A MALWARE RISK ANALYSIS AND DETECTION SYSTEM FOR MOBILE DEVICES USING PERMISSION-BASED FEATURES

MOHD FAIZAL BIN AB RAZAK

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

A MALWARE RISK ANALYSIS AND DETECTION SYSTEM FOR MOBILE DEVICES USING PERMISSION-BASED FEATURES

MOHD FAIZAL BIN AB RAZAK

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Mohd Faizal Bin Ab Razak

Matric No: WHA140021

Name of Degree: Degree of Philosophy

Title of Thesis: A MALWARE RISK ANALYSIS AND DETECTION SYSTEM FOR MOBILE DEVICES USING PERMISSION-BASED FEATURES

Field of Study: Security (Computer Science)

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every right in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

A MALWARE RISK ANALYSIS AND DETECTION SYSTEM FOR MOBILE DEVICES USING PERMISSION-BASED FEATURES

ABSTRACT

In recent years, the amount of malware targeting Android users has increased dramatically. Among many mobile operating systems, the Android operating system is most targeted by malware. In order to detect malware which causes immense chaos and problems to mobile device users, the Android mobile applications need to be analysed. Two types of malware analysis are available namely, static analysis and dynamic analysis. Static analysis examines the whole code of the applications thoroughly while dynamic analysis identifies malware applications by monitoring their behaviors. Although both types of analysis have been performed with some level of success, additional processes are needed to improve the malware detection system. This is because current technologies indicate that malware attackers find novel ways of avoiding detection while causing harm. This thesis aims to propose an efficient malware detection system which uses the machine learning approach and the risk analysis approach to analyse Android applications. This study focusses in particular on permission features which are able to disclose the sensitive information noted on Android mobile devices. This study uses data samples accessed from Drebin by collecting 5,560 applications from 179 different malware families. It also uses data samples accessed from Androzoo by collecting 5,000 benign applications. This study also proposes a novel quantitative security method for evaluating the risk analysis of malicious and benign applications based on Android permissions. The risk analysis helps users to understand the risk level of the applications. It also improves user attention by giving responses to the users regarding permissions that contain high-risk levels. More specifically, this study performs four experiments through to validate the proposed system for use. In particular, this study introduces the EZADroid for evaluating and zoning the Android applications which apply the Analytic Hierarchy Process (AHP) as a decision

factor to calculate the risk values and to assess the prediction performance through True Positive Rate (TPR), False Positive Rate (FPR), accuracy, f-measure and precision. Finally, a website was established to validate the prediction performance with machine learning approach that measures its efficiency and effectiveness. The outstanding results imply that this study has proven that the permission features are capable of classify malware applications.

Keywords: Machine learning, risk analysis, Android, static analysis, features selection

ANALISIS RISIKO DAN SISTEM PENGESANAN PERISIAN PEROSAK UNTUK PERANTI MUDAH ALIH MENGGUNAKAN CIRI KEBENARAN

ABSTRAK

Dalam tahun-tahun kebelakangan ini, jumlah perisian perosak yang menyasarkan pengguna Android telah meningkat secara dramatik. Di antara perisian mudah alih, sistem perisian Android paling disasarkan oleh perisian perosak. Perisian perosak ini menyebabkan masalah kepada pengguna peranti mudah alih, aplikasi mudah alih Android perlu dianalisis. Terdapat, dua jenis analisis iaitu analisis statik dan analisis dinamik. Analisis statik mengkaji keseluruhan kod aplikasi secara menyeluruh sementara analisis dinamik mengenal pasti aplikasi malware dengan memantau tingkah laku mereka. Walaupun kedua-dua jenis analisis telah dilakukan dengan beberapa tahap kejayaan, proses tambahan diperlukan untuk memperbaiki sistem pengesanan perisian perosak. Ini kerana teknologi semasa menunjukkan bahawa penyerang perisian perosak mencari cara baru untuk mengelakkan pengesanan dan menyebabkan bahaya. Tujuan tesis ini adalah untuk mencadangkan sistem pengesanan perisian perosak yang berkesan yang menggunakan pendekatan pembelajaran mesin dan pendekatan analisis risiko untuk menganalisis aplikasi Android. Kajian ini memberi tumpuan khususnya pada ciri-ciri kebenaran yang dapat mendedahkan maklumat sensitif yang dicatatkan pada peranti mudah alih Android. Kajian ini menggunakan sampel data yang diakses dari Drebin dengan mengumpulkan 5,560 perisian terdiri daripada 179 keluarga. Ia juga menggunakan sampel data yang diakses dari Androzoo dengan mengumpulkan 5,000 perisian baik. Kajian ini juga mencadangkan kaedah keselamatan kuantitatif novel untuk menilai analisis risiko perisian perosak dan baik berdasarkan kebenaran Android. Analisis risiko membantu pengguna memahami tahap risiko perisian. Ia juga meningkatkan perhatian pengguna dengan memberi maklum balas kepada pengguna mengenai kebenaran yang mengandungi tahap risiko tinggi. Lebih khusus lagi, kajian ini

menjalankan empat eksperimen melalui fasa dan langkah untuk mengesahkan sistem yang dicadangkan. Khususnya, kajian ini memperkenalkan EZADroid untuk menilai dan mengelaskan aplikasi Android yang menggunakan Proses Hierarki Analitik (AHP) sebagai faktor keputusan untuk mengira nilai risiko dan menilai prestasi ramalan melalui Kadar Positif Benar (TPR), Kadar Positif Palsu (FPR), f-ukur dan ketepatan. Akhirnya, sebuah laman web ditubuhkan untuk mengesahkan prestasi ramalan dengan pendekatan pembelajaran mesin yang mengukur kecekapan dan keberkesanannya. Hasil menunjukkan bahawa kajian ini telah membuktikan bahawa ciri kebenaran mampu meramal perisian perosak yang tidak diketahui termasuk analisis risiko pada perisian Android.

Kata kunci: Pembelajaran mesin, analisis risiko, Android, analisis statik, pemilihan ciri-ciri

ACKNOWLEDGEMENTS

First of all, I am thankful to the Almighty Allah for bestowing me with the strength and perseverance to carry on with my PhD journey even though at times I felt weary. I am very blessed to have endured it all and still be able to come out of it successfully by completing this study.

I am deeply indebted to my supervisors, Prof. Madya Dr. Rosli Bin Salleh and Prof. Madya Dr. Nor Badrul Anuar Bin Jumaat' for their invaluable guidance, supervision and encouragement throughout this study and this journey of endurance. Their continuous guidance and support has assisted me in conducting a valuable piece of study that is reported in this thesis. They had also provided me with the opportunity to broaden my professional experience and to prepare me for future challenges. Their countless efforts have further encouraged me to work hard so as to achieve the milestones in a defined time limit.

I would like to express my sincerest gratitude and appreciation to my family for their endless love and support during this doctoral stud pursuit especially my parents (Ab Razak Bin Taib, Wan Azizah Wan Abdul Rahman). Without their moral support, this thesis would not have been completed on time. No words can express my feelings and my gratitude towards my parents and siblings for all the sacrifices made. I dedicate the highest achievement of my student life to them.

I would also like to express my deep appreciation to my dearest lab friends who had been providing me with so much support and encouragement throughout this study and academic pursuit. I wish them all the best in their future undertakings.

Finally, I would like to thank the Faculty of Computer Science and Information Technology for its help in enabling me to deal with all sorts of matters during my studies.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ADB	:	Android Debug Bridge
AHP	:	Analytical Hierarchy Process
AI	:	Artificial Intelligence
APK	:	Android Package
Arff	:	Attribute-Relation File Format
CSV	:	Comma Separated Values
DT	:	Decision Tree
FP	:	False Positive
FPR	:	False Positive Rate
GUI	:	Graphical User Interface
IDS	:	Intrusion Detection System
KNN	:	K-Nearest Neighbors
ML	:	Machine Learning
MLP	:	Multi-Layer Perceptron
NB	:	Naïve Bayes
PSO	:	Particle Swarm Optimization
RF	:	Random Forest
SVM	:	Support Vector Machine
TN	:	True Negative
TPR	:	True Positive Rate
XML	:	Extensible Markup Language

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

This chapter introduces the theoretical framework by explaining the importance of the study. In order to give readers a glimpse into the study, this thesis is divided into six sections. Section 1.1 presents the background of the study. Section 1.2 explains the research motivation. Section 1.3 describes the problem statements and highlights the issues regarding the application risk and malware detection. Section 1.4 presents the research objectives. Section 1.5 explains the research methodology and Section 1.6 presents the thesis layout.

1.1 Background of the study

The explosive growth of Android mobile devices is most notable in the smartphone market. Android mobile devices are making smartphones more relevant than ever to people's daily lives as compared to ten or twenty years ago. However, the growing adoption of the Android mobile device has also brought about many security concerns and threats such as malicious software also called malware. It is a programme that harms the mobile system by injecting viruses such as Trojan Horses, root exploit, botnet, and spyware into Android applications. This malware has the capability to steal user credentials, read contact numbers and cause resource abuse. In 2015, the McAfee Labs discovered more than two million new malware (McAfee, 2016).

By September 2017, a total of 21.1 million Android mobile devices have been infected by malware (Dassanayake, 2017; Fox-Brewster, 2017) which sneaked its way into the Android mobile devices from Google Play Store (Fox-Brewster, 2017). According to the Trend Micro 2016 Security Predictions, China will be driving mobile malware growth to 20 million and most of the malware will be attacks on mobile payment methods (Clay, 2015). What the malware does is send fraudulent premium SMS messages and then charge the users for fake services. In the first half of 2017, about 235,000 Android

ransomware have been detected (Trend Micro, 2017). This number shows that the Android has become a high-risk mobile application (Clay, 2015).

Current but traditional approaches to detect malware include the anti-virus software product and Intrusion Detection System (IDS). However, unscrupulous authors apply sophisticated techniques such as a polymorphic and metamorphic techniques to prevent from anti-virus and the IDS. These sophisticated techniques are used to obfuscate and repackage the malicious codes so as to bypass the signature detection thereby defeating attempts to analyse their malicious intentions.

Of late, researchers (Firdaus et al., 2017) are focusing on malware detection by incorporating machine learning approaches to protect users from these novel threats. The machine learning approach allows the computer to train the data input while trying to detect malware. It uses the data to analyse the malware patterns. Without being programmed, it is also able to perform some specific tasks which produce reliable results.

There are two types of analyses which used on malware analysis namely, static analysis and dynamic analysis. Static analysis detects malware by extracting the code from the applications. It uses reverse engineering techniques (Razak et al., 2016). Dynamic analysis detects malware by running the applications and monitoring their behaviours. Its disadvantage is that it consumes high resources such as the central processing unit (CPU) processing time (Feizollah et al., 2015).

1.2 Motivation

This research was motivated by a number of reasons which are classified as follows:

a) **Trends on mobile devices:** The Android mobile device continues to lead in the mobile device market (Egham, 2017). To date, a total of 94 percent of mobile devices have been installed with the Android operating system (O'Shea, 2017). According to the IDC, the year 2020 will be seeing 1.5 billion Android mobile devices being shipped (International Data Corporation (IDC), 2016). In addition to this, 3.8 billions

of people are expected to be using the Android mobile device in 2022 (O'Shea, 2017). These statistics make the Android mobile device the most prominent and also a primary target of malware threats (Nokia, 2017).

b) **The increase of Android-based malware threat:** In the year 2017, the total number of malware threats recorded was 3.5 million with around 8,400 new malware being recorded. This trend is expected to continue every day the year until 2018 (Lueg, 2017). Reports indicate that, 87 percent of the Android mobile devices are exposed to malware threats and have become infected with a simple text message (Lab, 2017). This occurrence has caused a loss of MYR100,077,311.88 million to the mobile device especially in data breaches including operational losses and damages (Muncaster, 2017).

c) **The risk to mobile user:** Vulnerabilities and malware attacks in applications give attackers access to the mobile devices. This problem appears to affect mobile devices making users vulnerable to security risks. Malware can access sensitive information without user knowledge. One example is the Skycure Mobile Threat Risk Score which recorded that 30.23 percent of medium risks will be affecting mobile users (Skycure Mobile Threat Defense, 2016). Therefore, it is important to understand the risks and the severities caused to mobile devices so that users can be protected.

Despite the many research attempts to detect malware applications, there is still room for improvement in the malware detection system domain. The room for improvement can be attributed to current solutions which are still inadequate in providing users with protection from malware risks.

1.3 Statement of problems

As more sensitive information are being stored and accessed by mobile device users, the threat to these users also increases making them easy prey for malware attacks. In fact, 21.1 million Android mobile devices have been affected by malware applications that had been downloaded from Google Play Store (Dassanayake, 2017). Figure 1.1 presents the distribution of the types of mobile malware.

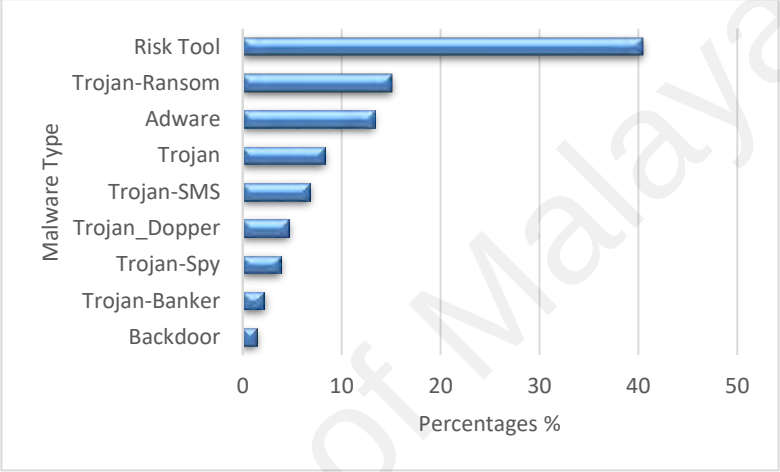

Figure 1.1: Distribution of mobile malware in 2017

The statistics indicate that Risk Tool (40.51%) was the most threatening (Unuchek et al., 2017) followed by Trojan-Ransom malware (15.09%). Clearly, most of the malware belong to the Trojan-Ransom type. This malware causes serious damages to mobile device users by making them subscribe to some unwanted premium services (Unuchek et al., 2017).

To analyse the risks and to detect malware applications, security analysts have implemented two type of analysis techniques, static and dynamic. However, these techniques were shown to be ineffective in analysing risks and for detecting malware applications when the attacker implements polymorphism into the application. Even though Google has introduced the Bouncer application (Oberheide et al., 2012) to detect malware applications, the threat cannot be alleviated as the threats seem widespread (Fox-Brewster, 2017).

Malware applications are capable of stealing users' account details, make them subscribe to premium messages via SMS and also compromise the hardware (Tam et al., 2017). The main problem with malware is that it conducts all these activities without the mobile device users' knowledge. Some benign applications in mobile devices may also carry a high-risk impact (Lookout, 2012; Song et al., 2016) thereby compounding the situation.

Malware detection achieved by deploying an Intrusion Detection System (IDS) using the static analysis or dynamic analysis approach. Nevertheless, both approaches also come with challenges. This calls for an urgent need to develop new risk analysis and new malware detection approaches that identify the risk of applications (Skycure Mobile Threat Defense, 2016; Saracino et al., 2016; Jackson, 2017).

1.4 Aim and objective

The aim of this study is to improve the current malware detection system for Android mobile devices and applications. The objectives of this study are thus:

i. To review the security vulnerabilities, challenges of each Android mobile application and establish the research gap by analysing the state-of-the-art malware detection system by investigating the properties of the mobile applications which are most critical with respect to the creation and sustainability of malware attacks on mobile applications.

ii. To propose a malware detection system that uses risk analysis to analyse the Android mobile applications, which is capable of analysing the structural properties of the Android mobile applications for detecting malware.

iii. To propose a malware detection system that is based on the time series approach by observing the behavioral properties of the Android mobile applications through time for the purpose of predicting future mobile malware.

iv. To evaluate the proposed system in terms of detection accuracy by using real-world Android malware and implement the prototype of the proposed system for a practical evaluation via a web-based assessment.

1.5 Research methodology

The entire study was carried out in four phases as shown in Figure 1.2. In the literature review phase, the security implications of the Android operating system was emphasised by focusing specifically on the state-of-the-art security solutions noted in Android risk analysis and malware detections. This study analyses the security vulnerabilities, risk analysis, and malware characteristics. It introduces the background of the malware analysis techniques and the detection methods in detecting malware including the IDS. A comprehensive taxonomy and the state-of-the-art IDS as well as a classification of mobile malware detections were then presented. This encompasses looking at the static and dynamic techniques, the signature approach, and the deployment approach. The chapter ends with the advantages and limitations of the study.

In order to carry out this study, several tools were deployed for running the experiments in the mobile malware tools phase. For example, the Androguard, ApkTool, R languages, and the IBM SPSS Statistics were employed. This study also introduced the features selection algorithms which include information gain, evolutionary algorithms, and bio-inspired optimisation algorithms in the tools used.

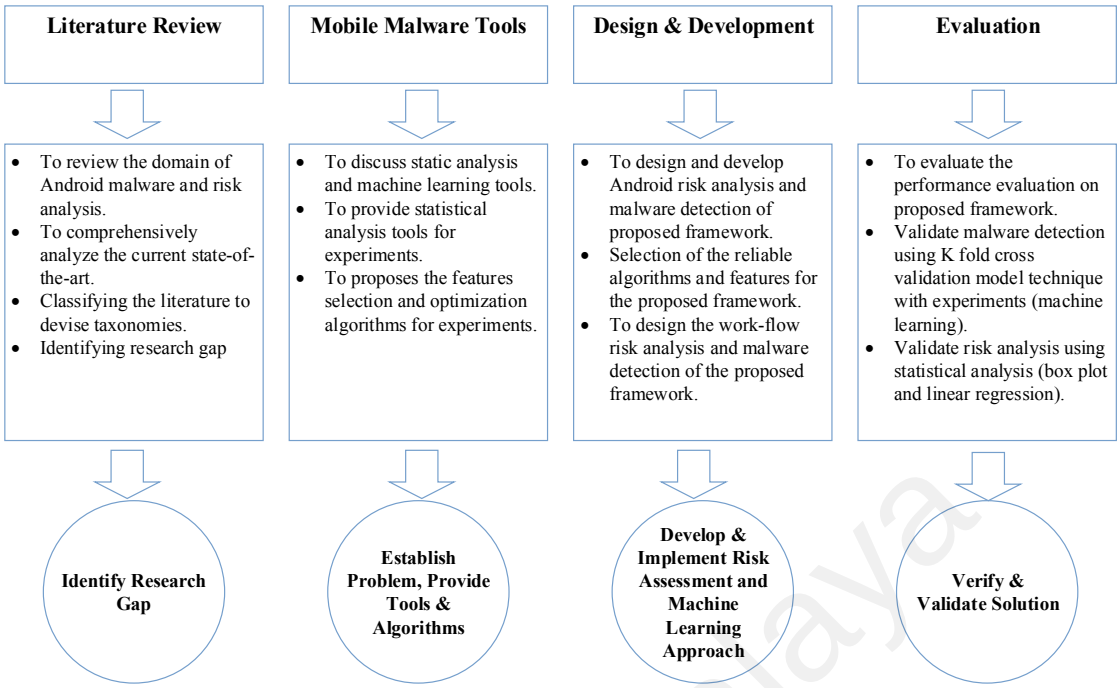

Figure 1.2: Proposed research design

The design and development of this study consists of four phases: data collection, features selection and extraction, and risk assessment evaluation. The data collection phase explains how dataset comprising benign and malware samples were gathered for use in the experiments. The samples were extracted and then labelled as accordingly as "Malware" and "Benign". The next phase selects static features (permission) while the final phase evaluates the risk analysis model.

Samples were retrieved from 5,560 malware samples from Drebin (Arp et al., 2014) and 5000 benign samples from the Androzoo dataset (Allix et al., 2016) and then evaluated. This was meant to show that permission features can project the effectiveness of the malware detection system.

The evaluation phase then evaluates the performance measure through seven benchmarks (i.e. accuracy, True Positive Rate (TPR), False Positive Rate (FPR), recall, precision, f-measure and Receiver Operating Characteristic (ROC). To show the significant performance and unbiasness of the proposed approach, this study employed a

ten-fold (i.e. k=10) cross-validation. A statistical analysis was then conducted to exhibit the performance of the proposed approach.

1.6 Summary

This chapter has provided the relevant information which encompass the background to the study, the motivation spurring this study, the research problem, the research methodology that this study incurs. The rest of this thesis is as laid out in Figure 1.3. This thesis is composed of seven chapters. Each chapter contains a part of the research work that was conducted to address the research problem and fulfill each objective of the study. Figure 1.3 illustrates.

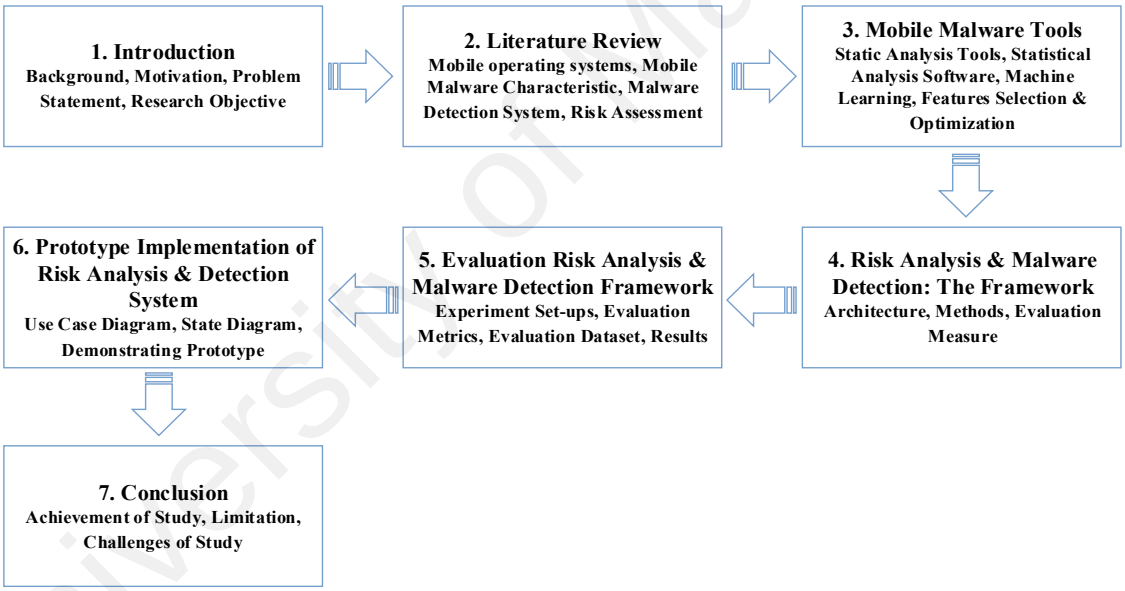

Figure 1.3: Thesis layout

Chapter 1 presents a brief overview of the study. It includes the background study outlining the Intrusion Detection System (IDS); it also discusses some of the proposed solutions. This chapter also states the problem statements that were formulated based on the findings of previous research by considering some gaps in the issues. A brief outline of the research methodology is then presented to show the steps used in achieving the objectives of this study and how the experiments were conducted.

Chapter 2 highlights the achievement of the first objective of this study. It introduces the various research undertaken in the field of Intrusion Detection System (IDS) discovery and the state-of-the-art mobile malware in Android mobile devices. This chapter expands on the horizon of malware detection by evaluating current literature that focusses on malware detection system. The classification of the malware detection system was devised by considering several aspects of the domain knowledge of the IDSs in Android mobile devices. This classification is necessary because it sheds light on how to discover malware and how to analyse malware threats that affect Android mobile devices. This chapter also identifies potential challenges that need important considerations in the future so as to develop a more effective malware detection system.

Chapter 3 discusses the tools used to conduct the experiments. It explains current approaches of the static analysis, machine learning tools, and other statistical analysis software. It continues with the review of relevant machine learning classifiers. This chapter also discusses the installation of the WEKA machine learning tool for malware detection. Finally, it looks at the feature selection and optimisation approached that helps to produce an effective Android malware detection system.

Chapter 4 presents the main contribution of this study which evolves around a novel framework that can be used as an Android malware detection system. The framework recommends using permission features with the machine learning and risk assessment approach. In presenting the framework, this chapter also introduces the characteristics and functionality of the framework as well as the rationale behind it. It also offers an insight into the evaluation measure, used method, and services offered by the framework.

Chapter 5 highlights the achievement of the second and third objective of this study. It focusses on the evaluation measurement that was applied in the experiments; it also analyses the effectiveness of the proposed method. The results highlight the performance

analysis and the ROC curve graph. The results obtained from the experiments were derived from using the selected classifiers of the WEKA machine learning tool. This chapter also describes the risk analysis through the risk assessment approach.

Chapter 6 highlights the achievement of the fourth objective of this study. The chapter presents the website development as a prototype which practically utilises the proposed features to detect the unknown malware. It provides an overview of the system development which consists of uploading and reversing the engineering applications. It also identifies and extracts the proposed features and the machine learning predictions. In addition, this chapter illustrates the use of different samples of malware extracted from a reliable source in testing the efficiency of the prediction.

Chapter 7 presents the conclusion to the study. It considers the results obtained as the achievement of the research objectives and the contribution of this research. It highlights the significance of the proposed solution. It also states the limitation of the research work. Finally, it discusses directions for future research that relevant to this area of discipline.

CHAPTER 2: MOBILE DEVICE EVOLUTION, MALWARE CHARACTERISTICS AND DETECTION SYSTEMS

This chapter covers the first objective of the thesis. It presents an overview of the security aspect of the Intrusion Detection System (IDS) as a leeway to discuss the vulnerabilities found in the Android mobile applications. The objective of this chapter is to highlight the significance of risk analysis and malicious detections on mobile devices which have been neglected thus far. The background of mobile malware is reviewed to gain insight into the problems faced by the Android mobile device. The classification of mobile detection systems such as analysis techniques, detection approaches, and various other deployments used is also included. The threats faced by mobile device users are discussed before the chapter concludes with a short summary.

2.1 Mobile device evolution

This section unveils the comprehensive information of the mobile operating systems, mobile devices, IDS and threats posed to mobile devices. It is important to describe the history and nature of a well-defined research problem with reference to the existing literature.

Personal computers (PC) and mobile devices are ubiquitous in today's landscape because of their highly personal and easy to use features followed by their portability and powerful attributes. Such devices are in high demand due to the advancement of technology. Between the two, mobile device shipments have surpasssed PCs (Egam et al., 2016, Egham, 2015b). Gartner, Inc. estimates that the use of worldwide mobile devices will reach 1933 million units in 2018, an increase of 1.2 percent from 2017 (Egam et al., 2016) while PC shipments are expected to exhibit a three percent increase in 2018. The mobile device market is maturing, reaching a global saturation with phones that are

increasingly high tech and more capable than before. Table 2.1 shows the worldwide device shipments between 2016-2018 (Millions of Units).

Table 2.1: Worldwide device shipments in 2016-2018 (Millions of Units)

Device Type	2016	2017	2018
Personal Computers (PC) Market	265	266	274
Mobile Device	1887	1910	1933
Total Device Market	2152	2176	2207

It seems clear that mobile devices will lead, surpassing others by the millions. This occurrence is caused by the polarisation of mobile devices with prices ranging between the high end to the low end market prices. Of the operating systems running the mobile devices, it appears that the Android and the iOS are in high demands. Gartner Inc. expect the market for mobile devices to grow 3.5 per cent in 2017. Accompanying this growth with newer designs and newer features that attractive enough to convince more buyers to replace their PCs with mobile devices (Egam et al., 2016). Expectations also indicate that mobile devices continue to do well globally in the next few years especially in developed countries thereby causing bigger shipments and generating more profits.

The world's mobile device shipments have expanded but the (Egam et al., 2016) the IDCs are noticeably experiencing a slowdown. This is explained further. The Android mobile device operating system is currently dominating the world market with 86.8 percent share in the third quartile of 2016 and Samsung tops them all (IDC, 2017). Figure 2.1 illustrates the statistics showing the trends of the mobile operating system from March 2016 to January 2017. Clearly, two (2) mobile device operating systems stood out showing the positive growth rate of the two rivals, the iOS and the Android operating systems. Between the two, Android has maintained a growth rate of more than 60 percent.

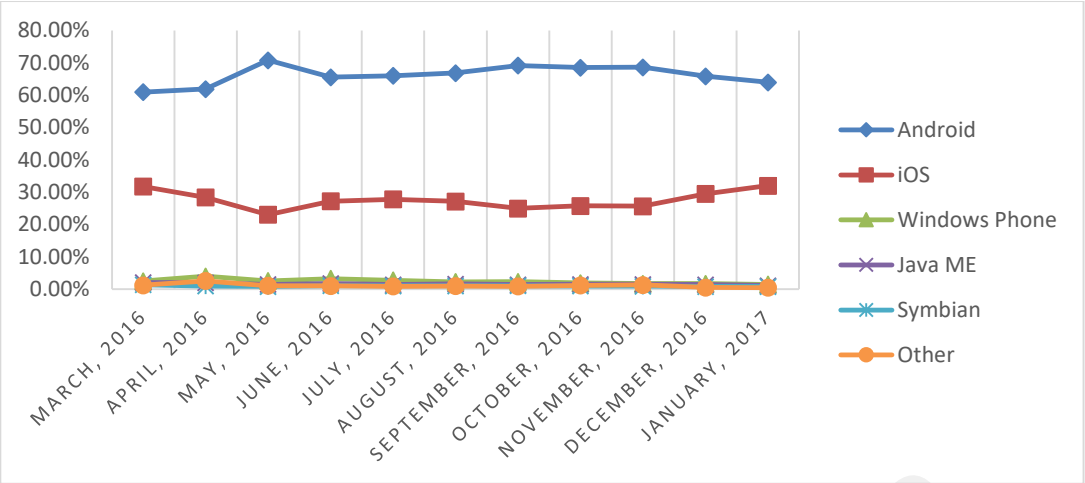

Figure 2.1: Mobile operating system trend

Android's exuberance sparked in May 2016. This is due to Google updating an Android version called Marshmallow (Kellex, 2016). This Marshmallow operating system had increased the popularity of the Android operating system because it offers many sophisticated functions. Other mobile operating systems such as Windows Phone, Java ME and Symbian, in comparison, showed a declining trend with below a 10 percent growth rate. Figure 2.2 presents the market share of the various mobile operating systems and the respective dominance owned by Android and iOS.

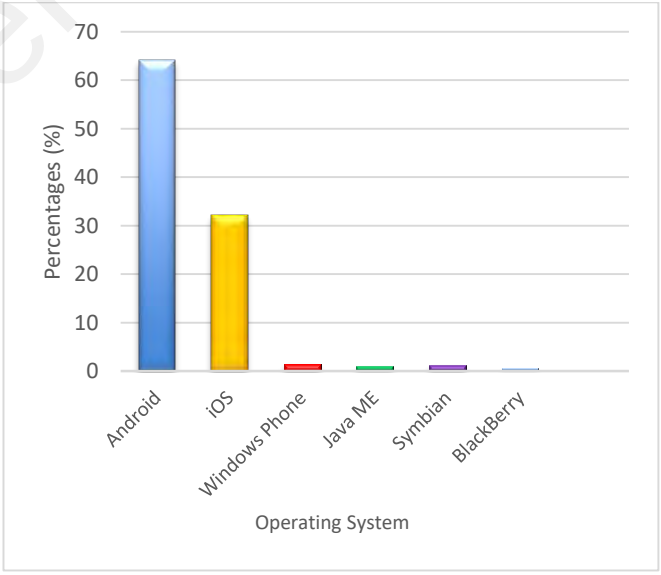

Figure 2.2: Percentages of market share in mobile operating systems in 2017

The figure also shows the top five (5) market shares with Android dominating the peak. The IDC (2017) claims that the global smartphone market has grown 1.1 percent on a yearly basis and in the third quarter of 2016, there had been 363.2 million shipments throughout the world (IDC, 2017). With Samsung currently dominating the smartphone market and Samsung continuing to climb the chart in the future, the Android operating system seems born to lead. In contrast, the iOS market shares for the third quarter of 2016 had grown by only 12.7 percent with 45.5 million shipments. This growth is attributed to Apple's newest smartphone model, the iPhone 7. Windows Phone, unfortunately, had experienced a decline of 35.2 percent with only 974.4 thousand units being shipped for the third quarter of 2016 while the Android market share had increased 7.1 percent across Europe in the first three months of 2016. Today, it holds 75.6 percent of the market shares compared to Apple's 18.9 per cent which had dropped from 20.2 per cent (Rhiannon Williams, 2016). In the operating system (OS) market, Android had surpassed a billion shipment of devices in 2014 continuing to grow at a double-digit pace in 2015 with a 26 percent increase year after year (Egham, 2015a). This undoubtedly makes Android the most prominently used mobile device operating system, as illustrated in Figure 2.3.

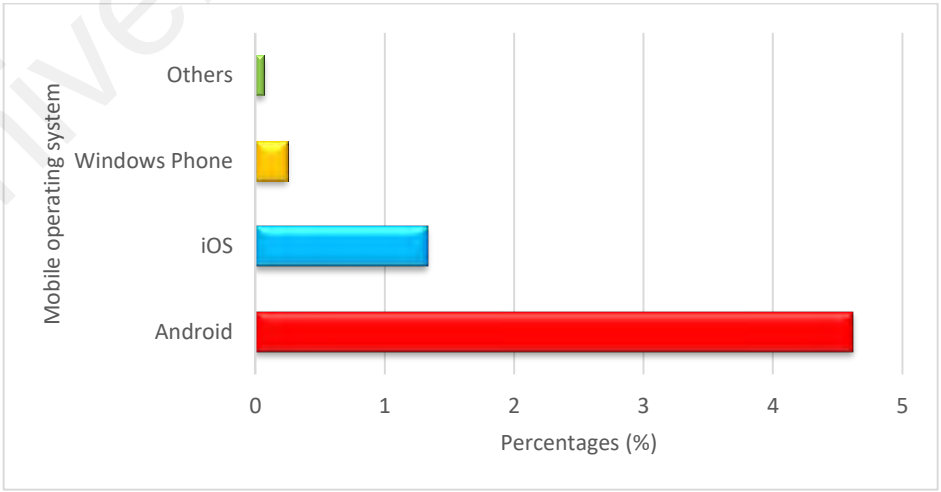

Figure 2.3: Percentage of usage in mobile operating systems

The Android operating system is an open source operating system. In 2014, a total of 204.4 million units of mobile devices were installed with the Android system. In European countries (EU5) alone, like France, the United Kingdom (UK), Germany, Italy and Spain, the leading operating system is led by Android (74%) as opposed to iOS (14.4%). This statistic shows that most users in the EU5 prefer Android-based devices. The popularity of Android based devices can also be traced to the 1.5 million units of mobile devices being installed on a daily basis (Amadeo, 2016) as portrayed in Figure 2.4.

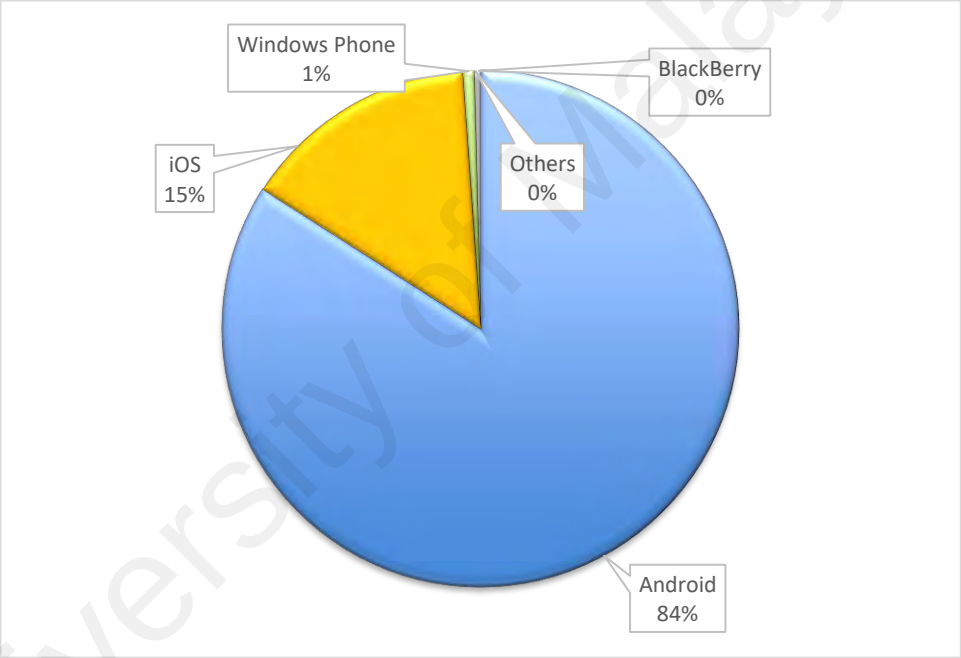

Figure 2.4: Percentages of worldwide mobile device sales by operating systems in 2016

The year 2008 saw the slowest sales and growth of global mobile devices (Egham, 2015c). Only 403 million units of mobile devices were successfully sold within the world in the fourth quarter of 2015 (Egham, 2015c). By 2014, the figure had increased with the value of 9.7 percent in the same period of time. On the whole, the sale of mobile devices reached 1.4 billion units between 2014 to 2015, showing an increase of 14.4 percent (Egham, 2015c). In the first quarter of 2016, there was an increase of 3.9 percent of global

sales of mobile devices which exceeded 349 million units (Egham, 2016). Since its introduction by Google in 2007, Android has become the leading operating system in the world as illustrated in Figure 2.4. Since its release, the Android has grown in strength with 78 percent mobile devices running on the Android operating system. This is equal to 220 million of Android mobile device sales in 2013 (Statista, 2017). By 2015, Android mobile devices had sold more than 1.16 billion units. In 2016, the Android mobile device had increase 85 percent of its sales worldwide.

Besides the Android, the second most popular smartphone operating system based on sales is Apple's iOS. This company has sold over 50 million units of mobile devices in the final quarter of 2013. For the whole of 2013, the Apple iPhones sold over 150 million sets worldwide. Nonetheless, Figure 2.4 indicates that Apple's iOS system remains to be behind Android.

2.2 Mobile operating systems

This section presents the general overview of the several outstanding mobile operating systems. This section is important because it provides information showing the differences of the various mobile operating systems as well as their advantages and disadvantages as projected in Table 2.2 and Table 2.3 respectively.

2.2.1 iOS operating system

The iOS is a proprietary operating system which belongs to Apple; it is only installed in Apple's devices. The strict requirement of Apple makes it challenging for developers to upload the iOS application into Apple Store. In addition, Apple's fees for applications are much higher than Android or Windows. It is different from the Android operating system that is introduced by Google that comes with an open source environment which enables multiple vendors to have access to its system. The iOS is a proprietary operating system that is controlled solely by Apple for Apple's own devices only.

2.2.2 Windows

Windows mobile operating system is similar to the iOS in that it is individually reviewed by system who then give the approval for all applications to be submitted to the store thereby eliminating malicious applications from gaining access to Windows Store. Due to the review ability, Windows mobile operating system does not require a dedicated anti-malware and software anti-virus.

