بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

Islamic University – Gaza
Deanery of Post Graduate Studies
Faculty of information technology

الجامعة الإسلامية – غزة
عمادة الدراسات العليا
كلية تكنولوجيا المعلومات

Malware Detection Based on Permissions on Android Platform Using Data Mining

By:

Mohammed I. Nasman

Supervised By:

Dr. Tawfiq S. Barhoom

Mar, 2016

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master in Information Technology

إقـــــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Malware Detection Based on Permissions on Android Platform Using Data Mining

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification

Student's name:	اسم الطالب/ة: محمد إسحاق نسمان
Signature:	التوقيع: محمد
Date:	التاريخ: 15/ 06 / 2016

بسم الله الرحمن الرحيم

الجامعة الإسلامية – غزة
The Islamic University - Gaza

مكتب نائب الرئيس للبحث العلمي والدراسات العليا هاتف داخلي: 1150

الرقم Ref ج س غ/35/.....

التاريخ Date 2016/03/30.....

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة شئون البحث العلمي والدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **محمد اسحاق إبراهيم نسمان** لنيل درجة الماجستير في **كلية *تكنولوجيا المعلومات*** **برنامج تكنولوجيا المعلومات** وموضوعها:

استكشاف البرامج الخبيثة اعتمادًا على الصلاحيات على منصة أندرويد باستخدام تنقيب البيانات

Malware Detection Based on Permissions on Android Platform Using Data Mining

وبعد المناقشة التي تمت اليوم الأربعاء 20 جمادى الآخر 1437هـ، الموافق 2016/03/30م الساعة الثانية مساءً، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

د. توفيق سليمان برهوم	مشرفاً و رئيساً	.....
أ.د. علاء مصطفى الهليس	مناقشاً داخلياً	.....
د. تامر سعد فطاير	مناقشاً خارجياً	.....

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *تكنولوجيا المعلومات*/ برنامج تكنولوجيا المعلومات.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق،،،

نائب الرئيس لشئون البحث العلمي والدراسات العليا

أ.د. عبدالرؤوف علي المناعمة

Acknowledgements

First and foremost I thank “Allah” for all the health and power that led me to grow and to learn till finish this step in my life.

This thesis is the result of years of work whereby I have been accompanied and supported by many people. It is wonderful that I now have the opportunity to express my gratitude to all of them.

I'd like to thank everyone who has helped me in completing this work, starting from my thesis advisor Dr. Tawfiq S. Barhoom, who helped me in through all the program and thesis, to Prof. Alaa Al Halees, his courses were very helpful thought my study and my work, and to all academic stuff in the Information Technology program at Islamic university of Gaza.

Also, I would like to take this opportunity to express my sincere gratitude to my beloved family, especially my father, mother, wife, my lovely kids (Sohaib, Batool, Sara, Ahmed), brother and sisters - without whom I would never have been able to achieve so much.

I am so grateful to two of my friends: Fadel O. Shaban and Emad Kehail, for their help and support through all the master courses and thesis.

Last but not least, I want to thank my colleagues, for their support during my years of study.

Abstract

With the spreading of smart mobile devices to nearly every person, Android Operating System is dominating the mobile's operating systems.

Due to the weak policy of submitting application to Google Play store, attackers developed malware to attack the users of the Android operating system with malware application or by including malicious code into applications.

Researches have been done in this area, but solutions required installing the applications to monitor the malware behavior, or by taking actions after installing the application.

We proposed a new method using Data Mining to detect newly and unknown malware using the applications' permissions as base features.

In order to create binary dataset we collected from benign and malware android app samples, the dataset consist of five different features collected based on different number of attributes and conditions.

Different evaluation measure used to evaluate the proposed method, the results show that we achieved 96.74% with f-measure and 99.3% with area under the Receiver Operating Characteristic ROC curve.

Keywords: Malware, Android, Data mining, Permissions, APK, Classifications.

الملخص

إكتشاف البرامج الخبيثة على نظام تشغيل أندرويد من خلال الصلاحيات بإستخدام تنقيب البيانات

مع تسارع إنتشار استخدام أجهزة الهواتف الخلوية الذكية لتصل تقريباً لجميع سكان الكرة الأرضية، يعتبر نظام تشغيل أندرويد النظام الأكثر إستخدماً في هذه الهواتف الذكية.

لكن مع ضعف السياسة المتبعة من شركة جوجل لإستقبال البرامج الخاصة بالهواتف الخلوية لنظام أندرويد على متجرها، أدى ذلك إلى جعله فريسة سهلة للمهاجمين الذي يبنون برامجيات خبيثة أو برامج عامة تحتوي بداخلها على أكواد خبيثة تضر المستخدمين أو تنتهك خصوصيتهم.

تمت دراسات عديدة في هذا الجانب من قبل العديد من الباحثين، لكن الحلول المتبعة تعتمد في أغلبها على تنزيل برامج من أجل فحص البرامج لمعرفة إذا كانت نظيفة أم خبيثة، أو من خلال قيام هذه البرامج بمراقبة أفعال البرامج ، لكنها في مجملها لا تمنع المستخدم من تنزيل برامج خبيثة جديدة وإكتشافها قبل ما يتم تنزيلها من قبل المستخدم.

في هذا البحث قمنا بتقديم طريقة جديدة بإستخدام تنقيب البيانات لكشف البرامجيات الخبيثة والتي لم تكتشف من قبل وذلك إعتماداً على قراءة الصلاحيات المستخدمة من قبل البرمجيات.

قمنا ببناء قاعدة بيانات مكونة من 103 برنامج نظيف وخبيث، وتم تقسيمها إلى مجموعات تختلف بإختلاف الصفات بناء على مجموعة من الشروط المذكورة ضمن البحث.

وقد وصلت دقة النتائج إلى أن الطريقة المقترحة حققت دقة لتميز البرامج الخبيثة والنظيفة وصلت إلى 96.74% للمصنف حسب مؤشر قياس ال F-Measure و0.993 لمؤشر ال AUC.

الكلمات المفتاحية: برامج خبيثة، تنقيب البيانات، أندرويد، الصلاحيات، التصنيفات.

Contents

List of Tables

List of Figures

List of Equations

List of Abbreviations:

APK - Android Application Package.

XML – Extensible Markup Language.

DEX – Dalvik Executable Format.

DDOS – (Distributed Denial of Service).

VM – Virtual Machine.

SDK – Software Development Kit

NDK - Native Development Kit

JAR - Java Archive.

AAPT – Android Asset Packaging Tool.

OS – Operating System.

ROC - Receiver Operating Characteristic

SVM – Support Vector Machines

kNN – K-Nearest Neighbor

NB – Naïve Bayes

AUC – Area under the Curve

EULA - End User License Agreements

SSD – Solid State Drive

RM – Rapid Miner

PE – Portable Executable

Chapter 1
Introduction

1.1. Introduction

With the repaid growing of Android application every day, there are growing threats for the mobile users by installing more malwares without ability to detect them before installing the applications to the user device.

Malware name came from “Malicious Software”, its software designed to secretly access a system without the owner's device knowledge.

Malware can effects mobile resources, or just make the devices not responding to the users, it may go to dangerous behaviors like steal private information, without the user notice any harmful action [**1**].

According to data from the International Data Corporation (IDC) the worldwide smartphone market grew 27.2% year over year in the second quarter of 2014, just over a third of a billion shipments at 335 million units.

2014 promises to close at nearly 1.3 billion shipments, with Android taking the lion's share, spread across over 180 tracked vendors [**2**].

Market research firm Strategy Analytics has given the numbers for the second quarter of 2014 that estimate the market share of Android platform’s on the global market has reached 84.6 percent. [**3**]

For the mobile devices that use Android as its platform, the official way to install the applications is the Google play store [**4**], which serve as repository of the application developed for Android, and it installed by default with all the Android devices.

The current reviewing process for the applications submitted by developers to Google Play store took only two hours [**5**], compared to process for Apple AppStore takes 6 days [**5**].

Google may phase out the discovered malware but after it's spreading, for example: More than 50 applications on Google's Android Market have been discovered to be infected with malware called "DroidDream" which can compromise personal data by taking over the user's device, and have been "suspended" from the store [**6**].

Currently mobile malware detection tools uses pattern recognition to identify the malware, but it fails to distinguish the threats.

Android gives accessing to the device's resources(such as writing files, accessing the internet, locations, SMS, etc.), with permissions system, which they defined on each Android Application Package (APK) in special file called "AndroidManifest.xml".

Any application needs to access any of these resources will define the resources required on "AndroidManifest.xml" on development time, after the application compiled and uploaded to Google play Store, it will show to the users the permission required for installing the application.

But with lack to understanding and knowledge for most of users, they can install the application that has access to special resource and it may be has a harmful use.

According to above, the need to a new method to recognize the malware applications before installed by the users is important to prevent the malware attack their mobile resources and Data.

This research focus on new method for detecting Malwares based on permissions required by the applications, using classification techniques to detect malware apps from benign.

1.2. Problem Statement

Anti-malware usually can detect the known malwares based on some techniques, which mostly are signature base, but these techniques are ineffective to detect new malwares.

Most Android phone users using Google play which is the main provider for Android OS's to install Apps to their phones.

Due to the easy and weak policies used from Google Play for adding Apps to their repository, it allows malware Apps to be added and not detected during the submit period.

The problem with current methods of detecting the malware, it's detect after it's installed on the mobile and may it harm users before it's can be detected.

There's an urgent need to have mobile malware detection method capable to distinguish the unknown malware that's not previously detected by other methods, before installing the application into user mobile.

1.3. Objectives

1.3.1. Main Objectives:

To propose a new method to use the applications' permissions to detect malware before installing them from Google play store.

The proposed method is based on Data Mining techniques.

1.3.2. Specific Objectives:

- Review the current malware detection techniques.
- Review the Android Malware and benign applications to collect samples applications.
- Study the Android application structure to extract the permissions from APKs.
- Building a dataset that consist of permissions extracted from malware and safe applications.
- Find appropriate classifiers to build the method.
- Apply the proposed method based on selected classifier.
- Testing & evaluating the performance of the proposed method.

1.4. Scope and Limitations:

- The method will developed on Android Platform only and will cover versions from 2.3 to 5.1.
- The method will be based on Classification.
- The method will use a dataset and will not interact with applications on Google play store [**4**] or other sources.
- The collecting for benign application will comes from Google play store.

1.5. Importance of the research:

- Installing malware applications on the mobile device is very dangerous for users, because it may steal their private data, or affect the performance of the device, therefor we need to find a method to protect them.
- Add a significant contribution to scientific research in the field of finding effective solutions in worms' detection.
- Find a better way to detect the malware before installing them to user's device.
- Evaluate multiple classifiers to find a better classification method to classify the malware in high accuracy.

.

1.6. Methodology

In this section we talk about the proposed methodology to achieve the work, as following steps:

1.6.1. Data set collection and preprocessing:

1.6.1.1. Collect sample Apps for the dataset:

Dataset will contain numbers of malware and benign applications, the Malware dataset that contain malwares for android platform as APK files and downloaded from "Free Range Security" website [**7**].
benign applications will be downloaded from Google Play Store [**4**].

1.6.1.2. Data processing:

We will extract the permissions from the benign and malware applications, and store them into text files, we will build a tool to automate this job, and then we will use official Google's Permission that used on Android OS as base attributes.

Based on the exported text files that contain the application's names and the extracted permissions, another tool will be implemented to store all the required data into database file to be used as the Dataset.

1.6.2. Applying the classification:

To distinguish the Malware applications from the dataset, we will use Classification method such as naïve bayes, kNN,etc [**8**].

1.6.3. Generate the results:

Based on the experimental processing in the previous steps.

1.6.4. Evaluation:

We will use different evaluation measure to evaluate the performance from different views.

1.7. Thesis Organization

The thesis is divided into six chapters, chapter one include the introduction, chapter two provides Literature Review, chapter three provides the some related work of malware and spyware detection, chapter four provides description about the proposed methodology including data collection, and feature extraction and selection, chapter

five talks about experiments and results, and the analysis of the experimental results. Future work will be reviewed in chapter six.

Chapter 2
Literature Review

In this chapter we will review the Android OS, and it's VM, the content of APK files and the permission used with Android application, later we will talk about malware types and the detection methods, then we will review different types of malware and the anti-virus techniques used to detect the malware, later we will review the data mining and it's and the performance evaluation.

2.1.Android Operating System:

Android is an open source operating systems for Mobile & smart devices, originally was founded on 2003 by Andy Rubin, Rich Miner, Nick Sears and Chris White, as an operating systems for digital cameras, but later due to lake of market for this type of devices, they start to target it into Mobile devices [**9**].

On 2005 the company was bought by Google, Google worked to enter it into the mobile OS game, since then it's start to be very popular operating system.

On 2011 Android applications on Google play store has more application than Apple AppStore [**10**].

Currently Android has more than 80% of total operating systems market share [**11**]. (Figure 2.1), and many of large Mobile manufactures uses it as main operating systems such as: Samsung, High Tech Computer Corporation (HTC), Huawei, etc.

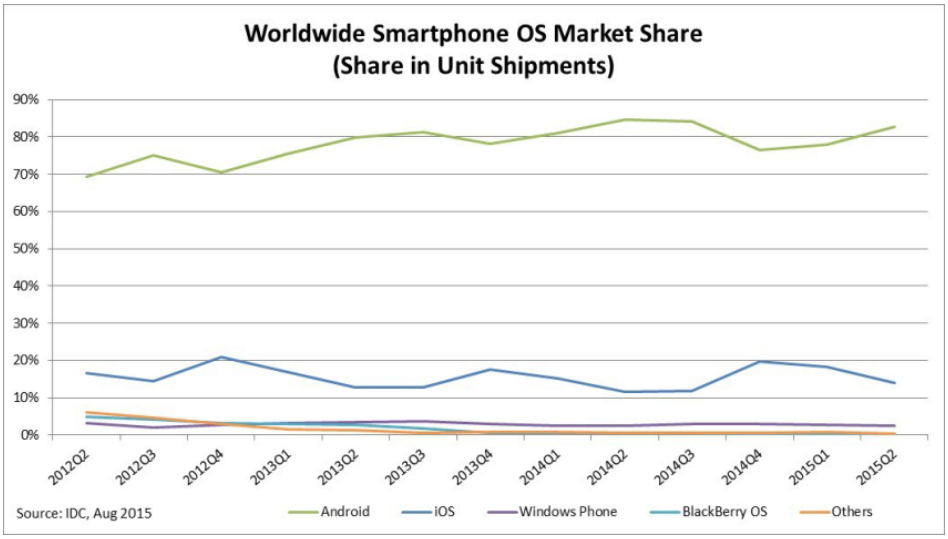


Figure 2.0.1 Mobiles OS Market share [11]

2.2.Dalvik Virtual Machine:

Dalvik is Virtual Machine (VM) that executing the .Dex files that generated form Java Byte code, it designed for specifically mobile devices to run on slow CPUs with little RAM compared to Java VM which target desktop [**12**], which require higher resources than mobile devices.

