarEntry(100.000f, 5); // Jun
        valueSet1.add(v1e6);

        ArrayList<BarEntry> valueSet2 = new ArrayList<>();
        BarEntry v2e1 = new BarEntry(150.000f, 0); // Jan
        valueSet2.add(v2e1);
        BarEntry v2e2 = new BarEntry(90.000f, 1); // Feb

```

```

        valueSet2.add(v2e2);
        BarEntry v2e3 = new BarEntry(120.000f, 2); // Mar
        valueSet2.add(v2e3);
        BarEntry v2e4 = new BarEntry(60.000f, 3); // Apr
        valueSet2.add(v2e4);
        BarEntry v2e5 = new BarEntry(20.000f, 4); // May
        valueSet2.add(v2e5);
        BarEntry v2e6 = new BarEntry(80.000f, 5); // Jun
        valueSet2.add(v2e6);

        BarDataSet barDataSet1 = new BarDataSet(valueSet1, "Brand 1");
        barDataSet1.setColor(Color.rgb(0, 155, 0));
        BarDataSet barDataSet2 = new BarDataSet(valueSet2, "Brand 2");
        barDataSet2.setColors(ColorTemplate.COLORFUL_COLORS);

        dataSets = new ArrayList<>();
        dataSets.add(barDataSet1);
        dataSets.add(barDataSet2);
        return dataSets;
    }

    private ArrayList<String> getXAxisValues() {
        ArrayList<String> xAxis = new ArrayList<>();
        xAxis.add("JAN");
        xAxis.add("FEB");
        xAxis.add("MAR");
        xAxis.add("APR");
        xAxis.add("MAY");
        xAxis.add("JUN");
        return xAxis;
    }
}

```

SECOND ACTIVITY.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) {

    }

}

//add in the machine learning part::
//.....

//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_SERVICE")
== 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_SERVICE") ==
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[j].compareTo("android.permission.BIND_INCALL_SERVICE") == 0){
        perFlag[17] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_INPUT_METHOD") == 0){
        perFlag[18] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_MIDI_DEVICE_SERVICE") ==
0){
        perFlag[19] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_NFC_SERVICE") == 0){
        perFlag[20] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_NOTIFICATION_LISTENER_SERVI
CE") == 0){
        perFlag[21] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_PRINT_SERVICE") == 0){
        perFlag[22] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_REMOTEVIEWS") == 0){
        perFlag[23] = 1;
        }else
if(permission[j].compareTo("android.permission.BIND_TELECOM_CONNECTION_SERVICE")

```

```

== 0) {
    perFlag[24] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_TEXT_SERVICE") == 0){
    perFlag[25] = 1;
    }else
if(permission[j].compareTo("android.permission.BIND_TV_INPUT") == 0){
    perFlag[26] = 1;
    }else
if(permission[j].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_REMOVED") ==
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_OUTPUT") ==
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_ENABLED_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_STATE") ==
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_UPDATES") ==
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_PROVIDER") ==

