

ANALYSING ANDROID APP PRIVACY USING CLASSIFICATION ALGORITHM

Submitted in partial fulfilment of the requirements for
The award of
Bachelor of engineering degree in Computer Science and Engineering
By

NEKKANTI VISHNU SURYANARAYANA (37110504)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

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

Accredited with Grade "A" by NAA

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI - 600 119**

APRIL - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai –
600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **NEKKANTI Vishnu** (37110504) who carried out the project entitled as **Analysing android app privacy using classification algorithm** under my supervision from August 2020 to March 2021.

Internal Guide

Mr. VIGNESH. R, M.E., (Ph.D.,)

Head of the Department

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

Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **NEKKANTI VISHNU (37110504)** hereby declare that the Project Report entitled "**Analysing android app privacy using classification algorithm**" done by me under the guidance of **Mr. Vigneshan. R, M.E., (Ph.D.,)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

SIGNATURE OF THE CANDIDATE

PLACE:

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks to **Board of Management of Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr. Vigneshan. R, M.E., (Ph.D.)** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of our project work.

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

ABSTRACT

The interface allows the user to search for an arbitrary application on the Play Store; the permissions list and the privacy policy are then automatically retrieved, whenever possible. The user has then the ability of selecting a specific permission, and a list of relevant sentences are extracted by the privacy policy and presented to them, along with an accurate description of the permission itself. Such an interface allows the user to quickly evaluate the privacy-related risks of an Android application, by highlighting the relevant sections of the privacy policy and by providing useful information about sensible permissions. We presented a novel approach to the analysis of privacy policies in the context of Android applications. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. The tool is developed with expandability in mind, and further developments in the approach can easily be integrated in order to increase the reliability and effectiveness. In addition, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws.

TABLE OF CONTENT

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	IV
	LIST OF FIGURES	vii
1	INTRODUCTION	01
	1.1 OVERVIEW	01
	1.2 PURPOSE OF ANDROID APP	01
	1.3 OBJECTIVES	02
2	LITERATURE SURVEY	03
3	AIM AND SCOPE	07
	3.1 AIM OF THE PROJECT	07
	SCOPE	07
	ADVANTAGES	07
	3.4 SYSTEM REQUIREMENTS	08
	3.4.1 HARDWARE REQUIREMENTS	08
	3.4.2 SOFTWARE REQUIREMENTS	08
4	SYSTEM DESIGN	08
	4.1 PROPOSED WORK	09
	4.2 ARCHITECTURE DIAGRAM	09

	4.3 ALGORITHMS USED	10
	K-MEANS ALGORITHM	10
	NAÏVE BAYES ALGORITHM	
	CLUSTER ALGORITHM	
5	SYSTEM IMPLEMENTATION	11
	5.1 MODULES	11
	5.1.1 LOGIN/REGISTRATION	11
	5.1.2 APPLICATION DOWMLOAD	12
	5.1.3 ANDROID APPLICATION SCAN	12
	5.1.4 MALWARE DETECTION	13
	5.2 RESULTS AND DISCUSSION	14
6	CONCLUSION AND FUTURE WORK	16
	6.1 CONCLUSION	16
	REFERENCES	17
	APPENDIX	18
	A. SOURCE CODE	18

CHAPTER 1

INTRODUCTION

OVERVIEW

Android software development is the process by which applications are created for devices running the Android operating system Google states that "Android apps can be written using Kotlin, Java, and C++ languages" using the Android software development(SDK), while using other languages is also possible.

The interface allows the user to search for an arbitrary application on the Play Store; the permissions list and the privacy policy are then automatically retrieved, whenever possible .The user has then the ability of selecting a specific permission, and a list of relevant sentences are extracted by the privacy policy and presented to them, along with an accurate description of the permission itself. Such an interface allows the user to quickly evaluate the privacy-related risks of an Android application, by highlighting the relevant sections of the privacy policy and by providing useful information about sensible permissions. We presented a novel approach to the analysis of privacy policies in the context of Android applications. The tool we implemented greatly eases the process of understanding the privacy implications of installing third party apps and it has already been proven able to highlight worrisome instances of applications. The tool is developed with expandability in mind, and further developments in the approach can easily be integrated in order to increase the reliability and effectiveness. In addition, if your app handles personal or sensitive user data, please also refer to the additional requirements in the "Personal and Sensitive Information" section below. These Google Play requirements are in addition to any requirements prescribed by applicable privacy or data protection laws .