Applications for Android built using Java, but with special compiler that convert the Bytecode to Dalvik code (.DEX) instead of JAR file which used with Java VM [**13**] , as shown in figure 2.2.

Google release NDK to allow the C & C++ programmers to create native android application without converting the source code into Dalvik code, but to machine language specific for each CPU, this allow to create high performance application but with more complexity in the development, it's mostly used for game engine and games development.

On our research we will work on APK files that work under Dalvik VM.

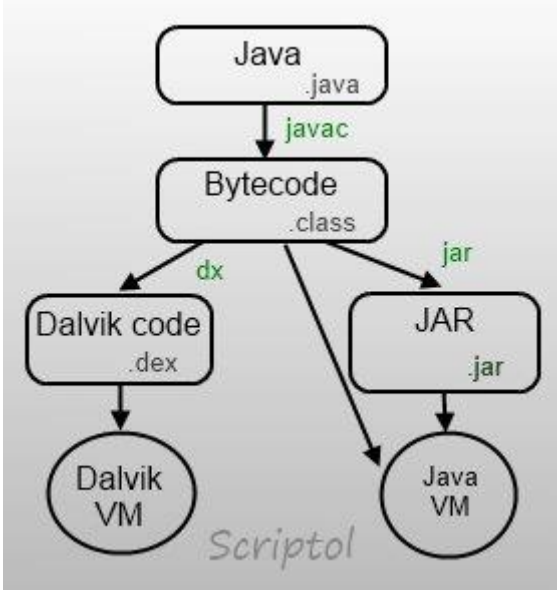


Figure 2.2 Java code to Dalvik VM [14]

2.3.Android Application Package (APK).

Android application package APK is package file format used by Google for the Android Operating System for the distribution of the mobile apps on the Android platform [**14**].

APK files is the executable file that run on the Dalvik Virtual machine, it's similar to .exe files on windows or deb packages on Debian based Linux distribution such as Ubuntu.

APK files is compressed files using zip format based on Jar file format, it could be opened by any Archive tools such as: winrar, 7zip, etc (see figure 2.3).

The APK file consists of multiple folders and files, that contain the different binary version of App (to run on different CPUs), resources required and AndroidManifest.XML as binary file.

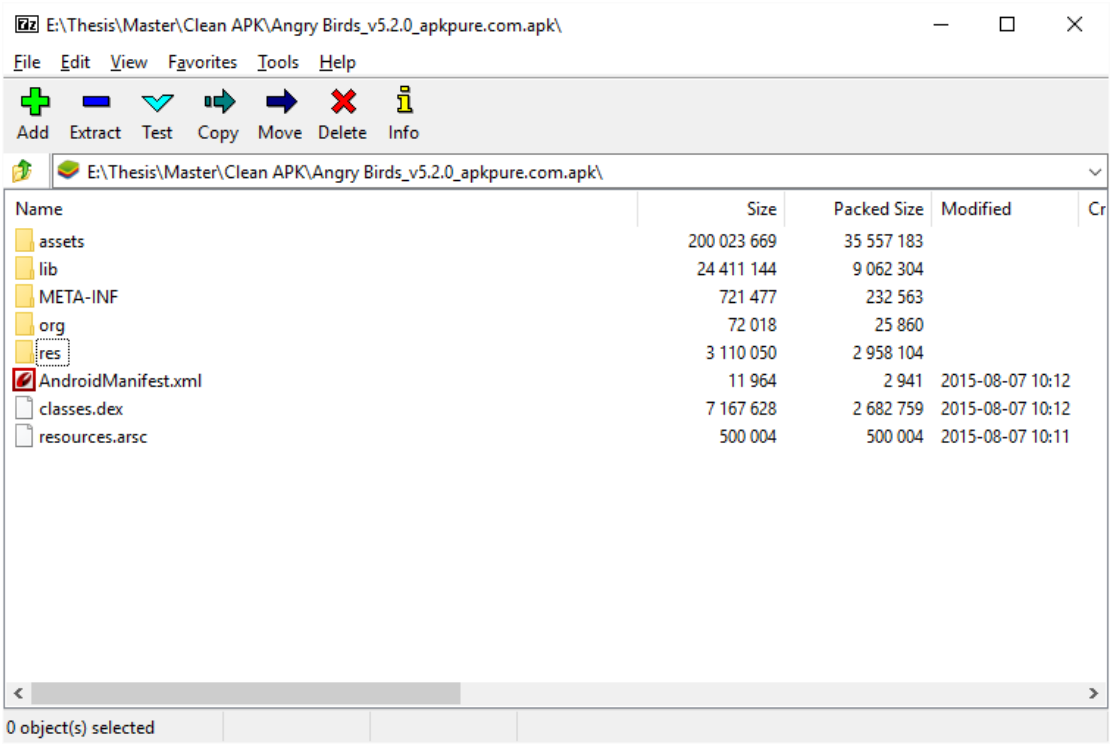


Figure 2 Error! No text of specified style in document.**.3 Angry Bird APK File opened in 7zip**

Usually APK file contain 6 items as following [**15**]:

1. META-INF folder: which contain MANIFEST.MF file, the certificate of the App (*.RSA) and (*.SF) file which is contain all the files with their checksum in SHA1(figure 2.4).

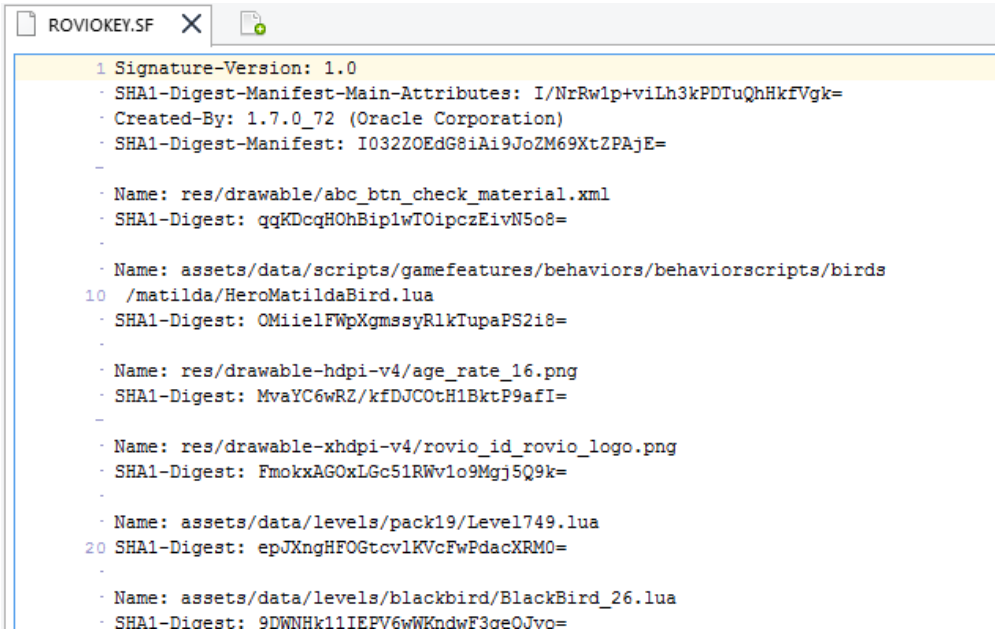


Figure 2.4 content of *.SF file

2. Lib folder: This folder contain the compiled in specific binary format for different processors, it may contain one or more of the format described in table 2.1 [**14**]:

Table 0 Description of Lib Folder

Format	**Description**
armeabi	compiled code for all ARM based processors only
armeabi-v7a	compiled code for all ARMv7 and above based processors only
arm64-v8a	compiled code for all ARMv8 arm64 and above based processors only
x86	compiled code for x86 processors only.
x86_64	compiled code for x86_64 processors only
mips	compiled code for MIPS processors only

3. Classes.dex: the classes compiled in .dex format code for Dalvik Android.
4. resources.arsc: contain the compiled resources of the application such XML files.
5. Res Folder: contain the resources that not compiled in binary format, such as images, icons, etc…
6. AndroidManifest.xml: additional manifest file that contain the file name, version and access rights required to run the App, it's distributed as binary format.

2.4.Android Permissions :

Applications that runs under Android operating system can't access some specific devices hardware sensors or features such as: location, camera, network or even access for the storage outside the allocated storage for the application.

On the contrary on Windows any application may access other files and drivers located on the same computer without notifying the users.

On Android if there's a special permission required it should asked from the operating system, and the application will define the required permission on the "AndroidMainfest.xml", and it will be shown to the user before it's installed on the user's device.

The permission required for the application to work properly, or it may work but some functions will not work without getting the permission required, for example Google Map: will require "ACCESS_FINE_LOCATION" permission and "INTERNET", without

"INTERNET", the application will not work at all, but if the application has active connection to the internet it will work, but for detecting the user location on the map, it will not work without turn on location setting on the mobile.

Needless to say, when you install the Google Maps it will ask for the two permissions before installing with other required permission.

Usually these permission needed to the application to work properly and to give the users all the features built in it, but some of the malware ask for permission not needed to work, but needed by malware to steal some information or access unauthorized files stored on the mobile storage.

Permissions on android could be on of "136" permission [**16**], any application require any of these permission, will add it when it's under development, and will be prompted to user on the install time.

2.5.Dangerous Permissions

Google has specified some of the permission as dangerous permissions [**17**] as listed on table 2.2, because these permissions allow the application to access private data of the devices without users permissions of he/she allowed the application to be installed, also some malware may use these features for making calling to sending SMS to international numbers, which may cost users money without he notice that.

Table 0 Dangerous permissions and permission groups

Permission Group	Permissions
CALENDAR	• READ_CALENDAR • WRITE_CALENDAR
CAMERA	• CAMERA
CONTACTS	• READ_CONTACTS • WRITE_CONTACTS • GET_ACCOUNTS
LOCATION	• ACCESS_FINE_LOCATION • ACCESS_COARSE_LOCATION
MICROPHONE	• RECORD_AUDIO
PHONE	• READ_PHONE_STATE • CALL_PHONE • READ_CALL_LOG • WRITE_CALL_LOG • ADD_VOICEMAIL • USE_SIP

	• PROCESS_OUTGOING_CALLS
SENSORS	• BODY_SENSORS
SMS	• SEND_SMS • RECEIVE_SMS • READ_SMS • RECEIVE_WAP_PUSH • RECEIVE_MMS
STORAGE	• READ_EXTERNAL_STORAGE • WRITE_EXTERNAL_STORAGE

2.6.Data Mining:

Data mining is the process of different queries and getting the useful information that's not previously known or unexpected [**18**].

Data mining is "refers to the nontrivial extraction of implicit, previously unknown and potentially useful information from data in databases" [**19**]

While data mining and knowledge discovery in databases (KDD) are usually treated as substitutes, data mining is a part of the knowledge discovery process.

Data mining can be separated into two general categories [**20**]:

- **Predictive**: implicates predicting unknown data based on given dataset.
- **Descriptive**: implicates finding patterns explaining the data.

Data mining is also an essential part of knowledge discovery (KDDs), including these steps [**21**]:

2. **Data cleaning**: Known as cleansing, these step remove the noise and outlier from the data.
3. **Data integration:** This step combines data from multiple and different sources into one database.
4. **Data selection**: This step the user will reduced representation of the dataset to reduce the size of original data but with keeping the integrity.

5. **Data transformation**: Data can be transformed into other suitable formats.
6. **Data mining**: Data Mining tools are applied to discover possibly useful patterns.
7. **Pattern evaluation**: Using validation measures to recognize interesting and useful patterns.
8. **Knowledge representation**: The final phase of the knowledge-discovery process, which is presented to the users in visual forms.

2.7.Data Mining Classifications methods:

In our research we evaluating a variety of classification methods such as: k-Nearest Neighbor(kNN), Naïve Bayes, Support Vector Machine (SVM) and Decision Tree, we used these classifications methods with different feature sets.

2.7.1. k-Nearest Neighbor(kNN):

kNN is one of oldest nonparametric classification algorithms, it's find groups of K objects in training set that are close to test object, the value of K usually find by using cross-validation [**22**].

2.7.2. Naïve Bayes:

Naïve Bayes classifier based on Bayes' theorem, one of the main advantages of NBC is it doesn't require large dataset of training set to find the means and variances of the variables needed for classification *[**23**]*.

2.7.3. Support Vector Machine (SVM):

Support Vector machine is supervised learning methods that analyze data and recognize patterns, it's used for classification and regression analysis [**22**].

2.7.4. Decision Tree:

Decision tree is one of the supervised learning algorithms that follow the "Divide and conquer" approach to solve the problem by learning from autonomous cases [**24**].

The structure of tree includes: root node, branches and leaf, each node represent a test for an attribute, and the branch fork

the result, and each leaf node represent a class label [**25**] as shown in figure 2.5.