```

```

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_PROCESSES") ==
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_FILESYSTEMS") ==
0){
    perFlag[75] = 1;
    }else
if(permission[j].compareTo("android.permission.MOUNT_UNMOUNT_FILESYSTEMS") ==
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_PACKAGES") == 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_MESSAGE") == 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);

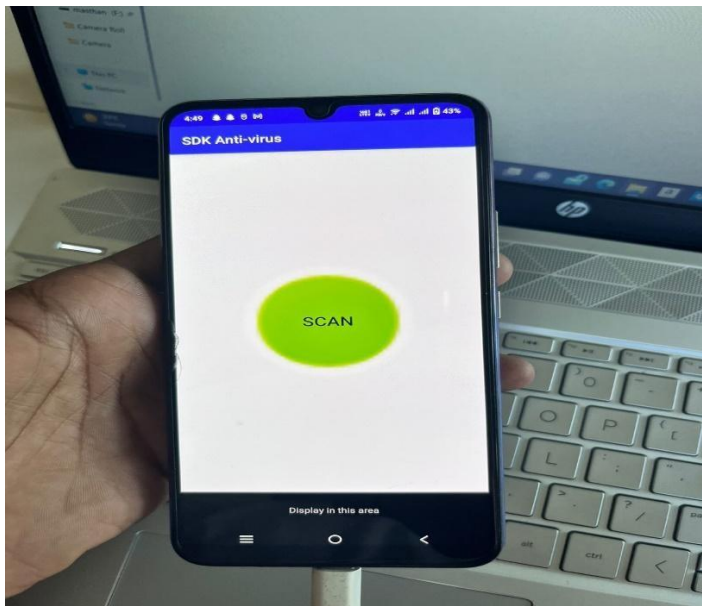
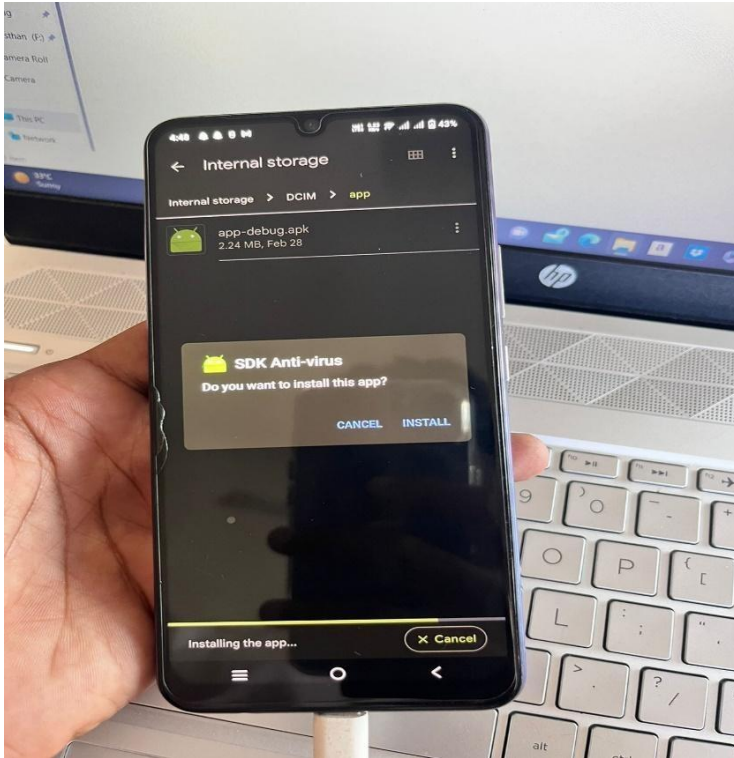
    //method to change the status bar percent
    if((pStatus +
Math.ceil(100.0/pkgAppsList.size()))<=100 ) {
        pStatus += Math.ceil(100.0 / pkgAppsList.size());
    }else{pStatus = 100;}
    handler.post(new Runnable() {
        @Override
        public void run() {
            progressBar.setProgress(pStatus);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

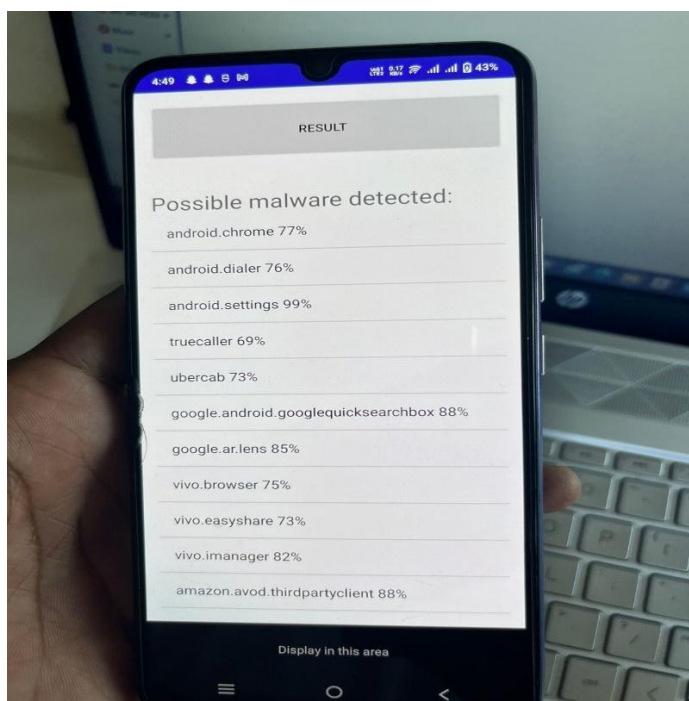
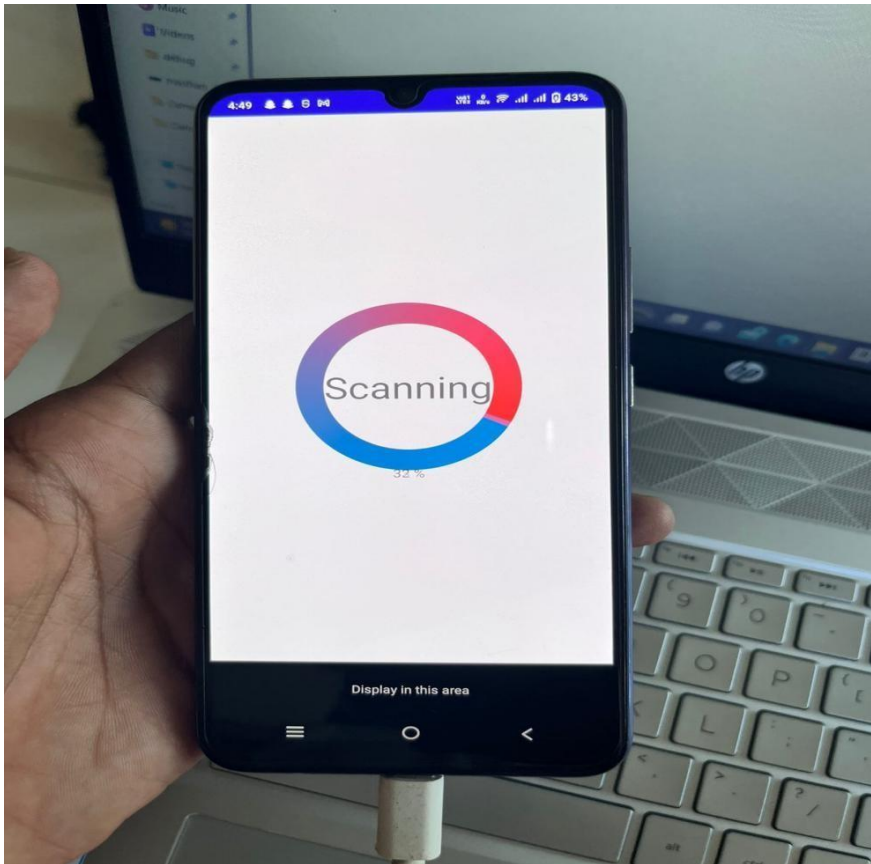
    // goto the next activity
    if(i == (pkgAppsList.size()-1)){
        //String com = "Complete";
        //Scan.setText(com);
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        Intent third = new
Intent("com.example.kevinwu.project.activity_third");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

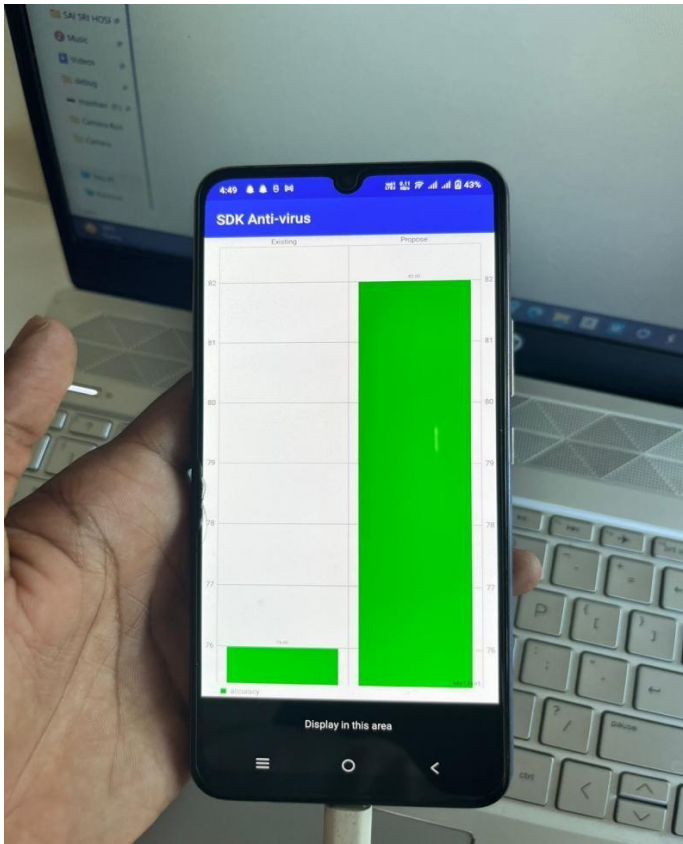
```