PURPOSE OF ANDROID APP

Android SDK. The development teams can use the material design from these SDKs to build interactive apps. However, the developers/development teams are required to pay a one-time registration fee for application distribution

Mobile app development is the creation of software intended to run

on mobile devices and optimized to take advantage of those products' unique features and hardware. The types of mobile apps that developers create include native apps, hybrid apps and HTML5 apps.

OBJECTIVE

- The main goal of this project is to automate efficiently the profiling and detection of android malware.
- With the help of Naïve bayes classifier, Kmeans and clustering algorithm we can analyse and detect it.
- Produce a light weight tool that can be installed on smart phones assisting users in the identification of malware.
- Study of machine learning concepts to apply for the detection of malware as well as research works done using this approach on android.
- Study of the malware landscape on android as well as the recent made to analyse and detect it.

CHAPTER 2

LITERATURE SURVEY

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.

inside the instance of the long sentences. Many proposed models don't appear to be so effective inaccurately grouping long content information. On the opposite side models are consolidating CNN organizations and showing great outcomes which are a result of its capacity of managing more extended content information.

K. Mohammed et al [6], this paper depicts which Recognition of characters enormously relies on highlights utilized. A few highlights of the handwritten Arabic characters are chosen and examined. A disconnected recognition framework dependent on the chosen highlights which was fabricated. Framework was prepared and tried with practical examples of Handwritten Arabic characters. Assessment of significance and exactness of chosen highlights are made. The recognition dependent on chosen highlights given normal exactness of 88% and 70% for the numbers and letters, separately. Further upgrades were accomplished by utilizing highlight freight dependent on bits of knowledge acquired from exactness of individual highlights. G. Pirlo et al [10], introduced another class of participation capacities, which are called Fuzzy membership capacities (FMFs), for drafting based characterization. These FMFs can be handily adjusted to a particular qualities of characterization issue to augment order execution. In this exploration, a real coded hereditary calculation is introduced to discover, in a solitary improvement strategy, the ideal FMF, along with the ideal drafting depicted by Voronoi decoration. Test results, which were completed in the field of Handwritten digit and character recognition, demonstrate which ideal FMF performs better compared to other participation capacities dependent on theoretical level, positioned level, and estimation level weighting models, which can be found in writing.

Author	Title	Goals	Perspective
Yerima, S. Y., Sezer, S., & McWilliams, G. (2014).	Analysis of Bayesian classification-based approaches for Android malware detection	The Bayesian-based classifier consists of learning and detection stages. The learning stage uses a training set of known malicious samples in the wild and another set of benign Android applications, collectively called the app corpus.	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.
Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012, June)	Android Permissions: A Perspective Combining Risks and Benefits	To achieve this, static analysis involves various binary forensic techniques, including decompilation, decryption, pattern matching and static system call analysis.	Analysis of Permission Based Risk signals. In order to get a baseline we first apply the only other mechanism that has been published to identify risk based on permissions, namely Kirin.
Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012, February).	Detecting Repackaged Smartphone Applications in Third-Party Android Marketplaces	Our system adopts a fuzzy hashing technique to effectively localize and detect possible changes from app repackaging.	The results call for the need of a rigorous vetting process for better regulation of third-party smartphone application marketplaces.
Zhang, W., Li, X., Xiong, N., & Vasilakos, A. V. (2016)	Android platform-based individual privacy information protection system.	the Android platform-based individual privacy information protection system architecture and the key execution techniques;	the system could satisfy user functional and non-functional requirements, with stable operation and high task execution efficiency.