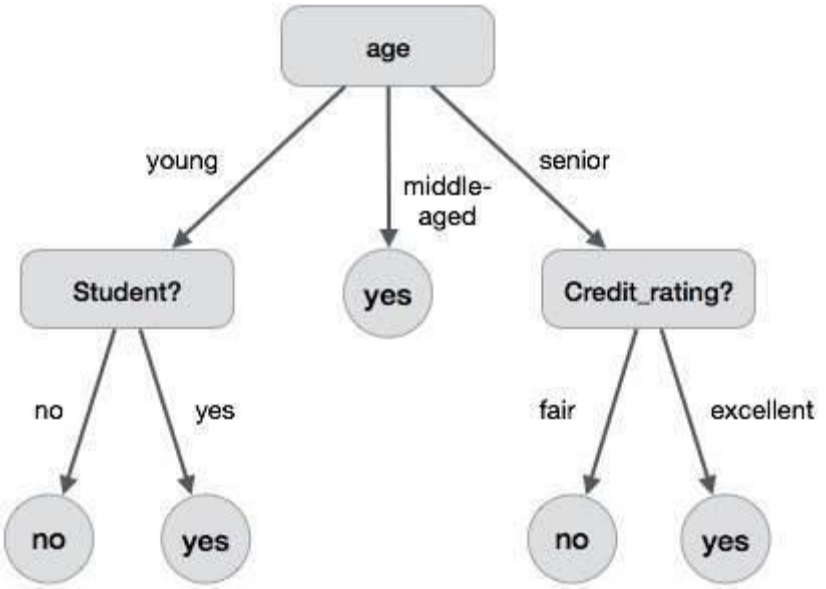


Figure 2.5 Decision tree [43]

2.8.Performance Evaluation

2.8.1. Coincidence Matrix

For the classifications problems the main source of performance measurement is the coincidence matrix see figure 2.6 (also known as classification matrix or a contingency table) [**26**]

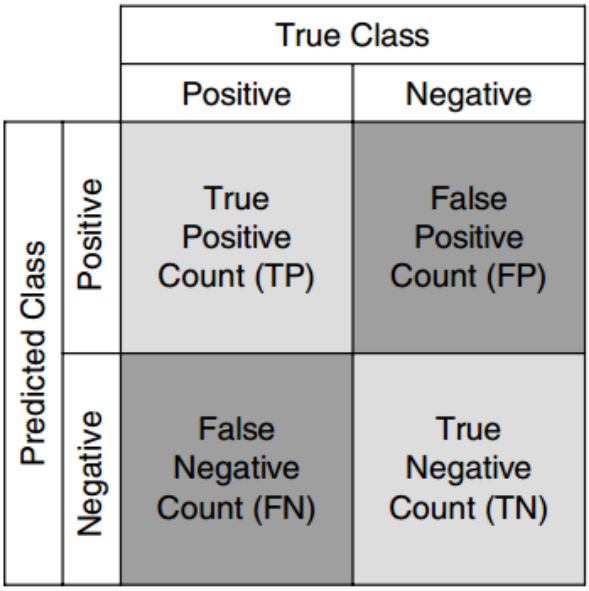


Figure 2.6 coincidence matrix

we can calculate most commonly used metrics equations from coincidence matrix as shown in eq 2.1, eq 2.2, eq 2.3, eq 2.4, eq 2.5.

$$\text{True Positive Rate} = \frac{TP}{TP + FN}$$ Eq. 2.1

$$\text{True Negative Rate} = \frac{TN}{TN + FP}$$ Eq. 2.2

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$ Eq. 2.2

$$\text{Precision} = \frac{TP}{TP + FP}$$ Eq. 2.3

$$\text{Recall} = \frac{TP}{TP + FN}$$ Eq. 2.4

2.8.2. Accuracy

Accuracy is the percentage of true results (true positives or true negatives) between the total number of cases examined *[**27**]*

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}}$$

Eq. 2.6

2.8.3. Precision

Precision is the correctly retrieved instances of query [**28**].

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

Eq. 2.7

2.8.4. Recall

Recall is part of the documents that relevant to the query that have been successfully retrieved *[**28**]*.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Eq. 2.8

2.8.5. AUC

Receiver operating characteristic is created by comparing the true positive rate (TPR) against the false positive rate (FPR) at various sill settings.

The ROC recently introduced to evaluate ranking performance of machine learning algorithms [**29**]

The AUC combine all of the features of ROC into single value, by calculating the area of inclination shape below the ROC, the closer ROC get into optimal point of prediction, the AUC gets closer to one *[**30**]*.

$$FPR(\theta) < FPR(\theta') \implies \theta > \theta' \implies TPR(\theta) \leq TPR(\theta')$$ Eq. 2.9

2.8.6. F-Measure:

F-Measure considered as weighted average between precision and recall, it's calculated as see eq 2.1:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$ Eq. 2.10

2.8.7. Cross Validation

In k-fold cross-validation, initial data are indiscriminately divided into k reciprocally exclusive subsets or "folds", D1, D2, …, DK, each one is approximately same size, training and testing done k times, in loop i , division Di, is set for test set, and the other divisions are used to train the model, and then for other Di, until DK (see figure 2.7).

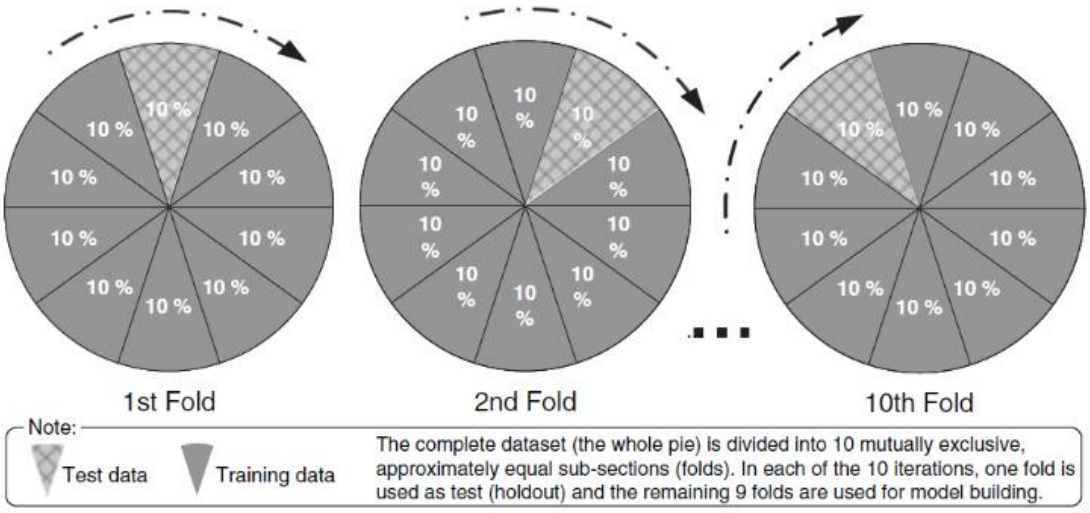


Figure 2.7 10 Fold Cross Validation

2.8.8. Identification methods for the malware:

Mainly the malware detection techniques fall into these categories:

- **Signature based detection:** It's search for sequence of unique bytes that defined the malware, and compare it to the database of other malware data, most of Anti-Malware use this technique [**31**].

- **Behaviors based detection**: By monitoring many factors of the malware such as the source, target and other statistical properties, then evaluating the damage of the system under controlled environment using dynamic behaviors. [**31**]

2.9.Malware Detection types:

The detection methods used to find the malware are categories into two categories: static and dynamic [**32**], on the static analysis, it's uses the source code or the binary format of the application (Machine code or Byte Code) to find a pattern, on the dynamic approach, it uses the behavior of the application on runtime to find if it a malware or not.

2.9.1. Antivirus:

Antivirus is a software that detect, prevent and delete the malware application form the computer and mobile devices, they are different malware types such as: Keylogger, Spyware, Trojans and worms [**33**]

2.9.1.1. Signature Based Detection:

Most of Antivirus uses signature-based detection to find the virues infected in the system, virus has special digital signature, which antivirus has a database for all known virus based on their digital signature, for that antivirus using this method require regular updates to download new signatures for newly detected virus by the developing company.

2.9.1.2. Heuristics

Some Antivirus uses the heuristic methods to detect the new variants of existing malware or new unknown malware.

Heuristic algorithms uses virus signature databases to detect malware by verifying not exact signatures of the malware, also it read the files to find random combinations of signatures to find malware.

2.9.1.3. Rootkit Detection

Rootkits are some type of malware that has the ability to alter the operating system to hide itself from the Antivirus software, also it may alter the Antivirus itself to hide from it. Powerful Antivirus may detect this type of malware but with limited success.

2.9.1.4. Real Time Protection

Most antivirus software provides real-time protection, often under any number of clever synonyms such as resident shield, background guard, autoprotect, and so on. The real-time protection feature of these antivirus programs monitors all activity in a computer for processes and activities that appear suspicious. This is done at all load and read times; any time a file is accessed, loaded, or downloaded. This feature helps protect against malware that has made it onto the computer system but has not yet been activated.

2.10. Types of Malware:

Malware has different types, PCs and Mobiles has the same types, which can be listed on different categories such as: spyware, virus, worms and more [**34**]:

2.10.1. Adware

Adware is special type of malware that work Advertisement Software, and during use the software it may show some

windows for the Ads, on starting of the software or open links to specific Ads websites.

2.10.2. Bot

Bots are software that act as rebotect to do a special operations, these operations may be used maliciously such as DDOS attacks, or spamming.

2.10.3. Bug:

Bug isn't a malware itself, but it's usually may cause a big problem by freezing or crashing the software or the entire operating system.

Bug is a flaw of the software that produce unwanted behavior of the functionally it should doing.

Some bugs may cause a security leaks which will allow the hackers to gain access to devices without permissions to steal the data or cause damage to the device or the entire network.

2.10.4. Rootkit

A rootkit is a special type of malware that infect the operating system and modify it to be hidden from the Antivirus or the operating system itself.

2.10.5. Spyware

Spyware is one type of malware that spying on user device and monitor and store his activity such as keystrokes, login information or credit cards numbers.

Some type of spyware may be installed with normal software and hide itself as Ads programs to support the software writer.

2.10.6. Trojan Horse

A Trojan horse or a "Trojan," is kind of malware that hide itself with other programs to installed itself with that software without informing the user.

Then it allows the attacker behind it to gain access to the device, and then it will have full access of the device and the files.

Also attacker can get screenshot of the victim device and may install key logger or other harmful tools.

2.10.7. Virus

A virus is a type of malware that can spread by coping itself to other hard disk partitions, USB storage or other computer on the network.

Then it can do harmful operations in the infected device such as: deleting files, formatting H.D, stealing information or just annoying users by showing some unwanted behaviors.

2.10.8. Worm

Worms are common types of malware. They spread on networks by utilizing the operating system vulnerabilities. They can consuming the bandwidth or overloading web servers.

Worms may classified as type of virus, but the main difference worms can spread itself without any human interaction (by using the vulnerability of OS) while virus need to be spreading using other files or by hide itself on special hard disk sectors.

2.11. Summary:

In this chapter, we presented the android operating system, and its virtual machine Dalivk, then we talked about the Android Application Package APK and what's the content inside it.

After that we review the android permissions and the dangerous permissions specified by Google.

then reviewed the data mining and it's categories, and the knowledge discovery part of the data mining.

malware detection techniques, which is signature based or behavior based, then we talked about the.

Finally we reviewed the different types of malware.

Chapter 3
Related Work

In this chapter, we will review previous studies in subject of malware detection, researches used different approaches to detect the malware, some of the methods require process on the mobile devices, and other methods do the processing on the cloud from the data collected on the mobile device.

3.1.Android Malware:

Sanz et al [**35**], presented detection method using string analysis that will get the strings from android application by disassembling the Android application and then extract the strings in const-string and using machine learning to training the dataset and assign category (malware or goodware).

The problem with method, that developers of malware may using non English languages in const-strings will not make them detectable by this method, also if the developer of the malware application encrypt the strings, they will not be detectable in this method too.

Burguera and Zurutuza [**36**], have developed a framework for detecting malware on android platform, the framework consist of multiple components: Data acquisition which using application developed "Crowdroid" is small application installed from Google Play store, and it will monitor Linux calls on the device, and compare it from same application that downloaded from other resources, then it may detected if the application is modified with some malware code, the other component is Data manipulation: this component will manage and parse the data collected from the android users, and Malware analysis and detection component: which is used to analyzing and clustering the feature vectors extracted from the other components.

The method developed consist of several tools on client and server side, the main problem with this method that if the malware application submitted to Google Play store and has no other resources, it will not detect the application is malware.

Yerima et al [**8**], proposed and evaluated a new approach for detecting Android malware by reverse engineering the Android applications

using APK Analyzer, and building the dataset from set of 58 properties from API call, Commands and Permissions, then used a Bayesian based classifier for learning and detection stages.

The result of study showed the proposed method has better detection rates then signature based anti-virus, but the method require disassembly of application and then extracting the used features which may not suitable as preventable method.

Cheng et al [**37**], has presented a collaborative virus detection and alert system for smartphones (SmartSiren), they used behavioral analysis of smartphone viruses by ontology, the certainty factor function (CF function) generation by the certainty factor theory and the reasoning process of detecting viruses by a FPN model.

They developed mobile malware detection system (MMDS), which will filter the files received by SMS or MMS by extracting their behaviors and determine the danger level and if the users have confirmed them danger of these files, the system will reject the files sent by the SMS or MMS.

The presented method require an application to be installed on the smart phone (MMDS), and also require an interactive from the users to confirm the danger of the files, novice users will have hard time determining if the received SMS or MMS has danger file or not, especially if that received from known numbers.

Koundel et al [**38**], proposed a method to build a dataset from installed application on user mobile phone, the method using an application that will be installed on user mobile and send list of the applications installed, and the permissions and applications battery's usage, then sent to server to as csv file, then serve will parse the csv file and build it into the dataset.

The downside of this method it's require an application to be installed to gather the data from the end user mobile, also the application may itself drain the battery, which is another downside of this method.

Liu et al [**39**], proposed general Malware detection method called VirusMeter, it's monitor the usage of battery power on mobile devices, and compare it to pre-defined power consumption model to identify the abnormal usages of the battery power, using the OS Api it will calculate how much power used by the running services, and compared to the pre-defined model.

The proposed model monitor only the power usage of the system to determine if there's a malware on the mobile device or not, it may gave misleading alarm based on s normal services may require more power for various reasons such as background updating, or downloading the data.

Sahs and Khan [**40**], in this paper the authors used an open source application called "Androguard" to extract features of the APK file and used Scikit-learn framework to train vector machine to generate as much as positive marked as negative if there's enough differences from the training data.