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

B. OUTPUT SCREENSHOTS







C.RESEARCH PAPER

International Journal of Machine Learning and Cybernetics

ANDROID MALWARE DETECTION USING MACHINE LEARNING

--Manuscript Draft--

Manuscript Number:	
Full Title:	ANDROID MALWARE DETECTION USING MACHINE LEARNING
Article Type:	Original Article
Corresponding Author:	shaik masthan masthan valli, B.E Sathyabama Institute of Science and Technology Dental College and Hospital:Sathyabama Institute of Science and Technology School of Dental Sciences chennai, INDIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Sathyabama Institute of Science and Technology Dental College and Hospital:Sathyabama Institute of Science and Technology School of Dental Sciences
Corresponding Author's Secondary Institution:	
First Author:	shaik masthan masthan valli, B.E
First Author Secondary Information:	
Order of Authors:	shaik masthan masthan valli, B.E
Order of Authors Secondary Information:	
Funding Information:	
Abstract:	The interface lets in the consumer to search for an app in the Play Store; restores the license listing and privateness coverage as quickly as viable. The person then has the potential to pick out a specific permission, and a listing of counseled diversifications is an in depth description of the permission itself. This interface lets in the person to quick verify the privateness dangers of an Android application, highlighting the applicable sections of the privacy policy and offering beneficial data about reasonable permissions. We draw from the privateness coverage and present to them a new approach to the analysis of privacy coverage inside the context of Android programs. Atool applied to apprehend the privateness implications of putting in 1/3-celebration apps has already established its potential to focus on instances of app disruption. The device is designed to be extensible and further tendencies may be easily included into the approach to improve balance and efficiency. In addition, if the utility handles non-public or sensitive consumer records, you may also see extra necessities within the Personal and Confidential Information section under. These Google Play requirementsare in addition to any requirements underneath applicable privacy or records protectionlegal guidelines. We suggested: The user who desires to set up and use any third- celebration utility does now not apprehend the importance and which means of the asked utility permissions, and therefore honestly allows all the sources, inflicting awful applications to be mounted and to perform mischief behind the curtain.
Suggested Reviewers:	

ANDROID MALWARE DETECTION USING MACHINELEARNING

¹Shaik Masthan Valli, ²Shaik Ismail, ³R Indhu, ⁴S Annie Angeline Preethi

^{1,2}Student, ^{3,4}Assistant Professor

^{1,2,3,4}Sathyabama Institute of Science and Technology, Chennai, Tamil Nadu.

[*indhu.ece@sathyabama.ac.in](mailto:indhu.ece@sathyabama.ac.in)

Abstract

The interface lets in the consumer to search for an app in the Play Store; restores the license listing and privateness coverage as quickly as viable. The person then has the potential to pick out a specific permission, and a listing of counseled diversifications is an in depth description of the permission itself. This interface lets in the person to quick verify the privateness dangers of an Android application, highlighting the applicable sections of the privacy policy and offering beneficial data about reasonable permissions. We draw from the privateness coverage and present to them a new approach to the analysis of privacy coverage inside the context of Android programs. A tool applied to apprehend the privateness implications of putting in 1/3-celebration apps has already established its potential to focus on instances of app disruption. The device is designed to be extensible and further tendencies may be easily included into the approach to improve balance and efficiency. In addition, if the utility handles non-public or sensitive consumer records, you may also see extra necessities within the Personal and Confidential Information section under. These Google Play requirements are in addition to any requirements underneath applicable privacy or records protection legal guidelines. We suggested: The user who desires to set up and use any third-celebration utility does now

not apprehend the importance and which means of the asked utility permissions, and therefore honestly allows all the sources, inflicting awful applications to be mounted and to perform mischief behind the curtain.