2.2.3 Android

Android mobile operating system is an open source system used on mobile devices such as smartphones and tablets. Opened to multiple vendors, the Android operating system is also the most used among all mobile devices. This, inevitably, has attracted many malware attackers who want to penetrate the system by taking advantage of the users. Unlike Apple and Windows, Android is the easier prey for attackers because it is much easier to submit and to get applications accepted into Google Play Store. The Google Play Store contains Google Bouncer which is a malware scanner. It was developed to protect users. It main function is to analyse and identify available applications in the Google Play Store. Table 2.2 lists the comparison of the various mobile operating systems.

Table 2.2: Comparison of mobile operating system

Type	Android	iOS	Windows
Proprietary	Open source	Close source	Close Source
Application store	Google Application Store	Apple Store	Window Phone Store
Device manufacture	No	Apple only	No
Operating system based	Linux	Darwin	Window
Access to external storage	Yes	No	Yes

The above information indicates that majority of the software used are closed source software. None of these mobile operating systems produces its own mobile device except

for Apple. Table 2.3 lists the advantages and disadvantages of the various mobile operating systems. Here, it appears that Android has lesser secure features comparatively.

Table 2.3: Pros and cons of the mobile operating systems

Operating Systems	Pros	Cons
Android	• Available on a large range of devices • Open source operating system • Anybody has capability to submit application to Google Play Store	• Android holds the majority of smartphone users making them more susceptible to malicious attacks • Since Android is run on many different devices, not all of them support the newest OS. This is problematic due to security updates • Not as secure as iOS and windows OS
Window	• Provide support from Microsoft Services • It is more secure compared than Android and iOS because it has sandboxing, secure boot and data sync.	• As more users adapt to this OS, there would likely be more vulnerabilities that are found. • Sharing function is less than Android and iOS
iOS	• Proprietary operating systems • Improving on secure app submission process, whereas required the applications are signed by certificates that are checked using Apple's servers.	• Difficult to integrate and sharing file with different manufactures • Like Android, a large number of mobile users also own Apple devices. This alone poses a risk as it is more susceptible to being a target for attackers.

From the table above, it derived that mobile devices with the iOS, Windows or Android operating systems are capable of doing similar functions such as messaging, calling, connecting to the Wi-Fi and taking photos. However, the open source system of the Android and its capability to be installed by a number of mobile manufacturers make it an easy target for malware attackers.

2.3 Android operating system

As technology becomes more woven into the fabric of society, the mobile device landscape also continues to grow and evolve. This has been accelerated by the improvement in technology, the increase in power, the abundance in storage space and the multitude of applications available, thereby making the mobile device susceptible to malware attacks. Current mobile devices offer many accessibilities such as online banking, online shopping, online applications for jobs, gaming, music and e-purchasing of air tickets or hotel reservations. The number of users installing the Android system is also multiplying enormously. The International Data Corporation (IDC) has predicted that the Android operating system powered by Google will experience a more positive exponential growth than the iOS (International Data Corporation (IDC), 2016).

In 2016, Android's operating system had grown 6.2 percent garnered by 1.24 billion shipments. It is expected to increase to 1.57 billion in 2020. In contrast, the iOS system is expected to decline by -2.0 percent (International Data Corporation (IDC), 2016). The growing trend illustrates the dominance of the Android operating system. As an open source system, Android runs on the Linux-based operating system that was developed by Google (Gheorghe et al., 2015). This has transformed the Android operating system to be more popular than ever besides its unified approach in application development. This means that all Android applications are able to run on any Android devices.

Mobile users using the Android operating system easily download a variety of Android applications from Google Play store. These applications include a mix of free as well as premium applications that require payments. In total, there are 2,449,044 numbers of Android applications (AppBrain, 2016). In this regard, Google Play store is its official market.

To constantly support the wave of new technologies, Google Play store also constantly updates its version of Android software (Razak et al., 2016; Firdaus et al., 2017). Most Android versions are themed with sweets and desserts and sorted in alphabetical order. The latest version of Android provides a great API for applications. For example, the new version of the Android Marshmallow aims to save battery life; is user friendly and it provides more control to users as demonstrated in Table 2.4 which also highlights their codenames (Developer, 2016c).

Table 2.4: Android version

Version	Codename	API
2.2	Froyo	8
2.3.3 - 2.3.7	Gingerbread	10
4.0.3 - 4.0.4	Ice Cream Sandwich	15
4.1x	Jelly Bean	16
4.2.x	Jelly Bean	17
4.3	Jelly Bean	18
4.4	KitKat	19
5.0	Lollipop	21
5.1	Lollipop	22
6.0	Marshmallow	23
7.0	Nougat	24
8.0	Oreo	27

As seen in the table, the API in the Android versions has increased. This increase is important because it helps users and developers to install an Android application based on mobile device characteristics such as screen size. Furthermore, the current Android API also supports the older versions as well. This makes it easier for users and developers, thereby expanding market growth. The Android system is made up of a certain architecture.

2.3.1 Android architecture

Android is designed in software stacks which are customised for mobile devices. It has six (6) layers: Linux kernel, Hardware abstraction, Libraries, Android runtime, Java API framework and System applications. Each layer provides different services to users to perform their functions. Figure 2.5 illustrates the major components of the Android system.

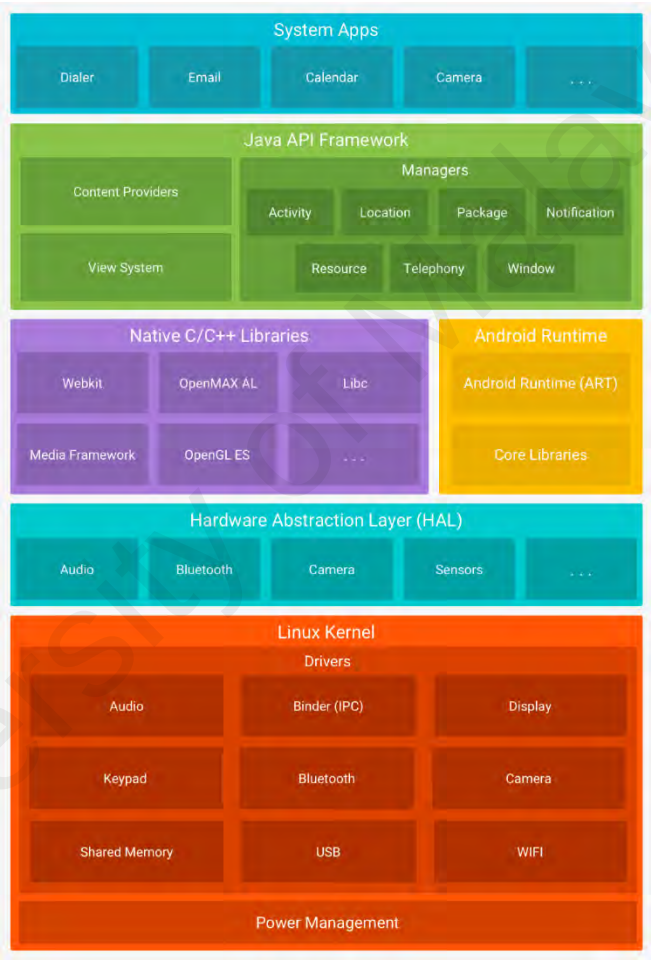

Figure 2.5: Android system architecture

The detailed architecture of the Android begins from bottom up. The figure shows that each layer of the stacks and the corresponding elements within each layer are tightly integrated and carefully tuned so as to provide users with optimal application development and execution. Table 2.5 describes the Android system's architecture (Developer, 2016b).

Table 2.5: Description of the Android system's Architecture

Type of layer	Description
Linux kernel	Linux kernel is at the bottom of the entire layer and represent as the heart of Android architecture as well as foundation of Android platform. This layer is important because it responsible for device driver and allows Android to take advantage of key security features.
Hardware abstraction layer (HAL)	The hardware abstraction layer (HAL) defines a standard interface for implementation between hardware and driver. It allows implementing functionality to the higher-level Java API framework. When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component. It also consists of multiple library modules, each of which implements an interface for a specific type of hardware component, such as the camera or Bluetooth module.
Android Runtime	Android runtime provides core libraries and Android Runtime (ART). The core libraries enable Android developers to write Android applications using standard Java programming language. ART is responsible to run Android application. For Android version 5.0 (API level 21) or higher, each application running within their own ART and process. The ART able to execute multiple virtual machines on low memory device using DEX files.
Native C/C ++ Libraries	Top of HAL consist of native libraries such as Webkit, OpenMax AL, Libc, media framework and OpenGL ES. The Android system component and services are built from native code written in C and C++. Webkit library is used for browser support.
Java API Framework	The operating systems of Android are written in Java language while Android API provides classes and interface for development Android application. Java API framework consists of content provider, view system and managers.
System applications	Android system applications are on the top of Android architecture. It consists a set of core applications for contact, email, camera, web browsing and SMS messaging. Various applications created by developers like tools, games, browser and social media are installed in this layer.

Based on the architecture and description, developer able to develop applications and to become a good Android developer, a clear understanding of the Android system's architecture is necessary. Since the Android operating system is an architecture of stacked software encompassing the Linux kernel, hardware abstraction layer, Android runtime, native C/C++ libraries, Java API framework and system applications, users are protected from resource consumption. This is because Android's system architecture was built to ensure that it functions with efficiency and offers a great performance.

2.3.2 Security model in Android devices

Android's mobile device security has always been a crucial topic. Aside from the calling and messaging functions, Android users also use mobile devices for connecting their digital life such as photo sharing, social networking, emailing and internet banking. As a result, the user stores valuable data information on their mobile devices. This valuable information is confidential and used irresponsibly by others for impersonation and blackmailing purposes, hence, attracting attackers. These attackers are interested in using the valuable information to harness profits for themselves. For examples, attackers apply social engineering mechanisms to attract users to subscribe to premium SMS services. This service is very costly and when users fall prey to this scam, they encounter many financial losses as well as problems.

As the aforementioned problem becomes widespread, a more significant security mechanism is needed to overcome threats faced by mobile device users. Threat has the potential to cause serious harms to mobile devices. Among these threats, mobile malware remains a significant cyber security threat. Suarez-Tangil et al. (2014) highlighted three (3) security features which are incorporated into mobile devices: a) security measure implemented at the market level, b) security measure implemented at the platform level and c) others types of security mechanisms. Market protection is a primary defense against malware applications, preventing them from entering the distribution market. Two (2) protection approaches are applied at the market level. They are application reviews and signing. Both protections are, however, insufficient to protect mobile devices from malware. Security at the platform level aims to restrict the malware application from executing on mobile devices while other Android security mechanisms applied at the platform level (e.g. permission) includes sandboxing followed by interactions between application platforms. Other security mechanisms being offered by others are the research works done on analysis and in detecting malware on mobile devices.

It is important to develop an Android security mechanism for mobile device users because no individual able to survive without his/her mobile device. The individual's basic, personal and important information are all located within that mobile device. Such information and identities stored in the mobile device can be compromised by attackers. Application permission, application signing, and component encapsulation are types of security mechanisms used to thwart a malware threat on the Android (Gheorghe et al., 2015). As is understood, all Android applications must be signed with a certificate before they are installed. However, the certificate is unable to control applications installed in the mobile device. The important point for users to understand about application signing is that the Android system is unable to run an application that is not signed in properly.

Furthermore, the Android application encapsulates different components which unable to access from an external entity (Gheorghe et al., 2015). This external entity needs permission to access certain application components. Another security mechanism is sandboxing which is used to isolate running applications by using mandatory access control policies. Sandboxing has the capability to protect the mobile system from malicious applications to a certain extent. However, this sandboxing has flaws. When user grants permission to install the applications, it is unable to protect the mobile system from exploiting attackers.

Application permission is a special privilege that needs to be granted before an application is installed. This permission system always asks permission from users to access sensitive information such as picture, contact number, email, location, and documents. This sensitive information is used by irresponsible users to threaten the user's privacy. By default, application has the permission unable to perform any operations that would adversely impact the user, other applications and the operating system (Developer, 2016a). This includes reading and writing the user's sensitive information, keeping the

device awake or performing a network access (Developer, 2016a). There are more than 100 predefined permissions stored in the Android architecture but developers are able to declare custom permissions (Gheorghe et al., 2015). The Android permission system is divided into several protection levels: normal, dangerous, and signature or system. Android applications request permissions based on the defining permission of the AndroidManifest.xml file. This permission is used to notify users about the risk of installing the application (Felt et al., 2012). Table 2.6 lists the levels of Android protection.

Table 2.6: Level of Android level protection

Level protection	Descriptions
Normal	A lower risk permission that give granted permission to access isolated application level features with minimal impact to other system, applications or user. This state, the system automatically granted access to install application without user approval.
Dangerous	A higher risk permission whereas require to grant access from user to access on confidential information and have negative impact. Due to the high risk, the system may not automatically grant access to the requesting applications.
Signature	Provide grant access to the application whereas already signed with the same certificate as the applications declared the permissions. The system automatically grant access the permission when the certificate is match.
SignatureOrSystem	This type of level protection is used on specific task only when multiple vendor requires to build an application into system image. The permission only granted when the application has sign with same certificate as declare on permissions.

Table 2.6 illustrates the characteristics of the risks implied in the permission. It is important to note that this level of protection determines whether or not to grant requesting permission from an application for security purposes. This security feature has the capability to reduce the impact and frequency of the application's security issues. In addition, it makes application developers follow the system design that comes with the default system and the file permissions. The following section discusses the threats posed to mobile devices.

2.3.3 Threats on mobile devices

In 2014, Symantec detected more than 317 million new malwares. In the same year, PandaLabs was able to neutralise 75 million malware (Lopez, 2015). This figure demonstrates that nearly one (1) million of malware are released every day (Symantec, 2015). This explosive growth in malware will continuously infiltrate the system and mobile devices thereby posing threats to privacy issues as well as financial losses. This occurrence makes mobile devices a more vulnerable target for cybercriminals. In 2016, the Android operating system was suffering from major vulnerabilities (McAfee, 2016) posed by Remote access tool (RATs) and ransomware involving bank frauds. Due to this problem, Google took a very serious stand in updating the security system.

Mobile devices face three types of threats. It is first threatened by the growing presence of malware that is able to slip into the application store without being noticed. Second, it is threatened by the slow security updates. For instance, the Android monthly security system gets updated but it is always late in rolling out the updates for users. Finally, it is threatened by attackers who are now more sophisticated in expanding their targets into the mobile environment. According to the McAfee Mobile Security Report, some favorite mobile applications may actually be running some malicious behaviors such as tracking users' location data, oversharing users' personal information, looking for users' contact information as well as installing malware. These behaviors are used by attackers who sell the stolen information for profit incentives. Figure 2.6 illustrates the type of information collected from mobile devices.

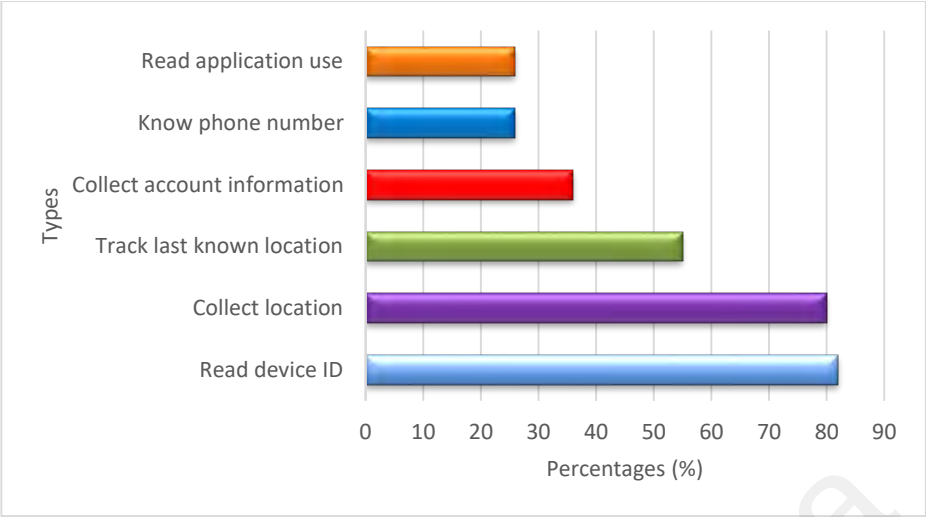

Figure 2.6: Percentages of information collected from mobile devices

The figure above shows how applications invade users' privacy on mobile devices. In 2014, a total of 82 percent of applications were used to track users' mobile activities and 80 percent of these had actually collected users' location information (Caetano, 2014). The location information that was collected comprised users' exact location, general location and the last known location. Some major application had also collected information about users' Wi-Fi and data networks. This behavior of applications places mobile users under great risks. To avoid these risks, all mobile applications that accessed sensitive information should be analyse in host-based or network-based deployment approaches.

2.4 Mobile malware characteristics

Malware is a programme bearing malicious intents which are posed by generic forms of hostile applications. Malware has become a severe threat to interconnected mobile devices for years. It is a particular type of virus used by attackers to infiltrate the mobile devices of users so as to collect user activities and sensitive information that can be used to perform unauthorised operations. All the information gained are sent to unscrupulous authors through the network connection who use these for profits. Guarding against malware attacks is becoming an increasingly complex process. For example,

unscrupulous authors apply obfuscation through stealthy techniques such as polymorphism, encryption and metamorphism to escape from anti-malware detection systems (Gandotra et al., 2014). Unscrupulous authors can design various types of malware such as botnet, Trojan, rootkit and worm (Karim et al., 2014) for their benefit. Each malware has its own goals and usually causes undesirable results (Wu et al., 2014). Table 2.7 illustrates the common types of malware.

Table 2.7: Common malware types

Type	Description
Worm	Worm infect the operating systems by multiplying itself to affect the operating systems and sending copies of itself through networks.
Trojan	Trojan able distinguish as a normal application to attract user for run its. After successfully run, Trojan take over the resources and able to disrupt the availability of operating system with denial of service.
Rootkits	Rootkits is difficult to detect because it start the malware activities while the user is not using the computer.
Botnet	Botnet allow attacker to take control over the infected computer. The infected system known as a zombie and always spread themselves through the network.
Spyware	Malware application that uses eavesdropper technique to reveal user's private information.
Backdoor	Specialize Trojan horses that masquerade itself to enable remote access and bypass authorize authentication to take control of the infected system.

As seen, the most dangerous malware are those which silently infect mobile devices through fake websites, counterfeit software updates, spam email attachments and fraudulent applications. These malwares appear as attractive and useful tools for users. This is how malware manipulate users into executing malware applications. By the time users realise this, it would have been too late. The situation becomes worst when sensitive information leads to financial losses (Fang et al., 2014). In order to thwart the break-neck exponential growth of malware, it needs a significant approach to detect and analyse these applications as well as give quick responses to users.

2.4.1 Research on mobile malware

There are existing approaches such as firewall, antiviruses and the Intrusion Detection Systems (IDSs) which aim to overcome malware attacks but the noticeable spikes of the aforementioned malware statistics suggest that current mechanism is still inadequate. Honeypot and computer forensics are able to expose the malware behaviors by analysing malware behaviors and researchers are able to identify the attack mechanism and goals through analysis. These activities enable them to implement an effective malware detection approach that protect mobile device users from malware. Figure 2.7 illustrates the publication trends that are related to mobile research.

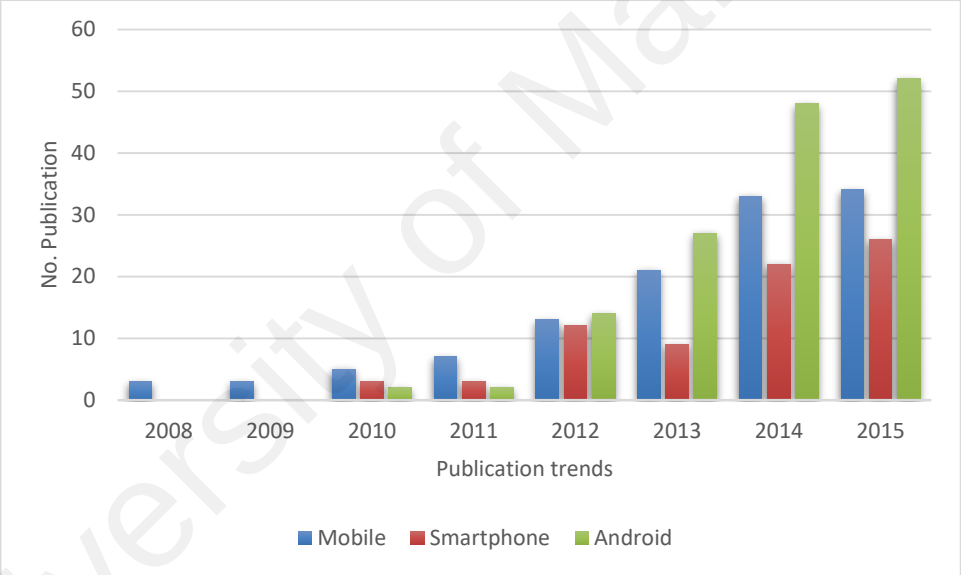

Figure 2.7: Publication trends

The above statistics show three (3) categories of publication trends which include mobiles, smartphones and Android. Presently, Android has become a popular target for malware research encompassing 42.8 percent of publications. This shows that the current issue involves Android, a trend that has been growing since 2012 until 2015. It is also expected to increase for the next few years. Android malware is best described as the new direction for research in security.

As described previously, Android is a mobile operating system made by Google (Schmeelk et al., 2015). It is installed on a variety of mobile devices and it offers Google services like Google search, Gmail, YouTube, and Google maps. The Android also delivers a free application for downloads which are easily installed on mobile devices. Such services fascinate user's attention and so it further encourages them to use the Android mobile operating system. Android, as discussed earlier, is more popular than other operating systems (Apvrille et al., 2012) and Gartner estimates that 60 percent of the mobile devices are installed with the Android operating systems (Egham, 2015a).

Besides downloading applications from its official website Google Play store, Android able to download applications from third party markets such as SlideMe, GetJar, and Amazon's Appstore (Narudin et al., 2016). Android applications are free for downloading but some payment may be required for full premium versions. It is hereby mentioned that applications downloaded from third party markets are normally done so manually without going through a store. In this regard, it makes Android a trendy mobile operating system that is the main target of malware.

2.4.2 Infected vectors

This section presents the methods for preventing malware infection on the mobile device system. It is important to show how infected vectors play their role in ensuring that their attacks penetrate the mobile device system.

a) **The insecurity of mobile device:** The mobile device threat landscape has continued to grow and evolve with several contributing factors such as the increasing speed of technology, the expansion of data storage, the increase in power and the vast mobility offered by mobile devices. All these factors have certainly made user's daily lives more convenient as they use their personal mobile devices for various transactions such as online shopping, online banking, e-payment of utility bills and also for social

interactions such as playing games. These conveniences have certainly made mobile devices more prone for attacks by unscrupulous cybercriminals. The reality of the matter is that mobile devices and their core applications and operating systems are important. Therefore, these become vulnerable when the Android operating system is also open sourced. In addition, the altered configurations happening on mobile devices also make the system more vulnerable for attacks.

b) **Insecure application threat:** Currently, there are millions of applications that are available in application stores such as Amazon and Play Store. The newer applications are being created by developers, the easier it is for attackers to attack. This is because application developers do not focus on security. On top of that, application developers are only good with their application designs and specifications as well as usability. They tend to overlook application security particularly on private information. Added to this is the false sense of security that these applications provide to user. Most application stores claim that their sites are secure giving users this false impression that an application is safe to be downloaded and installed in mobile devices. In reality, some of these application stores actually contain malware applications that expose users to risk. Furthermore, most of the applications require the Internet to update and communicate their services. Indirectly, malware applications uses the Internet to get the data during photo and other exchange of details. Moreover, malware such as Botnet is able to open up Web services and give the applications permission to access personal information.

c) **Network based threat:** Spam, phishing and adware come under the same type of malware that uses the Internet. Mobile devices that are infected with these malwares, as a result, also impact the reputation of the application stores and the mobile's user. Besides that, free Wi-Fi hotspot offered by scammers because it is a source for capturing user's personal data such as online banking passwords, credit card and

contact numbers. As an example, worm uses the push based scheme to find vulnerability on network services so as to infect mobile devices. Based on this, it was said that mobile users should avoid using free Wi-Fi hotspot if this unprovided by reputable mobile operators or business agencies.

d) **Drive by download:** The drive-by download is the most common web browser which also contains malicious web pages that able to infect mobile users. It basically refers to users downloading applications without knowing the source. This accomplished by users triggering a drive-by download which click on some malicious links in text messages or pictures. In this case, malicious applications pretend to look like legitimate applications thus when clicked, it exposes users to unwanted applications and services. This kind of services activate the SMS message from the user's mobile device and then sending a request for premium services without user knowledge. As a result, users are charged and at the same time, the attacker deletes any receiving SMS messages which acknowledge the charges. In order to protect mobile devices from these threats, the user should stop immediately when getting any suspicious pop-up screens that asks users to click on it. Besides that, users need to make sure that the anti-virus and operating system is up to date.

e) **Social engineering:** Any kind of action where a user is lured into executing malicious codes in his/her mobile device rather than using technical hacking techniques. This is known as social engineering attacks. In the past years, social engineers trying to fool users by getting them to give up sensitive information; they pretend to be a good friend on the social network which lures the users into clicking into malicious web sites. In mobile devices, this method is used to penetrate the user's smartphone and gain access into their confidential information.

2.5 Malware detection system

Malware is a malicious software that can access mobile and computer devices with the aim of extractjng personal information thereby causing serious damage to the system. Although existing systems such as firewall, antiviruses and the IDS are available in overcoming malware attacks, more novel approaches are necessary. This is because through new technologies, malware authors are able to use novel approaches to avoid detection. Studies Razak et al. (2017) exploring malware domains are some mechanisms that contribute to this detection. The study of malware is about investigating and analysing malware characteristics in order to propose a new approach that detect malware. Studies such as Tang et al. (2014) and Sahs et al. (2012) had applied the machine learning approach to detect malware. In contrast, Nadeem et al. (2014) applied adaptive responses as an approach to halt attacks, mitigate damages and prevent attacks in a mobile ad hoc network (MANET). These studies demonstrate that research activities conducted in this domain are significant. Figure 2.8 shows the various classifications of the malware detection system.

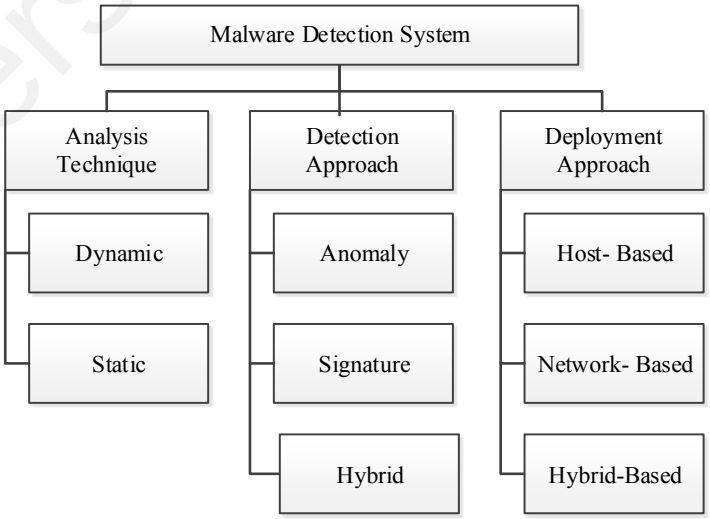

Figure 2.8: Classification of malware detection system

In this study, the existing malware detection systems are classified based on three components: a) analysis technique, b) detection approach and c) deployment approach. These classifications are important in showing the relationship of malware detection system.

2.5.1 Analysis technique

Malware analysis is a process of examining the malware code and identifying the dynamic characteristics of the malware. Unscrupulous authors strive to avoid malware analysis with obfuscation (Sharif et al., 2008) or the packer and anti-debugging technique (Rad et al. 2012; Xie et al., 2013; (Mamoun Alazab et al., 2010). These techniques make the examining of malware analysis harder thus enabling the malware to better hide their devious intentions.

A malware analysis is performed for the purpose of examining the components through the dissection of application codes and behaviors (Zhou et al., 2012; Platforms et al., 2013). Malware analysts need to apply caution during such examinations so as to alleviate the spread of contamination. To analyse the malware, analysts need a proper environment setup that ensure security and prevent infection (Razak et al., 2017). The process of a malware analysis begins with an isolated environment such as a virtualisation software (Damopoulos et al., 2012; Gonzalez et al., 2014). Two most commonly known malware analysis techniques are static analysis and dynamic analysis (Ravula et al., 2013; Gandotra et al., 2014). Table 2.8 further illustrates.

Static analysis applies reverse engineering, similarity and command techniques (Veerwal et al., 2013) whereas dynamic analysis analyses the malicious behaviors and error programmes through observations conducted in the controlled environment (Ghiasi et al., 2015). Static analysis has fast detection but its major problem is its unable to detect when the malware applies obfuscation technique. It is able to examine malware without

being able to execute it. This technique is also able to read the code programme, determine the goals and then detect malware (Talha et al., 2015). Unscrupulous malware authors apply other techniques like polymorphism, metamorphism, and encryption to evade such detections (Rad et al., 2012).

The dynamic analysis is capable of detecting unknown malware, executing the malware through monitors in a controlled environment (Egele et al., 2012; Seideman et al., 2015) even when the malware applies obfuscation.

Table 2.8: Types of malware analysis

Analysis technique	Advantages	Disadvantages
Dynamic	• Able to detect unknown malware	• Time intensive • Resource consuming
Static	• Fast detection	• Unable to detect malware with obfuscation technique

2.5.2 Detection approach

The two common detection approaches seen in the IDS are anomaly (Feizollah et al., 2013; Elshoush et al., 2011) and signature (Yassin et al., 2012; Hubballi et al., 2014).

a) **Anomaly-based approach:** The Anomaly-based approach detects malicious activities by monitoring the level of activities seen in the network traffic and systems (A Shabtai et al., 2014; Narudin et al., 2016). The anomaly detection approach is better in comparison because it is able to detect new and unfamiliar attacks through the normal and abnormal patterns noted. Any abnormal pattern noted in the mobile device is considered a malware attack. The changing behavior of users moving from normal to abnormal patterns is also considered an attack. This approach however, is limited by several factors. It requires a complex process to examine the effective features for the learning and training processes. To optimise the process of

determining malware behaviors, it is important to collect specific mobile malware features. For example, mobile applications have the capability to provide features to researchers who are investigating malware activities. Table 2.9 presents the anomaly approach.

Table 2.9: Anomaly approach

Reference	Objective	Algorithm	Result
(P. Wang et al., 2014)	To develop an automatic malware detection system by based on behavior signatures	Support vector machines (SVM)	Accuracy = 97.67%
(D. W. Kim et al., 2015)	To identify fake AV web pages in the Internet.	Random forest, SVM and Gradient-Boosted Tree	Accuracy = 90.4%, FPR = 0.2%.
(Cui et al., 2015)	To identify the malicious behaviors of the mobile applications using data mining packet	Naive Bayes and Decision tree	Accuracy = 60%
(Lin et al., 2015)	To select and extract malware features	SVM	Accuracy = 0.98, Precision = 0.85, TPR = 0.92, TNR = 0.98
(Ghiasi et al., 2015)	To find similarities of run-time behaviors based on the assumption that binary behaviors affect registers values	Random forest, Decision tree, Bayesian logistic regression	Accuracy = 95.9%, FP = 4.5%

b) Signature-based approach: Another type of approach is called the signature approach as illustrated in Table 2.10. This approach detects malicious activities by matching the normal pattern with abnormal signatures. It discovers malware patterns by using the signatures which are stored in the database. Nonetheless, this approach is unable to detect unknown malware if the signatures are unavailable in the database. Moreover, this type of approach needs to frequently update the signatures database so as to ensure that able to detect new variants of malware. It also helps to define some possible pattern variations (Feizollah et al., 2013). Any mistake in defining the malicious pattern cause a false alarm thereby decreasing the accuracy of the detection technique.

Table 2.10: Signature approach

Reference	Objective	Algorithm	Result
(Elish et al., 2015)	To advocate the approach of benign property enforcement	Trigger based API dependence	FP = 2%, FN = 2.1%
(Talha et al., 2015)	To characterize and classify Android applications as benign or malicious.	Statistical score	FPR = 0.050, TPR = 0.101, FNR = 0.898
(Sheen et al., 2015)	To design malware detection using multi feature collaborative decision fusion (MCDF).	Naive Bayes, Decision tree, SVM, IBk (Instance based learning), JRip (Rule based learning)	Precision = 83%, TPR = 97%
(Choi et al., 2015)	To detect the act of leakage internal private information	Context Ontology Reasoning	Condition reasoning (high, low, active, available)
(Faruki et al., 2014)	To detect unknown malware	Clustering algorithm	Accuracy = 76%, TPR = 80.65%
(Cen et al., 2015)	To develop effective technique for malware detection	Naive Bayes	Accuracy = 0.95, TPR = 0.95, FPR = 0.05
(Clemens, 2015)	To classify architecture of computer object code	SVM, Decision tree, Random Forest, Naive Bayes, Neural network	Accuracy = 90%

c) **Hybrid-based approach:** The hybrid-based approach combines the signature database with anomaly patterns so as to detect known or some new variants of malware attacks (Inayat et al., 2016; Wang et al., 2015). This approach is able to perform dynamic analysis during the running applications; it also uses statistically analyse data by using the signatures database (Arshad et al., 2016). This approach is able to overcome the weaknesses of both the signature-based approach and the anomaly detection approach. Its disadvantage is that it needs more research to be conducted subject to the malware detection designs. Machine learning is also used to trace the normal and abnormal patterns (Inayat et al., 2016; Haq, 2015) in this approach.

Machine learning is a type of artificial intelligence that provides computational learning theories to predict the data. Machine learning focuses on prediction making; it acts without being explicitly programmed. In addition, machine learning is an approach

that examines the data so as to look for patterns. Supervised and unsupervised classifiers in machine learning have been used to trace the model and to analyse the features (Narudin et al., 2016). This approach helps to determine the validity of the normal and malicious activities. Decision trees, random forest, and SVM are the types of algorithm classifiers used on supervised learning for this purpose. Table 2.11 lists the advantage and disadvantage of the detection approach..

Table 2.11: Advantage and disadvantage of the detection approach

Detection approach	Advantages	Disadvantages
Anomaly	**a)** Dynamically adapt to new, unique, or original attacks. **b)** Less dependent on identifying specific operating system vulnerabilities **c)** Effective to detect new and unforeseen vulnerabilities	**a)** Higher false alarm rates. **b)** Usage patterns that change often and not be static enough to implement an effective behavior-based IDS.
Signature	**a)** Lower false alarm rates. **b)** Alarms are more standardized and more easily understood than behavior-based. **c)** Simplest and effective method to detect known attacks (Liao et al., 2012)	**a)** Signature database must be continually updated and maintained. **b)** Ineffective to detect unknown attacks, evasion attacks, and variants of known attacks (Liao et al., 2012). **c)** Time-consuming to maintain the knowledge

2.5.3 Deployment approach

The deployment approach (hybrid, network and host-based) monitors and detects malicious activities (Inayat et al., 2016; Shameli-Sendi et al., 2014; Lar, 2011). The hybrid-based Intrusion Detection System is a combination of both the Network-based Intrusion Detection System (NIDS) and the Host-based Intrusion Detection System (HIDS) (Butun et al., 2014; Tahaei et al., 2018). The NIDS is used to analyse data over the network traffic by using deep packet analysers (Zhang et al., 2003). The packet analyser is able to identify any malicious activities during interactions between the network and computer (Patel et al., 2012). Table 2.12 shows the deployment approach.

Table 2.12: Deployment approach

Reference	Titles	Deployment approach	Detection approach	Year
(Asaf Shabtai et al., 2010)	Applying behavioral detection on android-based devices	HIDS	Signature	2010
(Grace et al., 2012)	Unsafe exposure analysis of mobile in-app advertisements	HIDS	Signature	2012
(Dini, Martinelli, Saracino, et al., 2012)	MADAM: A multi-level anomaly detector for android malware	HIDS	Anomaly	2012
(Zhao et al., 2012)	RobotDroid : A Lightweight Malware Detection Framework on Smartphones	HIDS	Anomaly	2012
(D.-J. Wu et al., 2012)	DroidMat: Android Malware Detection through Manifest and API Calls Tracing	HIDS	Signature	2012
(Feizollah, Shamshirband, et al., 2013)	Anomaly Detection Using Cooperative Fuzzy Logic Controller	NIDS	Anomaly	2013
(Narudin et al., 2016)	Evaluation of machine learning classifiers for mobile malware detection	NIDS	Anomaly	2014
(Cen et al., 2015)	A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code	HIDS	Signature	2014
(Gonzalez et al., 2014)	DroidKin : Lightweight Detection of Android Apps Similarity	HIDS	Signature	2014
(Gheorghe et al., 2015)	Smart malware detection on Android	NIDS	Anomaly	2015
(Chen et al., 2015)	Simple and effective method for detecting abnormal internet behaviors of mobile devices	NIDS	Anomaly	2015
(X. Wang et al., 2015)	Novel Hybrid Mobile Malware Detection System Integrating Anomaly Detection With Misuse Detection	HIDS	Hybrid	2015
(Chuang et al., 2015)	Machine Learning Based Hybrid Behavior Models for Android Malware Analysis	HIDS	Hybrid	2015

The HIDS monitors and analyses any intrusive activities by assessing the system resources. It focuses on the memory, the device, CPU consumption, the user, system activities and also the file system (Weiss et al., 2012). Andromaly (Asaf Shabtai et al., 2012) is an example of host-based malware detection. The HIDS collects resources from mobile devices, computers, and servers. Over the years, the booming mobile devices have stimulated users into replacing personal computers, in terms of the Internet usage, for

personal transactions such as online banking, games, emails, social media, and news articles. Mobile devices are also more appealing to users because applications are downloadable and free. Based on Table 2.12, it shows the Android malware detection system implements the HIDS and NIDS. The IDS method is used to identify and analyse the Android mobile malware (Corona et al., 2013).

2.6 Risk assessment

Risk assessment is the process which identifies the loss, vulnerability, damage and consequences of the action. It is very important because it presents studies with the evaluated impact of malware attacks. In terms of mobile risk, risk assessment helps mobile devices to establish the level of safety or the risk involved in some mobile applications. The risk mechanism protects mobile users from installing malware applications (Seo et al., 2014). This is achieved by informing the users in the permissive mode by ensuring the right trust. Potential risks able to mininmise (Chowdhury et al., 2012; Anuar et al., 2011). Risk is defined as one component that combines threats with vulnerabilities; risks cause an impact on an asset even though the vulnerabilities flawed. Some researchers (Ledermüller et al., 2011) treat a risk as a single entity but others treat risk as giving bad impacts such as costs, degraded performance as well as functionality. The aim of risk assessment is to provide security measures by impacting confidentiality, integrity and availability in security threat.