This method treat all applications as benign except if it's sufficiently different from training data, so this may mark malware application as benign because there's not previously added in the training dataset.

3.2. General Malware Detection:

Jacobsson et al [**41**], Built two models "bag-of-words" and "meta EULA model" to find spywares, they collected more than 1000 (900 clean, 100 Bad) of "End User License Agreements (EULAs)" and they apply the model with multiple classifiers such as: NaiveBayes, DecisionStump, J48, Etc , and results support their hypothesis that EULAs can be used as a basis for classifying the corresponding software as good or bad.

This method will not work if the spyware authors start to copy the good applications EULA and use them with same text.

Shaban [**42**], has built a model to detect the spyware using data mining for windows portables files (PE), the researcher collected many windows PE that include benign and spyware executable files, then exported the API calls and put them on categories, then apply data mining classification for detecting the spyware.

The proposed model in this study require the files to be saved first, after that the file need to be analyzed to extract the API calls, we need to find a way to find if the file is malware before install it.

Summary:

As listed in this section, the researches about malware used different approach and methods to detect the malware on the Android platform.

But the most of the researchers didn't talk about a preventable approach for the normal users, how can't detect the benign Android applications and Malware Androids applications because of lake of experience.

The normal users lake the understanding of harmful applications, so Based on that our method will target those people by proposing a new preventable method for the Malware on Android platform.

Chapter 4
Research proposal and Methodology

In this chapter, we present and explain our method for detecting the malware using the permissions of APK files, the chapter is divided into six sections, section one present the methodology steps of our method.

Section two contain the data collection of the malware and benign applications, section three contain data preprocess and feature selections, and section four contain the classifications algorithms that used for our method, and section five and six for applying the used classifiers and evaluation for the method.

4.1 Methodology steps.

The steps shown in figure 4.1 is base steps used for work on our methodology, and figure 4.2 show the overview of the method for detecting the malwares, we start by collecting samples app then extract the permissions from them, then we will select number of features based on special attributes, we will find a prober classifier and apply it, then we will evaluate the results we have.

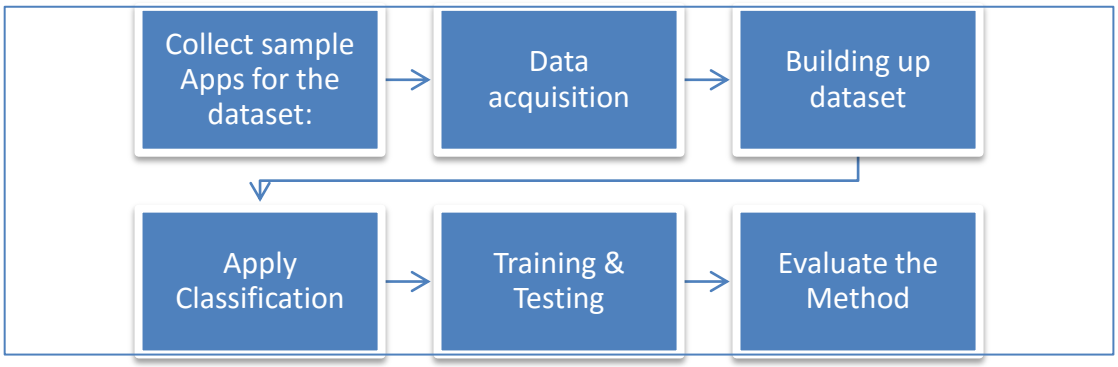


Figure 4.0.1 Steps of the work

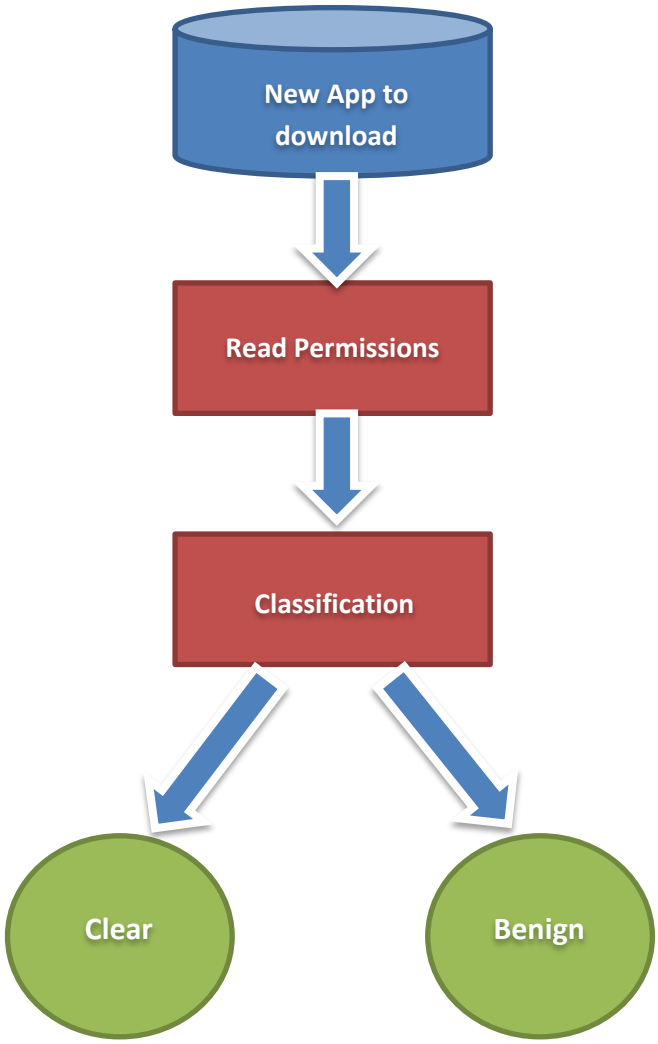


Figure 4.0.2 Overview of new method

4.2 Data Collection:

In this phase, we collected the benign and malware applications from different sources.

4.2.1 Benign Applications

The benign Applications has been downloaded from Google Play store, and due to Google policy it doesn't allow downloading the APK files directly from their website, but users can install them directly from the Play Store application on their Mobile device, also Android mobile phone doesn't allow to extract the APK files because they are hidden with the system files, so we used "APK Downloader" [**43**] website to download the APK files, it's simulate the mobile devices as it act as mobile, then offering the APK to be downloaded from the website directly to our PCs.

We have made some rules to ensure that the downloaded application for our sample dataset will benign and to eliminate any hidden malware may be found from downloaded application.

Rules to download the benign applications as following:

1. Application should be downloaded more than 1 Million times, because it will be sure it's speeded well and if it had any malware inside it will be discovered before reach this number of downloads.
2. Application rating should not be less than 4 of 5, to be ensure it tested by many users and it working well.
3. Application should pass the entire 53 virus tool scan on www.virustotal.com website.

Figure 4.3, show a sample of the Benign application that downloaded from Google play store.

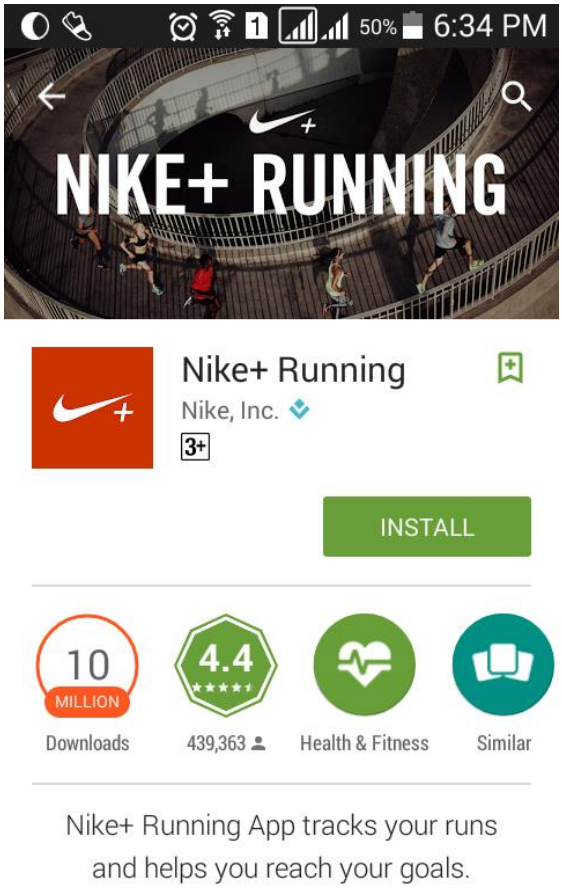


Figure 4.3 Sample for Benign application

4.2.2 Validating the Benign Applications:

The downloaded files with APK Downloader has been verified from virus using "Virus Total" [**44**] website, which verify the uploaded files with 53 Anti-Virus to make sure the applications has not been infected by any Malware.

4.2.3 Malware Applications:

The malware dataset has been downloaded from "Free Range Security" [**7**], it's containing 189 malware Applications.

4.3 Data Preprocessing:

In our method we rely on the permissions of Android application to recognize the benign and malware applications

In this section we will cover the steps to extract the permissions from the APK files, that required to build the Dataset.

4.3.1 Extract Permissions from the APKS:

On this step the permissions will be extracted from each APK.

APK file is a normal zipped file contain multiple files and folders As shown in figure 4.4.

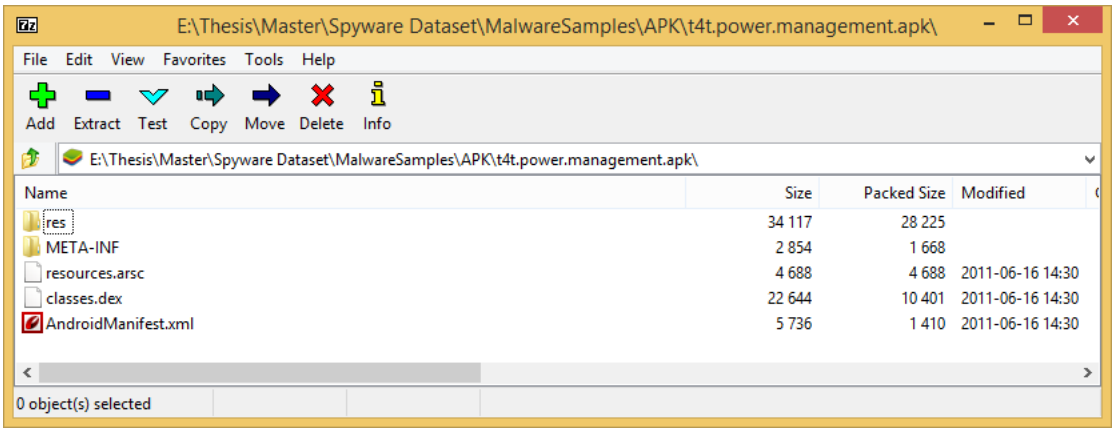


Figure 4.4 The APK file from inside

The APK file contain file called “Classes.dex” [**45**], its executable file of the application, Another file is the “AndroidMainfest.xml”, it’s binary file that every Android application should have (see figure 4.5).

The file contain the name of Java package for the application and it’s components, also it included the permissions required to the application.

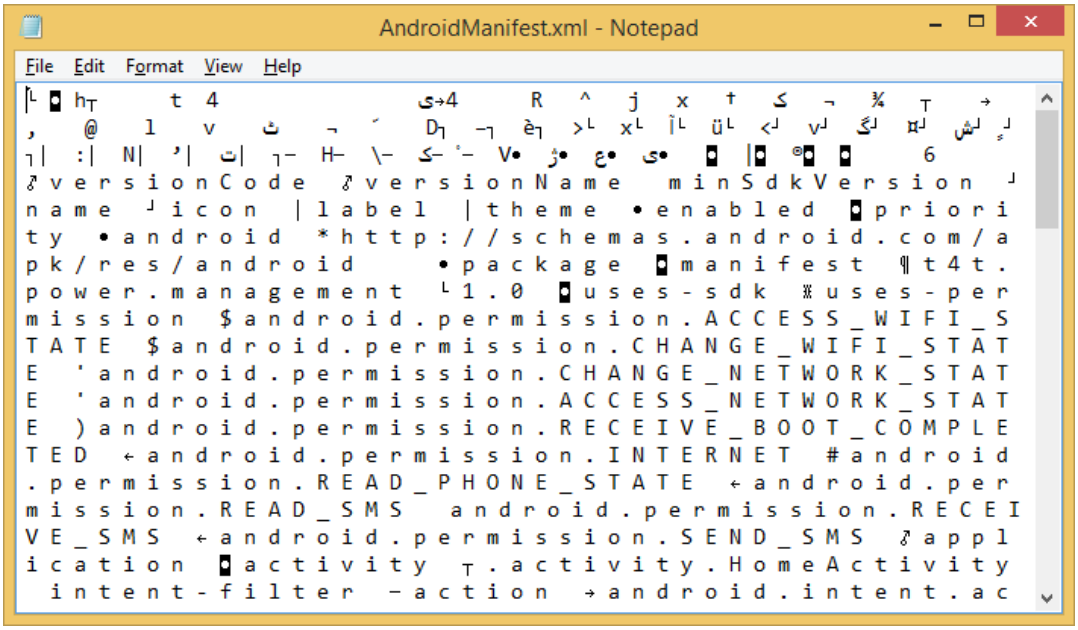


Figure 4.5 The AndroidManifest.xml (Packaged)

A special tool from Android SDK Called Android Asset Packaging Tool "AAPT.EXE" [**46**] used to extract the AndroidMainfest.xml.

And because it's work only for one APK per time under command line, we developed a special utility to automated the extraction of desired information from the "AndroidMainfest.xml".

This tool called "Permissions Extractor" (Figure 4.6), and it can export the output as text files, the data collected "The APK files" stored on two folders, malware & benign, this Permission Extractor point to the folder, then extract all the permissions from the both locations and save extracted permissions as text files in output folder for each one of them.