Keywords: Android, malware, benign, permissions-based, signatures-based.

I INTRODUCTION

Mobile devices with various operating systems have proven dramatic increase over the last decade, resulting in an growth within the variety and form of packages going for walks on cellular gadgets. Recently, smartphones have been used either as an addition to the computer laptop, or maybe instead of it. A nearer look at the motive for the usage of mobile telephones especially famous that they're commonly used for internet, social networking, and online reliance. In addition, they're used for unique mobile functions along with SMS messaging, random area analyzing, and ubiquitous get admission to. As the features of cell telephones (eg personal programs) increase, mobile phones are getting an increasing number of famous for a larger population. Market survey statistics show that during 2012, the range of telephone income international reached 208 million. This is 38.3% greater than in 2011. The development of smartphones and cell applications has led to primary changes inside the way we perform duties in diverse elements of our each day lives, along with enterprise and social communications. Mobile cellphone programs offer a huge variety now not handiest for day by day sports, however also for users with more unique wishes. From games to multimedia programs, navigation systems and fitness-related applications, emerging cellular application markets such as Google Play Market, Apple App Store

or Microsoft Windows Store offer a extensive range of packages for customers with exclusive wishes. Major application markets are constantly developing each in terms of applications that are supplied to users and in terms of downloads which can be made by using users. The rapid rise in popularity of smartphones has accelerated their capability as a target for malicious activities. This is mainly due to the reality that customers offer access to diverse private statistics the usage of cell applications. As a result, some applications within the marketplace are regarded to carry out malicious sports. Market policy providers additionally have an impact on the spread of malware. For example, Apple's App Store recently issued a strict registration and digital certification policy to the organisation earlier than freeing any app, ensuring proper protection scores for apps listed on their app platform. Others, along with the Google Play marketplace, supply developers more freedom to submit their apps to the marketplace. Drugs are then taken off the market if fraudulent pastime is reported. In specific, the Android operating gadget (inside the contemporary model) lets in you to get rid of the malicious application from the tool. A comparable elimination mechanism is also used in the Apple App Store. A put up-detection malware removal strategy calls for early detection of malicious applications.

II OBJECTIVE

The essential cause of this device is that the person who desires to deploy and use any third-party utility does not understand the which means and that means of the permissions required with the aid of the software, and consequently actually offers all of the sources, causing bad programs to also install and carry out their malicious activities.

III PROBLEM FORMATION

When the use of Android app improvement, it's far essential to hit upon malicious behavior in Android apps to open the cease of safety and privacy. The proposed machine is a gadget getting to know-primarily based malware detection gadget that classifies malicious and secure packages.

IV LITERATURE SURVEY

Literature evaluation is the most crucial step within the software program development manner. Before the device is advanced, the time factor, the economic system and the electricity of the organization ought to be decided. When all these situations are met, the subsequent step is to decide which operating device and language may be used to broaden the device. When programmers begin building a tool, they need a number of external guide. This assist can be received from older software program, from books, or from websites. Before growing a system, those considerations are taken into consideration while the system is being evolved. The most a part of the assignment improvement is considering and absolutely studying all the requirements vital for the improvement of the task. For any cause, literature overview is the most important part of the software program development manner. Before developing the relevant gear and strategies, it's miles important to decide and examine the importance of time, the want for sources, the hard work pressure, the economic system and the electricity of the enterprise. With this stuff satisfied and fully understood, the following step is to decide the specification of the software within the respective device, as to what kind of running device could be required for the cause, and what's going to be had to pass all of the essential software. To the next steps to broaden associated tools and sports.

A Bayesian Classification Analysis-Based Approach to Android Malware Detection Mobile malware is becoming larger and more complicated due to the steady proliferation of smartphones around the sector. Android is unexpectedly turning into the most famous cell platform, main to a dramatic boom in malware objectives

On the forehead In addition, Android malware is evolving too speedy to be detected by traditional signature-based improvement. Despite ongoing detection efforts, well timed detection of new malware remains a essential trouble. This calls for

new procedures to mitigate the developing chance of Android malware. The authors therefore expand and resolve a proactive gadget getting to know technique in Bayesian classification geared toward detecting unknown Android malware the use of static analysis. The have a look at, on a large range of malware samples, from maximum of the present malware families, demonstrates the excessive precision of the detection abilities. Empirical outcomes and comparative evaluation are provided, imparting useful data for growing effective solutions based totally on static Bayesian analytical classification for the detection of unknown malware for Android.

Android permissions: a perspective combining risks and benefits

The obvious improvement of the Android platform over the past few years has made it a totally profitable target for malicious application (utility) developers. There are many examples of malicious programs that ship SMS messages, display users' private facts, or applications that, even if they are not called malware, perform questionable movements that affect the consumer's privateness or cost him money. In this newsletter, we explore the potential to apply each in step with-app and in step with-app class permissions, in addition to which permissions are asked through other apps within the equal class, to better tell users whether or not the app is putting in a threat. Commensurate with its predicted benefits. The current method simplest considers the risks of permissions requested by way of an application and ignores each the benefits and permissions requested through different applications, with constrained effect. We provide several threat signs and examine them using two datasets, certainly one of 158,062 Android apps from the Android Market and the other of 121 malicious manufacturers. We reveal the effectiveness of our provide via tremendous analysis.