2.6.1 Threats

Threats exists in risk assessment. Becher et al. (2011) classify mobile threats into four classes: a) hardware centric attack where the attacker has direct access to the physical mobile devices, b) device independent attack which protects the device from the confidentiality violation, c) software centric attack which is the most used by attackers i.e. APK file with malicious application to exploit the vulnerabilities, and d) user layer

attacks. To defend mobile threats, an effective security mechanism which provides a response to the user must be available. The risk assessment approach is used to improve the effectiveness of risk evaluation by generating risk zones as a warning against malicious applications (e.g. very low, low, medium, and high). Table 2.13 presents the description of risk assessment.

Table 2.13: Description of risk assessment

	Description
AHP approach	One of the Multi-Criteria decision-making approach for analyzing decisions
Risk assessment phase	An approach that is derived from an evaluation of threat and impact of risk
Judgment matrix	This technique gives an analysis to construct the matrix consistency.

2.7 Risk assessment phase

The AHP uses a pairwise comparison of criteria to evaluate the weight of the criteria which is in line with the main objective of the hierarchical structure. This pair-wise comparison is performed using a matrix table which evaluates the consistency of the judgment (Model et al., 2015). The comparison matrix (A) takes the size $n^x n$ where n denotes the number of criteria being compared, which is relative to the specific elements. The elements of the matrix are a_{ij}. The table matrix A demonstrates the evaluation that is similar to Dweiri et al.(2016):

$$a_{ij} = a_{jk} \, x \, a_{jk} \quad \textbf{2-1}$$

$$a_{ij} = 1/a_{jk} \quad \textbf{2-2}$$

where i, j and k are any elements of the matrix A.

$$A = \begin{pmatrix} a_{11} \, a_{12} \, a_{13} \\ a_{21} \, a_{22} \, a_{23} \\ a_{31} \, a_{32} \, a_{33} \end{pmatrix} \quad \textbf{2-3}$$

$$\text{where } a_{ij} = 1 \; and \; i = j \quad \textbf{2-4}$$

In order to evaluate the consistency, the normalization table matrix of A is computed using N matrix

$$N = \begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{pmatrix} \text{ where, } w_{ij} = a_{ij} \quad \textbf{2-5}$$

$$w_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \quad \textbf{2-6}$$

$\sum_{i=1}^{n} a_{ij}\ w_i = 1$ is the sum of the columns **2-7**

Divide the sum of the value of n to find the relative weight

$$\text{The weight of } i = w_i = \frac{\sum_{j=1}^{n} w_{ij}}{n} \quad \textbf{2-8}$$

$$\text{Note that } \sum_{i=1}^{n} w_i = 1 \quad \textbf{2-9}$$

A is reflected as consistent if

$$A\ x\ w = n\ x\ w \quad \textbf{2-10}$$

The Eigenvalue problem could be solved by using this equation where the largest Eigenvalue is greater than or equal to n ($\lambda_{max} \geq n$). The value (λ_{max}) becomes closer to n, thus it is more consistent. (λ_{max}) is equivalent to the total of the criteria of the column vector AW. The consistency ratio (CR) is calculated as:

$$CR = \frac{Consistency\ Index\ (CI)}{Random\ Consistency\ of\ A} \quad \textbf{2-11}$$

$$\text{The value } CI = \frac{\lambda \max - n}{n-1} \quad \textbf{2-12}$$

The level of consistency is considered acceptable when the CR is less than or equal to 0.10. Otherwise, it requires a reexamination of the judgment for the values of aij. This judgment is necessary in order to keep the consistency

2.8 Judgement matrix

In AHP, the hierarchy of risk assessment is measured by using the pairwise comparison which is important for estimating the relative elements in the hierarchy and the preceding level. Pairwise comparison is measured through the ratio scale. The scale is used to define the intensity of important judgments. Table 2.14 shows the fundamental scale of the absolute numbers of the AHP approach (Saaty, 2008).

Table 2.14: Fundamental scale of the absolute numbers

Intensity of importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
2	Weak	Two activities contribute equally to the objective
3	Moderate importance	One activity is slightly being favored over another based on experience and judgment
4	Moderate plus	One activity is slightly being favored over another based on experience and judgment
5	Strong importance	One activity is strongly being favored over another based on experience and judgment
6	Strong plus	One activity is very strongly favored over another based its dominance in practice
7	Very strong or demonstrated importance	One activity is very strongly favored over another based its dominance in practice
8	Very strong	One activity is very strongly favored over another based its dominance in practice
9	Utmost importance	One activity is utmost important favored over another based on the highest possible order of affirmation
Reciprocals of above	If nonzero numbers assigned to activity I when compared with activity j, then when compared with I, j has the reciprocal value	A sensible assumption

2.9 Summary

This chapter has presented the mobile device evolution as well as the mobile operating systems. It highlights the Android operating system including its unique characteristics and architecture. It also discusses the types of malware detection systems by providing the detection taxonomies which encompass the analysis techniques and the detection and development approaches. It also presents the risk assessment approach by introducing the risk assessment phases and the judgment matrix which are used in the experiments of this study. The subsequent chapter presents the mobile malware analysis tools.

CHAPTER 3: MOBILE MALWARE ANALYSIS TOOLS

As detailed previously, this study adopts the static analysis as the analytical procedure because it consumes low resources, it checks for application code structures and it is a common practice of researchers. The main advantage of adapting this approach is that it is able to differentiate between a well-known coding standard and malicious codes. Without such tools the process of identifying malicious codes in Android applications is likely to fail because the number of codes in the Android application structure is so large that it is difficult to remember. This chapter details the static analysis tools by introducing their characteristics as well as their advantages. Overviewing various tools used in malware detection also helps to save a lot of effort in understanding the applications codes structure and making the analysis process more effective.

3.1 Static analysis tools

Several types of static analysis tools are presented in this section. They are helpful for analysing the codes in Android applications. They also help to shed light on the structure of the codes for malware detection systems. Details of the static analysis tools are described as follows:

3.1.1 Androguard

Androguard is a toolkit for malware analysis. It has the capability to reverse engineering in Android applications (Desnos, 2012). It is a python based tool which executed on Windows, Linux and OSX operating systems. It is a powerful tool that able to decompile and disassemble Android applications by using the static analysis approach. The advantages of the Androguard include its ability to reverse engineering on the Android applications, its compatibility with the open source malware analysis, it able to assess the static approach of the codes, and it visualises the applications with gephi outputs.

3.1.2 ApkTool

The ApkTool is also a reverse engineering tool used for Android applications (Wiśniewski, 2017). It has the capability to decode the resources to nearly its original form and to rebuild after making some modifications. It comes with features such as organising and handling the apks that depend on the framework resource; it also has the ability to turn the smali debugging and the rebuilding decode sources back to apks. Smali is a DEX code dissembler that transforms the byte code into a syntax. The aim of the syntax is to alleviate the complexity of the exploring Java Virtual Machine binary. Decompilation performs an inverse operation to that of the Dalvik's byte code compiler and the apk packaging.

3.1.3 Statistical analysis software tools

Statistical software are programmes which data analytics use for analysis, interpretation, and the presentation of data. It comes with unique features such as user interface which enhances its usability, its comprehensive and powerful programming language for in-depth analysis and its data exploration. With a graphical user interface and analysis capability, the software has been applied in research, academia, industries and government agencies. Statistical analysis software enhances the experience of developing statistical analyses and the interpretation of results.

3.1.4 R language

R language is used for statistical computing and graphics. It is a free software and it runs on different platforms such as Windows, UNIX and MacOS under the Free Software Foundation's GNU General Public License (Bryan et al., 2017). It provides a variety of statistical analysis like classification, clustering, linear and nonlinear modelling, time-series analysis and graphical techniques. The R language offers advantages such as effective data handling, graphical attribute provision, and integrated collection tools for

data analysis. It also contains an effective programming language which includes conditional and loops. Due to its capability in data manipulation, calculation and graphical display, the software is used by researchers especially in statistical methodology activities.

3.1.5 IBM SPSS statistics

The IBM SPSS Statistics is a statistical software used to solve research and business problems (IBM, 2017). This powerful software provides a variety of features, for example, it interprets data; it manages data; it enables hypothesis testing; it reports and performs analyses. It also provides a wide range of analytic capabilities including linear regression and descriptive analysis which enhance the presentation of results as well as uncover dataset relationships. In addition, it offers scalability, flexibility and cost effectiveness.

3.2 Machine learning classifiers

Machine learning is a type of artificial intelligence (AI) that the computer provides; it has the ability to learn without being explicitly programmed. It also has the capability to predict future decisions and to improve decisions when exposed to new data. The process of prediction is based on the search through the dataset which look for patterns. This is also known as learning. The learning process and prediction results are made according to the types of classifiers. This technique has been widely implemented for classifying samples especially in the intrusion detection system area (malware and benign). The two common types of machine learning are supervised and unsupervised machine learning. Supervised learning is where the dataset for training are labeled with the class (malware and benign). The class is used for training the dataset which are in the learning process of an algorithm learning. The algorithm makes predictions based on the training data. Learning stops when the algorithm achieves an acceptable level of performance. Linear

regression, random forest and support vector machine (SVM) algorithms are examples of supervised machine learning algorithms. The latter, unsupervised learning, only demands input data without comparing yield factors. The goals are to model the underlying structure or distribution in the data in order to learn more about the data. The algorithm presents the interesting structure in the data which show the information. This learning type has no right answers and they are unlabelled. The learning stage measures itself to predict and present the information. Apriori algorithm is an example of unsupervised machine learning algorithm.

This study applies the supervised machine learning approach because the sample dataset for each application come with labels (i.e. malware and benign). Besides that, supervised machine learning offers good potential results by reducing errors. In the interest to observe the distinctive results noted in the various machine learning classifiers, this study implements five (5) classifiers: random forest (RF), multi-layer perceptron (MLP), k-nearest neighbors (KNN), J48 and Adaboost. They are further explained below.

a) RF: Random forest is a combination of tree predictors. Each tree is capable of producing a response with the set of predator value. This classifier was developed by Leo Breiman and Adele Cutler (Gaviria et al., 2013). The principle of the random forest is a weak learner forms a group to create a strong learner. (Gaviria et al., 2013) and (Narudin et al., 2016). The random forest classifier is used for mobile malware detection.

b) MLP: The multi-layer perceptron is a supervised learning algorithm which consists of multiple layers of nodes. Each node interacts via the weighted connection (Narudin et al., 2016). The intermediate layer is known as a hidden layer with one or more non-linear layers.

c) KNN: The K-nearest neighbors is one of the simple machine learning classifiers that works well in classification. It is a lazy learning type of classifier. In order to predict the label of a new instance, the KNN classifier needs to find the K-nearest neighbor. The classifier uses training samples to predict the label whereas the user defines and classifies the K-nearest neighbor by labelling the sample. Gaviria et al. (2013) achieved a high detection accuracy by using permission features.

d) J48: This is an open source implementation of the machine learning tool (WEKA) which comprises the C4.5 algorithms. The tool predicts the sample training based on the various attribute values of the sample training data. It requires dependent and independent variables to predict data. The independent variable is used to predict the dependent variable. In addition, this classifier chooses the attribute that is most efficient in using the entropy in the data (Aung et al., 2013). This classifier obtained a TPR value of 90.7 percent and so the J48 is noted as a logic-based learning type classifier.

e) AdaBoost: This is a popular boosting classification algorithm. It resembles the learning type classifier. It is constructed from multiple learners for the purpose of building a stronger learning classifier. It also performs well in a variety of dataset except for noisy data. During the training data, the model attempts to correct the error so as to improve the predictable performance. Sheen et al. (2015) used AdaBoost for malware detection and they achieved high precision with a recalled value of more than 90 percent. Table 3.1 describes the four classifiers and their advantages and disadvantages, both of which were noted in the current study.

Table 3.1: Description of classifiers

Classifier	Learning type	Advantages	Disadvantages
Random forests	Ensemble	**a)** Used bagging and boosting to support a large number of training samples with efficiently. **b)** It able to improve the classification accuracy by growing an ensemble of the tree. **c)** Run efficiently on large datasets.	**a)** Over fit when dataset is unclean from noise.
MLP	Perceptron based	**a)** Capable of learning non-linear models. **b)** Capable of learning models in real time	**a)** Requires tuning a number of hyperparameters such as layer and number of hidden neurons. **b)** Ineffective to feature scaling
KNN	Instance-based	**a)** Able to use with categorical data. **b)** Robust to noise training set. **c)** Run effectively on large data sets.	**a)** Require to determine value of K (number of nearest neighbors). **b)** High computation cost because require to distance of each query.
J48	Ensemble	**a)** Applicable for continuous and categorical inputs. **b)** Data classification without much calculations.	**a)** High classification error while training set is small.
AdaBoost	Ensemble	**a)** A powerful classification algorithm that improves the prediction performance	**a)** Sensitive to noisy data and outliers

3.3 Machine learning tools

Machine learning tools provide functionality for data analysis that automates the analytical model building. This model enables a system to learn from the past or present dataset and in that learning process makes predictions or decisions. In general, there are two (2) types of machine learning tools: supervised and unsupervised. Implementing the machine learning tools in a system makes the analytical work easier and faster. In addition, it has the ability to automatically apply complex mathematical calculations in solving problems without requiring any machine learning technique or expertise.

3.3.1 WEKA

The Waikato Environment for Knowledge Analysis (WEKA) is a popular suite; it has a collection of machine learning algorithms written in Java (Waikato, 2017). It is used for data mining tasks, data analysis and for predictive modelling. It was developed at the University of Waikato, New Zealand. It is a free software licensed under the General Public License (GNU). It is made up of visualisation tools that comprise various types of algorithms such as random forest, multi-layer perceptron, regression, clustering, Adaboost, k-nearest neighbors and neural networks. These algorithms able to applied directly to a dataset or recalled from the Java code. Figure 3.1 shows the user interface of the WEKA.

Figure 3.1: WEKA GUI

The graphical user interface (GUI) of WEKA with four (4) tabs in the header and five (5) applications is illustrated. The program tab contains the system setting and memory usage while the visualisation tab is used to generate a graph like the ROC, a plot, and the boundary visualiser. Figure 3.2 further portrays the applications for features selection.

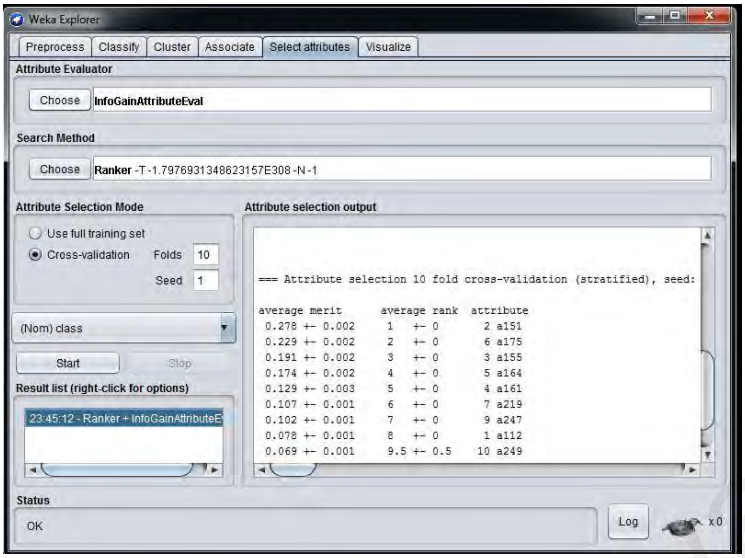

Figure 3.2: Features selection

The figure shows the information gain with ranker attributes for choosing the relevant features for malware detection from 10 fold cross validations. With good interface, this application is used even by normal users. All the algorithms for classifier and features selection is found in this application. The application is also able to show good quality result which is also easy to understand.

Figure 3.3 shows the list of classifiers which belong to the tree algorithms. In addition to this, there are bayes, functions, meta, misc, and rules. These classifiers are used for training and testing the sample dataset. The training and testing are used to create a model. It is well-suited for developing new predictive machine learning.

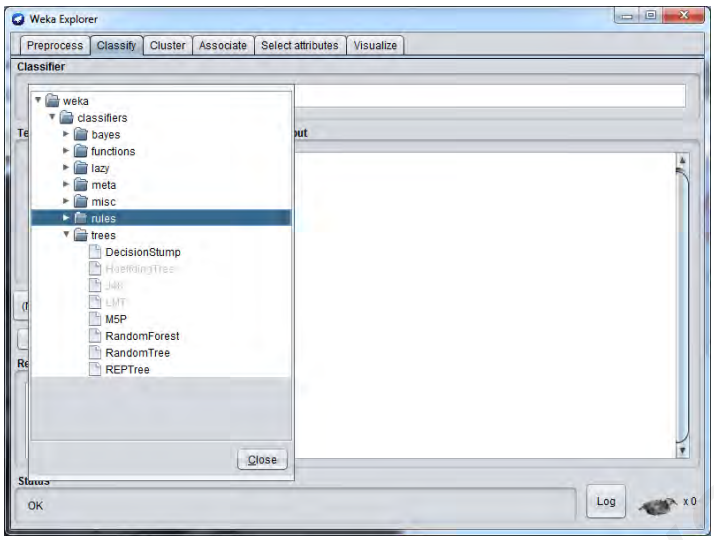

Figure 3.3: Examples of classifiers

The main objective of WEKA is to make machine learning approaches available, to design new predictive models for sharing with the world, and to contribute to the theoretical framework. The advantages of the WEKA are able to traced to its services to users, its user friendly values due to its graphical user interface, the comprehensive collection of modelling techniques and data analysis and its portability.

3.4 Online analysis tools

VirusTotal is a web analysis tool that provides free online services; it also analyses files and URLs (Quintaro, 2017). This web analysis has the capability to identify viruses like Trojans, worms, or any kinds of malicious applications through website scanners and antivirus engines. The purpose is to help to improve the antivirus security industry thereby making the internet a safer place accomplished through the development of free tools and services. Figure 3.4 shows the GUI for VirusTotal.

Figure 3.4: GUI of VirusTotal

As seen on Figure 3.4, the user is able to submit a file to VirusTotal through the Web browser and then receive results of the file scanning. The maximum file size is only 128MB for each application. This study used the Web analyser to analyse all the sample dataset so as to ensure that the data were correctly labelled with benign or malware applications. The user is able to select the application from the folder before scanning. Figure 3. 5 shows the results of the scanned applications.

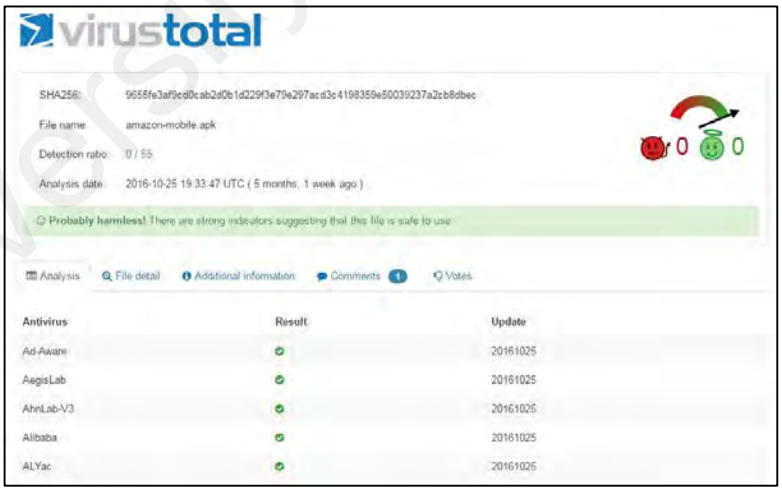

Figure 3.5: Examples of analysis results

Above, the result of the applications indicates whether the applications were benign or malware, based on the detection ratio. Any application with a detection ratio of more than 0 is known as a malware. It is worth noting that this web analyser runs multiple antivirus

engines and website scanners. Additionally, the malware signatures of the antivirus solution in VirusTotal are periodically updated as they are developed and distributed by an antivirus company. As a result, the report that is received from the submitted file will display the exact detection. Figure 3.6 shows the detail of the scanned applications.

This detailed information is used to create an attribute or features for the dataset. The information explains the role of permission on the scanned applications. The information is used to as a part for improving and advancing the testing sample dataset.

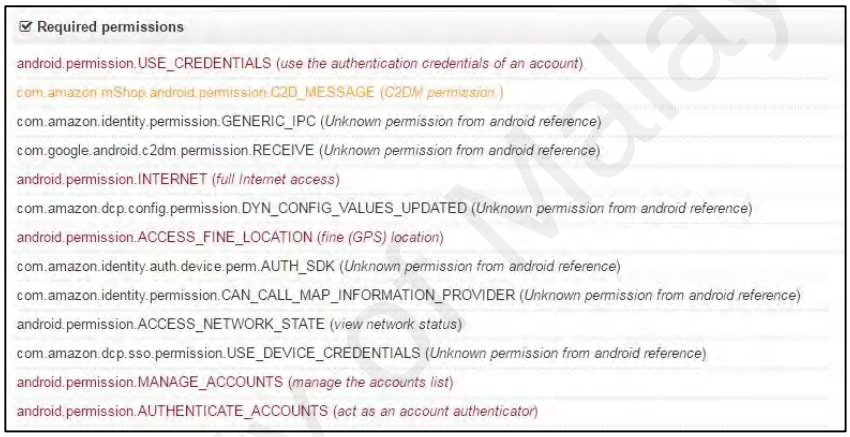

Figure 3.6: Details of scanned applications

3.5 Feature selection and optimisation method

Feature selection plays an important role in detecting malicious activities in mobile applications. The relevance of the extracted features depends on the topology and resilience of the malware in general. This study survey the features based on their categories, relevance and definitions. Relevant features is one of the important contributing factors of an excellent detection model. Features with a lack of exploration may lose their full potential for best analysis. For example, Android applications consist of various elements for analysis such as static, dynamic, hybrid and application metadata. Selecting the most relevant feature from the massive number of available features is crucial because feature selection also has other motivations which include:

i. General data reduction: It limits storage requirement and increases algorithm speed to save time and cost of experiments.

ii. Feature set reduction: It saves resources for data collection during utilisation.

iii. Performance enhancement: By removing the irrelevant features such as noisy data from the dataset, detection accuracy especially on machine learning algorithms is increased.

iv. Data understanding: It is easy to visualise data and monitor the experiment process.

An Android application consists of several elements that carry good potentials to be the features in Android malware detection. Therefore, this study selected the permission features for the experiment of Android malware detection. The permission features were selected because permissions are the entry points for every mobile application. Various dangerous permissions for Android applications have been highlighted but users are still unaware of the side effects of such risky permissions. Undoubtedly, dangerous permissions act as a gateway for attackers to install malware applications that interact with malware programmers who access users' mobile devices. A large portion of dangerous permissions requested by applications present malicious intentions (Firdaus et al., 2017). Permissions features have been widely explored and its significance and effectiveness have also been established. Thus, this study considered permission features.

For the purpose of recognising the dangerous permissions in Android applications, it is necessary to conduct a thorough static analysis. Static analysis is used in the experiment because it provides more insights into the coding patterns of the applications. The in-depth knowledge gained of the dangerous and risky permissions of Android applications are used to analyse Android applications in cloud analyses thereby serving as the proposed solution to the problem.

Static analysis is a lightweight approach when compared to dynamic analysis. It has the capability to dissemble the Android application so as to retrieve the codes. Thus it was applied on the benign and malware applications. The malware programmer deploy various evasion techniques on Android applications to secure premium message on permission features. This study attempts to solve the problem by relying on the entry-level structural information known as AndroidManifest.xml to collect the permission features.

The Android application which consist of data, resource files, and Java code are compiled using Android SDK tools. This process requires static analysis to decompile and extract the binary codes. During this phase, any permission listed on AndroidManifest.xml is considered as features.

Following this, the features are examined so as to remove the irrelevant data. The most relevant features containing malicious activities are then stored. This is important in ensuring the accuracy of the Android malware detection performance. To achieve this, the current study selected lesser features for the experiments involved because previous works (Razak et al., 2017) had also used fewer features to remove the irrelevant data from the dataset thereby improving the result of the machine learning algorithms. Table 3.2 shows the number of features used by previous works.

Table 3.2: Number of features used by previous works

References	Total of benign	Total of malware	Total of features
(Feizollah et al., 2017)	1846	5560	10
(Afifi et al., 2016)	20	1000	7
(Narudin et al., 2016)	20	1000	11
(Yuan et al., 2016)	880	880	13
(Gheorghe et al., 2015)	400	400	13

As seen in Table 3.2, previous studies used fewer than 20 features for testing and training their machine learning algorithms. The reason for using fewer features is because this will save the time and costs of the experiments thereby hastening the time for real world implementation. The following are the features selection approaches applied in this study:

3.5.1 Information gain

Information gain is also known as info gain; it is an attribute selection approach that uses the ratio of information gain to measure the intrinsic information. It measures the small bits of information obtained for predicting a class (c) by understanding the absence and presence of a term (t) in a dataset.

In the Intrusion Detection System (IDS), information gain is applied so as to improve the accuracy performance of the malware detection (Nadiammai et al., 2014). The features are ranked in descending order with the highest information gain being the most relevant feature ranked at the top. The features are then selected for training the model so as to maximise the classification accuracy (McWilliams et al., 2014). In their study, Asaf Shabtai et al. (2012) managed to achieve a 99.9 percent accuracy rate after applying information gain with the decision tree, J48. Similarly, Santos et al. (2013) performed features selection on 1000 opcode sequences using information gain. They also achieved 95.26 percent accuracy with the decision tree, random forest. These findings suggest that information gain has the capability to select relevant features and as a result, is able to increase the classification performance accuracy.

3.5.2 Evolutionary algorithm

Evolutionary algorithms (EAs) are inspired by the biological model of evolution and natural selection that was first proposed by Charles Darwin in 1859 (Dyer, 2010). In the natural world, evolution helps species to adapt to their environment. Environmental

factors that influence the survival prospect of an organism include climate, availability of food and the dangers of predators. Species change over the course of many generations and mutations occur randomly. Some mutations will be advantageous but many will be useless or detrimental. In that regard, progress comes from the feedback provided by non-random natural selection, for example, animals that can run fast will be more successful at evading predators than their slower rivals. If a random genetic modification helps an organism to survive and reproduce, that modification will itself survive and spread throughout the population via the organism's offspring. This is how evolutionary algorithms work.

Evolutionary algorithms are derived from a simplified model of this biological evolution. It is typically used to provide good approximate solutions to problems that cannot be solved easily when using other techniques. Many optimisation problems fall into this category. It may be too computationally-intensive to find an exact solution but a near-optimal solution is sufficient. In such situations, evolutionary techniques can be effective.

3.5.3 Bio-inspired Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation (PSO) was implemented in this study for the purpose of increasing the performance of the malware detection system. This is accomplished by minimising the error imposed on the swarm intelligence. PSO was developed by Dr. Kennedy and Dr. Eberhart in 1995 (Afifi et al., 2016). It is a popular bionic algorithm that was derived from looking at the social behavior of birds flocking together for formulating the optimisation problem. PSO is based on the principle particle of the swarm for each solution. Each particle has a solution in the search space where the vector $x_i = (x_{i1,}\, x_{i2} \ldots., x_{iD})$ and where D represents the dimensionality of the search space. Besides that, the particle has a velocity with $v_i = (v_{i1,}\, v_{i2}, \ldots., v_{iD})$. Each particle updates its

velocity and position during the movement based on its neighbours. The *pbest* and *gbest* obtained from the best population is known as the best position for the particle. These *pbest* and *gbest* help the PSO to search for optimal solution as it signifies the best swarm particle. The equation shown below illustrates how the PSO searches for optimal solution.

$$x^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad \textbf{3-1}$$

$$v_{id}^{t+1} = w * v_{id}^{t} + c_1 * r_1 * (p_{id} - x_{id}^{t}) + c_2 * r_2 * (p_{gd} - x_{id}^{t}) \quad \textbf{3-2}$$

In the evolutionary process, t serves as the *t*th iteration while *d*th is represented in d ϵ D within the search space. In order to control the impact of previous velocities, the inertia weight (*w*) is used. The $c_1$and c_2 show the acceleration constant. The random values r_1 and r_2 are generally distributed in terms of [0, 1]. The p_{id} and p_{gd} indicate the value of *pbest* and *gbest* in the *d*th dimension. The maximum velocity, v_{max} and $v_{id}^{t+1} \epsilon [-v_{max}, v_{max}]$ predefines the velocity. After all the predefined criteria are met, the algorithm is stopped. It then shows the best fitness value or the maximum number of iterations that have been set.

3.5.4 Distinctive features between application

This type of feature selection was motivated by the aim of conducting an experiment that calculates the range of the features noted in Android applications (malware and benign) and its effectiveness in detecting malware. For instance, malware applications would request more dangerous permissions than benign applications.

3.6 Summary

This chapter has explained some related and important static analysis tools which can be used to improve the effectiveness of analysing the applications structure. Addressing the advantages and functionality of these tools is important because the outcome of the current study is used as a guideline to construct an analysis process which aids in finding the relevant features for malware detection systems. The subsequent chapter discusses the framework of the proposed system along with its different modules and the rationale behind it.

CHAPTER 4: RISK ANALYSIS AND MALWARE DETECTION: THE FRAMEWORK

This chapter provides the details of the proposed framework that serve as a system that able to detect malware applications and also identify their risks. This framework was derived from two approaches: the machine learning approach and the risk assessment approach, both of which were used in combination with other selected methods and models. The proposed framework is necessary for detecting malware risks thereby protecting mobile device users. This chapter also explains the procedure of the malware detection process as well as the rationale behind the implementation of framework. The chapter also provides a detailed overview of the main and sub-models used to support the proposed framework.

4.1 EZADroid framework

The proposed framework attempts to identify whether the Android applications are malware or benign by using the Intrusion Detection System (IDSs) and risk assessment to detect malware and assess the risk level of applications posed to users. It is very important to choose the appropriate approach when dealing with the technical aspects. This study proposes to develop an EZADroid framework that is represented in Figure 4.1. The system is capable of detecting and analysing Android applications in mobile devices as well as on Web browsers. The proposed framework contains the following benefits:

i. **User awareness:** This framework is designed to identify malware applications.
ii. **Web module:** This framework contains different models including malware detection and risk analysis models to satisfy the objectives of this study.
iii. **Signature approach:** This framework relies on signatures; it examines the collected information which allows the framework to identify which applications act and look like malware.

iv. **Response:** This framework comes with a risk assessment approach which poses itself as a passive response to users until the threat is reduced.

v. **Intelligent:** This framework detects and analyses the malware including the effective responses given to users, based on risk levels.

vi. **Scalability:** This framework applies the machine learning model to classify Android applications as malware or benign. The system administrator has the capability to access the back end of the system which extends the framework to meet the current demand in detecting the malware applications. For instance, updating the model by re-training via the use of the machine learning classifiers and adding more functions to the system. As a result, the system becomes more effective in detecting Android malware applications.

The proposed framework is established through the combination of three (3) components: webpages, mobile devices and server. With the aid of the IDS, the risk assessment approach and the machine learning approach as noted in the previous chapters, the server thus acts as a model. It then supports the procedures for selecting the relevant features for malware detection by using the machine learning algorithms. Additionally, the server plays several important roles such as collecting dataset, analysing results and responding to users with the results.

The web pages offer online malware analysis services specifically for Android applications. This web pages are considered as a comprehensive online security service that is capable of identifying suspicious and malicious applications by using the static analysis technique. The technique analyses the applications with the machine learning algorithms. These web pages service users and security researchers by identifying their Android applications as malware or benign. Users activate this service by submitting their Android applications to these web pages through their personal computers.

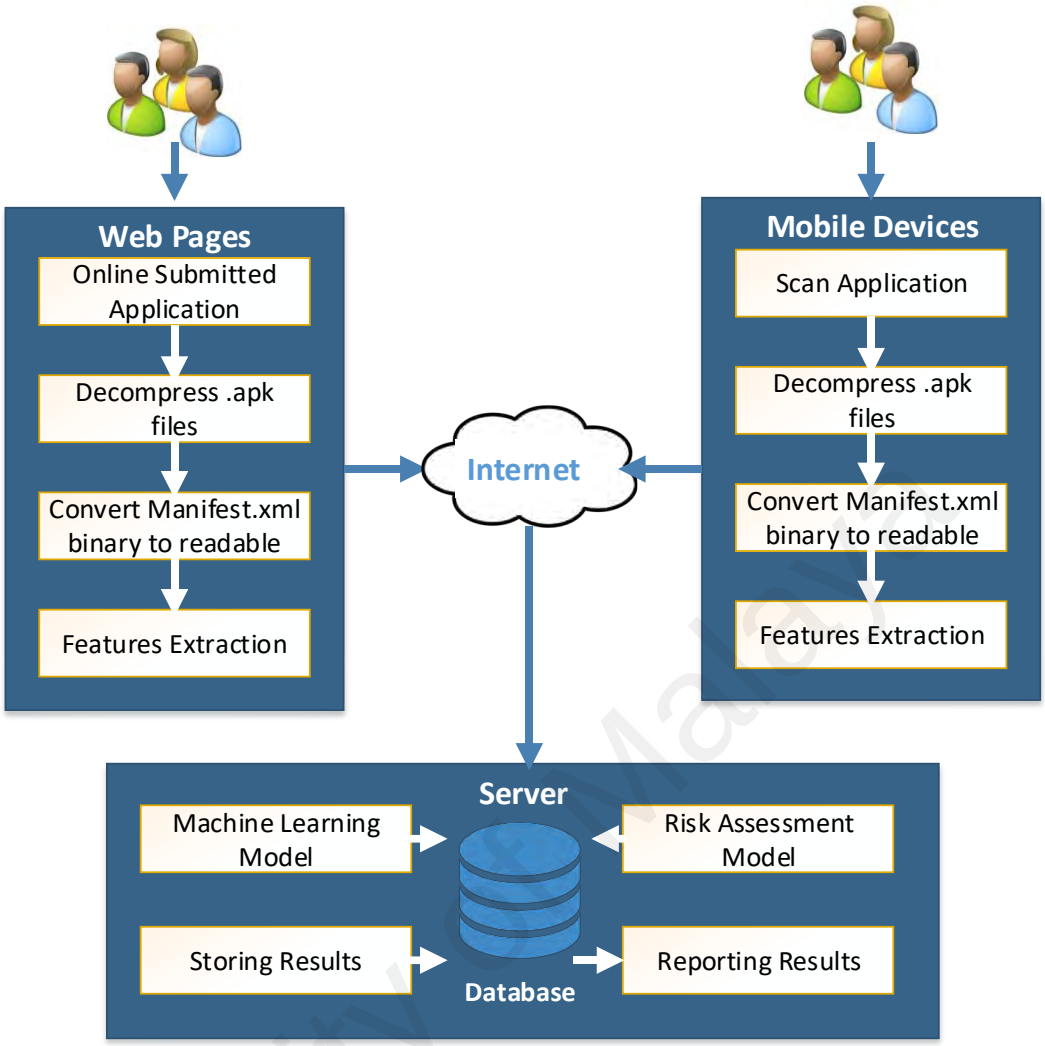

Figure 4.1: EZADroid Framework

Mobile devices are designed such that they able to assess the application risks and detect Android malware on mobile devices. Part of the EZADroid system offers a risk assessment and detection approach which utilizes applications that monitor the static analysis and is capable of protecting mobile devices from threats. Even though the detection and risk assessment works with the server, all the results obtained from scanning the applications are returned to users. These results also indicate the risk level of the threats and the ratio number of positives detected.

Servers are built with an IDS and a risk assessment approach. The main advantages of this server is that the submitted Android applications are analysed through the static analysis and risk analysis model, reducing the chance for a malware to attack users. In

addition, it is capable of collecting data, detecting the result and responding to users. The heavy process of identifying the malware is done on the server and the responses are sent back to users through the mobile devices or web pages. In this way, it is able to reduce processing time and consumption of resources.

The similarity in these three (3) parts is that they implement static analysis which is responsible for analysing the apk files of the application and for extracting related features. The apk files from mobile devices and web pages are sent to the server which extract the features of the applications. All the extracted features are then collected and stored in a database at the server for analysis purposes. The analysis helps to ensure that the mobile devices and the apk files are clean from malicious activities. This process is performed by submitting data to the machine learning model that is prepared offline by the system administrator.

This model is produced by training thousands of malware and benign applications using selected extracted features. These features are then fed into the machine learning algorithms. The best algorithms are selected based on their performance and accuracy. The entire process is able to create a good detection model which placed at the server module to determine the cleanliness of the new apk files which submitted by the mobile devices and web pages. All the results obtained from analysing the apk files are then sent back to the mobile devices and web pages. To ensure the accuracy and durability of the results, the system administrator accesses the server directly. It helps to maintain and update the model whenever it is required. The detailed process of creating the model and for testing the effectiveness of the model is discussed in the next chapter

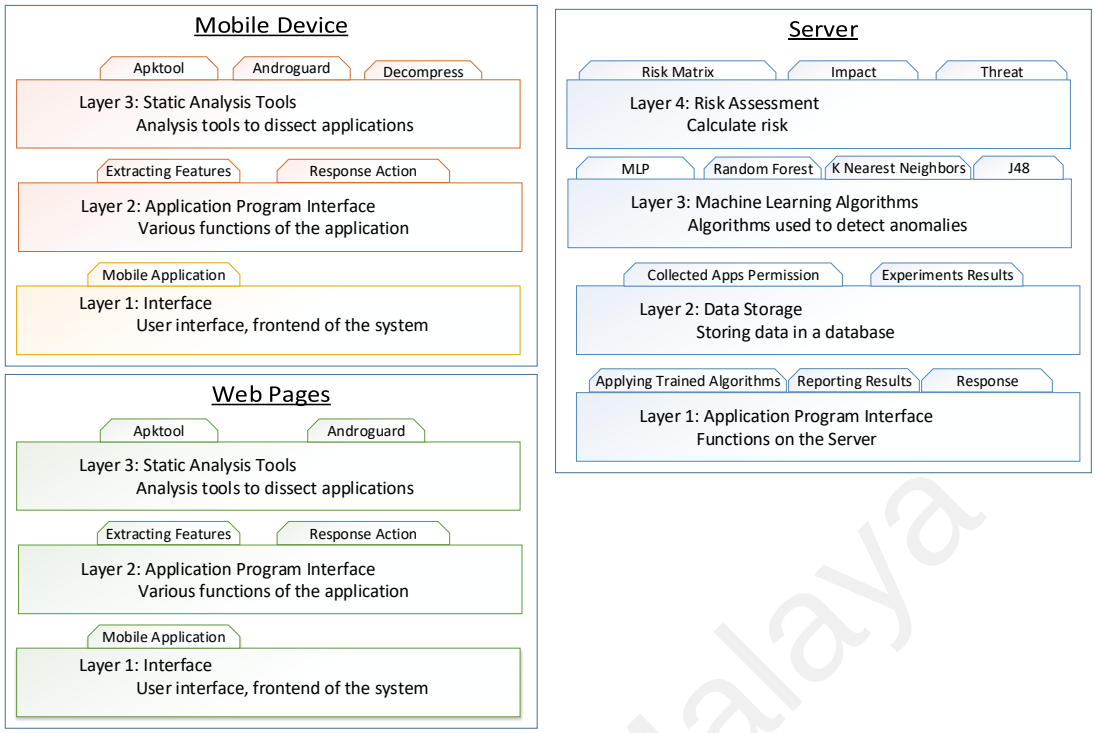

Figure 4.2: Layer Framework of the EZADroid System

Figure 4.2 depicts the structure layer of the EZADroid. Each layer represents a function within the respective system. There are four (4) structured layers: web pages, mobile devices, server and responses. Each layer has its own specific action to make sure the process functions effectively. For instance, the application programme interface layer in the mobile devices and Webpages perform activities such as collecting and submitting the apk files to the server. These activities continue at the server when dissecting the apk files and extracting the features through static analysis. These structured layers are important in presenting the flow process of the EZADroid. Figure 4.3 depicts the flow process and the layers of interactions.