CHAPTER 3

AIM AND SCOPE

AIM OF THE PROJECT

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

The main goal of this project is to automate efficiently the profiling and detection of android malware. With the help of naïve bayes classifiers, Kmeans and dclustering algorithm we can analyse and detect it. Produce a lightweight tool that can be installed on smart phones assisting users in the identification of malware. Study of machine learning concepts to apply for the detection of malware as well as research works done using this approach on android. Study of the malware landscape on android as well as the recent progress made to analyse and detect it.

SCOPE

- The proposed system identifies malware with efficiency and effectiveness. The novel techniques enabling effectiveness is the machine learning and data mining to automatically extract features to detect malware on both xml files and byte code.

ADVANTAGES:

- ❖ Security is high

SYSTEM REQUIREMENTS

HARDWARE REQUIREMENTS:

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented

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

SOFTWARE REQUIREMENTS:

Requirement analysis determines the requirements of a new system. This project analyses on product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

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

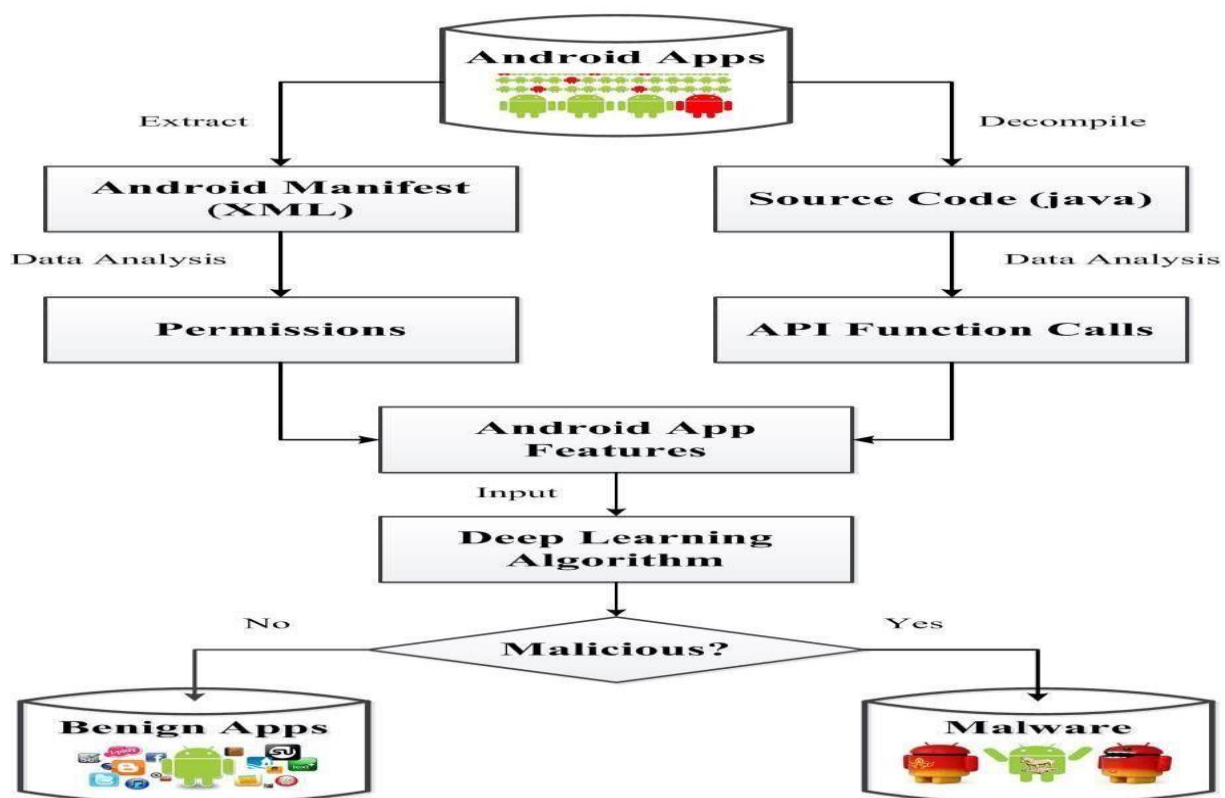
CHAPTER 4

SYSTEM DESIGN

ARCHITECTURE DIAGRAM

Identify the list of third party installed applications. Extract the complete list of permissions of each application. Identify android: protection level of each permission, normal or dangerous of App. Take android app permission dataset. To identify the harmful applications apply classification algorithms. Note the accuracy of spam classification given it by and time required for execution. Results as accuracy among different harmful and normal apps classifiers are analysed The main goal of this