The Android working machine has become exceedingly popular over the past few years. This platform has verified itself to be dominant no

longer best inside the cell global, however also in IoT technology. However, this popular protection currency has turn out to be a tempting target for malicious use. There is therefore an increasing want for classy, computerized and portable malware detection solutions [1].

Android structures have emerged as a goal for criminals due to the multiplied variety of Android users in 2017. More than eight,000 Android malware are detected each day, making it hard for malware analytics to stumble on. Traditional malware detection methods are now not dependable for detecting newly created malware in a brief time period. In this text, we use some other method to hit upon the Android checkout loop. The detection is completed and in comparison the use of 3 different classifiers, specifically okay-nearest (KNN), random bounce (RF), and choice tree (DT). Representation [2].

Recently Android malware threats are growing at a rapid price with utilization and reputation. Various research have highlighted the importance of Android permission pattern evaluation for effective Android malware detection. Optimizing Android malware detection with multidimensional license facts is a bottleneck this is an acute hassle. In this take a look at, nature-based totally metaheuristic algorithms such as firefly, bat, and whale optimization algorithms are explored to analyze numerous Android styles appropriate for malware detection. Various device gaining knowledge of classification algorithms together with Linear Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Decision Tree (DT);

Random Forest (RF), Gradient Boosting (GB) and Extreme Learning Machine (ELM) are evaluated. Shell-based feature choice the usage of the Firefly Algorithm (WFSFA) outperformed different algorithms in terms of feature reduction and marked a higher category accuracy of 95.28% when examined with the CICInvesAndMal2019 multivariate function set with 4115 functions [3].

Malicious software program is designed to

disrupt, disable, or take over a laptop system. Android malware in particular targets the Android OS by using leaking touchy facts and disrupting the system. Several tries have been made to discover MAS malware thru secret information leaks and machine crashes. Several tries were made to detect Android malware. But these equipment can not discover malware right away, and most of them rely on signatures that can't come across new ones. However, those groups can not mechanically detect malware, and most rely upon subscriptions that can not stumble on new malware variations. In our paintings we have explored one of a kind algorithms to find the first-rate algorithm for predicting malware and variations... [4].

The growing popularity of Android-primarily based smartphones has caused the proliferation of malicious programs developed with the aid of malicious actors, necessitating the usage of sophisticated malware detection techniques. Several methods had been proposed that use static and/or dynamic information extracted from an Android utility to discover malware. The use of system learning is tailored to diverse malware detection strategies to triumph over the manual overhead. Machine getting to know classifiers had been broadly used to discover Android malware styles based on static capabilities and dynamic behavior. In order to resolve the problem of malware detection, in this newsletter we proposed a gadget mastering detection machine for malware on the Android platform. We advise a device based totally on randomly accrued examples of proper and horrific programs to teach classifiers. The machine detects asked permissions, inclined API calls, and the presence of key application statistics together with; dynamic code, mirrored image code, local code, photo code and database required by way of the packages proposed in the preceding answers, and used them as capabilities in diverse gadget mastering classifiers to build a type version. To test the performance of the

proposed "Ran Droid" system, various experiments had been carried out, which showed that Ran Droid ought to obtain a high type accuracy of ninety seven.7%.

Keywords: Android cell safety, malware evaluation [5].

The Android working device has turn out to be the most popular working device on smartphones and capsules considering 2012. This popularity has caused the fast boom of Android malware in latest years. The complexity of Android's obfuscation and malware detection methods has significantly advanced, making many conventional malware detection strategies obsolete. In this newsletter, we advocate DL Droid, a deep studying device for the detection of Android malware via dynamic evaluation the usage of complete kingdom era. Tests were completed on greater than 30,000 applications (safe and malicious) on real machines. In addition, experiments were additionally performed to examine the detection performance and code coverage of the whole input country era technique with a in large part stateless approach the use of a deep studying machine. Our take a look at suggests that DL Droid can achieve as much as 97.8% detection rate (with dynamic functions simplest) and 99.6% detection (with dynamic + static capabilities) respectively, that is superior to traditional system gaining knowledge of strategies. In addition, the outcomes boom the importance of better enter technology for dynamic evaluation, as the DL Droid based totally on state input generation suggests that it's far advanced to the modern approach [6].

In trendy digital global, maximum anti-malware equipment are signature-based totally, that is ineffective for detecting unknown state-of-the-art malware, viz. Metamorphic malware. In this article, we take a look at the frequency of opcode occurrence for unknown malware detection using device studying strategies. For this, we used the Microsoft malware category dataset. The pinnacle

20 features derived from Fisher's scale, facts acquisition, advantage, selected chi-squared capabilities, and symmetric uncertainty are in comparison. We additionally tested numerous classifiers in the WEKA GUI-primarily based system learning device and found that five of them (Random Forest, LMT, NBT, J48 Graft, and Arbor) can discover malware with an accuracy of almost one hundred%.