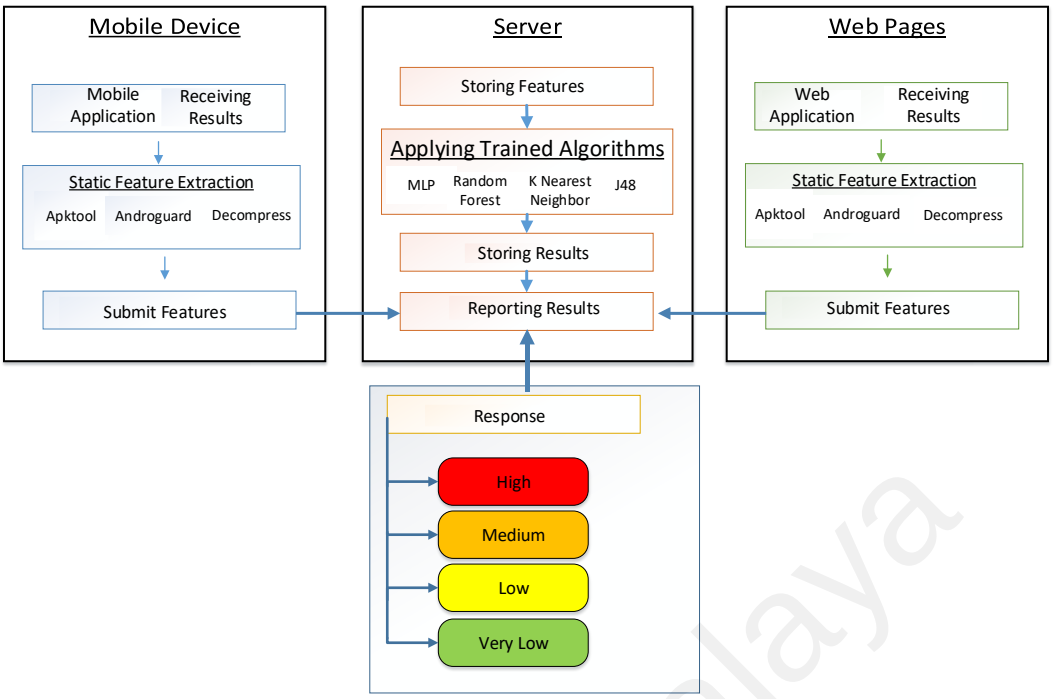

Figure 4.3: Layer Interactions

4.2 Machine learning classifiers

It is important to determine the best machine learning classifiers for Android malware detection because it able to produce models that can analyze more complex data and deliver faster results. This section discusses the output gained from the machine learning classifier. The output is derived from the results that were extracted after training and testing the sample dataset which comprised the malware and the benign dataset.

In this study, 10-fold cross validations and split percentages were conducted to evaluate the performance of the machine learning classifiers for the collected dataset. The 10-fold cross validations were used to construct 10 identical instances in the dataset. Finally, the results from each of these instances were combined into one composite final result. The significance of using the 10-fold cross validations is that they are able to produce a realistic result.

4.3 Evaluation measure

This section identifies the common evaluative measures noted in the research community. The effectiveness of the malware detection system is assessed based on accuracy, true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), f-measure, precision, and recall through the evaluation measures used (Narudin et al., 2016; Razak et al., 2017). In this study, the standard metrics were used to evaluate malware detection. A true positive (TP) refers to an instance where the detection is correctly noted as malicious. The higher the true positive, the better the result. A false negative (FN) represents an instance where the detection is incorrectly noted as benign. A true negative (TN) is a benign application detected correctly as benign. A false positive (FP) is a benign application detected incorrectly as malicious, as shown in Table 4.1.

Table 4.1: IDS confusion matrix

		Prediction Condition	
		Prediction positive	Prediction negative
Actual	Condition positive	True positive (TP)	False negative (FN)
	Condition negative	False positive (FP)	True negative (TN)

TPR, also called recall rate, is defined as: $TPR = \frac{TP}{TP+FN}$ **4-1**

TN) is defined as: $TNR = \frac{TN}{TN+FP}$ **4-2**

FPR is defined as: $FPR = \frac{FP}{FP+TN}$ **4-3**

False negative rate (FNR) is defined as: $FNR = \frac{FN}{FN+TP}$ **4-4**

Accuracy is defined as: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ **4-5**

Precision is defined as: $Precision = \frac{TP}{TP+FP}$ **4-6**

F-measure is defined as: $F - measure = \frac{2\ x\ TPR\ x\ Precision}{TPR+Precision}$ **4-7**

With the formula provided in assessing accuracy, the section below discusses the evaluation measures as demonstrated in Table 4.2.

Table 4.2: Evaluation measures

Evaluation measure	No. of tested apps	Type of analysis	Year	Reference
True positive rate	800	Dynamic	2015	(Gheorghe et al., 2015)
	1738	Static	2012	(Wu et al., 2012)
	2000	Static	2013	(Yerima et al., 2013)
	6863	Static	2015	(Suleiman Y. Yerima et al., 2015)
True negative rate	1100	Static	2014	(Deshotels et al., 2014)
	2000	Static	2013	(Yerima et al., 2013)
False positive rate	1000	Dynamic	2014	(Narudin et al., 2016)
	1257	Dynamic	2013	(Feizollah et al., 2013)
	120	Dynamic	2012	(Dini, Martinelli, Saracino, et al., 2012)
False negative rate	1100	Static	2014	(Deshotels et al., 2014)
	2000	Static	2013	(Yerima et al., 2013)
Accuracy	800	Dynamic	2015	(Gheorghe et al., 2015)
	1738	Static	2012	(Wu et al., 2012)
Precision	1000	Dynamic	2014	(Narudin et al., 2016)
	174971	Static	2015	(Cen et al., 2015)
F-measure	1000	Dynamic	2014	(Narudin et al., 2016)
	1738	Static	2012	(Wu et al., 2012)
	800	Dynamic	2015	(Gheorghe et al., 2015)

From the information given above, it shows that evaluation measures and the number of dataset used have a significant role in measuring the malware detection system.

4.4 Area under curve (AUC) performance

The area under the ROC curve, known as area under the curve (AUC), is widely used in optimising the problem and for weighing classifier performance on malware dataset. The AUC results identified were able to measure the detection approach as good or bad. An area of 1 indicates perfect prediction while an area of 0.5 indicates a bad prediction. The AUC levels were listed as: 0.9-1.0 = perfect prediction; 0.8-0.9 = excellent prediction; 0.7-0.8 = good prediction; 0.6-0.7 = fair prediction; and 0.5-0.60 = poor prediction. Table 4.3 defines the AUC threshold which describes the performance of malware detection system (Narudin et al., 2016).

Table 4.3: AUC performance threshold

Threshold	Description
1.0	Perfect prediction
0.9	Excellent prediction
0.8	Good prediction
0.7	Mediocre prediction
0.6	Poor prediction

4.5 Summary

This chapter has illustrated the proposed framework that was developed for this study. It also described the characteristics of the proposed framework. The schematic presentation of the proposed framework and its major layers of interactions were also noted in the proposed EZADroid. This chapter also described the layers of interactions that occurred between the modules (i.e. web pages, mobile devices and servers) by explaining the functional and non-functional characteristics of the main components of the framework. Added to this is the highlight of several significant features of the framework. The next chapter explains the comprehensive experiments conducted in the proposed EZADroid.

CHAPTER 5: EVALUATION OF RISK ANALYSIS AND MALWARE DETECTION FRAMEWORK

This chapter presents the performance of the evaluation methods which were used to evaluate and validate the execution performance of the proposed framework. The novelty of the framework is its ability to assess the risk of Android applications and to detect malware on mobile devices by sending them to the servers for processing. The process helps to reduce the resource consumption on mobile devices and avoid heavy processing issues. In this framework, all the detection processes were performed on the server without affecting the mobile device resources. The aim of this chapter is thus to evaluate the proposed framework in terms of validity and feasibility. The evaluation is carried out to verify how well the system meets the research objective.

Four (4) experiments were conducted on the proposed framework. These experiments involved the static analysis technique (bio – inspired analysis, features selection, time based detection and risk analysis). The first experiment describes the static analysis procedure. It analyses Android permissions for malware detection by using bio-inspired algorithms. The second experiment is also performed on Android permissions to see how feature selection plays an important role in determining the performance measures of malware detection. The third experiment discusses how future malware can be predicted by using time-series analysis. The fourth experiment assesses the risk of Android applications on mobile devices. The assessment produced are returned to users as a response.

5.1 Dataset descriptions

This section describes the collection of the dataset. The dataset is important; they are an integral and critical part of the research process. The dataset paves the way for understanding the malicious and benign activities. Upon further examination, the dataset

is then analysed and the results are used for predicting future events in Android malware. The growth of Android malware comes hand in hand with the proliferation of online repositories.

5.1.1 Malware Genome Project

The Android Malware Genome Project is a malware repository that focuses on the Android platform. Its aim is to characterise existing Android malware. The samples of this study were collected from August 2010 to October 2011. The total samples collected contained more than 1,200 malware samples that come from 49 malware families. This number covers the majority of the existing Android malware families. Zhou and Jiang (Zhou et al., 2012) had characterised the samples from various aspects including their installation methods, activation mechanisms as well as the nature of their carried malicious payloads. These characteristics of the dataset have been published by Zhou and Jiang (Zhou et al., 2012).

5.1.2 Drebin

Drebin dataset are made publicly available by the MobileSandbox project (Arp et al., 2014). They are considered the largest available malware dataset. The dataset collected from Drebin came to a total of 5,560 malwares which were composed of 179 families. The samples were collected between August 2010 to October 2012. The dataset was published as a means of filling in the gap for Android malware detection where malware dataset was required for experimental purposes. This malware dataset was used in the experiments which help in developing a more effective malware detection system. Prior to use, this dataset was scanned by anti-virus applications. The results showed that 90 percent of the samples detected were malware. Drebin is well-accepted among researchers (Karim, 2016; Feizollah et al., 2017) and previous studies (Firdaus et al., 2017; Feizollah et al., 2017) had also used similar dataset.

5.1.3 AndroZoo

AndroZoo comes from the Android Applications which were sourced from several sites such as the official Google Play app market. It contains 5,416,421 different applications and each application was analysed by 70 of different AntiVirus products so as to distinguish them as malware (Allix et al., 2016). This dataset contributes to the new potential research in Android malware detection. Figure 5.1 shows the interface of the AndroZoo website published by the University of Luxembourg.

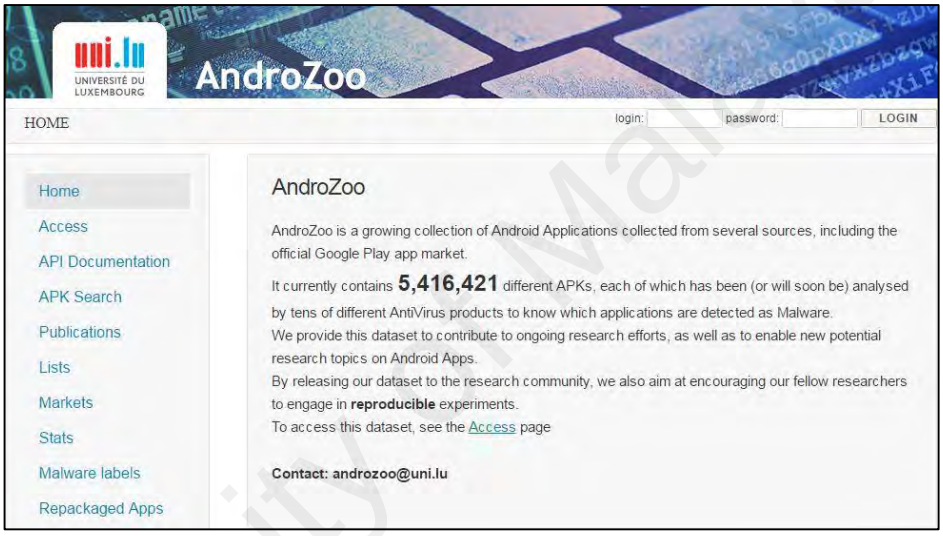

Figure 5.1: Website of AndroZoo

Figure 5.1 illustrates the AndroZoo website which provides access to researchers to download Android applications. The website also provides labels with their respective results for each apk. This type of result is important to researchers as it helps to ensure the validity of the apk by indicating whether they are malware or benign applications. This contributes to the efficiency of the system.

5.1.4 Google Play store

Google Play store is Google's official market source for Android applications and downloads (Google, 2017). It contains more than one million applications including games. Google Play store is a wide resource which allows researchers to search for specific dataset using keywords or by browsing applications in the Google Play library.

Most of the applications in this store are free for downloads. Some of applications need to be purchased as they contain more functions. This store is suitable for collecting benign applications because Google Play store provides an automated antivirus system known as Google bouncer (Tam et al., 2017). It has the capability to scan both the new and existing applications for malware and benign.

5.1.5 Benign dataset

To collect the benign dataset, clean applications from AndroZoo were gathered (Allix et al., 2016). In this study, the experiment only used benign applications taken from Google Play. This is because each submitted application must go through Google Bouncer before it is available for downloading. In addition, Google is continuously updating and improving the effectiveness of Google Bouncer which is responsible for the analysis of any malicious applications. Google Play applications have been categorized into 27 main categories while the games have been categorized into 17 subcategories (Feizollah et al., 2017). To ensure cleanliness, all the applications were scanned on VirusTotal and then analyzed by 70 of different AntiVirus products such as Avira, Comodo and AVG Technologies. Additionally, only applications with a score of 0/50 were used for the experiments as this indicate that there is no positive number for the malicious activities. By doing this, completeness of the genuine benign dataset is confirmed.

5.2 Experiment I: Evaluation of bio-inspired

The leaking of sensitive data on Android mobile devices poses serious threats to users. It occurs when unscrupulous attacks violate the privacy of users. However, detecting the attack is challenging due to the similarity of the permissions noted in malware and benign applications. This experiment aims to evaluate the effectiveness of the machine learning approach for detecting Android malware. In this experiment, we applied the bio-inspired algorithm as a feature optimisation technique for selecting the relevant permission

features that are able to identify malware attacks. A static analysis technique combined with machine learning classifiers was then developed from the permission features noted in Android mobile devices. This is used for detecting the malware applications. This study compares the bio-inspired algorithms (particle swarm optimisation (PSO) and the evolutionary computations with information gain as a means to detect the best feature optimisation technique in selecting the features.

5.2.1 Experiment setup and procedure description

In implementing a mechanism for Android malware detection, a machine learning approach which trains the sample dataset to learn the behaviors of the benign and malware applications was developed. The mechanism was also implemented to determine the severity of new applications as malware or benign.

The main components of the Android malware detection system are presented in Figure 5.2. Here, three (3) components are noted in the detection architecture. They include data collection, machine learning, and the database. The system describes the relationship between the components and the purpose of each component. Data collection begins by gathering all the permissions which include benign and malware applications. The process includes decompiling the apk file. This is followed by the process which extracts and filters the permissions. All the collected permissions are stored in a readable format and saved as a .arff file. This file contains all the feature attributes that are used for the feature optimisation approach. This approach is able to exclude irrelevance and noise in the dataset (Kumar et al., 2014). In the context of this experiment, the bio-inspired algorithms (particle swarm optimization (PSO) and evolutionary computation) and information gain play the role of identifying and selecting the best features. The main purpose of using the three (3) features optimisation technique was to locate the difference between the bio-inspired algorithms and the non-bio-inspired algorithms. The database

noted here also stores a collection of data and results. Figure 5.2 illustrates the malware detection architecture.

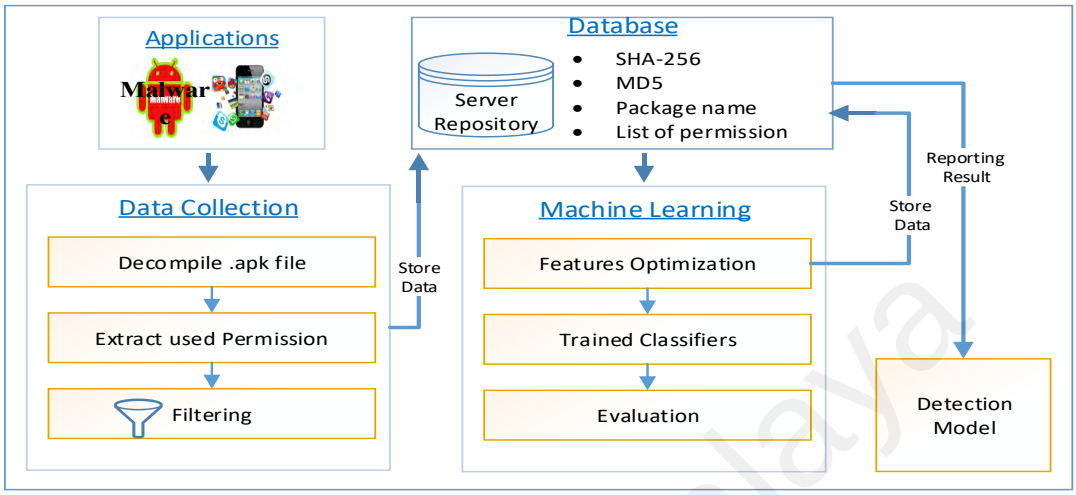

Figure 5.2: Malware detection architecture

In this experiment, data collection and the feature optimisation process are important for detecting malware. Specifically, the data collection process obtains the benign and malware behavior data from the application permissions. It then sends the information to the database. Here, the data are filtered based on the permission and package names. This filtering is important because it helps to ensure that the duplicate applications and features (used permission) are removed from the database. The filtered features are then sent to a machine learning process for the feature optimisation which provides the number of fold values for each permission.

5.2.2 Data collection phase

This section discusses the data collection process which compiles the benign and malware applications (.apk) datasets. The samples chosen for this experiment were randomly drawn from the Drebin and AndroZoo dataset. The dataset retrieved from AndroZoo was confined to the applications drawn from Google Play store. Table 5.1 depicts the summary of the dataset.

Table 5.1: Dataset summary

Dataset	Source	Total used in experiments
Benign	Androzoo	3500
Malware	Drebin	5000
Total		8500

a) Decompiling the apk file

This process begins by collecting the benign and malware applications which amounted to 8,500 samples dataset with 5,000 being malware and 3,500 being benign applications. The benign applications were downloaded from AndroZoo which belongs to Google Play store. The samples comprise a collection of more than three (3) million applications (Allix et al., 2016). Figure 5.3 illustrates the process of the data collection.

In the figure, the AndroidManifest.xml file is shown to contain essential information regarding the application information such as activities and permission. All the extracted permission must be labelled before they are stored in the database as a .arff file. The value of the permission is stored as a binary number (0 or 1).

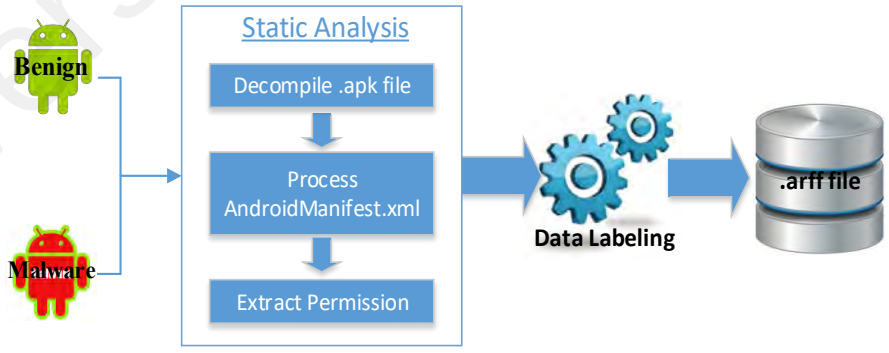

Figure 5.3: Data collection phase

For further investigation, feature optimisation was used. This helps to gain the best features among the 378 lists of permissions. Table 5.2 tabulates the top 10 permissions noted in the benign and malware applications.

Table 5.2: Top 10 permission in benign and malware applications

Benign application	Percentages (%)	Malware application	Percentages (%)
INTERNET	94	INTERNET	80
ACCESS_NETWORK_STATE	88	READ_PHONE_STATE	70
WRITE_EXTERNAL_STORAGE	66	WRITE_EXTERNAL_STORAGE	49
WAKE_LOCK	65	ACCESS_NETWORK_STATE	28
READ_PHONE_STATE	52	SEND_SMS	27
VIBRATE	47	RECEIVE_BOOT_COMPLETED	22
ACCESS_WIFI_STATE	43	ACCESS_WIFI_STATE	21
ACCESS_FINE_LOCATION	38	WAKE_LOCK	20
GET_ACCOUNTS	38	RECEIVE_SMS	19
ACCESS_COARSE_LOCATION	36	READ_SMS	18

From the table, six (6) similar permissions were used in the benign and malware applications. They include: ACCESS_WIFI_STATE, ACCESS_NETWORK_STATE, WAKE_LOCK, WRITE_EXTERNAL_STORAGE, READ_PHONE_STATE and INTERNET. Among the top 10 permissions, four (4) were dangerous permissions such as RECEIVE_BOOT_COMPLETE, READ_SMS, RECEIVE_SMS, and SEND_SMS. These dangerous permissions occur recurrently in malware applications but infrequently in benign applications. Permissions such as WRITE_SMS has the capability of subscribing to premium SMS service without user knowledge thereby directly providing profit to attackers (Elish et al., 2015; Somarriba et al., 2016). Although this study implements a large number of sample applications for analysis, the experiment is still considered small when compared to real world applications in the Android market. Therefore, the outcome is considered as a reflection of the permission trend in Android applications today. Figure 5.4 illustrates the total number of permissions requested by the benign and malware applications.

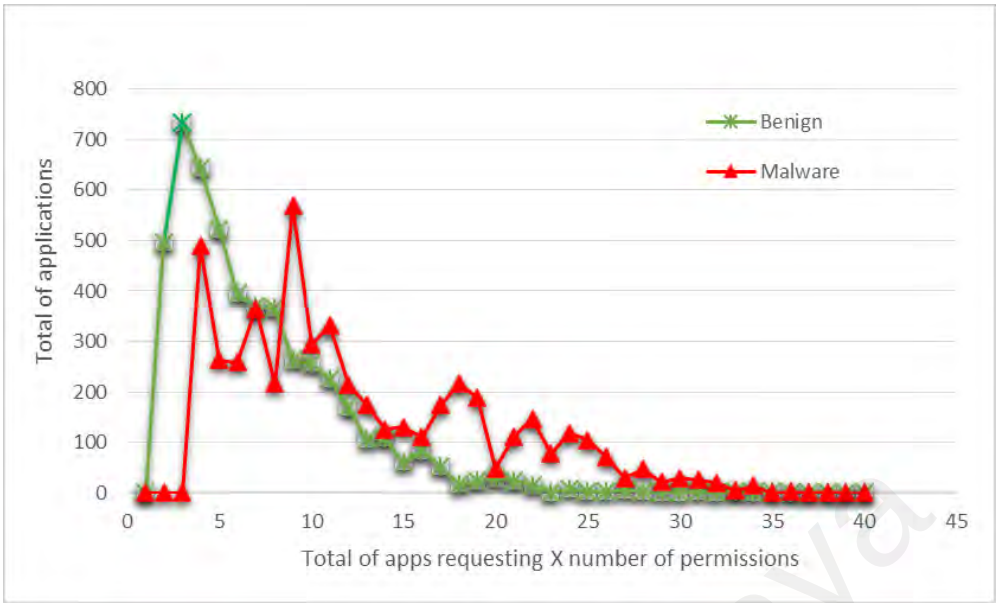

Figure 5.4: Total number of applications requesting permissions

The figure shows that malware applications requested more permissions than benign applications. It seems evident that malware applications have similar types of requested permissions as benign applications but its aim is malicious as it aspires to infect and disable the functionality of mobile systems. It is also noted that more than 500 malware applications requested only nine (9) permissions as are shown in the graph. This indicates that the technique used by an attacker to threaten mobile devices is by using permissions.

b) Machine learning phase

The use of the machine learning approach is to ensure that mobile device users are able to optimise the permission features through the feature optimisation approach. This approach reduces time for training and testing; it also reduces overfitting while also simplifying the malware detection system. It is also significant to data processing (Kumar et al., 2014). Inevitably, feature optimisation is a crucial core for building any malware detection system (Zhang et al., 2003). Without a good knowledge of the classifications, users will find it challenging to identify which features are relevant when detecting malware.

As a consequence of this, the dataset will contain redundant, irrelevant and relevant features. These features, if not attended to, can reduce the classification performance. To resolve this problem, feature optimisation was thus applied as a measure of improving the accuracy of experiments thereby serving as an effective component in the malware detection system (Ahmad, 2015). Feature optimization also has the ability to optimise the evaluation measure (Kumar et al., 2014).

Figure 5.5 summarises the selection of the significant features for Android malware detection. The feature optimisation approach was applied by using a specific metric which computes and returns a score for each feature individually (Shabtai et al., 2012). This process helps the feature optimisation approach to acquire significant features through the best accuracy rate coupled by the lack of over-fitting. The process begins with the cleaning of the dataset. This involves removing irrelevant and redundant features.

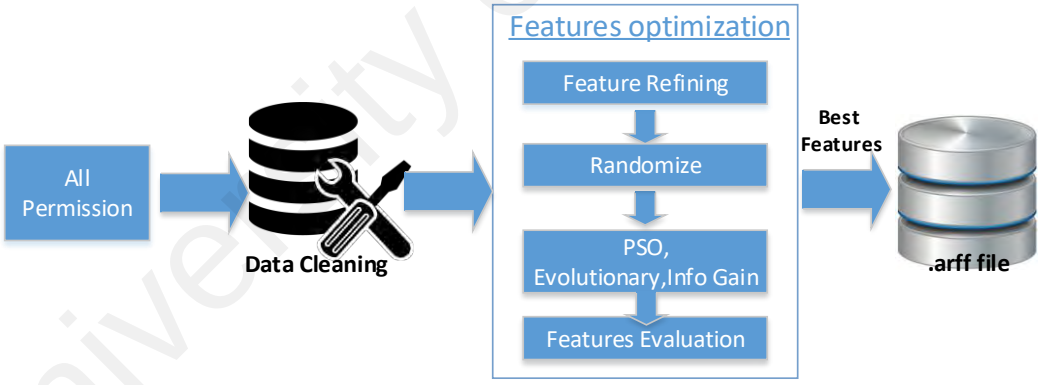

Figure 5.5: Machine learning phase

The next step of the experiment uses WEKA to acquire data randomisation which balances and controls the dataset. Doing so enhances the optimal sensitivity of feature optimisation. In this study, the bio-inspired algorithms (PSO) and the evolutionary computation algorithms were compared with information gain for feature optimisation.

Figure 5.6 shows the result of the comparison. The feature optimisation approach was based on the number of folds which totals to more than 90 percent. It is also based on the

information gain which uses a ratio of the information gain to select relevant features. This study chooses features with ranks ranging from 0 to 0.4 on the information gain. The PSO algorithm and information gain have 11 features while the evolutionary computation has 13 features. Here, it is seen that the number of the selected features is totally dependent on the type of algorithms used. The PSO algorithms carry a high performance in detecting malware. Table 5.3 lists the permission features noted after PSO was adopted in the approach.

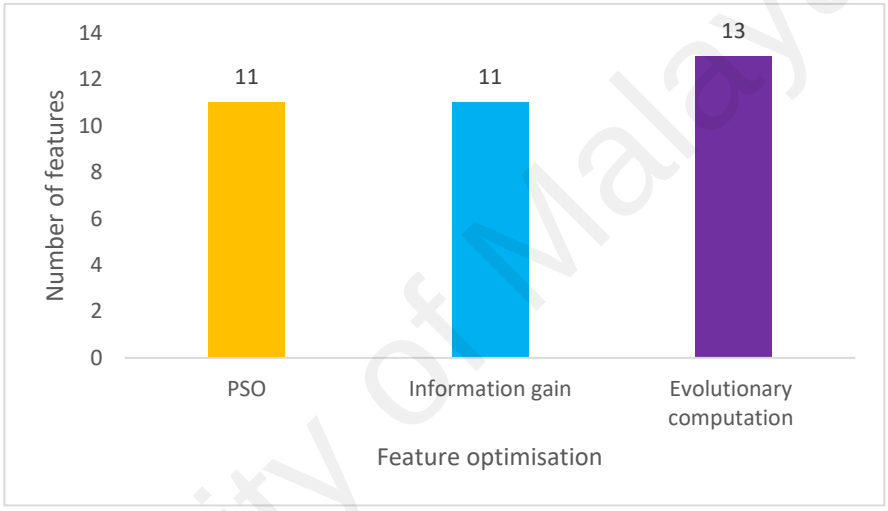

Figure 5.6: Comparison of feature optimisation approach based on number of features

Table 5.3 presents the list of feature permissions that were extracted after using the bio-inspired algorithm and PSO as the feature optimization approach. Here, it is noted that eight (8) of the 11 permission features belonged to the dangerous levels. These features are high-risk permissions which require application permissions to grant access and control over the mobile devices.

For instance, five (5) permissions were similar to the top 10 permissions noted in the malware applications of Table 5.2, READ_PHONE_STATE, READ_SMS, SEND_SMS, and RECEIVED_BOOT_COMPLETE ACCESS_WIFI_STATE. In this context, it is worth noting that the READ_SMS, SEND_SMS, RECEIVED_SMS and the BILLING permissions allowed the conveying of billing request and response between

applications on a mobile device with the Google Play server. Clearly, these were used by attackers to make mobile device users purchase premium rate messages.

If the user was unaware of this incidence, the permission granted able to incur a loss of money for the mobile device user and he user then ends up paying for expensive messages which are unrequired. It is also noted that READ_GSERVICES belongs to the Android.permission-group.ACCOUNTS. As malicious permissions, they have the capability to access the user's available accounts on his/her mobile device thereby incurring financial losses.

Table 5.3: List of permission features

List of features	Description	Protection levels
ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks	Normal
DELETE_PACKAGES	Allows an application to delete packages. Starting in Nougat (N), user confirmation is requested when the application deleting the package is not the same application that installed the package	Normal
READ_PHONE_STATE	Allows read only access to phone state	Dangerous
READ_SMS	Allows an application to read SMS messages	Dangerous
RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting	Normal
RECEIVE_SMS	Allows an application to receive SMS messages.	Dangerous
SEND_SMS	Allows an application to send SMS messages.	Dangerous
WRITE_APN_SETTINGS	Allow an application to change network setting, interpret and intercept all the network traffic	Dangerous
WRITE_HISTORY_BOOKMARKS	Allows an application to write (but not read) the user's browsing history and bookmarks	Dangerous
BILLING	Allows sending In-app Billing request and managing In-app Billing transaction using Google Play	Dangerous
READ_GSERVICES	Allows this app to read Google service configuration data	Dangerous

Further, the WRITE_APN_SETTINGS also seems to allow the malware applications to monitor, redirect or modify the network packet without the user's knowledge. The WRITE_APN_SETTINGS permission is used by malware to steal Transaction Authorisation Codes (TAC) during the communication between mobile devices and online banking transactions (Feizollah et al., 2017). To conclude, the list of permission features is comprehensive for malware detection since it contains malicious activities.

5.2.3 Evaluation and results

To evaluate the performance of the machine learning approach in detecting Android malware and to ensure that the machine learning approach generate a complex analysis of the permissions, this study conducted another experiment on two (2) public Android application dataset. Here, the benign applications comprise 3,500 randomly selected samples taken from the AndroZoo database which were downloaded only from Google Play Store. The 5,000 malicious Android application dataset came from Drebin. The total number of evaluations made for the Android applications thus amounted to 8,500.

a) Machine learning performance

This section provides part of the result taken from the experiment conducted. These applications were mixed together for training and testing the dataset. To train and test the models for machine learning, the parameters including the cross-validations need to be set. Table 5.4 illustrates the variation of the detection performance of machine learning as seen in the various categories of classifiers.

The detection performance of five (5) classifiers for Android malware detection is presented above. Each classifier performance was evaluated through five (5) performance metrics such as f-measure, recall, TPR, precision and FPR. The table indicates that the PSO presents a better performance when compared to evolutionary computation and information gain. It can also be noted that AdaBoost on the PSO provides a good detection

performance of 95.5 percent for TPR. The average TPR for each classifier was noted to be higher than 90 percent. Table 5.4 clearly shows that the machine learning approach is effective in detecting malware such as random forest, multi-layer perceptron, K-nearest neighbors, AdaBoost and J48. Therefore, it is worth noting that feature optimisation has a significant role in identifying the relevant features. This experiment evaluated the ROC curve, based on the PSO results for the classification accuracies, as explained below.

Table 5.4: Detection performance results

Features optimization	Classifier	TPR (%)	FPR	Precision (%)	Recall (%)	F-Measure (%)
PSO	Random Forest	93.6	0.155	89.6	93.6	91.6
	MLP	91.9	0.13	90.6	91.9	91.2
	kNN	93.7	0.15	89.4	93.7	91.5
	Adaboost	95.6	0.32	81	95.6	87.7
	J48	93.1	0.16	89	93.1	91
Evolutionary computation	Random Forest	88.6	0.09	93.3	88.6	90.9
	MLP	88.5	0.09	93.3	88.5	90.8
	kNN	88.7	0.09	93.2	88.7	90.9
	Adaboost	88	0.11	91.8	88	89.8
	J48	88.4	0.09	92.8	88.4	90.6
Information gain	Random Forest	90.1	0.11	91.7	90.1	90.9
	MLP	90.1	0.11	91.8	90.1	90.9
	kNN	90.3	0.11	91.7	90.3	91
	Adaboost	87	0.1	92.1	87	89.5
	J48	89.9	0.11	91.5	89.9	90.7

b) Receiver operating characteristics curve (ROC)

Focussing on the TPR, it is observed that the receiver operating characteristic curve (ROC) is a technique used for visualising the evaluated performance. This has been used in machine learning and data mining approaches (Fawcett, 2006). This technique also provides reliable information about the performance: the closer the apex of the curve is towards the upper left corner, the better the performance. In the context of this experiments, the ROC curve was used to measure the quality and the effectiveness of the

prediction classifiers. Figure 5.7 illustrates the tradeoff between the TPR and FPR. Cross validations with 11 features were applied on five (5) classifiers. The left corner represents the percentages of the samples which were correctly detected as malware. This statistic indicates the high performance of the occurrence. The results further confirmed that the experiments conducted had achieved high accuracy rates with minimal false alarms. Here, the curve is also indicated at the top border of the ROC curve. The five (5) classifiers demonstrate the good results of the ROC curve because it turned towards the upper left-hand corner of the plot.

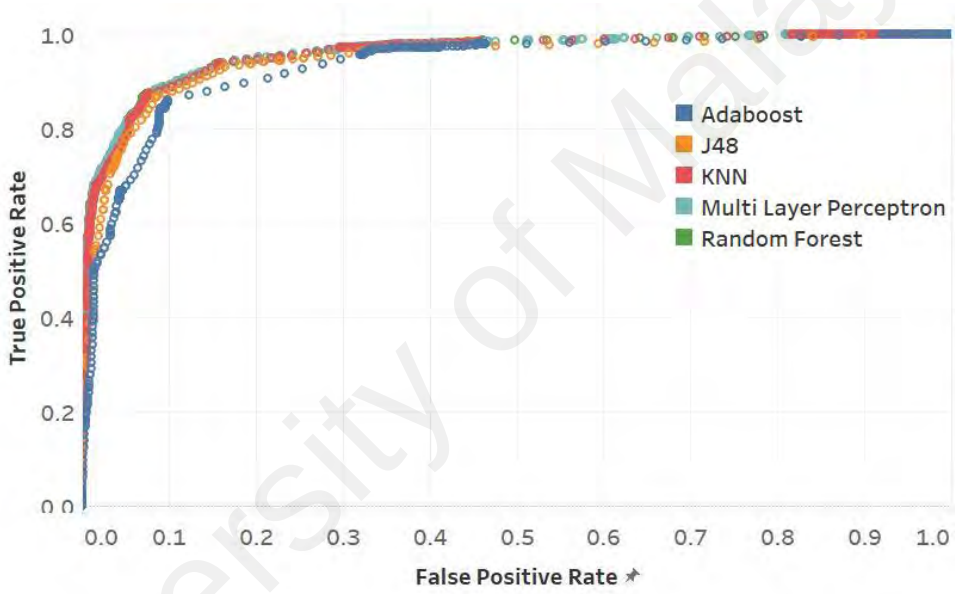

Figure 5.7: Performance of ROC curve

The multi-layer perceptron (MLP) outperformed all the classifiers. This shows that the MLP is effective in predicting any case as positive (malware). It also appears to be presenting an acceptable accuracy rate. The MLP is typically designed to minimise error rates; it measures how far the threshold to each sample training falls from its ideal value. Other classifiers such as AdaBoost, J48, KNN and random forest also provide good measures in mobile malware detection.

Table 5.5 lists the results of the AUC that was taken from the experiment which uses five (5) classifiers. Here, all the five (5) classifiers showed excellent prediction rates thus

the AUC values were acceptable for detecting mobile malware in the Android platform. This experiment specifically performed 10 cross validations which showed that the MLP classifiers had accomplished high detection accuracy. This performance is comparable to the random forest thereby indicating that the selected features were effective in detecting malware.

Table 5.5: Results of AUC

Classifier	AUC	Level
MLP	0.958	Perfect prediction
RF	0.957	Perfect prediction
KNN	0.956	Perfect prediction
J48	0.945	Perfect prediction
AdaBoost	0.932	Perfect prediction

c) Empirical assessment

The experiment applied in this study is similar to Allix et al. (2016). In the steps that followed, the approach was validated by using different parameters that were involved in the measurement process. In this regard, the current study used the results of the PSO because the results showed the best performance, with a minimum number of features. All the results of the testing process were saved into an .arff file. These were then saved again into the .csv file in order to get a result that bears consecutive numbers. Following that, the consecutive numbers were generated into box plot graphs for experimental assessment purposes.

The PSO experiments were run with 10-fold cross-validation experiments using a few types of performance metric to validate the performance of the Android malware detection approach. The validation of the experiments assessed five (5) classifiers of the machine learning model. Figure 5.8, Figure 5.9 and Figure 5.10 illustrate the results of the validation test for precision, recall, and f-measure. Each boxplot splits the result into a quartile. The boxplot size is made from the 25 percent score which depicts the minimum

value to the maximum value of the whisker. The results suggest that majority of the five (5) classifiers had revealed a high precision rate with an average value of over 0.89. The average value for the recall performance was at 0.63, demonstrating that 80 percent of the classifiers had a recall value that is equal or higher than 0.63. It also shows that five (5) classifiers were going from 0.0 to 1.0. In particular, the precision and recall value and the f-measure achieved a value of 0.72. All these results are important for measuring the performance of the classifiers in detecting malware.