project is to automate efficiently the profiling and detection of android malware. With the help of naïve bayes classifiers, Kmeans and dclustering algorithm we can analyse and detect it. Produce a lightweight tool that can be installed on smart phones assisting users in the identification of malware. Study of machine learning concepts to apply for the detection of malware as well as research works done using this approach on android. Study of the malware landscape on android as well as the recent progress made to analyse and detect it.



ALGORITHMS USED

K-means ALGORITHM :

K-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster

variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

The problem is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum. These are usually similar to the expectation-maximization algorithm for mixtures of Gaussian distributions via an iterative refinement approach employed by both k-means and Gaussian mixture modeling. They both use cluster centers to model the data; however, k-means clustering tends to find clusters of comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.

The unsupervised k-means algorithm has a loose relationship to the k-nearest neighbor classifier, a popular supervised machine learning technique for classification that is often confused with k-means due to the name. Applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means classifies new data into the existing clusters. This is known as nearest centroid classifier or Rocchio algorithm.

NAÏVE BAYES ALGORITHM:

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Cluster analysis:

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters). It is a main task of exploratory data analysis, and a common technique for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, data compression, computer graphics and machine learning.

CHAPTER 5

SYSTEM IMPLEMENTATION

MODULES

- Login/Registration
- Android Application Scan
- Malware Detection
- Android Permission Check

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.

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.

Application Download

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

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.

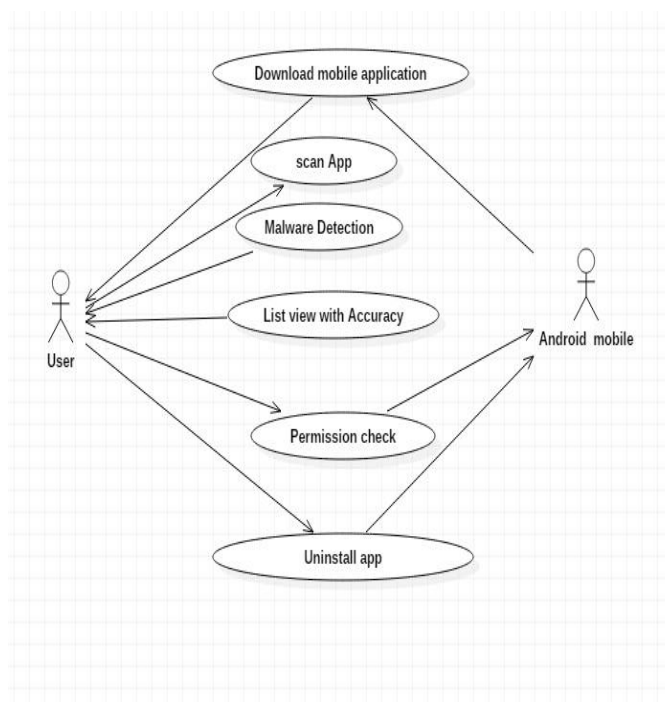
Malware Detection

- 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 kind of threat to mobile devices there should be some security mechanism to be implemented. Analysis of malware has got do with