Keywords: metamorphic, malware protection, WEKA, device gaining knowledge of [7]

In this look at we constitute malware as a series of opcodes and come across them the usage of the Deeps Trust Network (DBN). Compared to standard sparse neural networks, DBNs can use unstructured statistics to teach a generative model layer which can higher represent the traits of information patterns. We examine the overall performance of DBN with 3 basic malware detection fashions that use vector machines, selection bushes, and okay nearest associates as classifiers. Experiments display that the DBN model offers extra correct detection than the base fashions. When additional raw information is used to teach DBN, DBN plays higher than different discovery models. We also use DBN as an encoder to automatically extract the feature vector. Experiments display that the wide encoder can efficaciously reproduce the basic shape of the input statistics and extensively reduce the quantity of the function vector [8].

One of Android's main safety mechanisms for restricting get admission to to features at its core is permission manage. However, there is a big risk in regards to granting permissions as 97% of malware attacks target Android mobile devices. As malware becomes greater state-of-the-art, a greater sophisticated method has been proposed in a latest take a look at that tests the conduct of the evolved utility towards the behavior of the marketed software due to inconsistencies. In this newsletter, we discover such inconsistencies by means of matching the software's requested license with the application's natural language descriptions, which

gives intuitive insight into the software's expectations from the user. We consequently suggest which you use the prolonged app description for higher malware detection based on app descriptions and permissions. To compare overall performance, we ran numerous checks with the 56K. The proposed improvement reduces the fake positives of the cutting-edge processes, Why by way of, Auto Cog, CHABADA by at least 87% and TAP by means of at least fifty seven%. We proposed a brand new technique to evaluate the reliability of text descriptions for permission-based totally malware detection. Our experimental outcomes display a excessive detection fee of 98.72% for 71 malware households and an accuracy of 90% for benign and obfuscated malware samples. Our effects also show that touchy permissions and textual UI descriptions are required to offer promising talents for strong Android malware detection [9].

Anti-malware software program has traditionally used signature-primarily based detection techniques and heuristics. These detection structures should be manually up to date to locate new, unknown, or adapted malware. The goal is to create a new malware detection solution in an effort to serve three purposes: mechanically discover and record unknown documents based at the severity of the malware spectrum; enforcing a hybrid machine studying technique to stumble on malware change traces; and enhance the accuracy of the detection outcomes. Our answer is powered through metallic mining and machine getting to know standards and algorithms. We carry out styles of information mining on samples, n-gram and PE function extraction, which are used for system studying. We also present a new hybrid learning method that uses each unsupervised and unsupervised system gaining knowledge of in a duplicate protocol. A controlled set of rules is applied to suggest whether a document is taken into consideration malicious or secure. The malware files will then be classified after which assigned in step with the severity spectrum the usage of the SOFM algorithm [10].

Android malware has persevered to develop at an alarming rate in latest years. Many modern-day malware programs that use distinctly state-of-the-art detection prevention strategies are tons much less powerful than traditional gadget studying-primarily based malware detection techniques. In unique, Android can tolerate unique sorts of malware and come across boundaries because of the use of a unmarried category algorithm. To address this obstacle, in this article, we advocate a new technique that makes use of parallel mastering and information aggregation strategies called Android malware detection. To put in force this technique, we first extract eight feature lines from the Android app static evaluation and create types of function movies after feature selection. Then an apparent learning pattern detection is evolved to speed up the classification process. Finally, we discover an approach to information fusion based totally on probabilistic evaluation and theory that efficiently achieves detection results. To check our technique, different modern day invention work turned into decided on for assessment with real Android apps. Experimental effects show that it is able to offer a better detection accuracy and superior time to present malware detection answers [11].

In order to solve the trouble of malware detection, researchers have these days started to discover the possibilities of gadget studying techniques to recommend powerful processes. Several promising consequences were said in the literature, and lots of approaches had been evaluated using a not unusual "10-factor cross-validation" layout. This paper critiques the reason of malware detection to run the adequacy of the "10x" scheme of trying out strategies that don't surely perform well. To this quit, we developed a number of machine mastering classifiers, which were based on a new set of features superimposed on CFG packages. We use big facts set of over 50,000 Android apps, gathered from sources that we have selected the use of a country of the artwork method. We show that our approach is superior to current system gaining knowledge of methods. However, such excessive performance in

datasets does no longer suggest average high overall performance inside the real international.

Many present malware detection strategies are designed to locate malware primarily based on resource necessities including permissions, device calls, and suspicious API calls. The Android app tag is now used to tell customers about the dangers of putting in apps. Users from all around the global have started out to increase an increasing number of cellular applications in reaction to Android technology and wireless networks. The growth inside the number of programs leads to an growth inside the likelihood of putting in malicious packages and malware. When the person installs the app, they've the option to test the permission required for the Android app and cancel the set up if the permissions are unnecessary or excessive. As a result, present strategies are less effective at detecting many regular malware that require little or no suspect sources, but use the intercomponent verbal exchange (ICC) mechanism at the start of the assault. In an existing gadget, ICC Detector gives a reputation version with observe packages and a fixed of educated malware, and makes use of the educated version to stumble on malware. The approach we endorse proposes a framework which could hit upon malicious Android packages to assist the Android market. The proposed framework is designed as an engine-based malware detection system on Android to detect malicious programs and objectives to enhance the safety and privateness of telephone customers. This device monitors numerous traits and authority-primarily based and obtained results from Android applications and analyzes these characteristics the usage of system gaining knowledge of classifiers to decide whether the utility is right or malware[12]

With over 50 billion downloads and over 1.3 million apps at the respectable Google marketplace, Android continues to gain reputation among telephone users round the world. At the same time, there has been an growth in platform-

machine malware, with newer techniques the use of greater sophisticated detection-avoidance techniques. As conventional malware-primarily based methods become less effective at detecting unknown malware, an choice is wanted for strategic 0-day detection. Therefore, this examine proposes a getting to know synthesis method for Android malware detection. It combines the blessings of static analysis with the efficiency and performance of ensemble device learning to improve Android malware detection accuracy. Machine gaining knowledge of fashions are built using big repositories of malware styles and applications covered by means of main antivirus providers. The provided experimental consequences and analysis display that the proposed technique uses a large area to utilize the studying talents of clustering [13].