The increase of the precision rate indicates that the classification of the machine learning model is related to the low positive rate. It also shows an accurate result when detecting malware. However, the high recall shows that the malware applications have similar features to the benign applications. Thus, it was concluded that the results indicate an accuracy precision that carries a high recall value. In short, it is effective in detecting malware accurately.

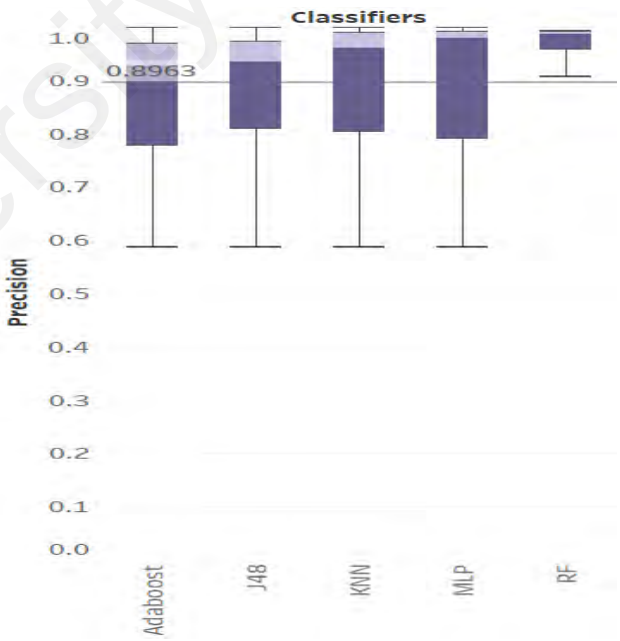

Figure 5.8: Precision

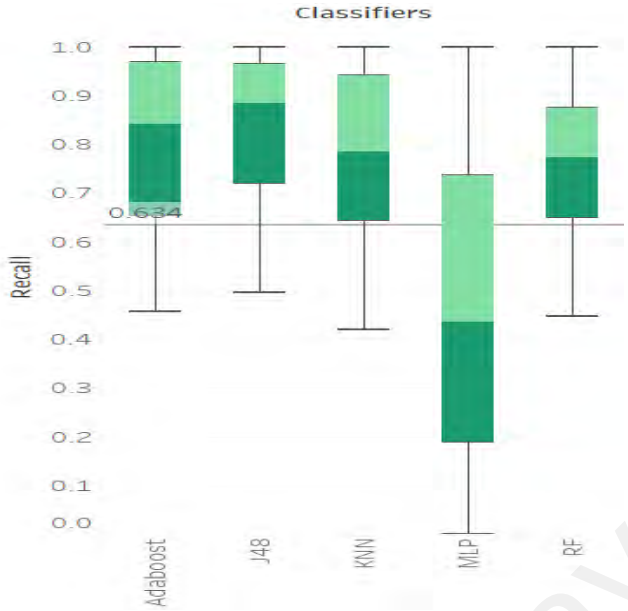

Figure 5.9: Recall

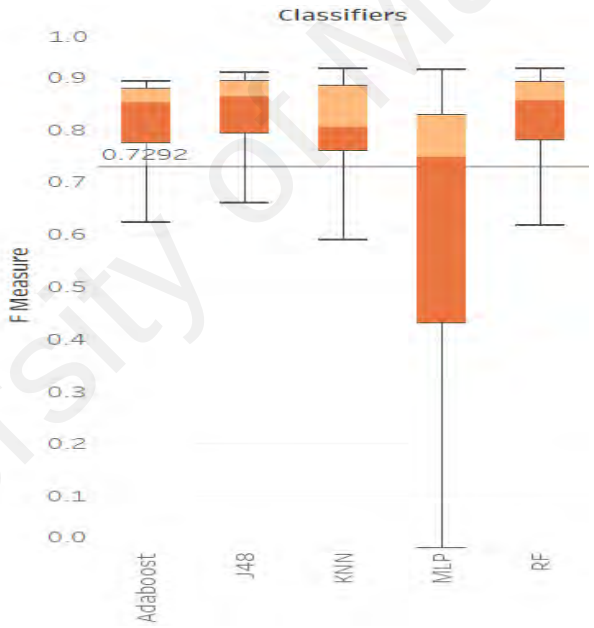

Figure 5.10: F-measure

This assessment is important for Android malware detection because the benign samples are more than the malware samples in real situations. The assessment implies that researchers require more malware samples with known features so as to be able to increase the performance evaluation for detecting malware.

5.2.4 Discussion

This section presents the discussion of the findings that are related to malware detection in brief. As has been mentioned, this study performs malware analysis by using a machine learning approach to detect Android malware. In doing so, it also considered other issues which helped the experiment to achieve a good performance evaluation during the Android malware detection. The first issue concerns feature collection. This issue was viewed as complementary to other studies. It appears that the performance evaluation had greatly benefited from the higher sets of data. The larger dataset would also give more features thus requiring more resources and constraints on the model. Therefore, the implication drawn from this study is that focus should be given to developing better features to increate malware detection performance.

This study also examines the role of the bio-inspired algorithm and the non-bio-inspired algorithm for feature optimisation. The results indicate that particle swarm optimisation (PSO), as compared to evolutionary computational and information gain, leads to better feature characterisation. This improves the performance evaluation of Android malware detection. It further shows that if the machine learning approach is unable to identify malware correctly, then the feature characteristics have not been selected properly. Besides that, any malware application should have some special characteristics before being defined as malware. Therefore, to detect more malware, a set of fine grain features need to be collected. This study collected 378 features from 8,500 Android applications but only 11 selected features were presented. Based on this, it can be concluded that fine grain features play an essential role in performance evaluation.

The second issue of this study relates to the machine learning approach. This study has applied several machine learning classifiers for the purpose of identifying the reliable predictive correlation in detecting malware. The experiment is important for the purpose

of finding effective classifiers that able to predict Android malware. The experiment showed that the AdaBoost outperforms the other machine learning classifiers.

The third issue of this study relates to the training samples which clearly still served as the major challenge in Android malware collections. It is deduced that more training samples are able to improve the accuracy of the classifiers in malware detection. Therefore, genuine malware dataset ought to be identified and researched for the purpose of sharing the outcome with the public and guided by some rules. This contribution can lead to more discoveries and so provide more improvement for areas related to Android malware detection. Nonetheless, all outcomes tend to depend on the goals of a research that is whether to detect new malware or signatures. In order to detect new malware, a study would require the latest dataset because attackers are constantly updating their techniques to steal information from mobile device users.

5.2.5 Conclusion

This study has presented the performance of the evaluation approach in detecting Android malware. The study used a machine learning classifier to identify the relevant permission features which were used to evaluate the learning types of classifiers. The study also used the ROC curve and the TPR to determine the effectiveness of the classifiers. This study has evaluated various categories of machine learning classifiers which were expected to improve the Android malware detection performance. For this purpose, large training samples were extracted and the most effective classifiers were identified. They include Random forest, J48, K-nearest neighbors, Multi-layer perceptron, and AdaBoost.

In all the experiments performed, a dataset consisting of real Android malware and benign sample applications were considered. In particular, this study evaluated a total of 5,560 malware samples drawn from Drebin and a total of 3,500 benign applications

collected from the Androzoo dataset. This experiment also used the static analysis technique to differentiate the benign and malware applications. The machine learning process comprised three (3) phases: (1) feature optimisation; (2) trained classifiers; and (3) the evaluation of machine learning classifiers.

The experiment results indicate a 95.6 percent detection rate for the TPR when using AdaBoost classifiers on the Drebin malware samples which were analysed using the PSO feature optimisation. The results also showed that the experiment which used the machine learning model had achieved a higher accuracy rate. Moreover, the evaluation of the machine learning approach in the experiment also indicated its efficiency in detecting Android malware. This proves that machine learning classifiers have the capability to detect the latest Android malware. Hence, it concluded that the greatest gains in detection performance will continue to be derived from an improved feature optimisation technique and from learning classifiers that are more efficient in detecting malware cases.

The significance of this study lies in use of the latest Android data collection and the evaluation of the different classifiers. Additionally, the results of the performance evaluation also showed that this mechanism as proposed by the current study, has the potential to detect well-known Android malware.

5.3 Experiment II: Evaluation of machine learning classifiers

In order to address the security concerns contained within Android applications, Experiment II introduces a novel machine learning approach which aims to detect malware by using the permission patterns used by Android applications with minimal features. The approach is based on the static analysis of the resource file of Android applications. The aim is to obtain permission patterns. For the sake of improving the detection accuracy rate, this study also used relevant features in minimal amounts. Specifically, this experiment applies permission features and the machine learning

approach to detect Android malware. For the purpose of evaluating the effectiveness of the proposed approach, this experiment compares the achieved results with results extracted from a baseline Android detection which uses a similar level of permission.

5.3.1 Experiment setup and procedure description

The methodology of Experiment II as performed in the current study is illustrated in Figure 5.11. Three (3) phases are involved: data collection, feature extraction and refining. These phases will help the experiment to select the relevant features noted among the overall features so that the effectiveness of the approach can be conducted. Here, the comparative analysis was used to detect the unknown malware. This is done by using the four machine learning classifiers of random forest, MLP, KNN and J48.

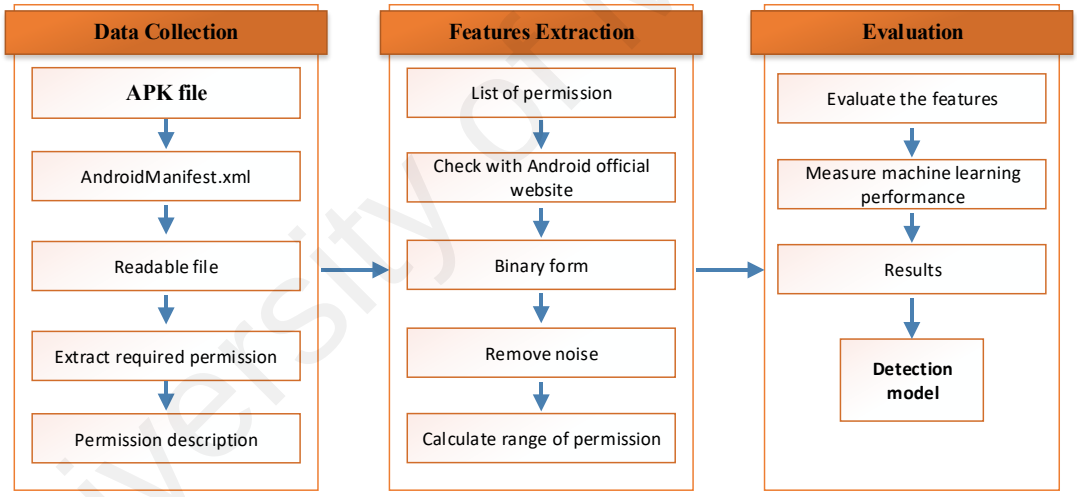

Figure 5.11: Methodology

Figure 5.11 depicts the construction of the Android malware detection approach based on the machine learning model which was implemented for application training and testing. It also helps to distinguish the benign applications from malware applications. Once the. apk file of an application is submitted, the approach analyses the permission and determines whether it is benign or malicious. The detection results, including the detailed information gathered from the machine learning performance evaluation, are then reported. This methodology is expected to detect unknown malware and to keep pace

with the evolution of Android malware. There are two types of experiments. Experiment I was conducted by using 12551 samples which include malware and benign applications and Experiment II was carried out by using 12551 samples which include malware and benign applications but this experiment used the feature selection approach instead.

5.3.2 Data collection phase

All the features were collected from the required permission types. A total of 274 different required permissions were collected from apk files through static analysis. This study uncompressed the apk file with Java code. The procedure then focused on the AndroidManifest.xml file which helped the experiment to obtain the permissions that were required by the applications. For instance, INTERNET served as the permission which allowed the applications to open the network sockets. It also allowed the applications to access the Internet. As a result of this, attackers were able to use this permission to download malware applications and then collect the sensitive information. Table 5.6 lists the summary of the dataset used in Experiment I and II.

Table 5.6: Dataset summary

Dataset	Source	Number used in experiments
Malware	Drebin	5551
Benign	Androzoo	7000
Total		12551

The features of the Android permission were first trained and then classified by using relevant features. When the machine learning classifiers have been trained well, some kind of regularity should appear in the extracted features. When this occurs, the features are then converted into binary forms of malware and benign applications, catering to two categorical features (1 and 0). In order to select the relevant features for effective malware detection, this study applies a similar feature selection approach that was used by Firdaus et al. (2017). In this way, the number of features was reduced from 274 features to 15

only. To ensure that there is a unique pattern occurring between the benign and malicious applications, it is necessary to look at the relevant features. By using a larger amount of data, this study was also able to classify the malware processes with more precision. Table 5.7 presents the list of permission used by the experiments.

Table 5.7: Lists of permission

Permission	Description
ACCESS_NETWORK_STATE	Allows applications to access information about networks
ACCESS_WIFI_STATE	Allows applications to access information about Wi-Fi networks
GET_ACCOUNTS	Allows access to the list of accounts in the Accounts Service
INSTALL_PACKAGES	Allows an application to install packages
READ_CONTACTS	Allows an application to read the user's contacts data
READ_PHONE_STATE	Allows read only access to phone state, including the phone number of the device, current cellular network information, the status of any ongoing calls, and a list of any Phone Accounts registered on the device
READ_SMS	Allows an application to read SMS messages
RECEIVE_BOOT_COMPLETED	Allows an application to receive the ACTION_BOOT_COMPLETED that is broadcast after the system finishes booting
RECEIVE_SMS	Allows an application to receive SMS messages
SEND_SMS	Allows an application to send SMS messages
WRITE_SMS	Allows an application to write SMS messages
READ_HISTORY_BOOKMARKS	Allows the app to read the history of all URLs that the Browser has visited, and all of the Browser's bookmarks
WRITE_HISTORY_BOOKMARKS	Allows the app to modify the Browser's history or bookmarks stored on your phone. This may allow the app to erase or modify Browser data
INSTALL_SHORTCUT	Allows an application to add Home screen shortcuts without user intervention
com.google.android.c2dm.permission.RECEIVE	Allows apps to accept cloud to device messages sent by the app's service. Using this service will incur data usage. Malicious apps could cause excess data usage.

5.3.3 Evaluation and results

The best results obtained from the experiments are further discussed. These are highlighted in bold in the following tables. The initial results indicate the comparison of the outcomes which were obtained from the four machine learning classifiers: random forest, MLP, KNN and J48. As this study also used the parameters of accuracy, FPR, recall, precision, and f-measure to examine the different measurements, the results are simultaneously provided. The results achieved from 30 percent of the testing set which used four (4) selected classifiers to perform the experiments are presented in Table 5.8. Here, the result illustrates the performance of each of the classifiers used in the two (2) experiment sets for the Android malware detection.

Table 5.8: Comparison with and without features selection approach

Experiment	Classifier	Without features selection				
		Accuracy (%)	FPR	Precision	Recall	F-Measure
I	Random forest	95.1	0.048	0.952	0.951	0.951
	MLP	93.8	0.061	0.939	0.938	0.938
	KNN	94.6	0.058	0.946	0.946	0.946
	J48	93.3	0.073	0.933	0.933	0.933
Experiment	Classifier	With features selection				
		Accuracy (%)	FPR	Precision	Recall	F-Measure
II	Random Forest	92.0	0.080	0.921	0.920	0.919
	MLP	91.6	0.084	0.917	0.916	0.915
	KNN	92.0	0.080	0.920	0.920	0.919
	J48	91.5	0.085	0.917	0.915	0.914

In Experiment I, the results showed that random forest classifier had achieved a higher accuracy result of 95.1 percent when compared to J48 which achieved only 93.3 percent. This outcome indicates that the random forest learning classifier is more effective than other selected classifiers in detecting the Android malware. Similarly, the instance-based classifier (KNN) also produced a high detection rate of 95 percent accuracy.

Table 5.8 illustrates the random forest and KNN classifiers both of which had achieved a 92 percent accuracy thus far, the highest precision value noted in the Android malware detection. This outcome indicates that feature selection plays an important role in determining the effectiveness of malware detection. The value for the FPR when using feature selection was also noted to be higher than the value of the FPR which was without feature selection. This happened because of the bias noted in the dataset as the benign samples have a common pattern as the malware samples. Most of the classifiers observed in Experiment II have similar FPR values of 0.08. The value for precision and the recall for Experiment I and Experiment II was more than 90 percent. The high precision rate indicates that the classifiers produced more relevant results. The high precision rate also indicates that the classifiers were producing accurate results, with majority being positive results.

Nevertheless, there are resource constraints on mobile devices, for instance, CPU, memory, battery and storage. In this regard, Experiment I and Experiment II were conducted based on the time spent by the classifiers. Time spent was tested because it is important for the malware detection system to detect any abnormal activity within minimal time without affecting the resource consumption on mobile devices. Table 5.9 shows the comparison of the processing time during the experiments - with feature selection approach and without feature selection approach.

The table Table 5.9 shows that by using minimal and relevant features, the experiments were able to improve the model for malware detection. When all the 274 features were used in the experiments, random forest classifier was shown to take a longer time to build the model, consuming 27.05 seconds. However, by using minimal features after the feature selection approach, the time was reduced to 2.27 seconds.

Table 5.9: Time taken to produce results (second)

Experiment	Classifier	Without features selection	
		Build model	Test model
I	Random forest	27.05	0.24
	MLP	25.64	0.03
	KNN	0.01	14.35
	J48	7.0	0.01
Experiment	Classifier	With features selection	
		Build model	Test model
II	Random Forest	2.27	0.14
	MLP	14.18	0.01
	KNN	0	2.09
	J48	0.29	0.0

In addition, the other classifiers also showed a decrease in processing time following the feature selection approach. Consequently, the results imply that when more features were used, the processing time taken to build the model also increased. Nonetheless, the size of the dataset also played a crucial role in the detection of malware, as this study indicates.

a) Confusion matrix

A confusion matrix is a table that describes the performance of a classification model. In this regard, the table provides two (2) possible predictable classes: “malware” and “benign”. For instance, if a model predicts the presence of malicious activities, the result would show “malware” and likewise “benign” if it does not detect any malicious activities. A confusion matrix also describes the information of the prediction in the testing phase as correct or incorrect. Table 5.10 shows the performance of the classifiers for the two experiments.

Table 5.10: Confusion matrix of classifiers

Experiment	Classifier	Without features selection		
		Actual	Predicted	
			Predicted malware	Predicted benign
I	Random Forest	Actual malware	1526	134
		Actual benign	49	2056
	MLP	Actual malware	1498	162
		Actual benign	70	2035
	KNN	Actual malware	1537	123
		Actual benign	81	2024
	J48	Actual malware	1499	161
		Actual benign	91	2014
Experiment	Classifier	With features selection		
		Actual	Predicted	
			Predicted malware	Predicted benign
II	Random Forest	Actual malware	1421	214
		Actual benign	88	2042
	MLP	Actual malware	1407	228
		Actual benign	89	2041
	KNN	Actual malware	1425	210
		Actual benign	93	2037
	J48	Actual malware	1398	237
		Actual benign	84	2046

The statistics above indicate that without using feature selection, the experiment had produced correct and magnificent results in predicting the unknown malware, with 1537 for the KNN classifier. In the incorrectly predicted perspective, the KNN also showed the most minimal value whether with or without feature selection. The outcome was 210 and 123 respectively. Meanwhile, random forest classifier also showed a prominent result in predicting malware. Consequently, the outcome suggests that random forest and the KNN classifiers were able to predict the unknown malware more accurately.

b) Receiver operating characteristics curve (ROC)

In the approach used in this study, the processes were classified as malware and benign applications based on the requested permissions. Besides using the performance matrix, this study also calculated the receiver operating characteristics (ROC) curve for each of

the machine learning classifiers. In this context, the TPR was regarded as the detection rate which correctly predicted the malware process and the FPR was regarded as the detection rate which incorrectly predicted benign as malware. Figure 5.12 presents the curve for the machine learning classifiers.

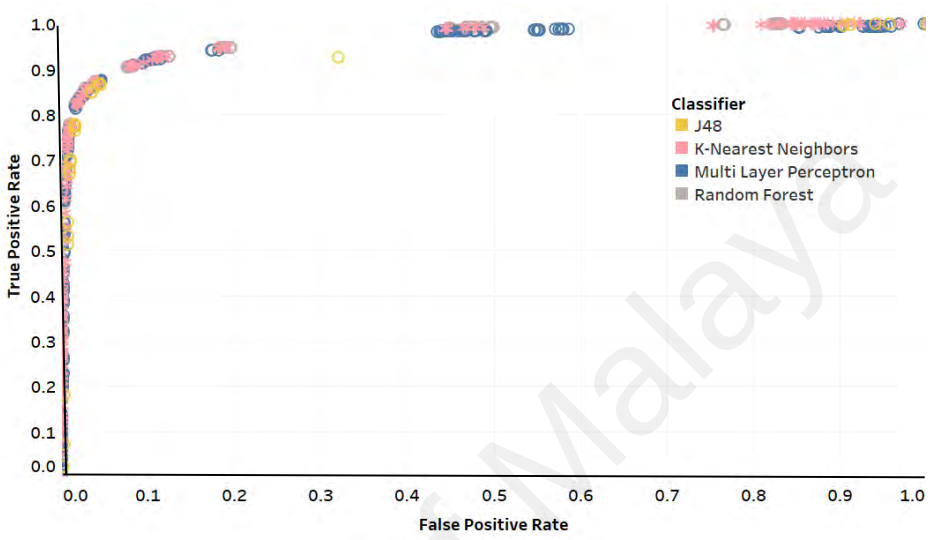

Figure 5.12: ROC curve

The horizontal axis in the above figure indicates the error detection rate; the vertical axis indicates the detection rate. Four (4) lines represent the individual ROC curve of the machine learning classifiers. The ROC curve is difficult to compare because it seems to be similar under the same conditions. Therefore, the area under the curve (AUC) was used to measure detection accuracy. The AUC results identified were able to measure whether the detection approach was good or bad. An area of 1 indicates perfect prediction while an area of 0.5 indicates a bad prediction.

The average AUC of random forest, MLP, KNN and J48 are 96.4, 95.8, 96.2 and 93.0 respectively. These results show that the approach applied in this study was able to detect the unknown malware processes with high precision. Table 5.11 shows the AUC performance.

Table 5.11: AUC results

Classifier	AUC	Level
Random Forest	0.964	Perfect prediction
MLP	0.958	Perfect prediction
KNN	0.962	Perfect prediction
J48	0.930	Perfect prediction

Table 5.11 illustrates that the random forest and KNN classifiers provided the best AUC value, with over 0.96. This signifies perfect prediction. The MLP classifier was next, with 0.958, denoting perfect prediction as well. Finally, the J48 classifier attained 0.930, which also signifies a perfect prediction. Overall, the ROC curve and the AUC values confirmed that the most recent malware experiments had provided compelling accurate results in the malware applications detection.

c) Threshold

Optimal threshold is defined as the value that best separates the two detection distributions that are relative to the malware and benign applications. The threshold value is used to determine whether the presence of behavior pattern indicator is malware (1) or benign (0). The threshold values for random forest, MLP, KNN and J48 are given in Figure 5.13. As the threshold values were obtained based on the real behavior patterns of the malware and benign applications, it can be said that the approach used in this study was able to detect malware with more than 90 percent accuracy rate.

Figure 5.13 also shows that the KNN classifier has an optimal threshold of 0.526 carrying an accuracy of 0.921. This is the point where the malware is finally detected. In other words, a threshold value of between 0 to1.0 needs to be seen in the system in order for the malicious behaviors to be identified.

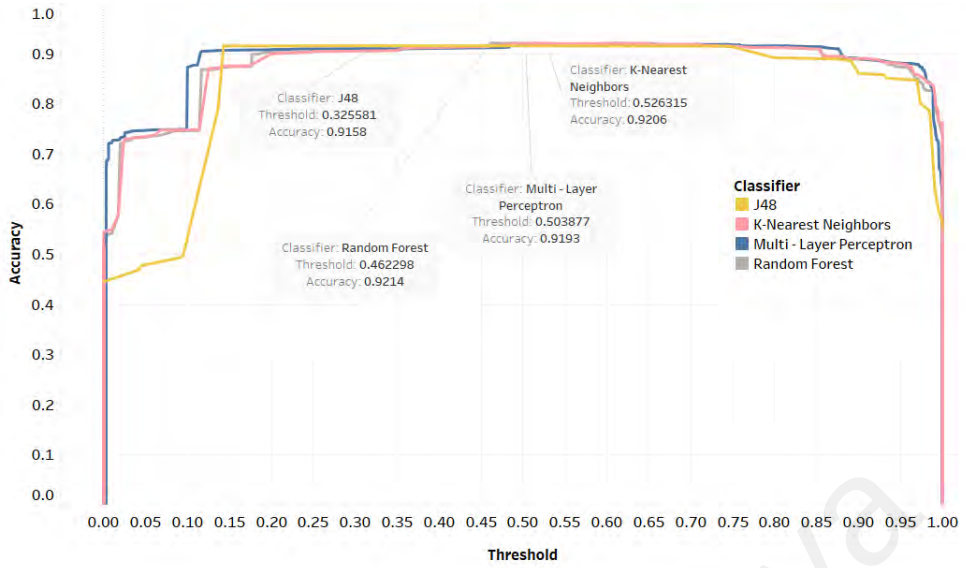

Figure 5.13: Classification threshold

Additionally, random forest also showed a similar accuracy as the KNN classifier. However, it has an optimal threshold value of 0.462. Thus, among other classifiers, it can be said that random forest and KNN both have high detection accuracy for malware detection. Therefore, it is deduced that different types of classifiers have different optimal thresholds in detecting malware. Table 5.12 shows the result of optimal threshold for classifiers.

Table 5.12: Optimal threshold

Classifier	Accuracy	Threshold
Random Forest	0.921	0.462
MLP	0.919	0.504
KNN	0.921	0.526
J48	0.916	0.326

d) Robustness

Besides evaluating the effectiveness of the approach, the robustness of the approach for producing more reliable results was also tested. Robustness is a method that characterises the effectiveness of the classifier while it is being tested on the new independent dataset. In other words, the robust performance of the classifier does not deteriorate too much when training and testing the dataset. In addition, this method shows

the stability of the approach used; it also demonstrates the good performance of the machine learning classifiers. Table 5.13 shows the result of the classifiers' performance.

Table 5.13: Performance result

Classifier	Accuracy (%)	FPR	Precision	Recall	F-Measure	ROC
Random Forest	91.8	8.1	91.9	91.8	91.8	96.6
MLP	90.9	9.0	91.1	91.0	90.9	95.9
KNN	91.6	8.3	91.7	91.6	91.6	96.4
J48	90.8	9.2	91.0	90.8	90.7	91.5

The results noted above shows that the approach applied in this study was able to detect unknown malware with over 90 percent accuracy rate. The comparison of results between Table 5.8 and Table 5.13 indicates that the approach has significantly more robustness by losing just 0.4 percent and 0.2 percent in accuracy for the random forest and KNN classifiers respectively with both drops being very low.

Further to that, it can be noted that the time taken to process the dataset when building the model was less than the time shown in Table 5.9. This implies that the KNN has the lowest model complexity since it uses minimal time to build the model. The robustness of the approach, based on the time taken to produce the model, is presented in Table 5.14 below. Consequently, it is deduced that to achieve acceptable accuracy and effectiveness in classifying unknown malware, robustness is important as it helps to determine the performance of the classifiers.

Table 5.14: Time taken to produce model (seconds)

Classifier	Build model	Test model
Random Forest	2.15	0.18
MLP	1.78	0.01
KNN	0	2.1
J48	0.18	0.01

5.3.4 Discussion

Permission in the dataset were analyzed from the standpoint of the current status or importance of the permissions. As mentioned earlier, permission requested has the capability to control access to the system's resources. Each application developer is required to declare the permission it needs. This permission is then notified to the user during the installation. As a result of the different types of permissions that were available and due to the large number of permissions being asked, the Android system becomes very complicated.

Consequently, this offers attackers more access into users' personal and sensitive resources which are available on mobile devices. Therefore, when an application notifies the permission request, there is no guarantee that all the permissions are needed in order to access the system's resources. A malicious application can request for more permissions than it actually needs. This permission is simply declared in the AndroidManifest.xml file where it provides essential information about the application to run on the mobile devices. This situation causes the malicious application to gain access to all the sensitive resources that are available on the mobile devices, inclusive of contact numbers, emails, photos, messages and passwords. When used irresponsibly, this lead to big personal and monetary losses to the mobile device user.

Current dataset indicate that malicious applications had used dangerous permissions more than clean applications did. However, clean applications like Facebook stimulated extra permissions such as com.facebook.home.permission.WRITE_BADGES and com.htc.launcher.permission.READ_SETTINGS which initiate their activities on mobile devices. Unlike the clean applications, each malicious application tends to ask for dangerous permissions more times. These requests for permissions enable the attackers to succeed in hijacking the sensitive information contained in the mobile devices.

In view of this threat posed by the malicious applications, it is suggested that mobile device users learn about such application risks so that they clearly understand the motive behind these requested permissions. This awareness is able to hinder the malicious applications from accessing users' sensitive information. In the context of this study, the current dataset was analysed. Findings indicate that the malicious applications had used the INTERNET as a medium to collect and transfer the sensitive information to attackers who camouflaged their malicious activities by confusing the mobile device users through fake installers. This made it easy for the attackers to spread the dangerous threats.

5.3.5 Conclusion

This study has highlighted the use of a machine learning approach which can effectively detect and analyse Android malware based only on requested permissions. The results obtained also offer a better understanding of the information derived from examining Android permissions. The crucial aspect of the Android malware detection was described in detail and the methodology was also detailed accordingly to demonstrate how the experiment processes were conducted according to phases. This was then followed by an exploration of the machine learning approach which was used for training and testing the dataset as well as for predicting and distinguishing the Android applications as malware or benign. In this regard, four (4) machine learning classifiers were implemented. Finally, the results obtained were discussed.

From the results obtained, it seen that Android security relies on permissions for controlling the applications' access to the software components as well as the mobile hardware. Android security has become a major challenge in current times where fine-grained permission control is necessary for Android applications. While Google implemented the Bouncer to analyse the submitted applications on Google Play store, this study discovered that existing permissions which were the focus of this study, had forced

the mobile users to either accept all the required permissions or to terminate the installation process. Consequently, most of the users who had proceeded with the installations ignored the warning permissions thereby, becoming targets for the malicious applications. Although, significant research work has been carried out to investigate the permission model of the Android malware detection, the privacy leak continues to happen.

To address this problem, an approach was proposed for the detection of malware. This was accomplished by analysing the permissions of Android applications. The aim was to evaluate the effectiveness of Android permissions by identifying the malicious applications through the use of a machine learning approach. In order to identify the relevant permission of the applications, the apk file was extracted and the required permissions were collected on AndroidManifest.xml file. Among the permissions required, it was noted that any application that used more permissions for its functionality could lead to malicious attacks. In addition, it was found that communication types such as READ_SMS, WRITE_SMS, SEND_SMS and RECEIVE_SMS were used malicious applications. These types of permission allowed the malicious applications to make premium message without the mobile user's knowledge and subsequently, caused money losses.

Applications with the permission READ_CONTACTS and GET_ACCOUNT were functional for accessing the list of accounts in the Accounts Service and for reading the mobile user's contact data. This occurrence highlighted the potential of malware threats to Android devices. The experiments conducted with the Android permissions showed that the approach used in this study had achieved a high detection rate and a low false positive rate. In addition, this study had also applied supervised learning using four classifiers: random forest, KNN, MLP and J48, on a collection of 5551 of malware and

7000 benign applications. Both were tested and validated. Thus it can be concluded that the approach with an accuracy of 92.0 percent and a prediction accuracy of around 92.0 percent is effective in detecting Android malware These outcomes suggest that the approach is capable of detecting almost all the malware applications. In addition, the AUC curve noted to be between 93 percent and 96 percent also implies that the approach carried an outstanding property that is reminiscent of Fawcett's (2006) work.

Nonetheless, some limitations exist because the malware detection approach only considered the permissions as features. The first limitation is that benign applications have several permissions which were likely to be considered as malware since they seemed to access several resources such as accounts, contact numbers, passwords, emails, and bookmarks. The second limitation is that the benign applications had also requested for the same permissions as malware applications did. In this regard, the detection is likely to be less precise.

5.4 Experiment III: Evaluation of time series detection

This section discusses the time series classification used to detect malware. A time series is a continuous sequence of discrete time spaced at time intervals (Patri et al., 2017; Tanaka et al., 2016). The aim in using the time series classification is twofold. First, it offers an understanding of the underlying structure that was produced by the observed data. Second, it is used to fit a model and to predict Android malware. In addition, classification algorithms, based on the time series, are more accurate and significantly faster than the state-of-the-art classifiers (Ye et al., 2011). The time series detection was applied to predict future unknown malware based on previously observed data.

5.4.1 Experiment setup and procedure description

The time series detection involves a few steps. The first step was to create the time series dataset. As mentioned above, the malware and benign dataset were collected from

the AndroZoo dataset. Each of them was then split according to the time series representation (i.e. 2010 - 2016). The dataset was then labelled as malware or benign applications. The second step was to create a time series model. Each model is used to test the future unknown malware dataset by using previous dataset. For instance, the dataset from the year 2010 were used to predict malware for 2011. Following this, dataset from 2010 and 2011 were combined so as to predict the future malware for 2012. The same preprocessing steps were conducted on the different time series dataset. The results were then collected. Finally, the machine learning technique was used to train and test the model as a means of predicting future unknown malware.

5.4.2 Data collection phase

This section describes the process of gathering and measuring the related information. Specifically, this study extracts the permission data from Android applications. The data collection phase is crucial for maintaining result accuracy. The real world applications were used to ensure data integrity. Typically, in the Android malware detection study, two (2) types of dataset were involved: benign (a normal application) and malware. Table 5.15 shows the categories of sample applications.

Table 5.15: Categories of application

Art Design	Dating	Food & Drink
Auto & Vehicle	Communication	Health & Fitness
Beauty	Education	House & Home
Book & Reference	Entertainment	Library & Demo
Business	Events	Map & Navigation
Comics	Finance	Medical
Music & Audio	Photography	Sports
News & Magazines	Productivity	Tools
Parenting	Shopping	Travel & Local
Personalization	Social	Video Player & Editors

Primarily, the dataset is a collection of related data that were used to initiate the experiment in the initial phase. It consists of all the information required for research

activities. The features used are similar to those noted in Section 5.3. Table 5.16 lists the summary of the dataset.

Table 5.16: Dataset summary

Dataset	Source	Number used in experiments
Malware	Androzoo	6942
Benign	Androzoo	7000
Total		13942

5.4.3 Evaluation and results

This section discusses the problem of malware detection for the time series classification process. This component is important because it is the basis for understanding how the performance of the approach relates to the nature of the unknown malware, also known as malware attack, in terms of time series. New malware applications taken from the AndroZoo dataset were used to test time series. This is based on the random forest model since the model shows high detection results. Findings indicate that the malware applications showed a positive ratio ranging from 1 to 50 on the VirusTotal evaluation. In this context, time series generated from the year 2010 until 2016 were tested using random forest classifier. Results are presented in Table 5.17.

Here, the best average prediction result achieved was produced when using training sets from the year 2010 until 2012. The trained model was used to perform prediction on the dataset for year 2013 until 2016 and it achieved the highest average accuracy rate of 90.70 percent. This shows that the model was able to perform prediction on future malware effectively. The assumption behind this result is that the unknown malware time series were different from the training sets. This can be attributed to the different time series that was generated; it showed different results in terms of conditions and patterns although the dataset may be almost similar. Consequently, irregular patterns emerged and digressed from achieving good results.

To generate an exact pattern within the time series, this study implemented a dataset that are from various time series. The performance of the approach used in this study is highly dependent on the time series dataset which were not easy to select. Nonetheless, the results shown in Table 5.17 reveal interesting insights about the performance of the malware detection approach on time series. Based on this, it is deduced that the results generated from this study imply that the approach used is superior in predicting future Android malware.

Table 5.17: Time series detection

Training	Testing	Accuracy (%)
2010	2011	62.85
	2012	87.23
	2013	88.25
	2014	78.39
	2015	83.60
	2016	85.89
	Average	**81.04**
2010-2011	2012	92.06
	2013	93.71
	2014	83.64
	2015	86.18
	2016	86.39
	Average	**88.39**
2010-2012	2013	94.84
	2014	86.99
	2015	89.56
	2016	91.42
	Average	**90.70**
2010-2013	2014	91.61
	2015	89.17
	2016	90.51
	Average	**90.43**
2010-2014	2015	89.74
	2016	90.36
	Average	**90.05**
2010-2015	2016	89.74
	Average	**89.74**

5.4.4 Discussion

This section briefly presents the synthesis results of the findings derived from predicting unknown malware using time series detection. The experiment results presented in the previous section are encouraging as the classifier (random forest) had successfully predicted the unknown malware. The accuracy result obtained from the random forest classifier using sample dataset proves that the results were very promising. Overall, the experiment results accumulated from using the training model between 2010 – 2012 had shown the highest accuracy (90.70%). This outcome demonstrates the efficacy of the machine learning-based classification method on time series as representations of the Android permissions. The reason is because random forest is mostly dependent on the sample dataset; this means that the training dataset of 2010 – 2012 had worked perfectly well with the random forest classifier.

5.4.5 Conclusion

Determining the best similarity measure for the different types of time series dataset was not easy. A performance measure of detection algorithms such as the random forest was observed to be performing well as it was highly sensitive to malware patterns. Thus, the approach used in this study is unsuitable when the time series are of different years. Finally, the disadvantage of the time series approach is its lack of ability to detect applications with no permissions since it had used permissions as features.

5.5 Experiment IV: Evaluation of application risk

This section introduces the EZADroid, which was used to evaluate Android applications which used less features or carried very minimal features. This study zoned the applications into several categories (i.e. high, medium, low and very low). This "zone" approach helps to inform users about the specific risks of Android applications, as seen in their criticality. Specifically, this study applies risk assessment and the AHP approach

to allow mobile device users to identify the risk zones noted in Android applications. This approach is better in helping users to make decisions and in using the appropriate method to identify which incidents are important and which are trivial.

The AHP was used to calculate the risk of the applications. The AHP is a structured technique for the multi-criteria decision-making approach that was developed by Saaty (Lo et al., 2012). With the ability for multicriteria decision making, AHP has used in many studies. It is broadly applicable because each application is well structured and effective in making a decision. For example, in fields of operation studies such as (Nikou et al., 2013) and (Nikou et al., 2011), they applied AHP to investigate the most applicable mobile service for the consumer. AHP method is used to identify the linkage between perceived performance benefit with and good practice in Small and Medium Enterprise (SME) (Thanki et al., 2016) but (Khalil et al., 2016) applied risk analysis to rating the building based on excellent, good, medium, low and poor as well as to lessen users' safety.