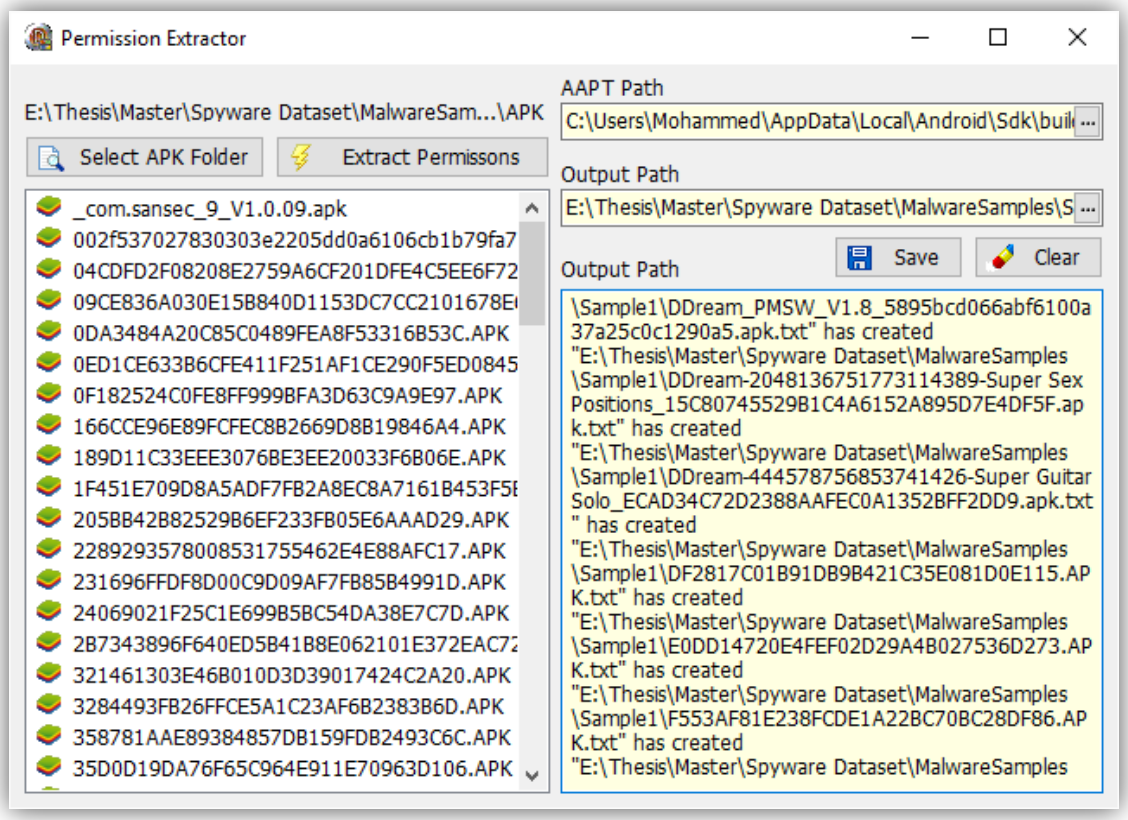


Figure 4.6 Permissions Extractor Tool

Another tool has been developed to read the permissions from the extract files and compare it for each application with Google official permissions list, we called the tool "Read Permissions" tool figure 4.7, then we combine all the files into memory and use the tool to specify the permissions for each application that it uses as Google permissions list, as shown in figure 4.8.

After the all the applications joined with permission, the file exported as CSV file for benign and another one for the malware.

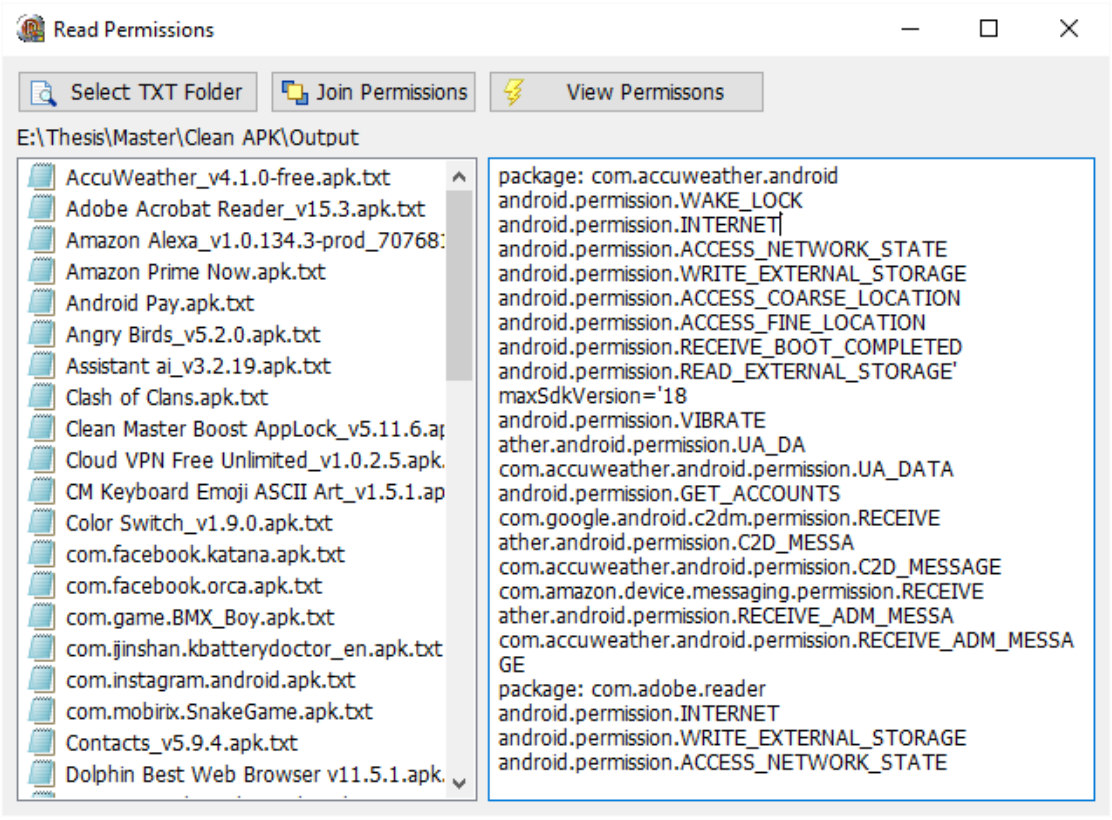
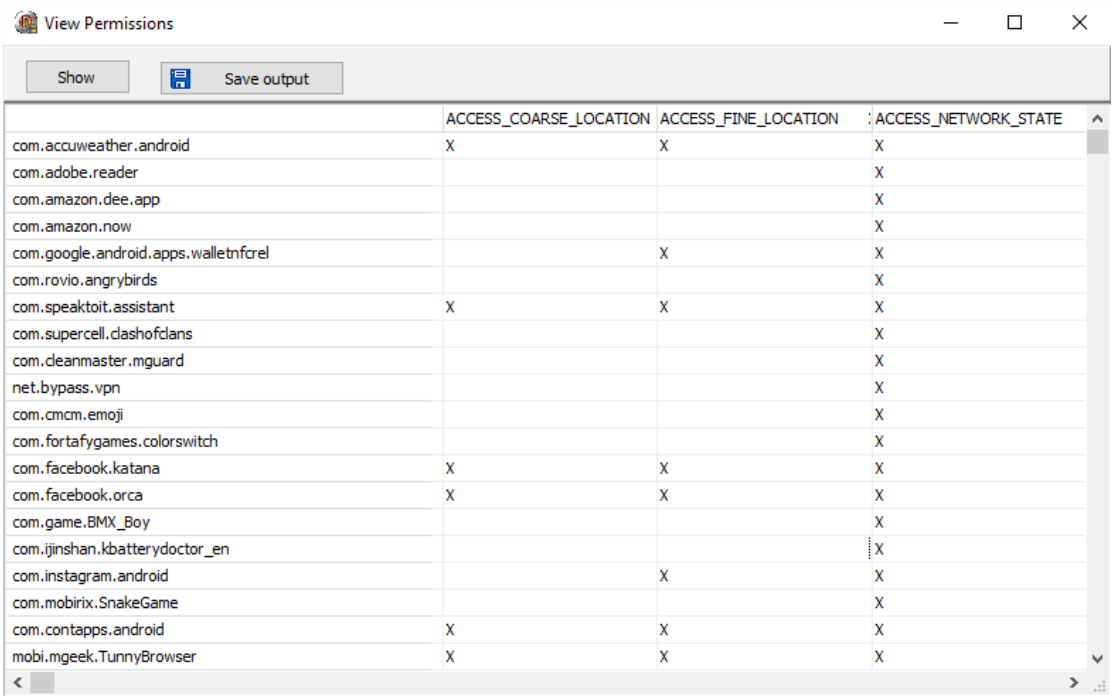


Figure 4.7 Read Permission Tool



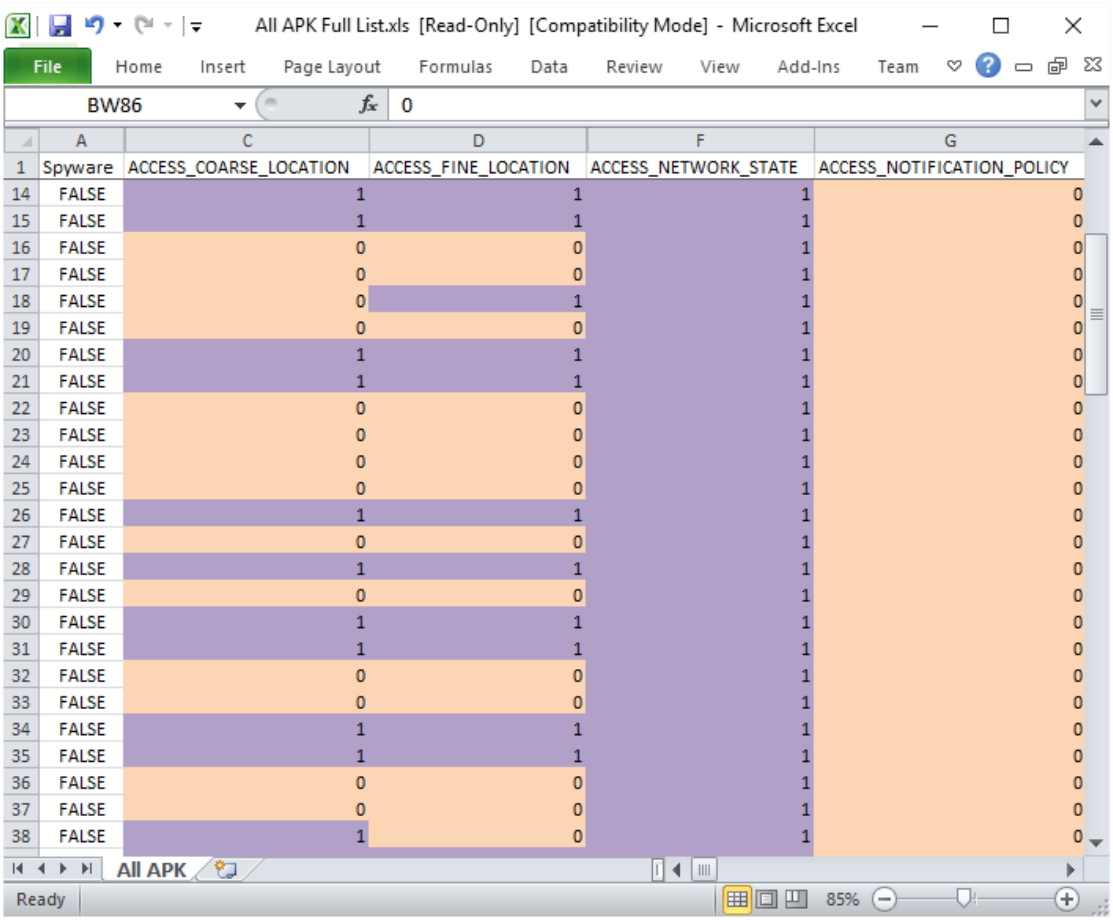
	ACCESS_COARSE_LOCATION	ACCESS_FINE_LOCATION	ACCESS_NETWORK_STATE
com.accuweather.android	X	X	X
com.adobe.reader			X
com.amazon.dee.app			X
com.amazon.now			X
com.google.android.apps.walletnfcrel		X	X
com.rovio.angrybirds			X
com.speaktoit.assistant	X	X	X
com.supercell.clashofclans			X
com.cleanmaster.mguard			X
net.bypass.vpn			X
com.cmcm.emoji			X
com.fortafygames.colorswitch			X
com.facebook.katana	X	X	X
com.facebook.orca	X	X	X
com.game.BMX_Boy			X
com.ijinshan.kbatterydoctor_en			X
com.instagram.android		X	X
com.mobirix.SnakeGame			X
com.contapps.android	X	X	X
mobi.mgeek.TunnyBrowser	X	X	X

Figure 4.8 Read Permissions Tool

4.3.3 Combine and clean the data:

In this step we took the CSV file and open it on Excel 2010 to remove many duplicates of the malwares(we found same malware "com.km.launcher" with same package name and permission has more than 34 files, but with same package name),also we removed any application that has no permissions at all. Then changed the "X" mark to number "1" and empty value to "0" figure 4.9.

Next step was to combine the both files and mark the spyware attributes with "True/False" to be used as label, and export it one Excel file, which has "103" samples divided to (61 benign and 42 malware), which is the dataset we will use.