Malicious software program abounds in the global of infinite pc customers who're constantly confronted with these threats from various assets, consisting of the Internet, local networks and portable drives. Malware is a low or high threat ability and can reason structures to malfunction, steal facts, or even crash. Malicious software program can be an executable file or a gadget library file inside the shape of viruses, worms, Trojans designed to violate machine security and violate consumer privateness. Typically, antivirus software program is primarily based on a signature detection system this is constantly up to date from the Internet and consequently keeps music of virus signatures. While this can be sufficient for home users, the safety threat from a brand new virus ought to threaten the entire company community. This article proposes a new and progressed anti-virus engine that not best scans files, but also accumulates expertise and detects files as capacity viruses. This is carried out with the aid of extracting machine API calls made through various ordinary and malicious executables and the use of machine gaining knowledge of algorithms to become aware of and therefore area files on the

safety risk scale. Although the sort of system is CPU intensive, it is very effective whilst used within the middle to defend the corporate network, which may be extra vulnerable to such recognized virus threats. While this could be sufficient for domestic customers, the security danger from a new virus should threaten the complete corporate community. This article proposes a new and progressed anti-virus engine that not simplest scans files, but additionally accumulates information and detects documents as capability viruses. This is achieved by extracting system API calls made with the aid of numerous normal and malicious executables and the use of system studying algorithms to perceive and therefore vicinity documents on the security chance scale. Although the sort of system is CPU extensive, it's miles very effective while centrally the usage of a company network to shield in opposition to such threats [14].

VEXISTING SYSTEM

GP PP (General Permissions - Private Invasive Permissions) is a useful example for breaking down permissions into fashionable and private invasive permissions. The version presents users with a easier manner to determine which apps are risky to install. Based at the set of permissions that a specific utility has asked; the GP PP model identifies the application as privacy-enhancing if most of the permissions asked are privateness-enhancing. This manner customers can determine which parish license is risky. We test the GP PP version to peer if the model applies to the proposed license utility requested with the aid of the application.

VI DISADVANTAGES OF THE EXISTING SYSTEM

1. Security is low.
2. The slowest manner to hit upon malware in your smartphone.
- Three. There are many elements, such as time, cost, records, training, and so forth.

VII PROPOSED SYSTEM

Determine the list of installed third-celebration packages. They elicit a full listing of permissions for every application. Define Android safety stage: security degree of each permission, i.E. Regular or

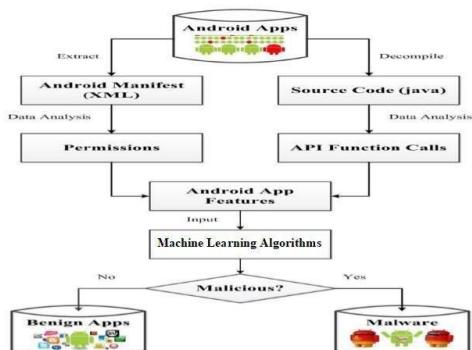
dangerous for each utility. Get Android app permissions set. Classification algorithms are used to come across malicious packages. Pay interest to the accuracy of the junk mail classification it gives and the time it takes to complete it. Accurate outcomes among distinctive malware and not unusual packages. Classifiers are advanced.

VIII CONVENIENT PROPOSAL SYSTEM

1. High fitness.
2. All types of statistics assets may be located right here and lots of strategies to use
3. More strong and flexible than the cutting-edge machine.

IX SYSTEM ARCHITECTURE

On cellular phones, the applications could be numbered. There may be a -step technique: applications could be extracted the use of XML code and programs will be compiled into Java code. In XML, after parsing the statistics, many permissions might be generated, and then in Java, after parsing the data, there may be function calls to the API. Android apps could be scanned the usage of a choice tree set of rules. After the scanning procedure, malicious and secure applications can be positioned. Using the tree algorithm, if sure, the utility is malicious, and if no longer, the utility isn't malicious. With this structure we constitute the model of our undertaking.



X SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

System : Pentium Dual Core.

Hard Disk : 120 GB.

Monitor : 15"LED

Input Devices : Keyboard, Mouse

Ram : 1GB.

SOFTWARE REQUIREMENTS:

Operating system: Windows 7. Coding

Language: Android, Java

Toolkit : Android 2.3 Above

IDE : Eclipse/Android Studio

XI MODULES

- 1 Login/Registration
- 2 Android Application Scan
- 3 Malware Detection
- 4 Android Permission Check
- 5 Privacy Policy Database

LOGIN MODULE

This is the first movement that opens while the consumer opens the website. The person is required to offer the ideal contact number and password that the consumer enters in the course of registration to enter the site. If the records from the person fits the information within the database, then the consumer is efficaciously logged into the web page, in any other case a failure message is displayed and the user desires to re-input the precise information. A registration link is likewise provided for new person registrations.