Within computer security studies, (Dini et al., 2018) and (Dini et al., 2012) adopted AHP to evaluate the trustworthiness of Android applications. Moreover, the methodology has proven to be very suitable for decision making and capable producing results that agree with expectations (Olson, 1996). According to a literature review by (Cegan et al., 2017) shows that AHP most frequently used in multicriteria decision making and has been thoroughly tested by thousands of organization around the world for the last 35 years (Opydo, 2013). The combination of AHP and risk assessment gives an advantage for decision making to assess quantitative of risk. In addition, a survey by (Gritzalis et al., 2018) shows that AHP is a popular method to assess and manage information security risk.

Figure 5.14 illustrates the main components of the EZADroid. The framework is categorised into three components: a) response option, b) response systems and c) risk zone. The proposed framework attempts to identify the risk of Android applications whether it malware or benign by assessing the risk zone. It is very important to choose appropriate approach, especially when dealing with the technical aspects. With the aid of the intrusion detection system (IDS), risk assessment and machine learning approach, EZADroid supports the procedures in selecting relevant features and response to the user with the risk zone.

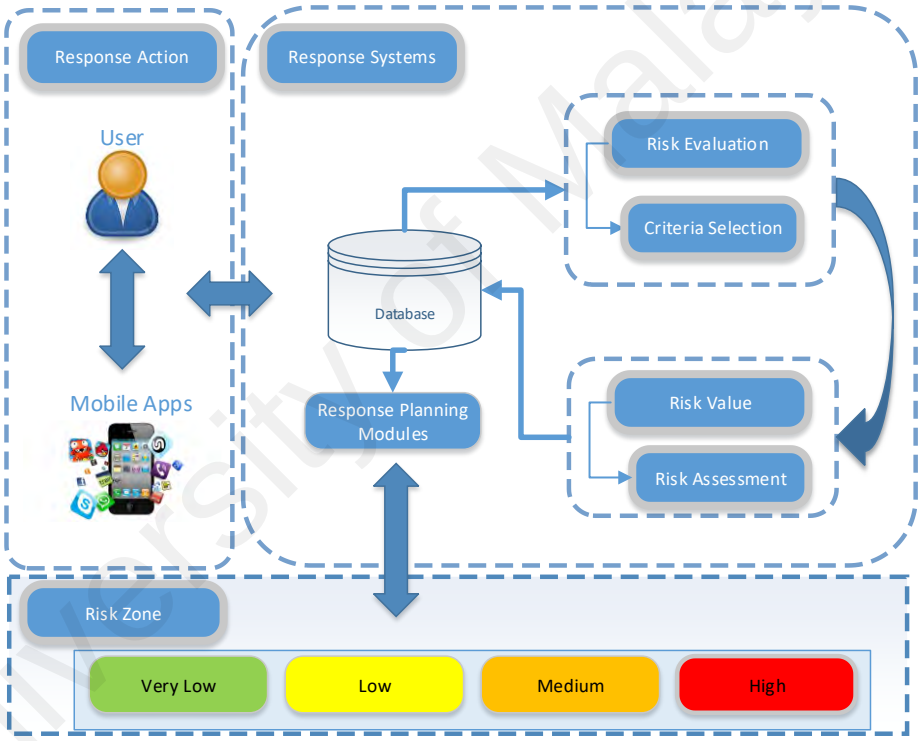

Figure 5.14: EZADroid framework

The response option component defines the boundaries of the response action which notifies the risk zone to users. In this component, the user is able to evaluate the risk on the Android applications and then respond to the risk either by accepting or removing the risk. The response system then conducts an analysis of the Android applications. This analysis consists of risk evaluation, risk value and the response planning module. Here, the first risk evaluation is associated with the selection of the criteria which consists of

the grouping permissions. The second is with respect to the risk value. In this regard, the criteria are evaluated using risk assessment. The risk value is stored in the database before it is submitted to the risk zone. This is achieved by using the response planning module. The risk zone categories have four (4) types of indicators: very low, low, medium and high risk. If the risk is greater than a certain threshold, the warning notification appears on the mobile device to notify the risk zone to the user. Each of the risk zones is presented in different colors to indicate the different levels of risk and to improve user awareness.

5.5.1 Experiment setup and procedure description

This section presents the overall workflow of the experiments. The risk assessment approach is used to improve the effectiveness of the risk evaluation by generating a risk zone for the user. It serves as a warning against malicious applications (e.g. very low, low, medium, and high). In recent works published in (Dini et al., 2018) is similar to our proposed approach, conduct a risk analysis method by considering permission as likelihood threat (Dini et al., 2018). They proposed user's rating developer's reputation and a number of application download as criteria for risk analysis. However, these criteria are less effective and untrusted. This is because user's rating is inconsistency (Sharma et al., 2013) and malware application also stored in Google Play Store (Liam Tung, 2017). This malware application has been downloaded up to 4.2 million (Liam Tung, 2017). These led to false sense accuracy. Though, our proposed approach submits Android applications to VirusTotal to validate and ensure the trustworthiness of dataset. In addition, the proposed approach utilized the relevant features as criteria in multicriteria decision making by implementing machine learning approach to significantly increase detection accuracy for risk analysis and malware detection. The proposed approach achieved higher accuracy with 89.82%, while (Dini et al., 2018) achieved 77.37% only. The results also validated using statistical analysis.

a) Risk assessment criteria

Android applications require permission granting from the mobile users in order to invoke the Android API successfully. The declared permission in AndroidManifest.xml file is important and effective for revealing potential risks; it is also useful as a warning message to notify users. Most of the risky applications require a combination of some permissions in order for attackers to launch the attack. Figure 5.15 illustrates the percentage of the top ten (10) most requested permissions by malware and benign applications.

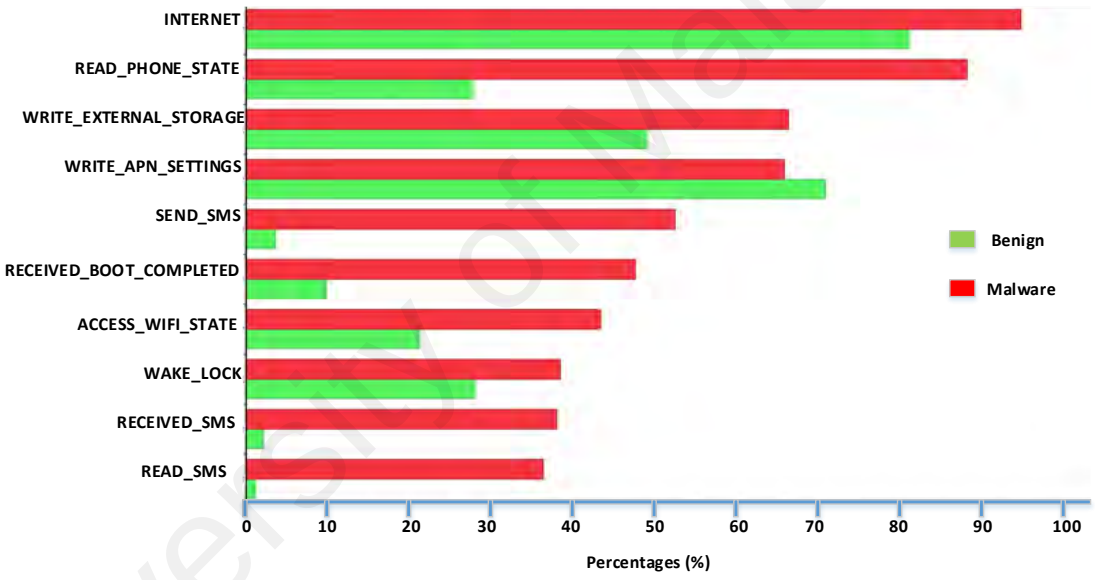

Figure 5.15: Percentage of the top 10 requested permission by malware applications

The above figure shows that the INTERNET is the most commonly used permission, both by the malware (94.06%) and benign (80.45%) applications. This is because the INTERNET permission is mandatory for applications to access the Internet especially for an application update. The permission access to READ_PHONE_STATE is also highly requested by the malware. This table shows an outcome that is similar to the results of Zhou and Jiang (2012) in detecting malware by permission, as the first research done on the Android malware family. Moreover, the malware noted in this study had also

requested more permissions on communication such as SEND_SMS, RECEIVED_SMS, and READ_SMS. This implies that the malware applications were more interested in acquiring dangerous permissions that offer access to sensitive information. It appears to be using the Internet as a medium.

Based on the observatory analysis, the related permissions were grouped. The individual permission joins the group through the same attributes of the permission (Developer, 2016a). The grouping process includes reading all the permissions in AndroidManifest.xml and extracting the permissions through grouping. This allows the number of comparisons to be reduced. A member of the group is then presented together as a criteria with each criterion used to calculate the risk of the Android applications.

b) Criteria selection

This study had used 10000 Android applications samples as the training set. The total number of benign and malware samples were thus 5000 each. The samples were manually predefined with their appropriate labels as benign or malware. However, it is important to note that we validated the labelling process by checking the Android application's status from VirusTotal. In other words, we labelled a sample as malware after running it through VirusTotal which is an online website that checks for viruses through the URL or through an uploaded file (Quintaro, 2017).

VirusTotal is highly reliable as it inspects the sample and aggregates the result of over 70 antivirus scanners. VirusTotal widely used by researchers (Huang et al., 2014; Boukhtouta et al., 2015) to provide the ground truth in their works. Following the use of the VirusTotal, the dataset samples were used for criteria selection.

The criteria selection phase makes use of a specific metric which computes and returns a score for each feature individually (Asaf Shabtai et al., 2012). Here, the WEKA

approach was implemented. It includes information gain which was used to select the best criteria. In this study, WEKA was applied as the machine learning platform. WEKA is a well-established software that has a collection of machine learning algorithms (Waikato, 2017). It is a well-rounded and complete software suite that fits the objective of the current study. information gain was implemented directly into WEKA to save time from manually coding the algorithms. The reliability and accuracy of WEKA's algorithms are also well recognized (Kaur et al., 2015; Deepa et al., 2015).

The criteria with a high value of information gain is selected. In this way, 10, 20 and 30 criteria were selected based on information gain values ranging from 0.033 to 1.0. The best criteria helped to improve the performance measure (Kumar et al., 2014). Table 5.18 shows a list of the criteria recorded when information gain was implemented. These were then stored in the database for the risk assessment process.

From the results presented in the table below, it appears that phone calls and messages were the top in the permission-based system. This represents the types of features noted on the Android developer (Developer, 2016a). It is necessary to assert that this group is significant in the risk assessment approach.

Table 5.18: List of criteria

Information Gain Value	Criteria	Permission Group
0.2776	android.permission.READ_PHONE_STATE	Phone Calls
0.2286	android.permission.SEND_SMS	Messages
0.1908	android.permission.READ_SMS	Messages
0.1739	android.permission.RECEIVE_SMS	Messages
0.1285	android.permission.RECEIVE_BOOT_COMPLETED	Application Information
0.1067	android.permission.WRITE_SMS	Messages
0.1016	com.android.launcher.permission.INSTALL_SHORTCUT	Properties
0.078	android.permission.INSTALL_PACKAGES	Application Information
0.0686	com.android.launcher.permission.UNINSTALL_SHORTCUT	Properties
0.0685	com.android.browser.permission.WRITE_HISTORY_BOOKMARKS	Personal Information
0.067	com.android.browser.permission.READ_HISTORY_BOOKMARKS	Personal Information
0.0631	com.lge.launcher.permission.READ_SETTINGS	Dev_Read_Setting
0.0631	com.motorola.launcher.permission.READ_SETTINGS	Dev_Read_Setting
0.0624	com.motorola.dlauncher.permission.READ_SETTINGS	Dev_Read_Setting
0.06	com.htc.launcher.permission.READ_SETTINGS	Dev_Read_Setting
0.0586	com.fede.launcher.permission.READ_SETTINGS	Dev_Read_Setting
0.0581	com.motorola.launcher.permission.INSTALL_SHORTCUT	Dev_Install
0.0579	com.android.launcher.permission.READ_SETTINGS	Properties
0.0578	com.lge.launcher.permission.INSTALL_SHORTCUT	Dev_Install
0.0575	com.motorola.dlauncher.permission.INSTALL_SHORTCUT	Dev_Install
0.056	org.adw.launcher.permission.READ_SETTINGS	Dev_Read_Setting
0.0526	android.permission.WRITE_APN_SETTINGS	Properties
0.0508	android.permission.RESTART_PACKAGES	Application Information
0.0484	android.permission.CHANGE_WIFI_STATE	Network Communication
0.0434	com.google.android.providers.gsf.permission.READ_GSERVICES	Dev_Service
0.0411	android.permission.ACCESS_NETWORK_STATE	Network Communication
0.0403	com.google.android.c2dm.permission.RECEIVE	Network Communication
0.0401	android.permission.GET_ACCOUNTS	Account
0.0359	com.android.vending.BILLING	Account
0.033	android.permission.READ_CONTACTS	Personal Information

Table 5.19 illustrates the judgment matrix for the decision factor and the indicator used in the risk zone threshold. The results of risk assessment are demonstrated in the evaluation and result section.

Table 5.19: Judgment matrix criteria

	Messages	Personal Information	Application Information	Properties	Phone Calls	Normalized
Messages	1	5	5	5	5	0.54
Personal Information	0.2	1	0.5	0.5	0.5	0.07
Application Information	0.2	2	1	2	2	0.16
Properties	0.2	2	0.5	1	2	0.12
Phone Calls	0.2	2	0.5	0.5	1	0.09

Consistency ratio = 0.043

c) Risk zone threshold

To establish a methodological approach for identifying the risk zone in the Android applications, the risk value and zoning process are explained. Four types of risk zones were applied: very low, low, medium and high. These risk zones have been used in security investigations for the purpose of evaluating the risk impact. The risk zone has been exemplified in the works undertaken by Theoharidou et al (2012) and Anuar et al. (2013b). They used the method for the Android platform and for incident prioritisation. The risk zones of the current study are presented in different colors depending on the levels of risk. The purpose is to increase awareness among Android users. This application of colors have also been applied by previous researchers (Theoharidou et al., 2012; Anuar et al., 2013b). Table 5.20 illustrates the description of the risk zones.

Table 5.20: Description of risk zone

Color	Risk Zone	Description
Red	High	High to critical risk. This application is able to harm the user.
Orange	Medium	Moderate risk. This application is capable to harm the user and should be put under observation.
Yellow	Low	Slight risk. This application is safe but could be misused if it has malicious activities or potential threats.
Green	Very Low	Very low to no risk. This application is safe for use.

As the table illustrates, the risk zones determine the levels of severity of the applications, thereby raising users' awareness of the severity of the risks involved. Very low and low-risk zone means that the risk of application is acceptable and safe to use. Figure 5.16 illustrates the threshold for the risk zone.

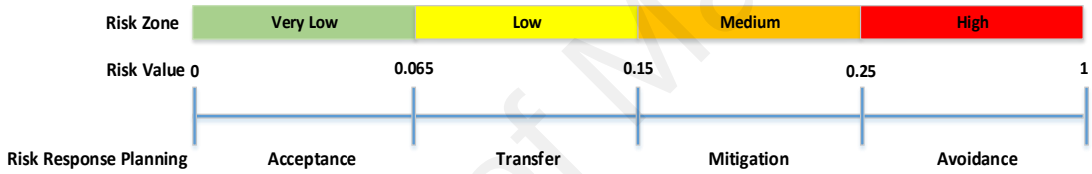

Figure 5.16: Risk zone threshold

This threshold is not a definitive value. It is subject to other reassessments thus different scales will produce a different distribution. The selection of this threshold is important in making a suitable and significant decision of the risk zones. The risk zone threshold is adapted from Table 5.21 which is the distribution analysis drawn from the box plot of ten (10) permissions. Furthermore, this threshold indicates the significance of the mapping process between the risk zone and the risk response planning for future works. Table 5.21lists the data analysis taken from the box plot Figure 5.7 which uses ten (10) permissions.

Table 5.21: Data analysis for 10 permission

	Malware	Benign
Min	0	0
1st Quartile	0.2306	0
Median	0.3738	0
Mean	0.3989	0.06447
3rd Quartile	0.5314	0.09468
Maximum	0.9176	0.79265

5.5.2 Data collection phase

This section describes the results of the experiment. This study evaluated the risk of 5,000 benign and 5,000 malware applications. Each application permission was extracted so as to collect their permissions from AndroidManifest.xml. Each of the permissions on the application was stored in the database as a criterion collection. This criterion is important in guaranteeing the accuracy of the proposed approach. The permission-based approach was used to identify the risk zone. After the EZADroid accepts the application's permission-based behavioral data, it computes the risk value and then determines the risk zone to see whether the application is very low, low, medium or high risk.

5.5.3 Evaluation and results

This section presents the experimental results and performance evaluation of EZADroid in generating the risk zones. The assessment consists of two (2) experiments. The first applies the risk assessment approach to evaluate the risk value. The second generates a risk zone. The experiment uses the box plot analysis to show the difference between the malware and the benign applications. The box plot analysis was able to discover the risk potential and to predict the risky malware. It plays an important role in determining how relevant the EZADroid. The risk zone was determined after the results were gathered.

a) Risk assessment evaluation

This section presents the risk assessment evaluation which uses the AHP approach. Table 5.22 shows the results obtained. It classifies the benign and malware applications accordingly. It appears that all the malware and benign applications had obtained the risk values from the AHP analysis which then classifies the risk zones accordingly. In order to validate the experiment, VirusTotal (Total, 2016) was used to strengthen the risk zone evaluation. Table 5.22 illustrates samples of the evaluation.

Table 5.22: Samples evaluation and risk zone on applications

Sha256	Virus Total	Family	Risk Value	Risk Zone	Class
78c1c57100bc14f9689c3f670d48405d9eb7487df1a34a846296f8dd4ab34e33	36	Plankton	0.760934	High	Malware
1ee4f5a8812ba86eda9f12f1e76a1a44e4d318bf4799eb7fd22c3c6a8a0f8ad2	34	Plankton	0.696939	High	Malware
3dc3631adbae697a10edfea65c06ebda751741302b6ba95c4f6c6031db71ce74	41	ExploitLinuxLotoor	0.656039	High	Malware
4419c922b2926246ffb5c4d427920b8785b7853f1f2c400914531ce99ad6164e	38	ExploitLinuxLotoor	0.656039	High	Malware
97d9acf46ba6e3bd759f74d0f2f312165f143a815ff41d26a212f3f99b20b8c6	35	FakeRun	0.559957	High	Malware
9e9fef1079a8d20a3074a1be16be029333d863add15dd0a44d67ab685bee7ea4	34	FakeRun	0.559957	High	Malware
06b53d3ea2aeee828123194b4cea8135f5b868296d8d7ab3cb839e34b2f04d6a	39	Adrd	0.557189	High	Malware
294cfb2bc890b65d7bc9135225369ab9bbd0ca81baa109f829e2c22478b4db2f	37	Adrd	0.557189	High	Malware
08ad6b366abf609018b1866f609d132ecdb66981aae540d3316c7584c816b179	38	BaseBridge	0.423385	High	Malware
09ac19bce6a6c98948ceb7db6398c0cddf2cb9167d547597731bc44411371478	38	BaseBridge	0.423385	High	Malware
00f24c9904ce23bae5a3cc4ca5a1bd13ed811b57ca772032530d415ccda02f04	0	Google Play	0.197446	Medium	Benign
00f28a5c4851f2702fc61753c21867c68916d00536d59b7c4e1d2bbbe8c7ca00	0	Google Play	0.031939	Very Low	Benign
00f2915b170f755efa3409c3ccd12fe5b1edd905592aad75d957295a1a616650	0	Google Play	0.115419	Low	Benign
00f29243375e2151947287b52f81a73a46a9e21b50a82e7cd7e3b8a8d6e6cafc	0	Google Play	0.197446	Medium	Benign
00f29ffd36c87e9138c65e09e5b455b4dbca29cbb5f37fc9bdd01c9ea73fd9a6	0	Google Play	0.031939	Very Low	Benign

Table 5.22 is classified into six (6) columns: SHA256, Virus Total, family, risk value, risk zone and class. The experiments applied in this study used a generated SHA256 hash to provide a unique key for each sample application.

i. The columns named *VirusTotal and class* presents the distinction between the malware and benign applications taken from the sample dataset.
ii. The *family* column shows the particular malware family of the sample application that was discovered during the experiment.
iii. The *risk value* column shows the weight of the risk that was measured by the risk assessment. In addition, the *risk zone* displays the level of risk generated from the risk value.

The proposed approach provides a solution to give rich insights into the application risk analysis by using a set of criteria that are combined with a multi-criteria decision approach. It automatically performs the permission analysis of applications. The level of risk provides insight information related to the risk of application. It also directly shows the user the risk zone. In fact, it improves the application security and the user awareness. Moreover, the ability to distinct permissions for malware detection provides an additional protection for the user. The results reveal that the proposed approach reduces a significant potential for several malicious applications from accessing mobile devices. It analyzes application permission in order to access the signature of malicious applications. The results of the analyses demonstrate that permissions to request message are requested by a high-risk application as well as the permissions, related to the application information (i.e. RECEIVE_BOOT_COMPLETE, INSTALL_PACKAGEs, and RESTART_PACKAGES) demonstrated risky permission.

Table 5.23 also indicates that applications infected with malware would pose as threats to the mobile device users. The high-risk application came from the Plankton malware

family. Once the Plankton malware is installed on mobile devices, it collects the device's ID and user information before sending it to a remote server (Idrees et al., 2017). The remote server then pushes payload dynamic onto the user's mobile device to exploit the root. The approach proposed in this study is effective in detecting the Plankton malware and in identifying the risk it poses. This is because the Plankton malware uses permissions such as messages, properties, phone calls and personal information. This was discovered by Zhou et al. (2012) and was consequently removed from the Android market by Google (Vanja Svajcer, 2011). Table 5.23 lists the malware families with their risk values. The full list of malware family and risk value is available in Appendix B.

Table 5.23: List of malware family and risk value

No.	Family	Total	Min Value	Max Value	No.	Family	Total	Min Value	Max Value
1	AccuTrack	9	0.018404	0.018404	9	Loicdos	1	0.063877	0.063877
2	Adrd	78	0.219138	0.557189	10	Loozfon	2	0.139663	0.139663
3	Adsms	3	0.394271	0.412675	11	Luckycat	5	0.101884	0.101884
4	Aks	5	0.08348	0.08348	12	Lypro	1	0.182683	0.182683
5	Ansca	1	0.174222	0.174222	13	Maistealer	1	0.056183	0.056183
6	Antares	2	0.180063	0.192626	14	Mania	6	0.188524	0.401693
7	Anti	2	0.210727	0.210727	15	Maxit	1	0.461164	0.461164
8	Anudow	1	0.341261	0.341261	17	MMarketPay	1	0.380736	0.380736

The extensive results displayed in Table 5.23 indicate the various Android malware families that had been presented in the sample dataset used in this study together with their risk values. Four (4) major malware families noted in the malware sample dataset were FakeInstaller, Plankton, DroidKungfu, and Opfake. Each of these malware families has a different risk value because each used different permissions. Table 5.23 also shows that the same malware family is unable to show the same risk value. Based on this, it is

deduced that the applications in the same malware family used different permissions. As a result, they showed different types of risk zones. To further discuss the risk zones, the following section discusses the rating threshold that was proposed in the RSM (Anuar et al., 2013a). This is applied to rate the risk zones.

b) Box plot analysis

In order to avoid any bias, an arbitrary number of criteria selection was used. This study used the criteria selection approach with three different configurations: 10, 20 and 30 criteria, as a measure to select the highest out of the 378 criteria featured by the feature selection algorithms (e.g. Information Gain). The box plot analysis shown in Figure 5.19 is related to the risk value for the malware and benign applications. The box plot analysis is also able to identify the criteria more effectively as it differentiates the benign from the malware applications. This difference will suggest that the two populations belong to different distributions. Figure 5.17, Figure 5.18 and Figure 5.19 illustrate the malware and benign applications. The trend illustrates that permission-based criteria are significant and relevant for conducting risk assessment.

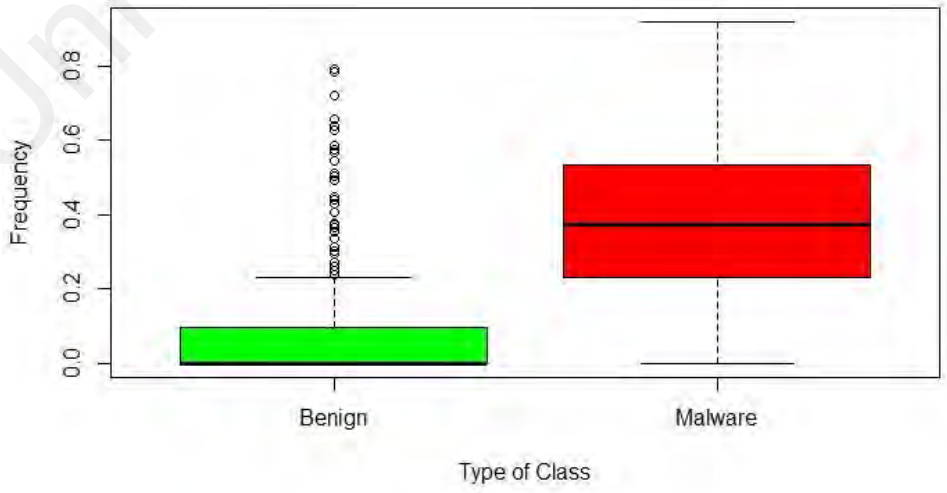

Figure 5.17: The boxplot of 10 permission

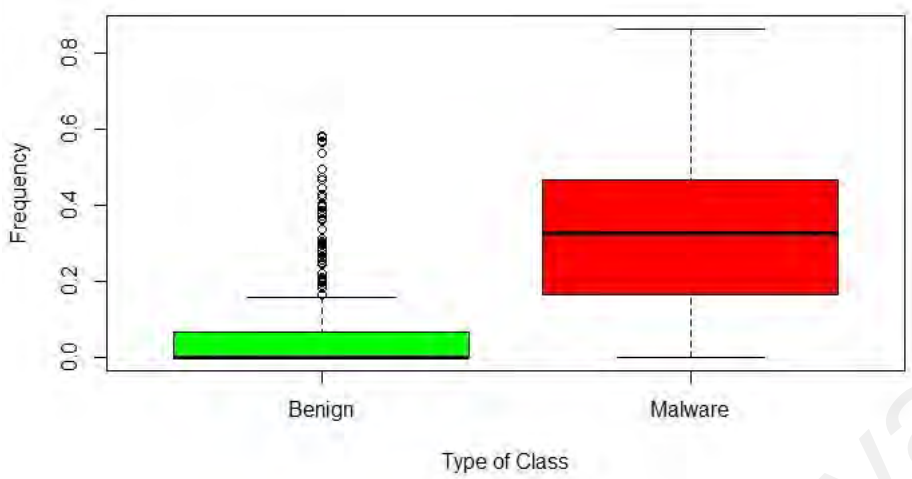

Figure 5.18: The boxplot of 20 permission

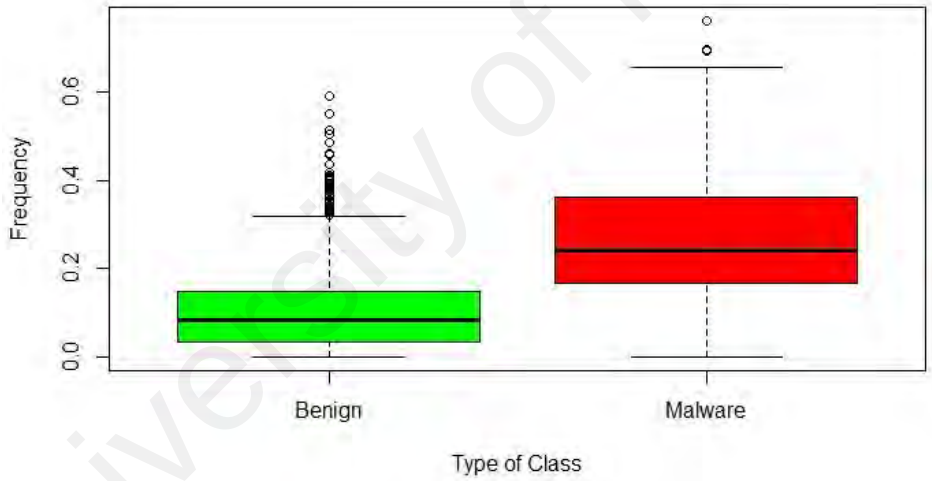

Figure 5.19: The boxplot of 30 permission

The distribution shown in Figure 5.17, Figure 5.18 and Figure 5.19 demonstrates different values. This means that there is a distinction between the benign and malware applications. This evidence strengthens the experiment evaluation with the result showing an accuracy of over 80 percent for the risk zone. It is fascinating to note that the implementation of the EZADroid is able to determine the risk zone based on 10,000 samples.

Table 5.24 shows the distribution of a dataset. The top 50 percent of the malware applications (2500) have high risks. They are represented by everything above the medium risk (the black line). The top whisker shows that 25 percent of the malware applications came from 625 applications. The maximum whisker represents the greatest value in the malware application (risk value). Returning to the aims of this study, it is possible to state that an application is malware or benign if it shows a good overview of the data's distribution. Table 5.24 illustrates the risk evaluation for 10, 20 and 30 criteria.

Table 5.24: Risk evaluation

	10		20		30	
	Malware	Benign	Malware	Benign	Malware	Benign
Very Low	95	2643	127	2928	127	2373
Low	414	1722	814	1679	697	1530
Medium	869	483	985	300	1800	922
High	3622	152	3074	93	2376	175

By indicating the applications as high and medium risk malware or as very low and low risk benign, the EZADroid had achieved an accuracy rate of 89.82 percent for the malware outcomes and 87.30 percent for the benign outcomes. This implies that the EZADroid is able to evaluate the risk of both applications effectively. In that regard, it improves the aim of this study in identifying the risk of applications and for detecting malware. Figure 5.20 shows evaluation of risk zone.

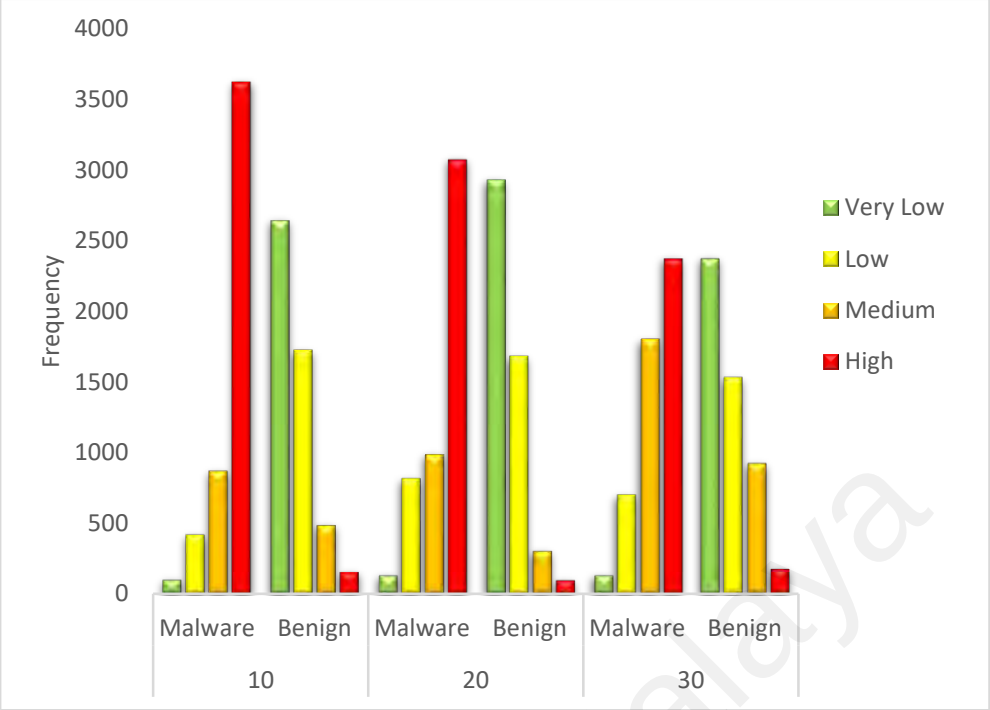

Figure 5.20: Risk zone evaluation in 10, 20 and 30 criteria

Figure 5.20 indicates that the total frequency of the application is measured by presenting the type of risk zones. In the 10 criteria selection, over 80 percent of the malware were detected as high risk. Based on this, it concluded that the proposed approach is efficient in evaluating the risk for most of the sample dataset.

In order to warrant the significance of the proposed approach, a statistical analysis of the 10 criteria was also conducted. The 10 criteria were selected because the results were more reliable, as shown on the box plot analysis. Table 5.25 illustrates the most famous in Android applications on Google Play store.

Table 5.25: Top free in Android applications

Applications	Categories	Risk Zone
Facebook	Social	High
Facebook Messenger	Communication	High
WhatsApp Messenger	Communication	High
WeChat	Communication	High
Instagram	Social	High

Table 5.25 shows that the top Android applications which belong to the social and communication categories have high risk because most of these applications requested dangerous permissions such as READ_SMS, RECEIVE_SMS and INSTALL_SHORTCUT. For instance, Facebook applications allowed Google to display contents from Facebook mobile applications including public profile information (Westenberg, 2015). This information could lead to cyber threat problems.

c) Statistical analysis

This section presents a component of the data analytics. In the context of research, statistical analysis scrutinises data and presents a selection taken from the population. Linear regression was applied to specify the nature of the relation between the malware and benign applications. A total of 10,000 applications taken from the sample dataset were applied. The experiment was able to manage the dependent (risk value) and independent (risk zone) variable score into the same row. Table 5.26 illustrates the variables used for the analysis and the results of the mean and standard deviation.

Table 5.26: Description statistics

	Mean	Std. Deviation	N
Risk Value	0.40	0.199	5000
Risk Zone	3.60	0.720	5000

In order to locate and interpret the relevant regression and correlation coefficients, the experiment needs to consider a variable Entered\Removed, model summary, ANOVA, and coefficient. Table 5.27 illustrates the independent variables. Table 5.28, Table 5.29 and Table 5.30 present the statistics of the data variable score. Table 5.28 demonstrates the correlation coefficient (r) and the coefficient of the determination (r square). It specifies the strength of the linear trend between the variables. Table 5.29 indicates the significant value of the independent-variable scores when compared to a predetermined

α. Finally, Table 5.30 also illustrates the y-intercept and the slope for the regression equation.

Table 5.27: Variables entered\Removed

Model	Variables Entered	Variables Removed	efficient
1	Risk zoneb	.	Enter

The model summary shows the correlation between the two variables (r): correlation coefficient (r) and the coefficient of determination (r square). The value of R represents the correlation coefficient that indicates the relation strength between the independent variable to the dependent variable.

Table 5.28: Model summary

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Durbin-Watson
1	0.750a	0.562	0.562	0.132	0.160

a. Predictors: (Constant), Risk zone;
b. Dependent Variable: Risk value

Table 5.28 indicates the correlation coefficient (r) with the value of 0.750. It suggests that the number of sample applications has a good linear relationship. Due to this result, the coefficient of the determination (r square) shows a different value of 0.562.

Table 5.29 lists the ANOVA statistics which indicate whether the regression equation explains the significant portion (sig.) variability of the dependent variable and the independent variable. It also presents the value of sig. to be 0.0. This indicates that the proposed approach is able to reject a null hypothesis where it demonstrates a significant and fit model. The table shows a p-value of 0.0 percent which obviously indicates that the number of malware changes significantly with respect to the number of malware applications. This change is found from the regression equation, y = -0.349 + (208)x.

Table 5.29: ANOVA

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	111.820	1	111.820	6421.648	0.000b
	Residual	87.030	4998	0.017		
	Total	198.851	4999			

a. Dependent Variable: Risk value

b. Predictors: (Constant), Risk zone

Table 5.30 shows one able dependent-variable score for each independent variable score. The y value is a result obtained from the regression equation which indicates that the pair falls on the regression line while the x value of the risk zone is substituted for the regression equation. This process is able to guess the number of application risk values and the risk zone.

Table 5.30: Coefficients

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Correlations
		B	Std. Error	Beta			Zero-order
1	(Constant)	-0.349	0.010		-36.680	0.000	
	Risk zone	0.208	0.003	0.750	80.135	0.000	0.750

Figure 5.21 illustrates the risk zone analysis where the high risk indicates the exponential line. This shows that the proposed approach had managed to evaluate the risk of the samples. The mean risk value for the medium risk zone is 0.2 while the high-risk zone is 0.5.

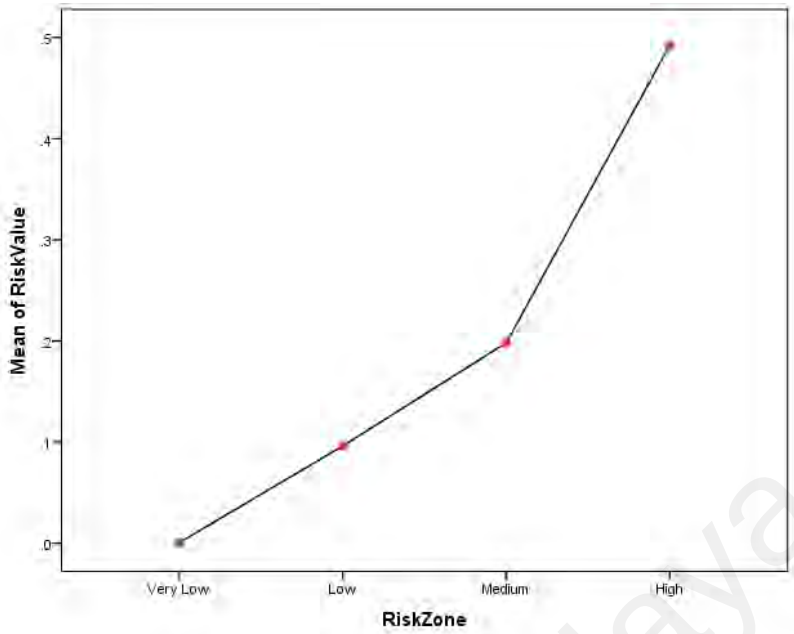

Figure 5.21: Risk zone analysis

5.5.4 Discussion

The EZADroid had implemented a permission-based application assessment. It collected 5,000 applications from a malware repository, Drebin, which was generally accepted as the Android malware dataset (Arp et al., 2014). The EZADroid also used 5,000 benign applications downloaded from AndroZoo (Allix et al., 2016) which belongs to Google Play. In total, this paper analyzed 10,000 applications by using the risk assessment approach. Both the benign and malware applications were analysed by VirusTotal (Total, 2016) which checks the validity of the malicious activities (Faruki et al., 2013). The VirusTotal provides significant results which also include a definition of all kinds of malware through 70 antivirus products (e.g. Symantec and Kaspersky; Talha et al., 2015). The experimental results demonstrate the effectiveness of the EZADroid in differentiating the malware from the benign applications. This is illustrated in Figure 5.17, Figure 5.18 and Figure 5.19.