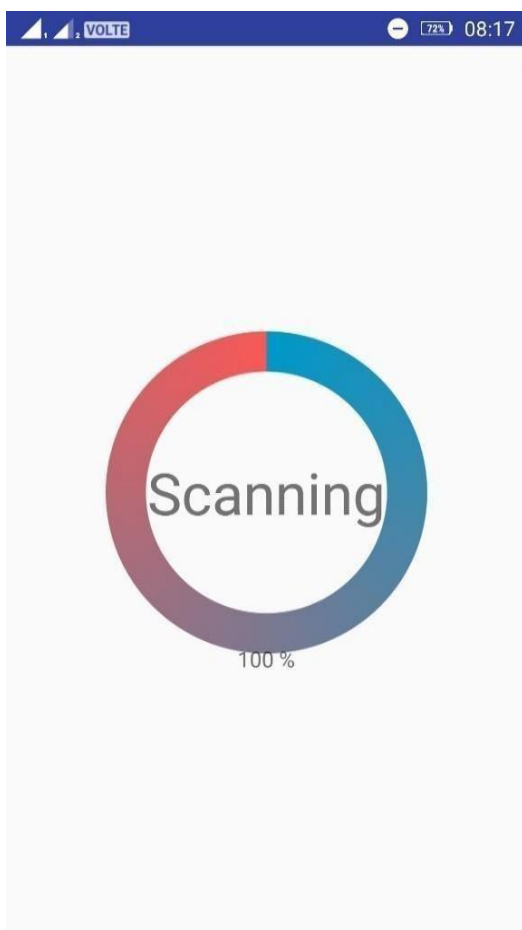
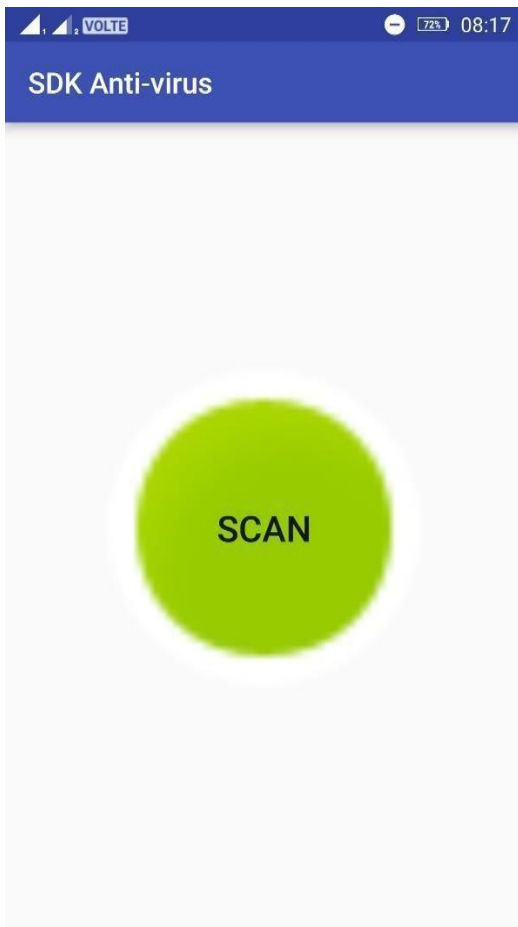
distinctive the instances of malware by completely different classification schemes by exploitation the attributes of well-known malware characteristics. Here also a prototype system named Malware detection system is presented to identify the mobile app's virulence or benignancy. Malware detection has 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 anti virus 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 cannot be detected by antivirus, supported signature-based scanner, however the utilization of applied math binary content analysis of file to detect abnormalous file segments.

USE CASE DIAGRAM

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



RESULTS AND DISCUSSION



CHAPTER 6

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.

REFERENCES

- [1] Yerima, S. Y., Sezer, S., & McWilliams, G. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security*, 8(1), 25-36.
- [2] Sarma, B. P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., & Molloy, I. (2012, June). Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies* (pp. 13-22). ACM.
- [3] Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012, February). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy* (pp. 317-326). ACM.
- [4] Chia, P. H., Yamamoto, Y., & Asokan, N. (2012, April). Is this app safe?: a large scale study on application permissions and risk signals. In *Proceedings of the 21st international conference on World Wide Web* (pp. 311-320). ACM.
- [5] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., & Wang, X. S. (2013, November). 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.
- [6] Zhang, W., Li, X., Xiong, N., & Vasilakos, A. V. (2016). Android platform-based individual privacy information protection system. *Personal and Ubiquitous Computing*, 20(6), 875-884.
- [7] Gartner, "Worldwide sales of smartphones grew 9 percent in first quarter of 2017," <https://www.gartner.com/newsroom/id/3725117>, 2017.
- [8] Symantec, "2018 internet security threat report," <https://www.symantec.com/security-center/threat-report>, 2018.
- [9] S. Website, "Internet security threat report," Cupertino, California, Symantec Website Technical Report, 2015.
- [10] W. Enck, M. Ongtang, and P. McDaniel, "Understanding android security," *IEEE security & privacy*, vol. 7, no. 1, pp. 50-57, 2009.
- [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).