REGISTRATION MODULE

A new person who desires to get entry to the application ought to sign in before logging in. Clicking the sign in button within the login action opens to check in the information. A new user is registered through entering their full call, password and call wide variety. The consumer have to re-input the password within the Confirm Password text field. When the consumer enters statistics in all of the textual content fields, when the login

button is clicked, the facts is transferred to the database and the person is directed to login once more. The user person must then log in to the software to get admission to it. Validations implemented to all textual content fields for proper operation of the utility. As the information in each textual content area is to be accomplished, that is, in each text field, be it name, contact, password or confirmation password, they will not be empty within the registration. If this kind of textual content area is empty, the utility will supply a message through the information that ought to be in each textual content. Also, the records inside the password and password affirmation fields must fit efficaciously. Another affirmation is that the touch quantity should be valid and incorporate 10 digits. If any such test is violated, the registration will fail and the user will have to sign up again. The application will show a message whilst one of the fields is empty. If all such statistics is authentic, the person could be directed to open the software.

Application Download

In this module, many packages have already been downloaded to the cellular smartphone thru the play store. Once installed, genuinely look for malware present in various packages.

Android Application Scan

In this module, many packages have already been downloaded to the cell phone through the play store. Once mounted, without a doubt search for malware found in diverse programs. This kind of malware can not be detected by using victimization the usage of a qualitative signature technique or through making use of traditional static or dynamic analysis.

The discovery is purely based on the application's network site visitors styles. For every application, a specific lively

model representing the nearby model (i.E. The device) is discovered.

Semi-supervisory gadget mastering strategies are used to study normal conduct patterns and detect errors expected from the conduct of the application. These packages are applied and evaluated on Android gadgets.

Malware Detection

In this proposed machine for improving cell software safety, one approach is proposed to evaluate the safety of cellular packages based on cloud computing and information mining. The challenge of malware detection will be divided into the evaluation, class, detection and very last containment of the malware. Several strategies of malware type are used to pick out their instances, and this makes it viable to pick out the sort and activities of malware and new variants. To cope with such threats to cellular devices, some kind of security mechanism should be applied. Malware evaluation is ready distinguishing malware instances primarily based on absolutely extraordinary category schemes through using specific malware characteristics. Also evolved here is a prototype machine referred to as Malware Detection System to determine the virulence or innocence of a mobile application. Malware detection is all about speedy detecting and blocking any times of malware from causing similarly harm to the machine. The closing a part of the job carries malware containment, which incorporates efforts to forestall its unfold and save you harm to the machine. Commercial antivirus particularly uses a statistics-primarily based approach, wherein the records need to be updated frequently to discover the brand-new virus detection gadgets. However, zero-day exploit malware can't be detected through an antivirus-based antivirus, however, the use of mathematical evaluation of the contents of a

binary file to hit upon anomalous file segments.

Android Permission Check

A person who desires to use the installation and application of any 1/3 party does now not recognize the that means and that means of the desired licenses of the utility, and therefore clearly presents all the licenses, which additionally presents for the set up of malicious packages and their malicious moves. Behind the scenes If an software needs to apply assets or statistics outdoor of its own sandbox, it should request the suitable permission. You claim that your app desires a license by list the permissions in the app show up after which prompting the consumer to confirm every license at runtime. If the utility calls for a risky license, you must take a look at to peer if you have the license each time you perform an operation that requires that license. The conduct of the device after the permission is declared depends on the sensitive permission. Some abilities are considered "regular", so the gadget offers them right now after set up. Other permissions are taken into consideration "unstable", which includes giving the user express access to the app. For greater information about the specific styles of permissions, see Security levels. If a user intends to apply functions that require a license but refuses to gain a license, it probably method the consumer does no longer apprehend why a license must be implemented to that characteristic..

XIICONCLUSION

In this machine, if the utility handles private or touchy person information, the requirements in the "Personal and Confidential Information" segment below also are necessary. These Google Play requirements are further to any requirements below relevant privacy or records protection laws. We advocate: The user who desires to install or use any 1/3-party software does now not recognize the

importance and meaning of the permissions asked by means of the application, and thus surely offers all of the assets, inflicting bad programs to install and run. His malicious activities behind the scenes.

REFERENCES

- [1] Rishab Agrawal , Vishal Shah , Sonam Chavan . Android Malware Detection Using Machine Learning. International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE).IEEE,(2020)
- [2] Darus,Fauzi Mohd,Salleh Noor Azurati Ahmad,and Aswami Fadillah Mohd Ariffin."Android Malware Detection Using Machine Learning on Image Patterns"2018 Cyber Resilience Conference(CRC).IEEE,2018.
- [3] Vrama,P.Ravi Kiran,Kotari Prudvi raj,and KV Subba Raju."Android mobile security by detecting and classification of malware based on permissions using machine learning algorithms."2017 InternationalConference on I-SMAC (IoT in Social,Mobile,Analytics and Cloud)(I-SMAC).IEEE2017.
- [4] Chang,Wei-Ling, Hung-Min Sun,and Wei Wu."An Android Behaviou Based Malware Detection Method using Machine Learning." 2016 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC).IEEE, 2016.
- [5] Koli, J. D. "RanDroid: Android malware detection using random machine learning classifiers." 2018 Technologies for Smart-City Energy Security and Power (ICSESP).IEEE,2018.
- [5] 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.
- [6] 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.

[7] 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.

[8] 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.

[9] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., & Wang, X. S. (2013, November)Appinttent: 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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).