Table 5.24 illustrates the results of the risk evaluation which evaluated the risk for the different kinds of malware family. Table 5.24 also shows that the EZADroid is effective

in identifying the risk for malware applications. The EZADroid had achieved an accuracy rate of over 89 percent thereby raising a challenge of the risk assessment for Android mobile malware. Specifically, the permission-based applications could preclude the use of the sophisticated risk assessment approach. Interestingly, a higher number of criteria had resulted in a lower performance of the risk assessment.

Using less criteria seems to show more promising results, as illustrated in Table 5.24. This outcome is also supported by Feizollah et al. (2015) where it was noted that less criteria resulted in less time thereby reducing the cost of the experiment. It is also prominent in showing more accurate results. From another perspective, the presence of the criteria (including benign and malware) provided some unique advantages in generating a risk zone and in preventing the malicious applications from affecting Android mobile users. It thus helped to increase users' awareness. One of the more significant findings that emerged from this experiment is that the number and types of criteria used, could have an effect on the effectiveness of the risk assessment. One possible solution recommended is to adapt a machine learning approach to create a different result in the risk assessment so as to ensure the effectiveness of the risk zone.

To demonstrate the superiority of the work, we compared the results with existing solutions. Kim et al. (2017) had conducted a study on risk assessment for Android applications to identify malware applications using Naive Bayes classification. Kim et al. (2017) proposed the Android application package (APK) Vulnerability Identification System (AVIS) that classifies an application into malware or benign based on the DEX file. The authors used 250 applications samples as the training set where the total number of malware and benign samples were 125 each. Their approach achieved approximately 94 percent accuracy. Although the current study, in comparison, had recorded a slightly lower accuracy rate, the size of the dataset used in the current study was much higher than

those of Kim et al. (2017), i.e. 10000 application samples. The approach used in this study had also managed to lower the risk of overfitting with better generalisations on unseen samples. In comparison, Rashidi et al. (2017) had proposed the XDroid which is an Android application and resource risk assessment tool. The XDroid was implemented with the dynamic analysis technique to detect malware by observing the applications' behaviors. Their results showed that the XDroid only managed to achieve 82 percent accuracy with 1000 samples training set. The XDroid also consumed higher consumption of computational time to train the model.

In contrast, our EZADroid solution had implemented the static analysis techniques which was effective in detecting known malware. The results of our experiments revealed the effectiveness of our approach. Detection rate showed 89.82 percent accuracy. This shows that our proposed approach had performed better than previous studies. The proposed approach of this study also involved using risk analysis on applications as well as to generate responses to users through the risk zone. This helped to alert security concerns for the users. Therefore, the sensitive data of the users can be safely protected via timely responses. Further to that, our approach was also able to reduce attacks from malware application and similarly minimise the risks.

5.5.5 Conclusion

This study has highlighted the significant findings of risk assessment and the risk zones for Android applications through the EZADroid which also implemented a permission-based application to determine the risk zones. Based on that knowledge and the effective risk evaluations, it was able to assess the mobile Android application into four (4) types of risk zones (e.g. very low, low, medium and high).

The data were collected from the permission-based applications using static analysis. The collected data were organized in a database. In order to select the effective criteria to

increase the effectiveness of the risk evaluation on the EZADroid, this study applied the criteria selection approach. The combination of the risk assessment approach and the AHP approach had improved the risk evaluation level as well as determined the risk zones for Android applications. The applications taken from Drebin and AndrZoo were used for fixing the validation and reputation of the EZADroid. It was noted that the EZADroid offered a good risk zone performance evaluation.

The EZADroid performed and achieved an accuracy rate of 89.82 percent in the experimental evaluation of 5,000 malwares and 5,000 benign applications based on the 10 criteria approach. The EZADroid was able to achieve an accuracy rate of over 80 percent on risk evaluation without using the machine learning classifier. This is the main advantage of the proposed approach. Moreover, the approach was also suitable to be installed on a mobile device as it provides good risk evaluation and increases user awareness about the risk of applications. This was accomplished through the illustration of the risk zone threshold.

The EZADroid approach has a limitation in running the risk evaluation for malware applications that do not use the criteria that were selected in its permission. This is a limitation of all permission-based malware detection mechanisms. Furthermore, the EZADroid was also unable to calculate the risk if the malware application does not have any permission. Nonetheless, this resolved by combining the permission-based applications with other different criteria. Another limitation is traced to the general static analysis applied. Here, the static analysis was less efficient in detecting the malware with an obfuscation technique. Considering the weaknesses of this study, an uninstalled or blocked application may be a good protection alternative for mobile devices. Therefore, more investigations and experimentations on Android risk assessments need to be conducted.

5.5.6 Summary

This chapter has discussed the evaluation study of the selected static features derived from the investigations and methods used in the proposed framework. The useful results from the experiments have demonstrated a combination of different aspects of evaluation, and they highlighted their unique findings and conclusions.

The key objective of describing the evaluation at different experiments of studies is to investigate the unique objectives at each experiment. The result presented has shown strong evidence to support the ability of the proposed framework to work robustly based upon its operational characteristics. In conclusion, the analysis made of the studies clearly defined their contribution as well as stating their limitations.

To further investigate the usefulness and feasibility of the proposed framework in a practical mode, the following chapter presents the prototype of the proposed framework and evaluates it using different datasets to the one used in this chapter, in order to test the efficiency in predicting unknown malware.

CHAPTER 6: PROTOTYPE IMPLEMENTATION OF RISK ANALYSIS AND MALWARE DETECTION SYSTEMS

After validating and evaluating the proposed framework, the next stage of the research is to design and implement a prototype system that demonstrates the main operation and also shows how these could be implemented in practice. This chapter discusses the prototype implementation of the proposed framework specifically, the administration detection module and risk assessment. The main features of the administration module have been embodied in the Web interface, EZADroid, which was used to manage, monitor, configure, detect and assess the risk on Android applications modules. Several modelling languages were used including case diagrams and state diagrams. These were used to provide a visual illustration of the prototype.

6.1 Implementation of EZADroid system

There are three parts in the proposed framework as illustrated in Figure 6.1 below. They include the Webpages, mobile device, and detection modules. These modules have been fully implemented whereas the data were collected from other sources such as AndroZoo and Drebin.

The rationale behind adopting the existing IDS and risk assessment approach rather than implementing them from scratch is twofold. Firstly, implementing these modules from scratch would have been out of the scope of this study and supporting input from existing IDS and risk assessment would provide a more realistic environment and strengthen the compatibility of the existing solutions. VirusTotal (Quintaro, 2017) as a popular free online service that analyzes file and URLs was used to identify the malicious contents that feed the prototype. Its only drawback is its inability to provide more information about the level of risk. The descriptions of the modules that were fully implemented in this study are as follows:

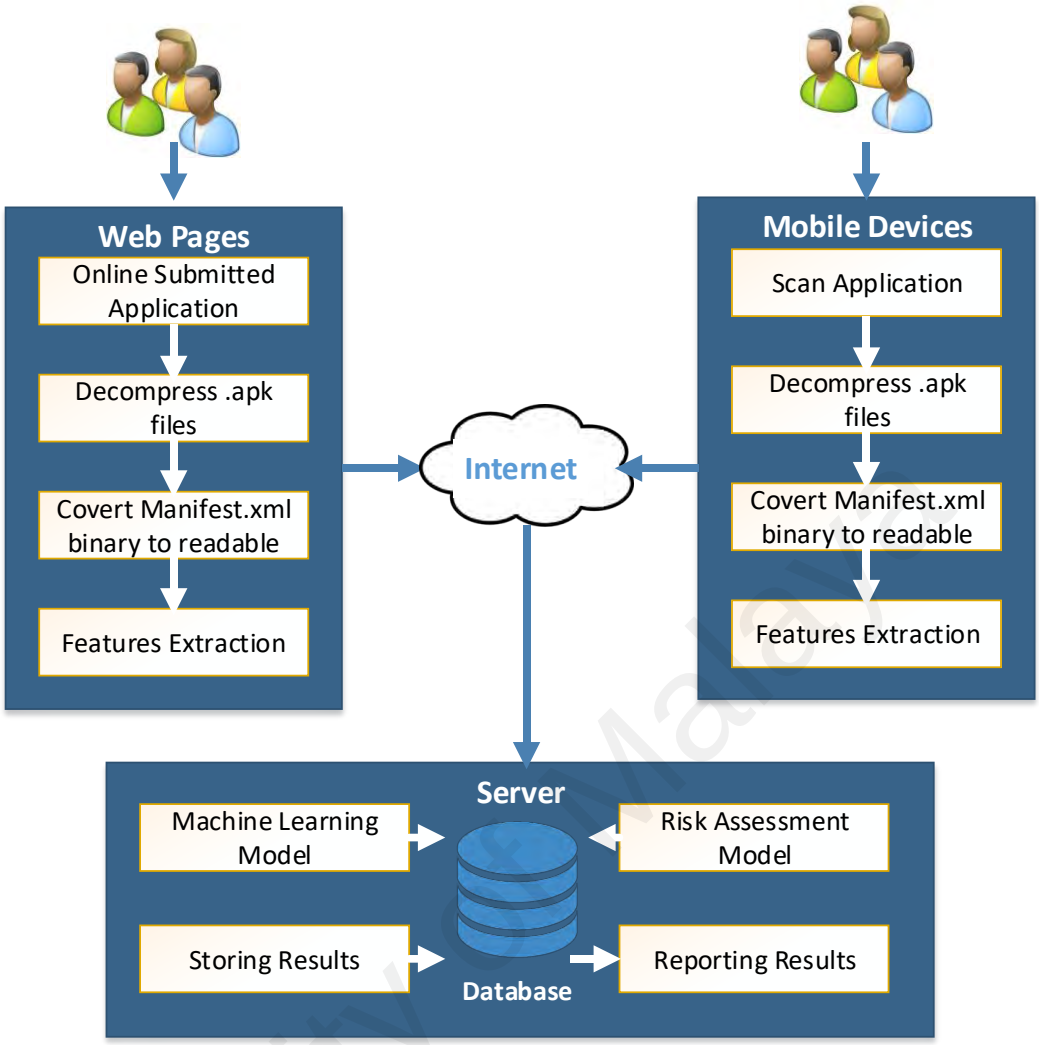

Figure 6.1: Web development framework

6.1.1 Use case diagram

Use case diagram has been commonly adopted to present a graphical overview of the functionality of a system and to show the relationship between actor and system. It is a set of actions, functions and services that is performed by the system. In this context, the "system" is the EZADroid while the 'actor" is the user. Use case diagram determines the characteristics of the developed systems with the actor without worrying about the details on how that functionality is implemented. Figure 6.2 demonstrates the system level and the relationship between the external systems.

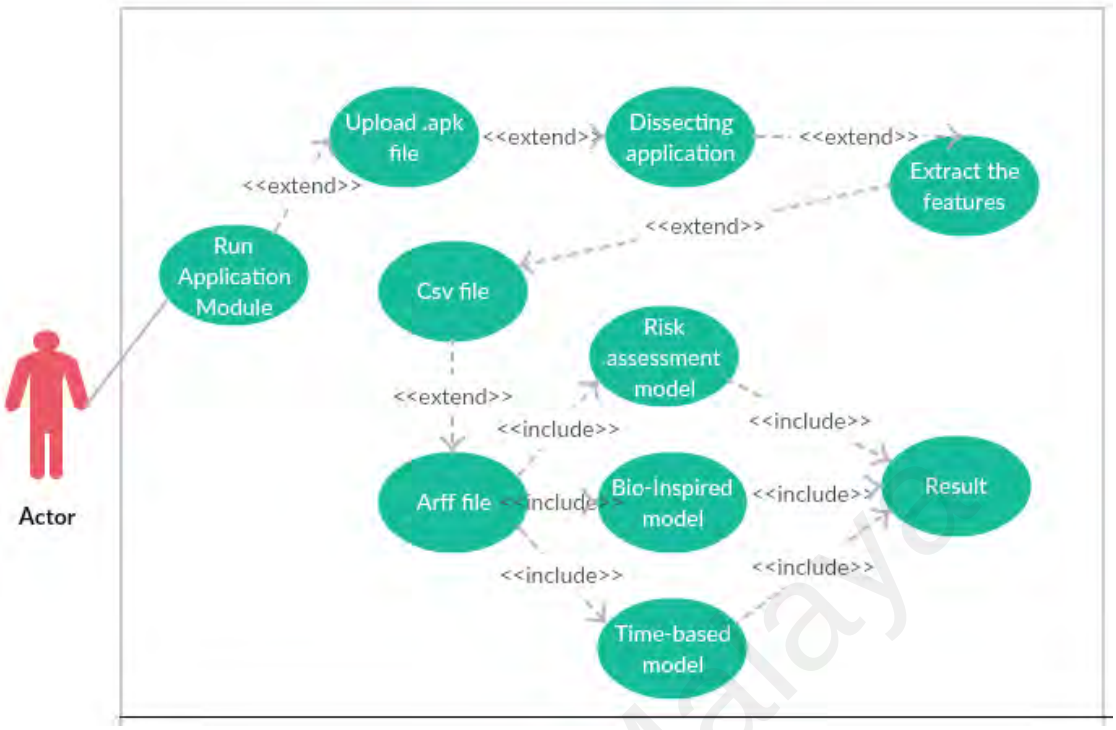

Figure 6.2: Use Case Diagram

The following explanations illustrate the role of the user in the above figure.

i. User as actor is an entity that performs a role in one given system. The actor is able to interact with the system to manage and run specific application modules. This includes the ability to update the detection model, analyse and configure the Web module. In addition, the actor or user is able to upload Android applications to the Web analyser to analyse the applications as benign or malware.

ii. The system defines the scope of the system as anything within the box that represents the functionality that is within the scope.

6.1.2 State diagram

The state diagram defines the different states of the system. These states are controlled by external or internal events. The purpose is to model the dynamic nature of a system and to respond to the external and internal events. In addition, it is used to control the flow process of one state to another. Figure 6.3 illustrates all the possible states in the proposed framework and it also summarises the characteristics of the running system.

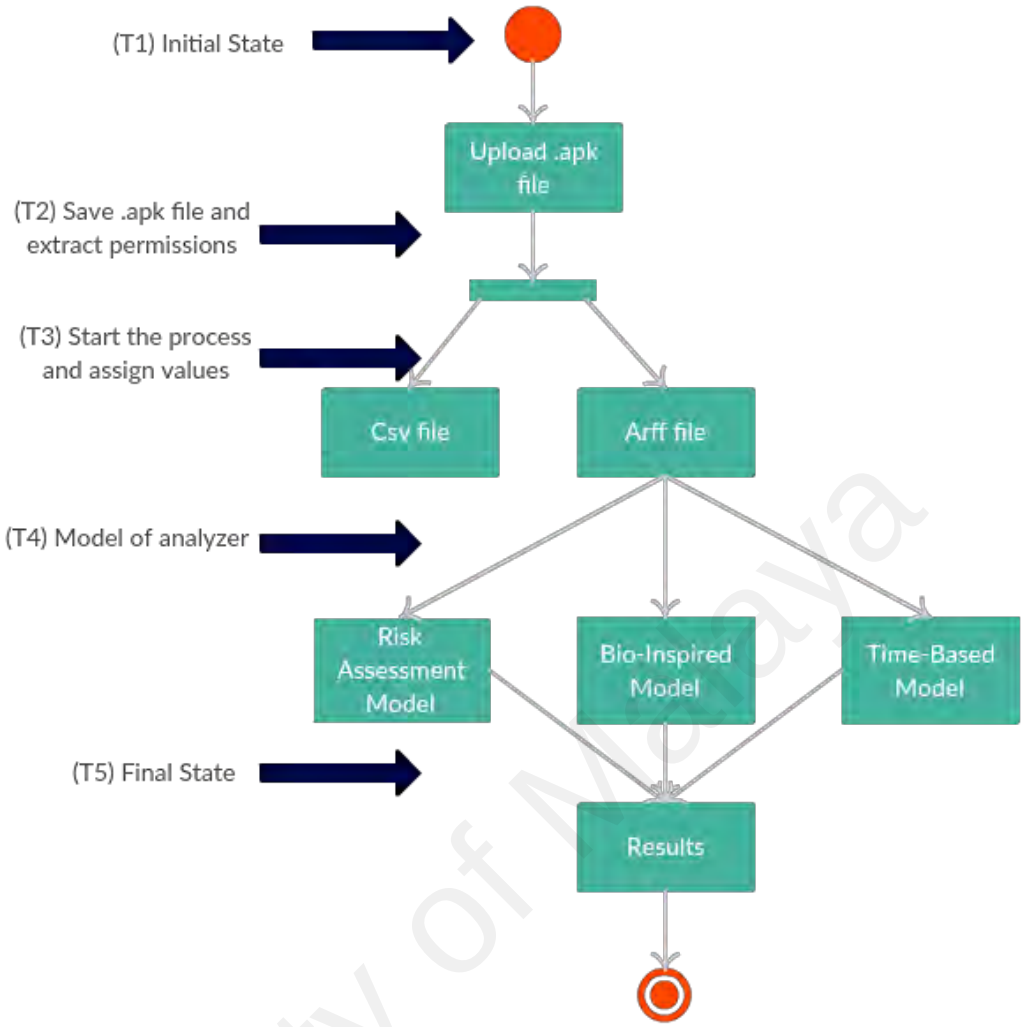

Figure 6.3: Prime-state Diagram

Figure 6.3 illustrates three sub-states. The summary of each is provided as follows:

a) **Save .apk file and extract permission:** The initial state is T2.1 when the user uploads the .apk file. There are four sub-states in this specific state namely, D*atabase stored .apk file, Identification of .apk file, Decompress .apk file* and *Stored permission features*. Figure 6.4 illustrates the storing .apk file state. In T2.2, the system identifies the file either as the .apk file or a different type of file. If the file is .apk, then the user is able to upload it in the system. The T2.3 state starts the process by decompressing the file to obtain application permissions. In T2.4, the system extracts the proposed features and stores them.

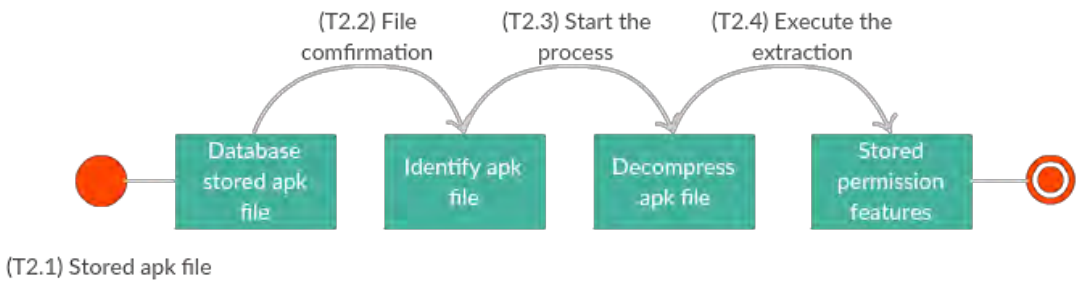

Figure 6.4: Storing of .apk file state

b) Assign value: This state assigns the features vectors by creating the csv and arff files. It is important for the model of the analyser to detect and assess the risk on uploaded. apk file. These files contain the information of permission features. Figure 6.5 illustrates the assign value state.

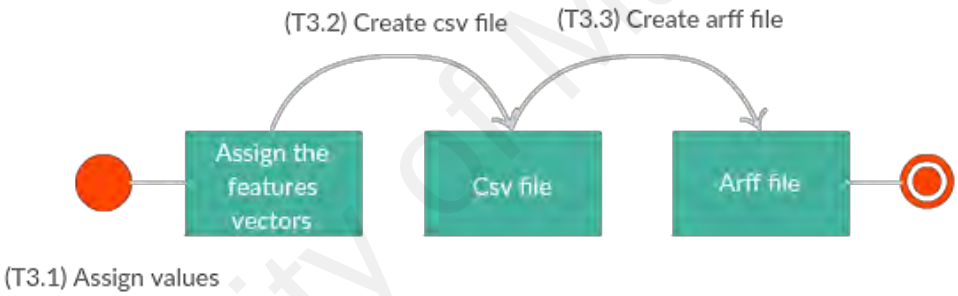

Figure 6.5: Assign value state

c) Model of analyser: This state processes the arff file to detect and assess the risks according to the analysers namely, Risk Assessment Model, Bio-Inspired Model and Time-Based Model. The Risk Assessment Model assesses the risk on the Android applications and provides risk responses indicating whether they are very low, low, medium or high risk. The Bio-Inspired model detects the unknown malware through the PSO feature selection while the time-series model detects malicious applications by using the proposed time based features. Figure 6.6 illustrates the model of the analyser state.

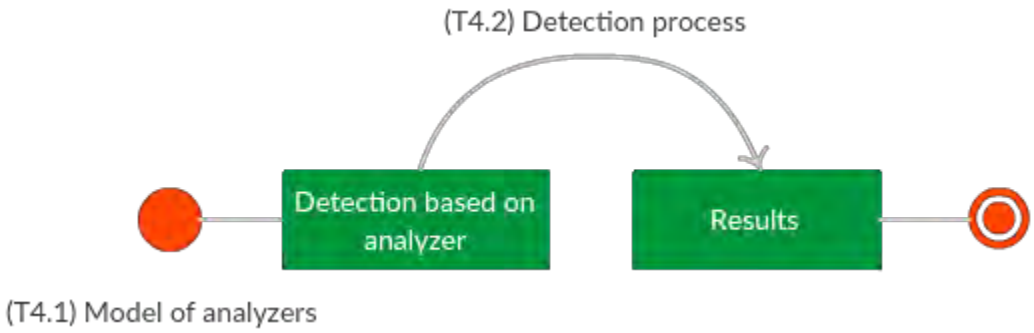

Figure 6.6: Model of analyser state

6.2 Demonstrating the risk analysis and malware detection system

Having presented the main functionalities of the proposed framework and its web module, this section demonstrates some examples to show how malware can be classified. The analysers have the capability to decompress the Android applications, feature extraction, detection and risk assessment.

a) **Decompress Android application:** This system requires the Android application package file (.apk) to be used as a sample to be analysed on the website and mobile devices. The system grants a unique identification number followed by the name of the file so as to avoid duplications.

b) **Feature extraction:** The next step sees the system collecting the information from the manifest.xml file and extracting the proposed features.

c) **Detection:** The system then uses the collected features as input for the machine learning classifier to predict the class of uploaded file as either malware or benign applications.

d) **Risk analysis:** Finally, the system measures the risk on applications and provides response to the user by showing the level of risk (very low, low, medium and high).

The difference between these analysers are in the proposed approach and the features. This analyser implements two approaches: risk assessment and machine learning. In order

to demonstrate the effectiveness of the analyser, these prototypes used similar hardware specifications which consist of desktop computer equipped with Intel Core i7-4770 CPU of 3.40 GHZ, 16 GB of RAM, and the operating system of Microsoft Window 7 Professional.

6.3 Risk analysis and malware detection system

This section demonstrates some examples to show how the risk analysis and malware detection were classified. It also presents the main features of the proposed framework and its module and the login web pages. Figure 6.7 further illustrates.

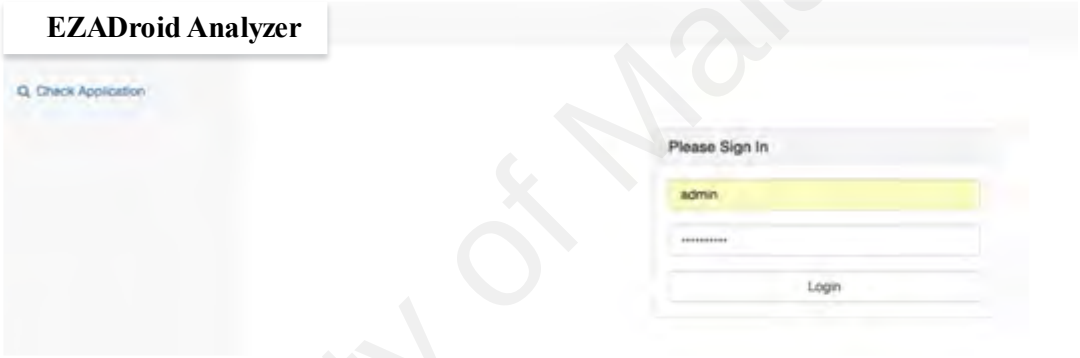

Figure 6.7: Login page

The login account system allows the user to give specific access to the system. Two types of login accounts are noted: administrator and normal user. The administrator account is a special account that is used for managing other users' accounts and for making changes to the system. It has the capability to view the figures and total number of applications submitted. In contrast, the normal user account is for normal tasks such as submitting Android applications and viewing the scanned results. To log into the Web module, administrator and normal users are required to use a legitimate username and password otherwise the Web module will prevent any access to other pages. The password is stored in the database. Once the login process is successful, the administrator and the normal user are redirected to the landing page which displays the main dashboard

of the module. To begin analysis, a user is required to upload Android applications into the system. Figure 6.8 illustrates the upload page for use by Android applications.

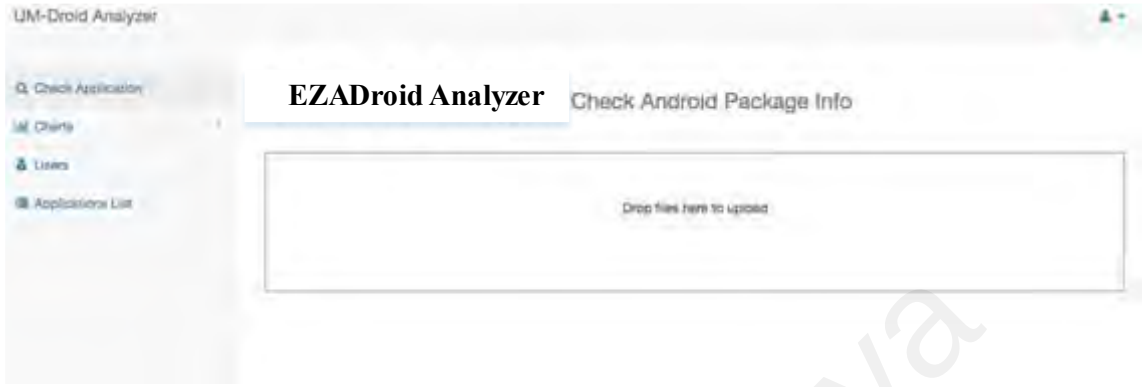

Figure 6.8: Upload page for Android applications

Once the process of the uploaded file is successful, a list of uploaded Android application is listed in the result page as shown in Figure 6.9.

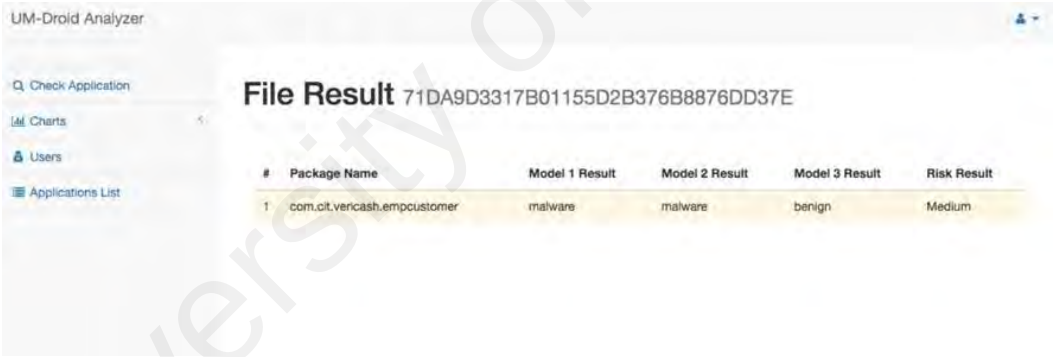

Figure 6.9: Result page

Figure 6.9 shows the result of the uploaded Android applications. This page shows the information regarding the uploaded applications including the name of the file using SHA-1 hashing, package name, result from the model and risk results. In the enthusiasm to substantiate the adequacy of experiment results, this study applied Drebin dataset for training and testing as the evaluation. Apart from the evaluation, the EZADroid was also evaluated through Malgenome malware dataset to expose the practicality of the framework and a scenario independence model. The obtained results from Malgenome

testing dataset showed that the framework recorded accuracy of 95.0%. It shows that the proposed approaches capable to detect malware even though using cross-dataset. In addition, after an analysis is successfully carried out, the user is able to view the entire result for the uploaded file, as shown in Figure 6.10.

Figure 6.10: List of application page

Figure 6.10 provides the risk analysis result which shows whether the Android applications have very low, low, medium or high risk. This risk result indicates the severity of the Android applications. The two models noted in Figure 6.10 refer to the evaluation of our approach as stated in Section 5.2 and Section 5.4.

Model 1 uses the Bio-inspired approach while Model 2 uses the Time-based approach. These models analyse the uploaded applications and determine whether the applications are malware or benign. As concluded, this system provides a summary of the detection as well as the risk of the applications. Figure 6.11 further illuminates.

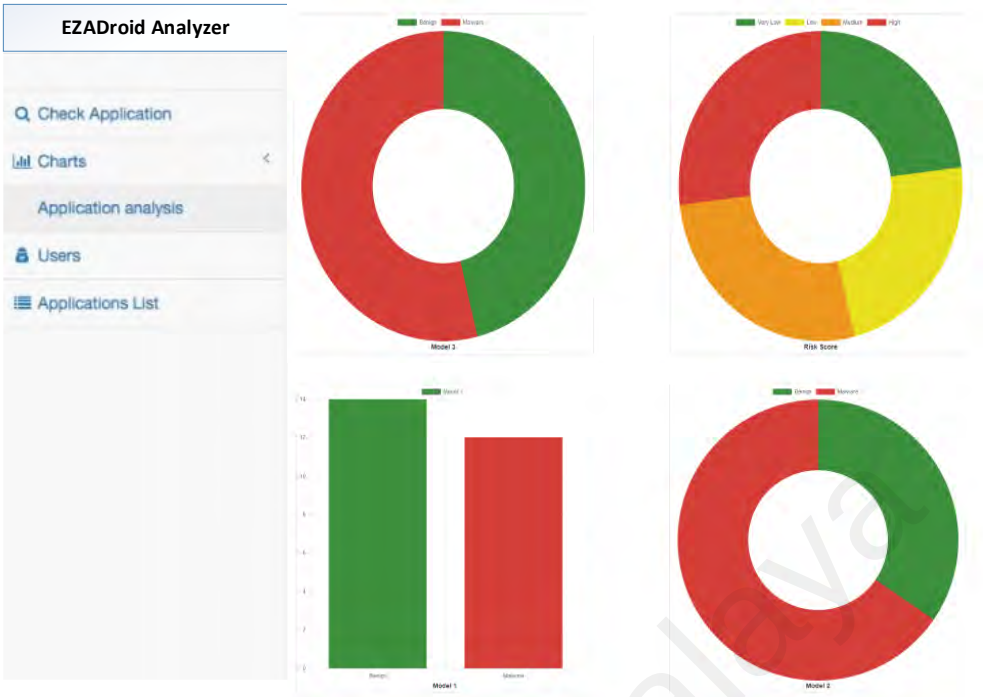

Figure 6.11: Summary of analysis

The EzaDroid offers a more detailed analysis based on the number of uploaded applications. It summarises the number of risks indicated by the benign and malware application as noted in the graphs. The EzaDroid provides a flexible platform for the administrator to configure, analyze and make wise decisions by using the analysis results. In particular, the Web page and mobile devices module gives the following advantages:

a) **Analytics results.** The Web and mobile device module offers a web analytics solution that gives rich insights into the malware detection process besides simplifying the analysis. The EzaDroid is easy-to-use, is customisable, flexible and able to optimise results. Hence, it allows administrators to analyse the entire detection process and examine the results in an online assessment mode.

b) **Simple graphical interfaces.** The Web and mobile device module also has the capability to analyse and respond by providing graphical interfaces in interpreting the malware analysis. It uses different colors and interesting graphs to improve user experience. This also allows the administrator to interpret the detection

results conveniently thereby giving an additional advantage to high-level management and non-technical people to understand the current situation.

c) **Easy management and user friendly interfaces.** The EzaDroid analyser also provides a user friendly interface where the administrator is able to customise the display settings.

d) **Able to utilize in mobile browser.** The EzaDroid module is Web based or mobile device based. Therefore, it is available for making analysis on the internet browser which may be installed in mobile devices. This advantage provides users with the opportunity to scan their applications quickly thereby minimizing the risk of malware attacks.

In addressing the advantages of the Web based module, some limitations are detected.

a) **Applications dependent.** As the Web module is served using a web server, it also relies on the efficiency of the web server to be efficient. In the case of the web server going offline, the detection process will be terminated and other analyses have to be halted. In addition, the network consistency is necessary for the communication and exchange of information to be processed.

b) **Inherit other vulnerabilities.** Due to the use of web applications, the Web module is henceforth vulnerable to some existing web applications such as HTTP Parameter Pollution (HPP), SQL injection, cross-site scripting and session hijacking. This increases the vulnerability of the Web and mobile device module to other hardware (e.g. web server and mobile devices) and software (e.g. browsers) elements.

In that regard, it is important to address these limitations by looking at other security precautions and countermeasures so as to ensure that future malware detection more efficient.

6.4 Summary

This chapter has presented the implementation stage of the proposed framework. It provided some examples and snapshots that were extracted from the Web and mobile device module which consists of the EzaDroid. The details of this module were explained by describing its system architecture and state diagrams.

The key objective of demonstrating and describing the details of the module was to shed light on how the EzaDroid works. A detailed explanation was given to demonstrate how its internal module may be affected by the external environment. Due to time constraint, it was impossible to fully implement the operation of some modules. This limitation, among others, is further exploited in the following chapter.

CHAPTER 7: CONCLUSION

Mobile devices in today's era are equipped with powerful computing and networking capabilities. In addition, they are easy to carry and have become the choice of most users with many preferring the tablet or smartphones. This phenomenon has led to mobile devices outselling the number of PCs worldwide. The increasing development of mobile devices and their widespread use among users throughout the world have simultaneously attracted unscrupulous authors for personal benefits such as money. Thus, these unscrupulous authors device some malicious software which can attack mobile users and consequently, collect their personal information from their mobile devices for their own benefits. Although a number of existing approaches such as firewall, antiviruses and Intrusion Detection Systems (IDSs) are available as a solution to overcome these malware attacks, the current advancement of technology indicates that there is still a need to develop a novel approach to detect malware. The availability of sophisticated techniques has encouraged these unscrupulous authors to become even more daring by taking more sophisticated steps to overcome the security and detection mechanisms so as to make malware attacks more difficult for detection. This thesis is based on a study that attempts to identify the problem of mobile malware attacks on the Android platform; it also aims to address several fundamental issues when automating its analysis on a large scale scenario.

This chapter summarises the study by reviewing the research aim and objectives. It highlights the most important findings of this study as well as the limitations. This chapter also discusses the potential of new research within the same domain showing how the proposed framework able to further enhanced for future implementation in mobile malware detection.

7.1 Research objectives

The aim of this study was to improve a malware detection system by using the static analysis technique for Android mobile applications. Section 1.4 had described the four research objectives of this study. In order to accomplish the aim, the four research objectives have to be accomplished so as to solve the problem.

Objective 1: To review the security vulnerabilities, challenges of each Android mobile application and establish the research gap by analysing the state-of-the-art malware detection system by investigating the properties of the mobile applications which are most critical with respect to the creation and sustainability of malware attacks on mobile applications.

The first objective was to critically investigate the current state-of-the-art malware detection specifically those of Android mobile devices. The research objective was accomplished by conducting a thorough review of the most crucial works published in online scholarly journals. They were extracted from digital libraries which were accessed through the University of Malaya's access portal. These journals include those published by the Institute of Electrical and Electronics Engineers (IEEE), the Association for Computing Machinery (ACM), Elsevier and the Web of Science portals. This objective was accomplished through Chapter 2 where all the related information regarding Android malware detection and risk assessment were presented. Chapter 2 also presented the malware detection taxonomy and the machine learning approach and algorithms. The static analysis technique using machine learning approach with anomaly based detection was reviewed as it offers a higher potential in uncovering and predicting unknown and future malware.

Objective 2: To propose a malware detection system that uses risk analysis to analyse the Android mobile applications, which is capable of analysing the structural properties of the Android mobile applications for detecting malware.

The second objective of this study was to propose a novel approach to facilitate the practical evaluation of mobile risk analysis by using the machine learning approach and to facilitate the analytical hierarchy process. The outcomes gathered from this study highlighted the importance of the risk assessment in determining the level of risk noted on Android applications. This study presented the threshold which assesses the risk; it also provided the risk levels which ranged from very low, low, medium to high risk. In addition, this study also searched for relevant features with minimum numbers to be used for assessing the risk noted on Android applications by using the feature selection approach (i.e. information gain). This is important for improving the assessment measurement results. This objective was accomplished in Chapter 5.

Objective 3: To propose a malware detection system that is based on the time series approach by observing the behavioral properties of the Android mobile applications through time for the purpose of predicting future mobile malware.

The third objective of this research was to propose the malware detection system based on the time series approach. The evaluation of the malware detection system was examined in two platforms: a) WEKA and b) Prototype. In the Weka simulation, the experiments tested the features in six evaluation measures: accuracy, True Positive Rate (TPR), recall, precision, f-measure and False Positive Rate (FPR). In the prototype platform, the experiment evaluated the features in terms of the accuracy and performance of each analyser model (i.e. Risk Assessment, Bio-Inspired and Time-based) in a practical environment. This assessment encompassed processes which include the decompressing

of applications, the extracting of features, the selecting of features and the measuring of the risks of Android applications. This objective was accomplished in Chapter 5.

Objective 4: To evaluate the proposed system in terms of detection accuracy by using real-world Android malware and implement the prototype of the proposed system for a practical evaluation via a web-based assessment.

The fourth objective of this study was to evaluate a prototype of the malware detection system based on a novel framework that was developed according to the relevant features and methodology approach. This study then devised a Web based system to detect Android applications to differentiate the files as malware or benign. The apk file was uploaded by the user to check the file application on the Web page or mobile device module. The system decompresses the apk file to extract features; it then identifies the classes of applications with the machine learning model. Based on the review of literature, this study had selected the best machine classifiers by comparing the performance of the prototype with the WEKA results. The proposed system contains three model analysers namely, Risk Assessment, Bio-Inspired and Time-Based model. This objective was accomplished in Chapter 6.