	A	C	D	F	G
1	Spyware	ACCESS_COARSE_LOCATION	ACCESS_FINE_LOCATION	ACCESS_NETWORK_STATE	ACCESS_NOTIFICATION_POLICY
14	FALSE	1	1	1	0
15	FALSE	1	1	1	0
16	FALSE	0	0	1	0
17	FALSE	0	0	1	0
18	FALSE	0	1	1	0
19	FALSE	0	0	1	0
20	FALSE	1	1	1	0
21	FALSE	1	1	1	0
22	FALSE	0	0	1	0
23	FALSE	0	0	1	0
24	FALSE	0	0	1	0
25	FALSE	0	0	1	0
26	FALSE	1	1	1	0
27	FALSE	0	0	1	0
28	FALSE	1	1	1	0
29	FALSE	0	0	1	0
30	FALSE	1	1	1	0
31	FALSE	1	1	1	0
32	FALSE	0	0	1	0
33	FALSE	0	0	1	0
34	FALSE	1	1	1	0
35	FALSE	1	1	1	0
36	FALSE	0	0	1	0
37	FALSE	0	0	1	0
38	FALSE	1	0	1	0

Figure 4.9 combine the data

4.3.3 Feature set Selection:

We used the final Excel file as main source for our featuresets, then

The total attributes number after excluding the 0 weight are 54, as listed on table 4.1:

Table 4.1 attributes with weight > 0

No.	Attribute	Weight
1	BROADCAST_STICKY	0.01
2	EXPAND_STATUS_BAR	0.01
3	DISABLE_KEYGUARD	0.01
4	SET_WALLPAPER_HINTS	0.01
5	INTERNET	0.02
6	RECEIVE_MMS	0.03
7	WRITE_SETTINGS	0.04
8	ACCESS_COARSE_LOCATION	0.04
9	CALL_PHONE	0.05
10	GET_PACKAGE_SIZE	0.05
11	READ_SYNC_STATS	0.05
12	RECEIVE_WAP_PUSH	0.07
13	CHANGE_WIFI_MULTICAST_STATE	0.07
14	FLASHLIGHT	0.09
15	CHANGE_WIFI_STATE	0.09
16	CHANGE_NETWORK_STATE	0.10
17	READ_LOGS	0.10
18	BATTERY_STATS	0.10
19	MODIFY_AUDIO_SETTINGS	0.11
20	DELETE_PACKAGES	0.11
21	RECEIVE_BOOT_COMPLETED	0.12
22	WRITE_SYNC_SETTINGS	0.12
23	READ_SYNC_SETTINGS	0.13
24	BLUETOOTH_ADMIN	0.15
25	READ_CALL_LOG	0.16
26	KILL_BACKGROUND_PROCESSES	0.16
27	BLUETOOTH	0.17
28	ACCESS_FINE_LOCATION	0.19
29	RESTART_PACKAGES	0.21
30	READ_CONTACTS	0.23
31	CAMERA	0.25
32	WRITE_CALL_LOG	0.25
33	SYSTEM_ALERT_WINDOW	0.26
34	VIBRATE	0.26
35	NFC	0.27
36	ACCESS_WIFI_STATE	0.27
37	WRITE_E1TERNAL_STORAGE	0.28
38	GET_TASKS	0.28
39	WRITE_CONTACTS	0.29

40	ACCESS_NETWORK_STATE	0.35
41	WRITE_APN_SETTINGS	0.36
42	FACTORY_TEST	0.41
43	SET_WALLPAPER	0.43
44	REBOOT	0.43
45	READ_PHONE_STATE	0.43
46	WAKE_LOCK	0.52
47	READ_SMS	0.52
48	READ_E1TERNAL_STORAGE	0.54
49	MODIFY_PHONE_STATE	0.57
50	MOUNT_UNMOUNT_FILESYSTEMS	0.59
51	RECEIVE_SMS	0.65
52	INSTALL_PACKAGES	0.70
53	SEND_SMS	0.71
54	GET_ACCOUNTS	1.00

Three feature sets built based on the attributes weights using "Weight by SVM" as shown in figure 4.10.

attribute	weight
BLUETOOTH	0.174
ACCESS_FINE_LOCATION	0.191
RESTART_PACKAGES	0.212
READ_CONTACTS	0.228
CAMERA	0.250
WRITE_CALL_LOG	0.254
SYSTEM_ALERT_WINDOW	0.262
VIBRATE	0.264
NFC	0.266
ACCESS_WIFI_STATE	0.270
WRITE_E1TERNAL_STORAGE	0.275
GET_TASKS	0.284
WRITE_CONTACTS	0.293
ACCESS_NETWORK_STATE	0.346
WRITE_APN_SETTINGS	0.361
FACTORY_TEST	0.414
SET_WALLPAPER	0.427
REBOOT	0.434
READ_PHONE_STATE	0.435
WAKE_LOCK	0.517
READ_SMS	0.522
READ_E1TERNAL_STORAGE	0.543
MODIFY_PHONE_STATE	0.568
MOUNT_UNMOUNT_FILESYSTEMS	0.589

0.174

Figure 4.10 Weight by SVM

After that we filtered the attributes using “Selection by Weight” (as shown in figure 4.11), the weight used are greater than “0.1, .2, and .3”

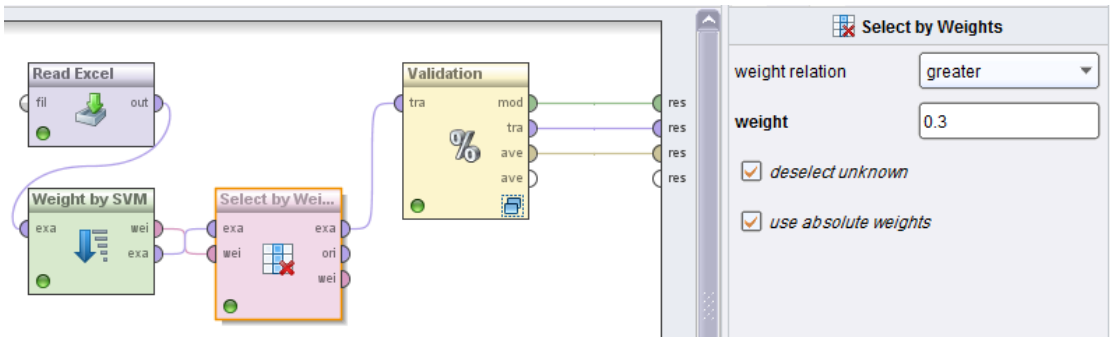


Figure 4.11 Feature Selection by Weight

We ignored any permissions not listed as “Android.Permission”, because they are application specific permissions, thou they will not take any security leak if they has given to the application.

Figure 4.12 show the permission required to “AccWeather” application which all inside large box, small box contain specific application, which will be excluded from our dataset, because they have no effect on the security for the application.

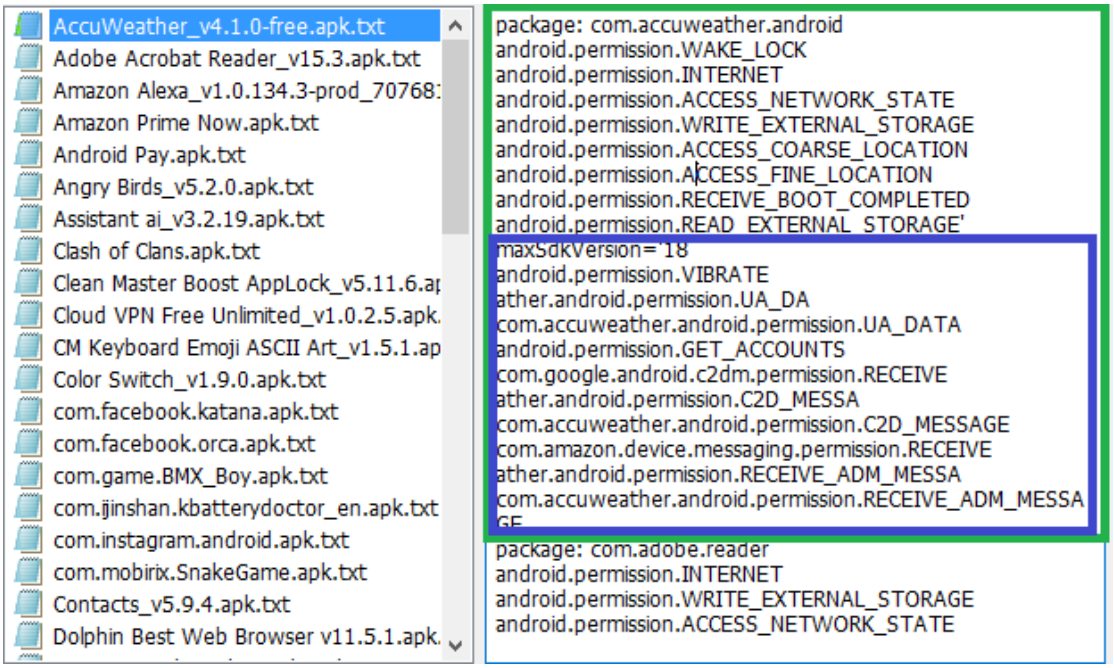


Figure 4.12 - sample of permission for an application

We build another feature set (see table 4.2) by emitting the attributes that not listed as dangerous permissions as Google specified [**17**], this feature has the following attributes (24 attribute):

Table 4.2 dangerous permissions Feature set

Attribute	Data Type
SPYWARE	Label
READ_CALENDAR	Integer
WRITE_CONTACTS	Integer
ACCESS_COARSE_LOCATION	Integer
READ_CALL_LOG	Integer
ADD_VOICEMAIL	Integer
SEND_SMS	Integer
READ_SMS	Integer

RECEIVE_MMS	Integer
WRITE_CALENDAR CAMERA	Integer
GET_ACCOUNTS	Integer
CALL_PHONE	Integer
WRITE_CALL_LOG	Integer
USE_SIP	Integer
RECEIVE_SMS	Integer
RECEIVE_WAP_PUSH	Integer
WRITE_EXTERNAL_STORAGE	Integer
READ_CONTACTS	Integer
ACCESS_FINE_LOCATION	Integer
READ_EXTERNAL_STORAGE	Integer
RECORD_AUDIO	Integer
PROCESS_OUTGOING_CALLS	Integer
BODY_SENSORS	Integer
READ_PHONE_STATE	Integer

Another feature set with dangerous permissions with additional attributes built, because we found they used by malware and has harmful effects see table 4.3, they not listed in dangerous permission list.

We calculated the total number of attributes used for each samples (benign and malware), we used the count of attributes that found in malware samples above benign samples, and also not listed in Google dangerous permissions.

Table 4.3 - Dangerous attributes for feature set 5

Attribute	**Description**
INSTALL_PACKAGES	Allows an application to install packages. Not for use by third-party applications.
RESTART_PACKAGES	Allows the app to end background processes of other apps. This may cause other apps to stop running
WRITE_APN_SETTINGS	Allows applications to write the apn settings. Not for use by third-party applications.

4.3.3 Building up Dataset:

We use the three features sets by weight and the one that contain Google's dangerous permissions as fourth Feature, the last feature set was the attributes used more in malware than benign and not listed in dangerous attributes list, this is the dataset as shown in table 4.4 that will be used in our research.

Table 4.4 Features selected

Feature sets	Weight	Number of Attributes
Feature set 1	➢ 0.1	38 regular 1 special (from attribute 16 to 54 as listed on table 4.1)
Feature set 2	➢ 0.2	26 regular 1 special (from attribute 29 to 54 as listed on table 4.1)
Feature set 3	➢ .03	15 regular 1 special (from attribute 40 to 54 as listed on table 4.1)
Feature set 4 (*dangerous permissions*)	No Weight	24 regular 1 special
Feature set 5 (extended *dangerous permissions*)	No Weight	27 regular 1 special

4.4.Find Appropriate Classifier:

We used well know classifiers used on previous studies for spyware and malware such as [**42**], the researcher used multiple classifier for his work to detect the spyware, in our research we used the same classifier, but we ignored the ones used by Weka, because our work done using RapidMiner.

4.5.Applying the Classifier:

After we prepared our dataset with 5 different feature sets, we applied the classification algorithms we previously selected to our dataset:

4.6.Evaluate the Method:

After we applied 4 different classifications methods, we used the AUC and F-measure evaluates performance for our method.

Summary:

In this chapter we started with data collections for the APK files, we used Google store to download benign asp, and "Free Security Range" for the malware, then we extract the permissions from APK files.

Two tools built for automate this work, and to export the extract permissions into one file, after we cleaned the data and built our dataset from five different feature set, based on weight for attributes and from dangerous permission listed provide by Google.

We find some dangerous permission used by the malware but not included in the list, we build another feature set with these attributes combined to dangerous permissions.

Chapter 5
Experimental Results and Evaluation

In this chapter, we will review the experimental settings and the software used for the work in this research.
Then will view the experimental results we had in the 5 scenarios.

5.1. Experimental Settings:

The experimental environment that used for all the experiments was laptop with core i7 CPU, 500GB SSD with 16GB Ram.

5.2. Software & Tools:

1. **Rapid Miner 5** [**47**]: An integrated environment for machine learning, data mining, text mining, predictive analytics and business analytics. It is used for business and commercial applications as well as for research, education, training, rapid prototyping, and application development and supports all steps of the data mining process including data preparation, results visualization, validation and optimization.

   Rapid Miner used to apply classifiers to our method, also used to test the data set with 10 fold cross validation.

2. **Microsoft Excel 2010** [**48**]: Is a spreadsheet developed by Microsoft for Windows, Mac OS X, and iOS. It features calculation, graphing tools, pivot tables.

   MS Excel used for the data cleaning and to combine the samples from the benign and malware into one feature set.

3. **PSPad** [**49**]: is a freeware text editor and source editor intended for use by programmers. It has many software development-oriented features, such as syntax highlighting and hex editing, and is designed as a universal GUI for editing many languages including PHP, Perl, HTML, and Java.

   PSPad used for open and edit the text files and other non-binary files, it can open large files very fast.

4. **Android SDK Tools** [**50**]: Android software development kit (SDK) includes a comprehensive set of development tools.[3] These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials.
   We use AAPT tools from Android SDK to extract the permission from the APKs files.

5. **Delphi 2010** [**51**]: Is a programming language and integrated development environment (IDE) for desktop, mobile, web, and console applications. Delphi's compilers use their own Object Pascal dialect of Pascal and generate native code for several platforms: Windows (x86 and x64), OS X (32-bit only), and iOS and Android (ARM).

   Delphi used to build two utilizes to automate the extraction of permissions from the APKs and to export them as CSV files.

6. **7-zip** [**52**]**:** is an open source software that used to create and extract compress files with LZMA and LZMA2 compression.

   7-zip used to open APKs files and to extract their content.

5.3.Evaluation

We choose two main performance measures for our evaluations, the AUC and F-Measure.