Appendix

A. Source code

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

```

ACTIVITY THIRD JAVA

```

package com.example.kevinwu.project;

import android.app.Activity;
import android.app.ListActivity;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

```

```

import android.util.Log;
import android.widget.AdapterView;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.TextView;
import android.view.View;
import java.math.*;
import java.util.*;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import android.widget.AdapterView;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.net.Uri;
import java.io.BufferedReader;
import android.widget.AdapterView.OnItemClickListener;
import java.io.FileReader;
public class activity_third extends ListActivity {
    LoginDataBaseAdapter loginDataBaseAdapter;
    private TextView Textresult;
    ArrayList<String> listItems=new ArrayList<String>();
    ArrayAdapter<String> adapter;
    Button b1;
    utile u=new utile();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_activity_third);
        b1 = (Button) findViewById(R.id.button2);

```

```

b1.setOnClickListener(
    new View.OnClickListener(){
        @Override
        public void onClick(View v){

            Intent intent = new
Intent("com.example.kevinwu.project.Main3Activity");
            startActivity(intent);

        }
    }
);

// db.setTransactionSuccessful();
// db.endTransaction();
// loginDataBaseAdapter.insertReportEntry(Pass,
Repass,Secu,mob);
ListView lv = getListView();

// on seleting single product
// launching Edit Product Screen
lv.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        // getting values from selected ListItem
        String val =(String) parent.getItemAtPosition(position);
        String[] vt=val.split(" ");
    }
});

```

```

        Uri packageUri = Uri.parse("package:com."+ vt[0]);
        Intent uninstallIntent =
            new Intent(Intent.ACTION_DELETE, packageUri);

        uninstallIntent.putExtra(Intent.EXTRA_RETURN_RESULT, false);
        //new Intent(Intent.ACTION_UNINSTALL_PACKAGE,
        packageUri);
        startActivity(uninstallIntent);

    }
});

```

```

TextView result = (TextView) findViewById(R.id.textView2);
adapter=new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1,
    listItems);
setListAdapter(adapter);
Intent in = getIntent();
Bundle bun = in.getExtras();

```

```

//show list of possible malwares if mcount != 0
if(bun.getInt("mcount")!= 0){
    String detected = "Possible malware detected:";
    result.setText(detected);
    String [] posList = bun.getStringArray("list");
    for(int i = 0; i < posList.length; i++) {
        if (posList[i] != null) {
            Log.d("Test:", i + " " + posList[i]);
            //add items into the listView
            test t=new test();

```



```

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();
}
}

```

DATABASEHELPER

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

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.util.Log;

public class DataBaseHelper extends SQLiteOpenHelper {
    public DataBaseHelper(Context context, String name,
        CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase _db) {

        _db.execSQL(LoginDataBaseAdapter.CREATE_LOGIN_TABLE);

        _db.execSQL(LoginDataBaseAdapter.CREATE_ADMIN_TABLE);

        _db.execSQL(LoginDataBaseAdapter.CREATE_REPORT_TABLE);

        _db.execSQL(LoginDataBaseAdapter.CREATE_Date_TABLE);
    }
}
```

```

        _db.execSQL(LoginDataBaseAdapter.CREATE_Voted_TABLE);

        Log.w("table Created","");
    }

    @Override
    public void onUpgrade(SQLiteDatabase _db, int _oldVersion,
int _newVersion) {
        // Log the version upgrade.
        Log.w("TaskDBAdapter","Upgrading from version " +
_oldVersion + " to " + _newVersion + ", which will destroy all old
data");
        _db.execSQL("DROP TABLE IF EXISTS " +
"TEMPLATE");
        onCreate(_db);
    }

```

KMEANS

```

package com.example.kevinwu.project;

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

public class KMeans
    1. {

```

```

//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;
    }
}

}

}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

```

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;
}
}
```