7.2 Achievement of the study

This research began by studying the mobile devices evolution and reviewing the different types of malware detection systems. It explored the issues about mobile device detection and risk assessment as well as the selection of relevant features. Several machine learning classifiers were explored and the performance results were collected. The study then evaluated the performance results so as to satisfy the aim of the study. Several points of interests were identified as noted in the following:

a) **A detection model for mobile malware.** This study has created a model which able to detect mobile malware applications through the static analysis approach. A machine learning approach was used as a better adaptive detection model. The model worked perfectly in detecting mobile malware based on permission features.

b) **Issues in mobile malware detection studies.** In Chapter 2, this study presented the state-of-the art technique on mobile malware detection and their significance in detecting mobile malware. By presenting the strengths and weaknesses of these issues, several strategies were identified for addressing the limitations. In order to improve the effectiveness of the malware detection system, research highlighting some of the limitation were performed. The aim was to search for the relevant features which used to develop a more efficient approach.

c) **Issue in mobile malware feature selection**. This study has shown a critical analysis of the different perspectives used when addressing the significant problems of feature selection in anomaly-based detection model. The aim was to improve detection performance and to minimise complexity.

d) **Implementation of the proposed model.** The investigation was extended by examining the feasibility of the proposed model so as to demonstrate its practical application on Android permissions with the risk assessment model. A proof-of-concept study was designed and realised (See Chapter 6). As an extension to the evaluation study, the implementation stage also developed a Web-based system, which concentrates on the Web module of the proposed model. To illustrate the implementation stage, the proposed model was presented with details using the modelling language. It includes the case diagrams as well as some snapshots extracted from the prototype pages.

e) **Risk Assessment model for mobile applications.** This study has created a model which has the capability to assess the risk of mobile applications through the risk assessment approach. The Analytical Hierarchy Process technique was applied to measure the risk and the rule was applied in the model. The model worked perfectly in assessing risks on mobile applications as well as generating the risk levels noted (i.e. very low, low, medium and high).

7.3 Limitation of the study

The discussions noted in previous chapters have validated that this study has adequately achieved its aims and objectives - the establishment of a novel framework that is useful for detecting unknown malware in anomaly-based detection environment. However, a number of limitations and challenges were encountered during the study and they are listed here for future references.

a) **The evaluation of the study was taken only from static-based detection model**. In conducting the experiment during the evaluation phase (see Chapter 5), this study found some practical limitations. In particular, all input features are gathered from static analysis. However, in practical solution, static and dynamic have their pros and cons. Therefore, comparing results from both analyses would be more meaningful.

b) **A practical proof-of-concept.** Although a practical evaluation study using web modules and a live simulation has been presented in Chapter 6, it is important to perform the entire prototype in a real mobile device with actual detection module because it able to strengthen the feasibility of the proposed approach. Moreover, the result from the experiments show a clear distinction between the ways mobile malware detected and risk analysis on Android applications.

c) **The usability of web module.** The performance evaluation study extended the implementation phase to demonstrate the practically of the web modules. However,

the usability of such module is not evaluated in this study. The snapshots of the prototype pages presented in Chapter 6 are considered adequate in order to demonstrate the usage of the web module.

7.4 Summary- suggestion for future works

The following are suggestions for future work outside the scope of this study have been identified as follows:

a) **Visualisation for risk analysis and malware detection.** This thesis has provided figures, tables and graphs which used to the advantage of security analysts and academic researchers in terms of an interactive approach that offer viewers an insight into the two dimensional aspect of the analysis. This thesis had also provided sizeable numerical figures and statistical data in an effective manner. The significance of these figure, tables and graphs is their ability to draw readers to the specific information in a short time span. Therefore, using different types of graph models offer further opportunities for future research.

b) **Improve false alarm rate.** False alarm rates continue to be an issues as long as it exists in the detection module. False alarms refer to the statistical measure of how well sample dataset correctly classify the malware applications. This means that the malware data have been falsely predicted as normal. This problem leads to false detection of applications and even a small rate of false alarms able to trigger huge impacts. Therefore, a reliable and effective detection module is necessary to overcome this issue.

c) **Select relevant features.** As data become more complex and larger, the selection of relevant and adequate features to improve detection performance becomes harder. The process will require further analysis to be conducted so as to examine the correlation between malware and benign applications. Doing so, it helps to

reduce instances of false alarms thereby increasing higher detection accuracy rate, especially when dealing with permission features.

d) **The malware detection approach could be combined to produce a hybrid approach.** Hybrid approaches have the capability to detect new variants of malware attacks. This hybrid approach when designed, will be able to perform static analysis and dynamic analysis using signatures and running applications. Moreover, this hybrid approach could also produce explicit learning and more comprehensible models. This hybrid approach can be used by future studies to improve the performance of malware detection.

REFERENCES

Afifi, F., Anuar, N. B., Shamshirband, S., & Choo, K.-K. R. (2016). DyHAP: Dynamic Hybrid ANFIS-PSO Approach for Predicting Mobile Malware. *Plos One*, *11*(9), e0162627.

Ahmad, I. (2015). Feature Selection Using Particle Swarm Optimization. *International Journal of Distributed Sensor Networks*, *2015*, 1–8.

Alazab, M., Monsamy, V., Batten, L., Lantz, P., & Tian, R. (2012). Analysis of Malicious and Benign Android Applications. *2012 32nd International Conference on Distributed Computing Systems Workshops*, 608–616.

Alazab, M., Venkataraman, S., & Watters, P. (2010). Towards Understanding Malware Behaviour by the Extraction of API Calls. *2010 Second Cybercrime and Trustworthy Computing Workshop*, (November 2009), 52–59.

Allix, K., Bissyandé, T. F., Jérome, Q., Klein, J., State, R., Le Traon, Y., … Traon, Y. Le. (2016). Empirical assessment of machine learning-based malware detectors for Android: Measuring the gap between in-the-lab and in-the-wild validation scenarios. *Empirical Software Engineering*, *21*(1), 183–211.

Allix, K., Bissyandé, T. F., Klein, J., & Le Traon, Y. (2016). AndroZoo: collecting millions of Android apps for the research community. *13th International Workshop on Mining Software Repositories - MSR '16*, 468–471.

Alzahrani, A. J., Stakhanova, N., Gonzalez, H., & Ali, A. (2014). Characterizing Evaluation Practices of Intrusion Detection Methods for Smartphones. *Journal of Cyber Security*, *3*(2), 89–132. Retrieved from Alzahrani2014

Anuar, N. B., Furnell, S., Papadaki, M., & Clarke, N. (2011). A risk index model for security incident prioritisation. *Australian Information Security Management Conference*, 24–39.

Anuar, N. B., Papadaki, M., Furnell, S., & Clarke, N. (2013a). A response selection model for intrusion response systems: Response Strategy Model (RSM). *Security and Communication Networks*, *7*(11), 71–81.

Anuar, N. B., Papadaki, M., Furnell, S., & Clarke, N. (2013b). Incident prioritisation using analytic hierarchy process (AHP): Risk Index Model (RIM). *Security and Communication Networks*, *6*(9), 1087–1116.

AppBrain. (2016). Google Play Stats. Retrieved October 12, 2016, from http://www.appbrain.com/stats/stats-index

Apvrille, A., & Strazzere, T. (2012). Reducing the window of opportunity for Android malware Gotta catch 'em all. *Journal in Computer Virology*, *8*(1–2), 61–71.

Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., & Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. *Symposium on Network and Distributed System Security (NDSS)*, 1–15.

Arshad, S., Ahmed, M., Shah, M. A., & Khan, A. (2016). Android Malware Detection & Protection : A Survey. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *7*(2), 463–475.

Aung, Z., & Zaw, W. (2013). Permission-Based Android Malware Detection. *International Journal of Scientific & Technology Research*, *2*(3), 228–234.

Becher, M., Freiling, F. C., Hoffmann, J., Holz, T., Uellenbeck, S., Wolf, C., & Horst, G. (2011). Mobile Security Catching Up ? Revealing the Nuts and Bolts of the Security of Mobile Devices. In *2011 IEEE Symposium on Security and Privacy* (pp. 96–111). Ieee.

Boukhtouta, A., Mouheb, D., Debbabi, M., Alfandi, O., Iqbal, F., & El Barachi, M. (2015). Graph-theoretic characterization of cyber-threat infrastructures. *Digital Investigation*, *14*, S3–S15.

Bryan, J., Cook, D., Josse, J., Kalibera, T., & Narasimhan, B. (2017). The R Project for Statistical Computing. Retrieved June 5, 2017, from https://www.r-project.org/about.html

Butun, I., Morgera, S. D., & Sankar, R. (2014). A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Sensors Journal*, *14*(5), 1370–1379.

Caetano, L. (2014). Are Your Apps Oversharing? 2014 Mobile Security Report Tells All. Retrieved February 16, 2017, from https://securingtomorrow.mcafee.com/consumer/mobile-security/mobile-security-report-2014/

Cegan, J. C., Filion, A. M., Keisler, J. M., & Linkov, I. (2017). Trends and applications of multi-criteria decision analysis in environmental sciences: literature review. *Environment Systems and Decisions*, *37*(2), 123–133.

Cen, L., Gates, C., Si, L., & Li, N. (2015). A Probabilistic Discriminative Model for Android Malware Detection with Decompiled Source Code. *IEEE Transactions on Dependable and Secure Computing*, *12*(4), 1–13.

Chen, P. S., Lin, S.-C., & Sun, C.-H. (2015). Simple and effective method for detecting

abnormal internet behaviors of mobile devices. *Information Sciences*, *321*(January 2012), 193–204.

Choi, J., Sung, W., Choi, C., & Kim, P. (2015). Personal information leakage detection method using the inference-based access control model on the Android platform. *Pervasive and Mobile Computing*.

Chowdhury, M. J. M., Matulevičius, R., Sindre, G., & Karpati, P. (2012). Aligning mal-activity diagrams and security risk management for security requirements definitions. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7195 LNCS*, 132–139.

Chuang, H.-Y., & Wang, S.-D. (2015). Machine Learning Based Hybrid Behavior Models for Android Malware Analysis. *2015 IEEE International Conference on Software Quality, Reliability and Security*, 201–206.

Clay, J. (2015). Continued Rise in Mobile Threats for 2016. Retrieved September 8, 2016, from http://blog.trendmicro.com/continued-rise-in-mobile-threats-for-2016/

Clemens, J. (2015). Automatic classification of object code using machine learning. *Digital Investigation*, *14*, S156–S162.

Corona, I., Giacinto, G., & Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences*, *239*, 201–225.

Cui, B., Jin, H., Carullo, G., & Liu, Z. (2015). Service-oriented mobile malware detection system based on mining strategies. *Pervasive and Mobile Computing*, *24*, 101–116.

Damopoulos, D., Kambourakis, G., Gritzalis, S., & Park, S. O. (2012). Exposing mobile malware from the inside (or what is your mobile app really doing?). *Peer-to-Peer Networking and Applications*, *7*(4), 687–697.

Dassanayake, D. (2017). ANDROID WARNING - Google Play apps infect millions of phones with dangerous malware. Retrieved October 7, 2017, from http://www.express.co.uk/life-style/science-technology/854529/Android-warning-Google-Play-malware-ExpensiveWall

Deepa, K., Radhamani, G., & Vinod, P. (2015). Investigation of feature selection methods for android malware analysis. *Procedia Computer Science*, *46*(Icict 2014), 841–848.

Deshotels, L., Notani, V., & Lakhotia, A. (2014). DroidLegacy : Automated Familial Classification of Android Malware. *In Proceedings of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop 2014*, 3.

Desnos, A. (2012). Androguard. Retrieved March 8, 2017, from http://doc.androguard.re/html/index.html

Developer, A. (2016a). Android Permission. Retrieved September 2, 2016, from https://developer.android.com/guide/topics/security/permissions.html

Developer, A. (2016b). Platform Architecture. Retrieved from https://developer.android.com/guide/platform/index.html#art

Developer, A. (2016c). Platform Versions. Retrieved October 12, 2016, from https://developer.android.com/about/dashboards/index.html

Dini, G., Martinelli, F., Matteucci, I., Petrocchi, M., Saracino, A., & Sgandurra, D. (2012). A Multi-criteria-Based Evaluation of Android Applications. *International Conference on Trusted Systems*, 67–82.

Dini, G., Martinelli, F., Matteucci, I., Petrocchi, M., Saracino, A., & Sgandurra, D. (2018). Risk analysis of Android applications: A user-centric solution. *Future Generation Computer Systems*, *80*, 505–518.

Dini, G., Martinelli, F., Saracino, A., & Sgandurra, D. (2012). MADAM: A multi-level anomaly detector for android malware. *Computer Network Security*, 240–253.

Dweiri, F., Kumar, S., Khan, S. A., Jain, V., Ahmed, S., & Jain, V. (2016). Designing an integrated AHP based decision support system for supplier selection in automotive industry. *Expert Systems with Applications*, *62*, 273–283.

Dyer, D. W. (2010). Evolutionary Computation in Java. Retrieved March 10, 2017, from http://watchmaker.uncommons.org/manual/ch01.html

Egam, & Egham. (2016). Gartner Forecasts Worldwide Device Shipments to Decline for Second Year in a Row. Retrieved February 13, 2017, from http://www.gartner.com/newsroom/id/3468817

Egele, M., Scholte, T., Kirda, E., & Kruegel, C. (2012). A Survey on Automated Dynamic Malware-Analysis. *ACM Computing Surveys (CSUR)*, *44*(2), 1–42.

Egham. (2015a). Gartner Says Tablet Sales Continue to Be Slow in 2015. Retrieved February 13, 2017, from http://www.gartner.com/newsroom/id/2954317

Egham. (2015b). Gartner Says Worldwide Device Shipments to Grow 1.5 Percent, to Reach 2.5 Billion Units in 2015. Retrieved February 13, 2017, from http://www.gartner.com/newsroom/id/3088221

Egham. (2015c). Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2015. Retrieved February 14, 2017, from http://www.gartner.com/newsroom/id/3215217

Egham. (2016). Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016. Retrieved February 13, 2017, from http://www.gartner.com/newsroom/id/3323017

Egham. (2017). Gartner Says Worldwide Sales of Smartphones Grew 9 Percent in First Quarter of 2017. Retrieved October 9, 2017, from http://www.gartner.com/newsroom/id/3725117

Elish, K. O., Shu, X., Yao, D. (Daphne), Ryder, B. G., & Jiang, X. (2015). Profiling user-trigger dependence for Android malware detection. *Computers & Security*, *49*(540), 255–273.

Elshoush, H. T., & Osman, I. M. (2011). Alert correlation in collaborative intelligent intrusion detection systems - A survey. *Applied Soft Computing Journal*, *11*(7), 4349–4365.

Fang, Z., Han, W., & Li, Y. (2014). Permission based Android security : Issues and countermeasures. *Computers & Security*, *43*(0), 205–218.

Faruki, P., Ganmoor, V., Laxmi, V., Gaur, M. S., & Bharmal, A. (2013). AndroSimilar : Robust Statistical Feature Signature for Android Malware Detection. *In Proceedings of the 6th International Conference on Security of Information and Networks, ACM*, 152–159.

Faruki, P., Laxmi, V., Bharmal, A., Gaur, M. S., & Ganmoor, V. (2014). AndroSimilar: Robust signature for detecting variants of Android malware. *Journal of Information Security and Applications*, *22*, 66–80.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*, 861–874.

Feizollah, A., Anuar, N. B., Salleh, R., Amalina, F., Ma'arof, R. R., & Shamshirband, S. (2013). A study of machine learning classifiers for anomaly-based mobile botnet detection. *Malaysian Journal of Computer Science*, *26*(4), 251–265.

Feizollah, A., Anuar, N. B., Salleh, R., Suarez-Tangil, G., & Furnell, S. (2017). AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection. *Computers and Security*, *65*, 121–134.

Feizollah, A., Anuar, N. B., Salleh, R., Wahid, A., & Wahab, A. (2015). A Review on

Feature Selection in Mobile Malware Detection. *Digital Investigation*, *3*, 22–37.

Feizollah, A., Shamshirband, S., Anuar, N. B., Salleh, R., & Kiah, M. L. M. (2013). Anomaly Detection Using Cooperative Fuzzy Logic Controller. *In Intelligent Robotics Systems: Inspiring the NEXT*, 220–231.

Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D. (2012). Android Permissions : User Attention , Comprehension , and Behavior. *Symposium on Usable Privacy and Security (SOUPS) 2012*.

Firdaus, A., Anuar, N. B., Karim, A., & Razak, M. F. A. (2017). Discovering optimal features using static analysis and genetic search based method for android malware detection. *Frontiers of Information Technology & Electronic Engineering*, 1–27.

Firdaus, A., Anuar, N. B., Razak, M. F. A., & Sangaiah, A. K. (2017). Bio-inspired computational paradigm for feature investigation and malware detection: interactive analytics. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-017-4586-0

Fox-Brewster, T. (2017). Google Is Fighting A Massive Android Malware Outbreak -- Up To 21 Million Victims. Retrieved October 7, 2017, from https://www.forbes.com/sites/thomasbrewster/2017/09/14/massive-google-android-malware-expensivewall/#60594e79477f

Gandotra, E., Bansal, D., & Sofat, S. (2014). Malware Analysis and Classification : A Survey. *Journal of Information Security*, *5*, 56–64.

Gaviria, J., Puerta, D., Sanz, B., Grueiro, I. S., & Bringas, P. G. (2013). The Evolution of Permission as Feature for AndroidMalware Detection. *International Joint Conference, Advances in Intelligent Systems and Computing*, 761.

Gheorghe, L., Marin, B., Gibson, G., Mogosanu, L., Deaconescu, R., Voiculescu, V.-G., & Carabas, M. (2015). Smart malware detection on Android. *Security and Communication Networks*, *8*(18), 4254–4272.

Ghiasi, M., Sami, A., & Salehi, Z. (2015). Dynamic VSA: a framework for malware detection based on register contents. *Engineering Applications of Artificial Intelligence*, *44*(0), 111–122.

Gonzalez, H., Stakhanova, N., & Ghorbani, A. A. (2014). DroidKin : Lightweight Detection of Android Apps Similarity. *Proceedings of the 10th SECURECOMM*.

Google. (2017). Google Play. Retrieved March 6, 2017, from https://play.google.com/store?hl=en

Grace, M. C., Zhou, W., Jiang, X., & Sadeghi, A.-R. (2012). Unsafe exposure analysis of mobile in-app advertisements. *Proc. 5th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, *67*(2), 101–112.

Gritzalis, D., Iseppi, G., Mylonas, A., & Stavrou, V. (2018). Exiting the Risk Assessment Maze : A Meta-Survey. *ACM Computing Surveys*, *51*(1), 1–30.

Haq, N. F. (2015). Application of Machine Learning Approaches in Intrusion Detection System : A Survey. *International Journal of Advanced Research in Artificial Intelligence (IJARAI)*, *4*(3), 9–18.

Houmansadr, A., Zonouz, S. a., & Berthier, R. (2011). A cloud-based intrusion detection and response system for mobile phones. *2011 IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 31–32.

Huang, H.-D., Lee, C.-S., Wang, M.-H., & Kao, H.-Y. (2014). IT2FS-based ontology with soft-computing mechanism for malware behavior analysis. *Soft Computing*, *18*(2), 267–284.

Hubballi, N., & Suryanarayanan, V. (2014). False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, *49*, 1–17.

IBM. (2017). IBM SPSS Statistics. Retrieved June 6, 2017, from https://www.ibm.com/us-en/marketplace/spss-statistics/details#product-header-top

IDC. (2017). Smartphone OS Market Share, 2016 Q3. Retrieved February 13, 2017, from http://www.idc.com/promo/smartphone-market-share/os

Idrees, F., Rajarajan, M., Conti, M., Chen, T. M., & Rahulamathavan, Y. (2017). PIndroid : A novel Android malware detection system using ensemble learning methods. *Computers & Security*, *68*, 36–46.

Inayat, Z., Gani, A., Anuar, N. B., Khan, M. K., & Anwar, S. (2016). Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications*.

International Data Corporation (IDC). (2016). Worldwide Smartphone Growth Forecast to Slow to 3.1% in 2016 as Focus Shifts to Device Lifecycles, According to IDC. Retrieved October 12, 2016, from http://www.idc.com/getdoc.jsp?containerId=prUS41425416

Jackson, J. (2017). New research reveals that 30 percent of malware attacks are zero day exploits. Retrieved November 14, 2017, from

http://www.itsecurityguru.org/2017/03/30/new-research-reveals-30-percent-malware-attacks-zero-day-exploits/

Karim, A. (2016). On the Analysis and Detection of Mobile Botnet. *Journal of Universal Computer Science*, *22*(4), 567–588.

Karim, A., Salleh, R. Bin, Shiraz, M., Shah, S. A. A., Awan, I., & Anuar, N. B. (2014). Botnet detection techniques: review, future trends, and issues. *Journal of Zhejiang University SCIENCE C*, *15*(11), 943–983.

Kaur, P., Singh, M., & Josan, G. S. (2015). Classification and Prediction Based Data Mining Algorithms to Predict Slow Learners in Education Sector. *Procedia Computer Science*, *57*, 500–508.

Kellex. (2016). Android Distribution Updated for May 2016 – Marshmallow Climbs to 7.5%. Retrieved May 3, 2016, from http://www.droid-life.com/2016/05/03/android-distribution-updated-may-2016/

Khalil, N., Kamaruzzaman, S. N., & Baharum, M. R. (2016). Ranking the indicators of building performance and the users' risk via Analytical Hierarchy Process (AHP): Case of Malaysia. *Ecological Indicators*, *71*, 567–576.

Kim, D. W., Yan, P., & Zhang, J. (2015). Detecting fake anti-virus software distribution webpages. *Computers & Security*, *49*, 95–106.

Kim, H., Cho, T., Ahn, G.-J., & Hyun Yi, J. (2017). Risk assessment of mobile applications based on machine learned malware dataset. *Multimedia Tools and Applications*, 1–16.

Kumar, V., & Minz, S. (2014). Feature Selection: A literature Review. *Smart Computing Review*, *4*(3), 211–229.

Lab, K. (2017). Android Mobile Security Threats. Retrieved October 9, 2017, from https://www.kaspersky.com/resource-center/threats/mobile

Lar, S.-U. (2011). Proactive Security Mechanism and Design for Firewall. *Journal of Information Security*, *2*(3), 122–131.

Ledermüller, T., & Clarke, N. L. (2011). Risk Assessment for Mobile Devices. *In Trust, Privacy and Security in Digital Business*, 210–221.

Lee, S., Lee, J., & Lee, H. (2015). Screening Smartphone Applications Using Malware Family Signature. *Computers & Security*, 1–31.

Liam Tung. (2017). Android malware in Google Play racked up 4.2M downloads: Are you a victim? Retrieved March 6, 2018, from http://www.zdnet.com/article/android-malware-in-google-play-racked-up-4-2-million-downloads-so-are-you-a-victim/

Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C., & Tung, K.-Y. (2012). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16–24.

Lin, C.-T., Wang, N.-J., Xiao, H., & Eckert, C. (2015). Feature Selection and Extraction for Malware Classification. *Journal of Information Science and Engineering*, *31*, 965–992.

Lo, C. C., & Chen, W. J. (2012). A hybrid information security risk assessment procedure considering interdependences between controls. *Expert Systems with Applications*, *39*(1), 247–257.

Lookout. (2012). Lookout Mobile Endpoint Security. *PC Magazine*, 178. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=a9h&AN=85818542&amp%0Alang=ja&site=ehost-live

Lopez, M. (2015). PandaLabs. Retrieved November 25, 2015, from http://www.pandasecurity.com/mediacenter/press-releases/pandalabs-neutralized-75-million-new-malware-samples-2014-twice-many-2013/

Lueg, C. (2017). 8,400 new Android malware samples every day. Retrieved October 9, 2017, from https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day

McAfee. (2016). *Mobile Threat Report: What's on the Horizon for 2016*. Retrieved from http://www.mcafee.com/us/resources/reports/rp-mobile-threat-report-2016.pdf

McWilliams, G., Sezer, S., & Yerima, S. Y. (2014). Analysis of Bayesian classification-based approaches for Android malware detection. *IET Information Security*, *8*(1), 25–36.

Model, H., Challenges, E. I., ShahidFarid, Ahmad, R., & MujahidAlam. (2015). A Hierarchical Model for E-learning Implementation Challenges using AHP. *Malaysian Journal of Computer Science*, *28*(3), 166–188.

Muncaster, P. (2017). Firms Face £18 Million Bill for Mobile Data Breaches. Retrieved October 9, 2017, from https://www.infosecurity-magazine.com/news/firms-face-18-million-bill-for/

Nadeem, A., & Howarth, M. P. (2014). An intrusion detection & adaptive response

mechanism for MANETs. *Ad Hoc Networks*, *13*(PART B), 368–380.

Nadiammai, G. V., & Hemalatha, M. (2014). Effective approach toward Intrusion Detection System using data mining techniques. *Egyptian Informatics Journal*, *15*(1), 37–50.

Narudin, F. A., Feizollah, A., Anuar, N. B., & Gani, A. (2016). Evaluation of machine learning classifiers for mobile malware detection. *Soft Computing*, *20*(1), 343–357.

Nikou, S., & Mezei, J. (2013). Evaluation of mobile services and substantial adoption factors with Analytic Hierarchy Process (AHP). *Telecommunications Policy*, *37*(10), 915–929.

Nikou, S., Mezei, J., & Bouwman, H. (2011). Analytic Hierarchy Process (AHP) approach for selecting mobile service category: (Consumers' preferences). *Proceedings - 2011 10th International Conference on Mobile Business, ICMB 2011*, 119–128.

Nokia. (2017). Nokia malware report reveals new all-time high in mobile device infections and major IoT device security vulnerabilities. Retrieved April 20, 2017, from https://www.nokia.com/en_int/news/releases/2017/03/27/nokia-malware-report-reveals-new-all-time-high-in-mobile-device-infections-and-major-iot-device-security-vulnerabilities

O'Shea, D. (2017). Forrester: 5.5B mobile users globally by 2022. Retrieved October 9, 2017, from http://www.retaildive.com/news/forrester-55b-mobile-users-globally-by-2022/447283/

Oberheide, J., & Miller, C. (2012). Dissecting the Android Bouncer. Retrieved September 5, 2017, from https://jon.oberheide.org/files/summercon12-bouncer.pdf

Olson, D. L. (1996). The analytic Hierarchy process. *Decision Aids for Selection Problems*, 49–68.

Opydo, D. (2013). 6 Reasons to Use Analytic Hierarchy Process for Collaborative Decision Making. Retrieved March 5, 2018, from https://blog.transparentchoice.com/analytic-hierarchy-process/6-reasons-to-use-ahp-for-collaborative-decision-making

Patel, R., Thakkar, A., & Ganatra, A. (2012). A Survey and Comparative Analysis of Data Mining Techniques for Network Intrusion Detection Systems. *International Journal of Soft Computing ...*, *2*(1), 265–271. Retrieved from http://www.ijsce.org/attachments/File/v2i1/A0432022112.pdf

Patri, O. P., Wojnowicz, M. T., & Wolff, M. (2017). Discovering Malware with Time Series Shapelets Abstract. *ACM Conference on Computer and Communications Security*, *4*(1), 229–240.

Platforms, N., & Threats, C. (2013). *Security Threat Report 2013*. *SOPHOS*.

Quintaro, B. (2017). VirusTotal. Retrieved April 11, 2017, from https://www.virustotal.com/

Rad, B. B., Masrom, M., & Ibrahim, S. (2012). Camouflage in Malware : from Encryption to Metamorphism. *International Journal of Computer Science and Network Security*, *12*(8), 74–83.

Rashidi, B., Fung, C., & Bertino, E. (2017). Android resource usage risk assessment using hidden Markov model and online learning. *Computers and Security*, *65*, 90–107.

Ravula, R. R., Liszka, K. J., & Chan, C. (2013). Learning Attack Features from Static and Dynamic Analysis of Malware. *Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 109–125.

Razak, M. F. A., Anuar, N. B., Othman, F., Firdaus, A., Afifi, F., & Salleh, R. (2017). Bio-inspired for Features Optimization and Malware Detection. *Arabian Journal for Science and Engineering*.

Razak, M. F. A., Anuar, N. B., Salleh, R., & Firdaus, A. (2016). The rise of "malware": Bibliometric analysis of malware study. *Journal of Network and Computer Applications*, *75*, 58–76.

Rhiannon Williams. (2016). Android roars back in strongest growth in two years, as iOS shrinks. Retrieved February 13, 2017, from http://www.telegraph.co.uk/technology/2016/05/17/android-roars-back-in-strongest-growth-in-two-years-as-apple-shr/

Saaty, T. L. (2008). Decision making with the analytic hierarchy process. *International Journal of Services Sciences*, *1*(1), 83. https://doi.org/10.1504/IJSSCI.2008.017590

Sahs, J., & Khan, L. (2012). A Machine Learning Approach to Android Malware Detection. *2012 European Intelligence and Security Informatics Conference*, 141–147.

Santos, I., Brezo, F., Ugarte-Pedrero, X., & Bringas, P. G. (2013). Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences*, *231*, 64–82.

Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2016). MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention. *IEEE Transactions on Dependable and Secure Computing*, 1–14.

Schmeelk, S., Yang, J., & Aho, A. (2015). Android Malware Static Analysis Techniques. *Proceedings of the 10th Annual Cyber and Information Security Research Conference.*

Seideman, J. D., Khan, B., & Vargas, C. (2015). Quantifying Malware Evolution through Archaeology. *Journal of Information Security*, *6*, 101–110.

Seo, S.-H., Gupta, A., Mohamed Sallam, A., Bertino, E., & Yim, K. (2014). Detecting mobile malware threats to homeland security through static analysis. *Journal of Network and Computer Applications*, *38*, 43–53.

Shabtai, A., & Elovici, Y. (2010). Applying behavioral detection on android-based devices. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, *48 LNICST*, 235–249.

Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., & Weiss, Y. (2012). "Andromaly": a behavioral malware detection framework for android devices. *Journal of Intelligent Information Systems*, *38*(1), 161–190.

Shabtai, A., Mimran, D., Rokach, L., Shapira, B., & Elovici, Y. (2014). Mobile malware detection through analysis of deviations in application network behavior. *Computers & Security*, *43*, 1–18.

Shameli-Sendi, A., Cheriet, M., & Hamou-Lhadj, A. (2014). Taxonomy of intrusion risk assessment and response system. *Computers and Security*, *45*, 1–16.

Sharif, M., Lanzi, A., Giffin, J., & Lee, W. (2008). Impeding Malware Analysis Using Conditional Code Obfuscation. *Informatica*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.126.9256&rep=rep1&type=pdf

Sharma, K., & Lin, K.-I. (2013). Review spam detector with rating consistency check. *Proceedings of the 51st ACM Southeast Conference on - ACMSE '13*, 1.

Sheen, S., Anitha, R., & Natarajan, V. (2015). Android based malware detection using a multifeature collaborative decision fusion approach. *Neurocomputing*, *151*, 905–912.

Skycure Mobile Threat Defense. (2016). *Mobile Threat Intelligence Report*. Retrieved from https://cg9j53d64gz46qncx41jvxq16p-wpengine.netdna-ssl.com/wp-

content/uploads/2016/06/Skycure-Q1-2016-MobileThreatIntelligenceReport.pdf

Somarriba, O., Zurutuza, U., Uribeetxeberria, R., Delosières, L., & Nadjm-tehrani, S. (2016). Detection and Visualization of Android Malware Behavior. *Journal of Electrical and Computer Engineering*, *2016*(i), 1–17.

Song, J., Han, C., Wang, K., Zhao, J., & Ranjan, R. (2016). An integrated static detection and analysis framework for android. *Pervasive and Mobile Computing*, *32*, 15–25.

Statista. (2017). Global smartphone sales by operating system from 2009 to 2015 (in millions). Retrieved February 14, 2017, from https://www.statista.com/statistics/263445/global-smartphone-sales-by-operating-system-since-2009/

Su, M. Y. (2011). Using clustering to improve the KNN-based classifiers for online anomaly network traffic identification. *Journal of Network and Computer Applications*, *34*(2), 722–730.

Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., & Ribagorda, A. (2014). Evolution, detection and analysis of malware for smart devices. *IEEE Communications Surveys and Tutorials*, *16*(2), 961–987.

Suleiman Y. Yerima, S. S., & Muttik, I. (2015). High accuracy android malware detection using ensemble learning. *IET Information Security*, *9*(6), 313–320.

Symantec. (2015). *2015 Internet Security Threat Report*. *Internet Security Threat Report* (Vol. 20). Retrieved from https://www4.symantec.com/mktginfo/whitepaper/ISTR/21347932_GA-internet-security-threat-report-volume-20-2015-social_v2.pdf

Tahaei, H., Salleh, R., Razak, M. F. A., Ko, K., & Anuar, N. B. (2018). Cost Effective Network Flow Measurement for Software Defined Networks: A Distributed Controller Scenario. *IEEE Access*, *6*, 5182–5198.

Talha, K. A., Alper, D. I., & Aydin, C. (2015). APK Auditor: Permission-based Android malware detection system. *Digital Investigation*, *13*, 1–14.

Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017). The Evolution of Android Malware and Android Analysis Techniques. *ACM Computing Surveys*, *49*(4), 1–41.

Tanaka, Y., & Goto, A. (2016). Analysis of malware download sites by focusing on time series variation of malware. *Proceedings - IEEE Symposium on Computers and Communications*, *2016–Augus*, 173–179.

Tang, A., Sethumadhavan, S., & Stolfo, S. (2014). Unsupervised Anomaly-based Malware Detection using Hardware Features. *Research in Attacks, Intrusions and Defenses*, 109–129. Retrieved from http://arxiv.org/abs/1403.1631

Tchakounte, F. (2014). Permission-based Malware Detection Mechanisms on Android : Analysis and Perspectives. *Journal of Computer Science and Software Application*, *1*(2), 63–77.

Thanki, S., Govindan, K., & Thakkar, J. (2016). An investigation on lean-green implementation practices in Indian SMEs using analytical hierarchy process (AHP) approach. *Journal of Cleaner Production*, *135*, 284–298.

Theoharidou, M., Mylonas, A., & Gritzalis, D. (2012). A Risk Assessment Method for Smartphones. *In Information Security and Privacy Research*, *376*, 443–456.

Total, V. (2016). Analyzes suspicious files and URLs. Retrieved September 8, 2016, from https://www.virustotal.com/en/#dlg-join

Trend Micro. (2017). Android Mobile Ransomware: Bigger, Badder, Better? Retrieved October 7, 2017, from http://blog.trendmicro.com/trendlabs-security-intelligence/android-mobile-ransomware-evolution/

Unuchek, R., Sinitsyn, F., Parinov, D., & Liskin, A. (2017). IT threat evolution Q2 2017. Statistics. Retrieved October 11, 2017, from https://securelist.com/it-threat-evolution-q2-2017-statistics/79432/

Vanja Svajcer. (2011). Plankton malware drifts into Android Market. Retrieved September 21, 2016, from https://nakedsecurity.sophos.com/2011/06/14/plankton-malware-drifts-into-android-market/

Veerwal, D., & Menaria, P. (2013). Ensemble of Soft Computing Techniques for Malware detection. *International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)*, *6*(2), 159–167.

Waikato, U. of. (2017). Waikato Environment for Knowledge Analysis (Weka). Retrieved March 8, 2017, from http://www.cs.waikato.ac.nz/ml/weka/

Wang, P., & Wang, Y.-S. (2014). Malware behavioural detection and vaccine development by using a support vector model classifier. *Journal of Computer and System Sciences*, *1*(Ml), 1–15.

Wang, X., Yang, Y., Zeng, Y., Tang, C., Shi, J., & Xu, K. (2015). A Novel Hybrid Mobile Malware Detection System Integrating Anomaly Detection With Misuse Detection. *Proceedings of the 6th International Workshop on Mobile Cloud Computing and*

Services, 15–22.

Weiss, Y., Fledel, Y., Elovici, Y., & Rokach, L. (2012). Cost-Sensitive Detection of Malicious Applications in Mobile Devices. *Mobile Computing, Applications, and Services*, 382–395.

Westenberg, J. (2015). Facebook now allowing Google to index its mobile app. Retrieved April 4, 2017, from http://www.androidauthority.com/facebook-allowing-google-to-index-its-mobile-app-655958/

Wiśniewski, R. (2017). Apktool. Retrieved March 8, 2017, from https://ibotpeaches.github.io/Apktool/

Wu, D.-J., Mao, C.-H., Wei, T.-E., Lee, H.-M., & Wu, K.-P. (2012). DroidMat: Android Malware Detection through Manifest and API Calls Tracing. *2012 Seventh Asia Joint Conference on Information Security*, 62–69.

Wu, F., Narang, H., & Clarke, D. (2014). An Overview of Mobile Malware and Solutions. *Journal of Computer and Communications*, *2*(2), 8–17.

Xie, P., Lu, X., Wang, Y., Su, J., & Li, M. (2013). An Automatic Approach to Detect Anti-debugging in Malware Analysis. *Trustworthy Computing and Services*, 436–442.

Yassin, W., Udzir, N. I., Muda, Z., Abdullah, a., & Abdullah, M. T. (2012). A Cloud-based Intrusion Detection Service framework. *Proceedings Title: 2012 International Conference on Cyber Security, Cyber Warfare and Digital Forensic (CyberSec)*, 213–218.

Ye, L., & Keogh, E. (2011). Time series shapelets: A novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, *22*(1–2), 149–182.

Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik, I. (2013). A New Android Malware Detection Approach Using Bayesian Classification. *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, 121–128.

Yuan, Z., Lu, Y., & Xue, Y. (2016). DroidDetector : Android Malware Characterization and Detection Using Deep Learning. *Tsinghua Science and Technology*, *21*(1), 114–123.

Zhang, Y., Lee, W., & Huang, Y.-A. (2003). Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, *9*, 545–556.

Zhao, M., Zhang, T., Ge, F., & Yuan, Z. (2012). RobotDroid : A Lightweight Malware Detection Framework on Smartphones. *Journal of Networks*, *7*(4), 715–722.

Zhou, Y., & Jiang, X. (2012). Dissecting Android Malware: Characterization and Evolution. *2012 IEEE Symposium on Security and Privacy*, (4), 95–109.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

i. Razak, M. F. A., Anuar, N. B, Salleh, R., & Firdaus, A. (2016). The rise of "malware": Bibliometric analysis of malware study. Journal of Network and Computer Applications, 75, 58–76.

ii. Razak, M. F. A, Anuar, N. B, Othman, F, Firdaus, A, Afifi, F, & Salleh, R. (2017). Bio-inspired for Features Optimization and Malware Detection. Arabian Journal Sciences and Engineering, 1 –19.

iii. Firdaus, A., Anuar, N. B., Karim, A., & Razak, M. F. A. (2017). Discovering optimal features using static analysis and genetic search based method for android malware detection. Frontiers of Information Technology & Electronic Engineering, 1–27.

iv. Firdaus, A., Anuar, N. B. N. B., Razak, M. F. A. M. F. A., & Sangaiah, A. K. A. K. (2017). Bio-inspired computational paradigm for feature investigation and malware detection: interactive analytics. Multimedia Tools and Applications.

v. Tahaei, H., Salleh, R., Razak, M. F. A., Ko, K., & Anuar, N. B. (2018). Cost Effective Network Flow Measurement for Software Defined Networks: A Distributed Controller Scenario. IEEE Access, 6, 5182–5198.

vi. Firdaus, A., Anuar, N.B., Razak, M.F.A., Hashem, I.A.T., Bachok, S., Sangaiah, A.K., 2018. Root Exploit Detection and Features Optimization: Mobile Device and Blockchain Based Medical Data Management. Journal of Medical Systems 42.

vii. Hazim, M., Anuar, N.B., Ab Razak, M.F., Abdullah, N.A., 2018. Detecting opinion spams through supervised boosting approach. PLoS ONE 13, 1–23