The reason for our chose, because several researcher found their performance is better than accuracy, for example Charles X. Lin said "AUC of ROC is a better measure than accuracy" *[**53**]*, in study *[**54**]* the researcher compared popular machine learning algorithms using AUC, and found that the AUC offer many of the desired characteristics compared to accuracy

AUC also used for measurement with spam detection *[**55**]*, so we think it suit for similar cases as malware detection.

Also F-measure is an alternative for Precision and recall, also its' not biased measure and can have a better result as state in [**56**].

5.4. Classifiers settings

The settings set in the evolution phase for each classifier as following (figure 5.1):

- **K-NN:**

The k value of the classifier set to 1.

- **Naïve bayes:**

Estimation mode: greedy

Minimum bandwidth: 0.1

Number of kernels: 10

- **SVM:**

We used the classifier with "The Laplace correction" is set, to prevent the influences of zero probabilities to be true [**42**].

Kernel type: dot

Kernel cache = 200

max iterations = 100000

- **Decision Tree:**

Criterion : gain_ratio

Minimize size of split: 4

Minimal leaf size: 2

Minimal gain: 0.1

Maximal depth: 20

Confidence: 0.25

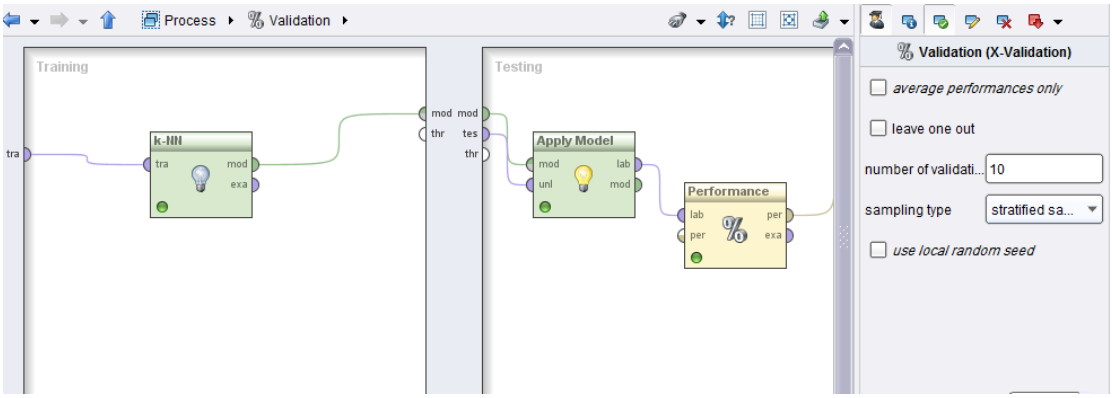


Figure 5.1 Rapid Miner with kNN and validation process

5.5. Experiment Scenarios and Result

In this section, we apply the 4 different classifiers to our dataset, which has 5 features, the detailed experimental results as following:

5.5.1. Experiment Scenarios 1 (feature set with Weight > 0.1)

We applied the 4 classifier to the dataset, the number of samples are 103, and number of attributes are 38, the SVM classifier was a higher in both AUC & F-Measure, see Table 5.1 and Figure 5.1:

Table 5.1 Experimental Result with feature set 1

Classifier	***AUC***	***F-Measure***
K-NN	0.5	88.58%
Naïve Bayes	0.989	92.31%
SVM	**0.993**	**95.20%**
Decision Tree	0.756	86.39%

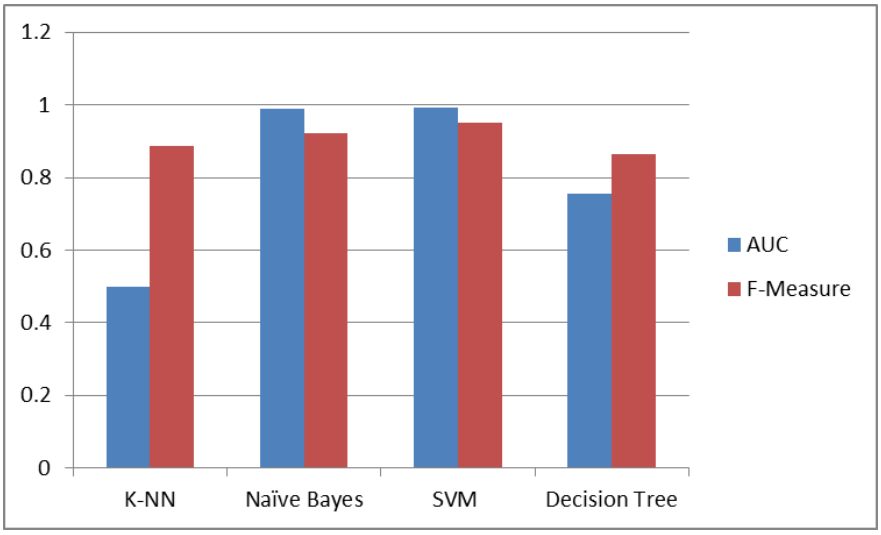


Figure 5.2 Experimental Results of feature set 1

5.5.2. Experiment Scenarios 2 (feature set with Weight > 0.2)

We applied the 4 classifier to the dataset, the number of samples are 103, and number of attributes are 36, the Naïve Bayes classifier was a higher in both AUC & F-Measure, but SVM gave the same value with AUC as NB see Table 5.2 and Figure 5.3:

Table 5.2 Experimental Result with feature set 2

Classifier	*AUC*	*F-Measure*
K-NN	0.5	88.89%
Naïve Bayes	**0.993**	**96.74%**
SVM	**0.993**	94.48%
Decision Tree	0.773	92.04%

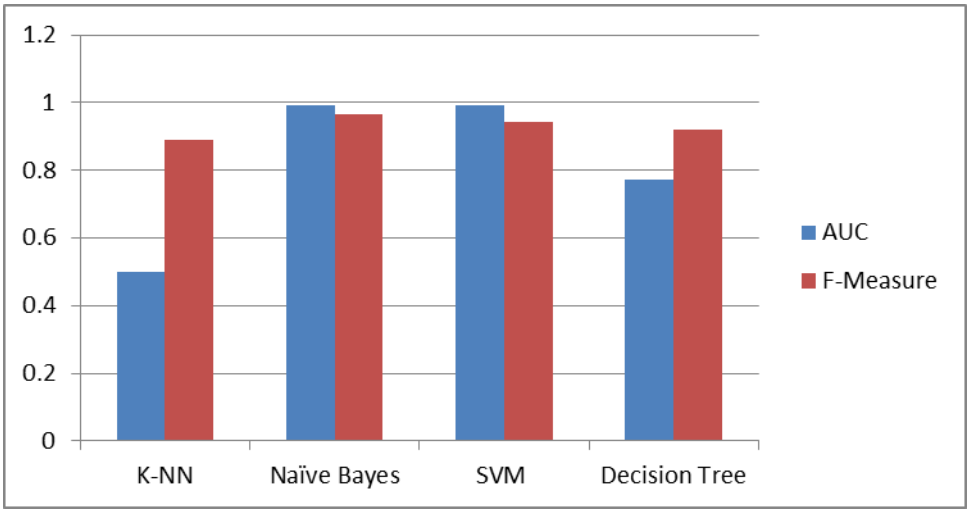


Figure 5.3 Experimental Result with feature set 2

5.5.3. Experiment Scenarios 3 (feature set with Weight > 0.3)

We applied the 4 classifier to the dataset, the number of samples are 103, and number of attributes are 15, the Naïve Bayes classifier was a higher AUC and SVM gave the higher value with F-Measure Table 5.3 and Figure 5.4

Table 5.3 Experimental Result with feature set 3

Classifier	*AUC*	*F-Measure*
K-NN	0.5	95.56%
Naïve Bayes	**0.988**	90.43%
SVM	0.985	**95.75%**
Decision Tree	0.766	92.17%

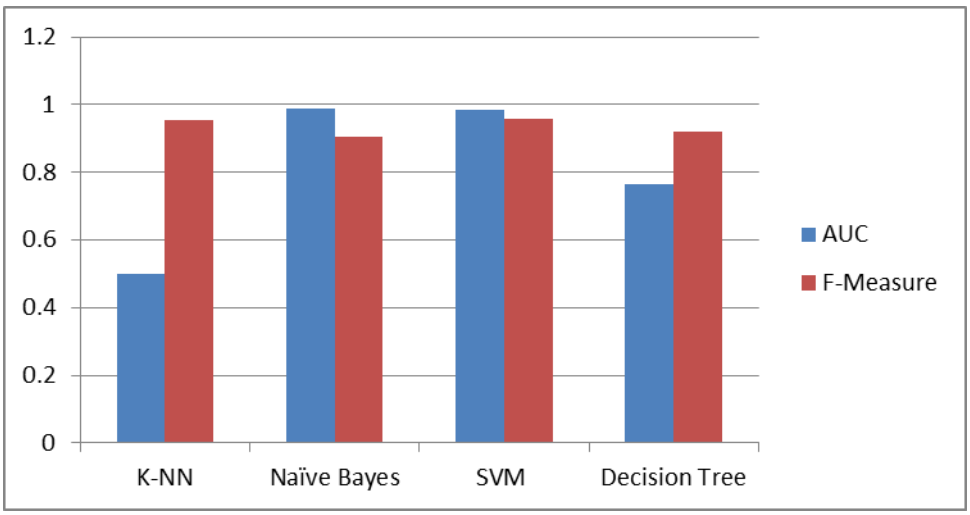


Figure 5.4 Experimental Result with feature set 3

5.5.4. Experiment Scenarios 4 (Dangerous Permissions)

We applied the 4 classifier to the dataset, the number of samples are 103, and number of attributes are 15, the Naïve Bayes classifier was a higher F-Measure and SVM gave the higher value with AUC Table 5.4 and Figure 5.5

Table 5.4 Experimental Result with Dangerous permissions feature set 4

Classifier	*AUC*	*F-Measure*
K-NN	0.500	85.93%
Naïve Bayes	0.979	**92.85%**
SVM	**0.985**	89.98%
Decision Tree	0.908	91.59%

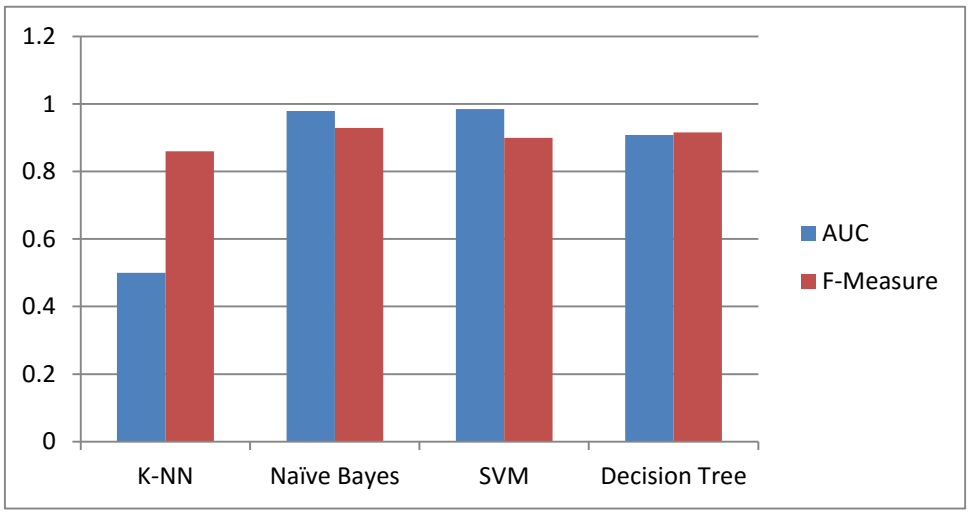


Figure 5.5 - Experimental Result with Dangerous permissions feature set 4

5.5.5. Experiment Scenarios 5 (Dangerous Permissions 2)

We applied the 4 classifier to the dataset, the number of samples are 103, and number of attributes are 27, the k value of kNN was 1, the naïve Bayes used with Laplace correction is checked, both SVM and Decision tree used with default values set by RM, table 10 show the detailed results :

Table 5.5 - Experimental Result with Dangerous permissions feature set 5

Classifier	*AUC*	*F-Measure*
K-NN	0.500	88.49%
Naïve Bayes	0.983	**94.64%**
SVM	**0.986**	92.79%
Decision Tree	0.894	92.13%

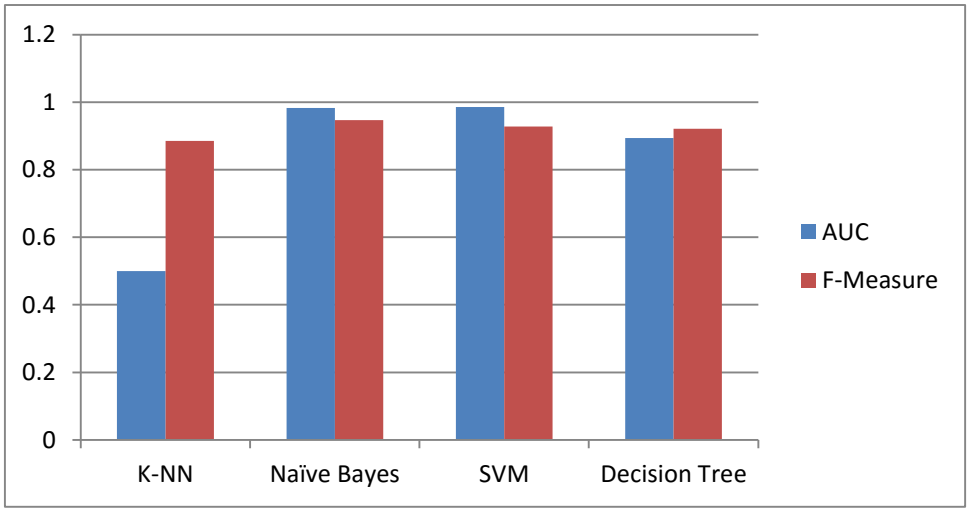


Figure 5.6 Experimental Result with Dangerous permissions feature set 5

5.6. Experiment Result Summary:

Based on our experimental, we found feature set 2 has the highest rates in the metrics we used for the evaluation (AUC and F-measure) Figure 5.7, and following the summary for our experimental:

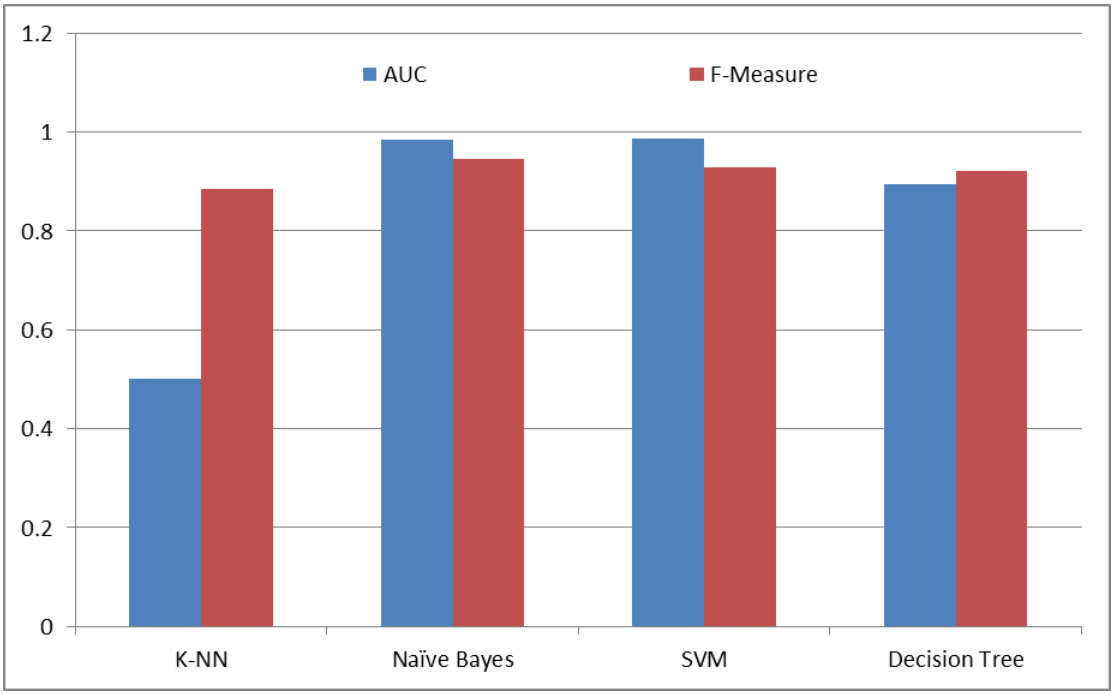


Figure 5.7 Experimental Result Summary

1. We achieved highest score in AUC (0.993) and F-Measure (96.74%) with Feature set 2 using Naïve Bayes classifier.
2. Feature set 1 gave same high accuracy as feature set 2 in AUC (0.993) using Support Vector Machine SVM classifier.

3. In our experimental k-NN classifier has the worst performance in both AUC & F-measure in all feature set.
4. Both Naïve Bayes and SVM, has the best performance in our experimental.
5. Feature set 5 (dangerous permissions with extended attributes) gave higher rates then feature set 4 (dangerous permissions specified by Google), these attributes should be consider by Google, to warn users about the dangerous effect of the attributes.

Summary:

In this chapter we review our work on the dataset we build in chapter 4.

We have done 5 scenarios with 4 classifications algorithms, then we used AUC and F-Measure as performance measurement, we reached 96.74% in our performance measure with f-measure and our highest AUC for the work is 0.993.

Finally we talked about the future work that we hope to work on future to make our method a feasible Prevention tool.

Chapter 6
Conclusion and Future Work

6.1. Future Work

In this research we worked on a new method to detect the malware before installing them to the user's mobile device.

However, with new thousands Applications added daily to Google play store, we need to find a better way to get the permissions of the applications without extracting them by downloading the APK first,

Currently, Google didn't provide any official API to access the information of the applications in Google play store information, some open source trying to achieve this [**57**], but may not work when Google change their protocol.

Other options to reverse engineer the Google's API to find better way to get the permissions from the store, or by doing website scarping to gather the information from play store website.

And for our method to effective as preventable tool, we need to build a layer on top of Google play store to warn the users about the possibility of the application to be malware before installing it on the user's mobile.

10. References

[1] Bharat Kumar Addagada, Intrusion Detection in Mobile Phone Systems Using Data Mining Techniques, 2010, Iowa State University.

[2] International Data Corporation. (2014) IDC. [Online]. http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[3] Strategy Analytics. (2014) [Online]. http://thenextweb.com/google/2014/07/31/android-reached-record-85-smartphone-market-share-q2-2014-report/

[4] Google. Play Store. [Online]. https://play.google.com/store

[5] Brendan Fitzgerald. (2014, Aug) appmakr.com. [Online]. http://www.appmakr.com/blog/how-long-app-approved/

[6] Charles Arthur. (2011, Mar) The Guardian. [Online]. http://www.theguardian.com/technology/blog/2011/mar/02/android-market-apps-malware

[7] Free Range Security. (2011) [Online]. http://cgi.cs.indiana.edu/~nhusted/dokuwiki/doku.php?id=datasets

[8] Suleiman Y. Yerima, Sakir Sezer, Gavin McWilliams , and Igor Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," *IEEE*, pp. 121-128, 2013.

[9] Wikipedia. (2005) Wikipedia. [Online]. https://en.wikipedia.org/wiki/Android_(operating_system)#History

[10] Erika Chin, Steve Hanna, Dawn Song, David Wagner Adrienne Porter Felt, "Android Permissions Demystified," *CCS '11 Proceedings of the 18th ACM conference on Computer and communications security*, pp. 627-638, Oct 2011.

[11] International Data Corporation. (2015, Aug) IDC. [Online]. http://www.idc.com/prodserv/smartphone-os-market-share.jsp

[12] Dan Bornstein (Google). (2008, May) 2008 Google I/O Session Videos and Slides. [Online]. https://14b1424d-a-62cb3a1a-s-sites.googlegroups.com/site/io/dalvik-vm-internals/2008-05-29-Presentation-Of-Dalvik-VM-Internals.pdf?attachauth=ANoY7cqDKcObEfRaofwSVempV1QZBREUlAmg9slrBtchjcxJh9XBEax1VnHYFPHh52U1uSL0ecQGXHrGaez4d4pYs9yWHbgvDgaO58y9viYfdsZ1

[13] the virtual machine of Android Dalvik. (2014) scriptol. [Online]. http://www.scriptol.com/programming/dalvik.php

[14] Wikipedia. Android application package. [Online].

https://en.wikipedia.org/wiki/Android_application_package

[15] Ken Christensen Nikolai Samteladze, "DELTA++: Reducing the Size of Android Application Updates," *IEEE Computer Society*, vol. 18, pp. 50-57, Mar 2014.

[16] Google. Manifest.permission. [Online]. http://developer.android.com/reference/android/Manifest.permission.html

[17] Google. System Permissions. [Online]. http://developer.android.com/guide/topics/security/permissions.html

[18] Latifur Khan, Bhavani Thuraisingham Mehedy Masud, *Data Mining Tools for Malware Detection*.: CRC Press, 2011.

[19] Osmar R. Zaïane, *Principles of Knowledge Discovery in Databases*. USA: University of Alberta , 1999.

[20] MUAZZAM AHMED SIDDIQUI, "DATA MINING METHODS FOR MALWARE DETECTION," University of Central Florida , Orlando, PhD Thesis 2008.

[21] S. Dua and X. Du, "Data Mining and Machine Learning in Cybersecurity," *The Auerbach*, 2011.

[22] N.Eswari C.Gunasundari, "A Study of Web Content Mining," *International Journal of Innovative Research in Science & Engineering*, vol. 3, no. 6, pp. 253-259, 2013.

[23] RapidMiner Documentation. (2016) http://docs.rapidminer.com/studio/operators/modeling/predictive/bayesian/naive_bayes_kernel.html. [Online]. http://docs.rapidminer.com/studio/operators/modeling/predictive/bayesian/naive_bayes_kernel.html

[24] Eibe Frank Ian H. Witten, *Data Mining Practical Machine Learning Tools and Techniques*, Second edition ed.: Morgan Kaufmann, 2005.

[25] Tutorials Point. http://www.tutorialspoint.com/data_mining/dm_dti.htm. [Online]. http://www.tutorialspoint.com/data_mining/dm_dti.htm

[26] Dursun Delen David L. Olson, *Advanced Data Mining Techniques*.: Springer, 2008.

[27] Michal Kepski Bogdan Kwolek, "Improving Fall Detection by the Use of Depth Sensor and Accelerometer," *Neurocomputing*, vol. 186, no. C, pp. 637-645, Novmber 2015.

[28] Wikipedia. Wikipedia. [Online]. https://en.wikipedia.org/wiki/Precision_and_recall

[29] Tom Fawcett Foster Provost, "Using AUC and Accuracy in Evaluating," in *Proceedings of*

the Third International Conference on Knowledge Discovery and Data Mining, pp. 43-48.

[30] Raffael Vogler. (2015) R news and tutorials. [Online]. http://www.r-bloggers.com/illustrated-guide-to-roc-and-auc/

[31] Iman Lotfi Sara Najari, "Malware Detection Using Data Mining Techniques," *Science Publishing Group*, vol. 3, no. 6, pp. 33-37, Oct 2014.

[32] "On the Comparison of Malware Detection Methods Using Data Mining with Two Feature Sets," *International Journal of Security and Its Applications*, vol. 9, no. 3, pp. 293-318, 2015.

[33] eBay. (2014) http://www.ebay.com. [Online]. http://www.ebay.com/gds/Different-Types-of-Antivirus-and-Security-Software-/10000000177629318/g.html

[34] Veracode. (2012) veracode. [Online]. https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101

[35] Igor Santos, Javier Nieves, Carlos Laorden, Iñigo Alonso-Gonzalez, Pablo G. Bringas Borja Sanz, "MADS: Malicious Android application Detection through String analysis," *Lecture Notes in Computer Science*, pp. 178-191, 2013.

[36] Urko Zurutuza Iker Burguera, "Crowdroid: behavior-based malware detection system for Android," *ACM - Association for Computing Machinery*, pp. 15-26, OCT 2011.

[37] Starsky H.Y. Wong, Hao Yang, Songwu Lu Jerry Cheng, "SmartSiren: Virus Detection and Alert for Smartphones," *ACM (Association for Computing Machinery)*, pp. 258-271, June 2011.

[38] Suraj Ithape, Vishkha Khobaragae, Rajat Jain Deepak Koundel, "Malware Classfification using Navie Bayes Classifier for Android OS," *The International Journal of Engineering and Science*, vol. 3, no. 4, pp. 59-63, 2014.

[39] Guanhua Yan, Xinwen Zhang, and Songqing Chen Lei Liu, "VirusMeter: Preventing Your Cellphone," *Association for Computing Machinery*, pp. 244 - 264, Oct 2009.

[40] Justin Sahs and Latifur Khan, "A Machine Learning Approach to Android Malware Detection," in *European Intelligence and Security Informatics Conference*, Dallas, 2012, pp. 141 - 147.

[41] Martin Boldt, Paul Davidsson, Andreas Jacobsson Niklas Lavesson, "Learning to detect spyware using end user license agreements," *Knowledge and Information Systems*, vol. 26, no. 2, pp. 285-307, January 2010.

[42] Fadel Omar Shaban, Spyware Detection Using Data Mining for Windows Portable Executable Files, 2013.

[43] APK Downloader. [Online]. http://apps.evozi.com/apk-downloader/

[44] Virus Total. [Online]. https://www.virustotal.com/

[45] Google. (2014) Android Open Source Project. [Online]. https://source.android.com/devices/tech/dalvik/dex-format.html

[46] Google. Build System Overview. [Online]. http://developer.android.com/sdk/installing/studio-build.html

[47] Rapid Miner. Rapid Miner. [Online]. http://www.rapidminer.com

[48] Microsoft. Microsoft Excel. [Online]. https://products.office.com/en/excel

[49] PsPad. PsPad. [Online]. http://www.pspad.com/en/

[50] Google. Android SDK Tools. [Online]. https://developer.android.com/studio/index.html

[51] Embarcadero. Delphi. [Online]. https://www.embarcadero.com/products/delphi

[52] 7-Zip. 7-Zip. [Online]. http://www.7-zip.org/

[53] Jin Huang, Harry Zhang Charles X. Ling, "AUC: a Better Measure than Accuracy in Comparing Learning Algorithm," Ottawa-Carleton Institute for Computer, Science,.

[54] Andrew P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, pp. 1145-1159, july 1997.

[55] indico. Stack exchange. [Online]. http://datascience.stackexchange.com/questions/806/advantages-of-auc-vs-standard-accuracy

[56] Nong Ye, *THE HANDBOOK OF DATA MINING*.: Human Factors and Ergonomics, 2003.

[57] Chad Remesch. GitHub. [Online]. https://github.com/chadrem/market_bot

[58] The Palestinian Central Bureau of Statistics. (2014, May) The Palestinian Central Bureau of Statistics. [Online]. http://www.pcbs.gov.ps/site/512/default.aspx?tabID=512&lang=en&ItemID=1115&mid=3171&wversion=Staging

[59] Rob Schapire, COS 511: Theoretical Machine Learning, 2008.

[60] Tom M. Mitchell, The Discipline of Machine Learning, 2006.

[61] Wikipedia. [Online]. https://en.wikipedia.org/wiki/Accuracy_and_precision#In_binary_classification

[62] Rapid Miner Doc. [Online]. http://docs.rapidminer.com/studio/getting-started/5-evaluating-model.html

[63] D. R. Wilson and T. R. Martinez, "Improved Heterogeneous Distance Functions," *Journal of Artificial intelligence Reseach*, pp. 1-34, 1997.

[64] Christopher Leckie Kingsly Leung, "Unsupervised anomaly detection in network intrusion detection using clusters," in *ACSC '05 Proceedings of the Twenty-eighth Australasian conference on Computer Science*, 2005, pp. 333-342.

[65] Juan Manuel Corchado Rodríguez, Sara Rodríguez González, Juan F. de Paz Santana Ajith Abraham, *International Symposium on Distributed Computing and Artificial Intelligence*. Berlin , Heidelberg: Springer, 2011.