SUDAN UNIVERSITY OF SCIENCE & TECHNOLOGY

COLLEGE OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

COMPUTER SCIENCE DEPARTMENT

USE STATIC ANALYSIS TO PROTECT USERS DATA FROM MALICIOUS ANDROID APPLICATIONS

THE DISSERTATION SUBMITTED AS A PARTIAL FULFILLMENT FOR THE REQUIREMENT OF BSC (HONOUR) DEGREE IN COMPUTER SYSTEMS AND NETWORKS

October 2015

بسم الله الرحمن الرحيم

SUDAN UNIVERSITY OF SCIENCE & TECHNOLOGY

COLLEGE OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY

COMPUTER SCIENCE DEPARTMENT

USE STATIC ANALYSIS TO PROTECT USERS DATA FROM MALICIOUS ANDROID APPLICATIONS

October 2015

PREPARED BY:

Mohamed Emadaldin Satti

Muhannad El Tayeb Musa

Suheil Ali Abdalmounim

Supervisor by:

T.Mohamed Osama

Signiture……...

الآية

لِإِيلَافِ قُرَيْشٍ ﴿١﴾ إِيلَافِهِمْ رِحْلَةَ الشِّتَاءِ

وَالصَّيْفِ ﴿٢﴾ فَلْيَعْبُدُوا رَبَّ هَذَا الْبَيْتِ ﴿٣﴾ الَّذِي

أَطْعَمَهُمْ مِنْ جُوعٍ وَآمَنَهُمْ مِنْ خَوْفٍ ﴿٤﴾

صدق الله العظيم

قريش الاية (1-4)

الحمد لله

الحمد لله على سابغ نعمه وعظيم إحسانه ومننه لا نحصي ثناءً عليه هو كما أثنى على نفسه .. فلك الحمد اللهم ربنا كما خلقتنا ورزقتنا وهديتنا وعلمتنا وأنقذتنا وفرجت عنا لك الحمد بالإسلام والقرآن ولك الحمد بالأهل والمال والمعافاة كبت عدونا وبسطت رزقنا وأظهرت أمننا وجمعت فرقتنا ورددت غائبنا وفككت أسرانا وشفيت مرضانا وأحسنت معافاتنا ومن كل ما سألناك ربنا أعطيتنا , لك الحمد والشكر والثناء الحسن بكل نعمة أنعمت بها علينا في قديم أو حديث أو سر أو علانية أو خاصة أو عامة أو حي أو ميت أو شاهد أو غائب , فاللهم لك المحامد كلها كالذي نقول , لك الحمد حتى ترضى , ولك الحمد إذا رضيت , ولك الحمد بعد الرضى ملء السموات وملء الأرض وملء ما شئت من شيء بعده.

لك الحمد يا رب والشكر ثم ... لك الحمد ما باح بالشكر فمْ
لك الحمد في كل ما حالة ... فقد خصني منك فضل وعمْ
من الماء أنشأتني نطفة ... ومن بعد ذلك لحم ودمْ
وأسكنت في جسدي روحه ... وأجعلتها في طباق الرحمْ
وأخرجتني بعد في عالمي ... وبلغتني درجات الفِهمْ
فمنك لي البصر المقتفي ... وسمع وذوق ونطق وشمْ
وحس صحيح وتميز ما ... خلقت بأنواعه من أممْ
ومكنتني من فنون العلوم ... ببادي الكلام وخط القلمْ
وعلمتني الحكم في هل وما ... وأطلعتني طلع كيف ولمْ
وحد الحقائق ميزت لي ... من الباطل المتقى في الكلم

ببرهان صدق يليح اليقين ... وينفي المحال ويبدي الحكم
ويوفي بيان اسمه ... ويحتد بالوصف ما لم يسم

DEDICATION

For the one who hit the message and led the Secretariat advised the nation to the Prophet of mercy and the light of the Worlds
Prophet Muhammad peace be upon him

To the one who teach me tender without waiting, to carry his name proudly, I ask God to reach at your age to see the fruit picking has come after a long wait and your star will remain guided by today and tomorrow and forever
to our dear fathers

To the greatest unconditional and infinite love we will ever experience in our existence...
To our dear mothers

To our brothers and sisters: For their unconditional love, faith, Understanding, and support**.**

ACKNOWLEDGEMENT

Praise be to God first and last that gave us strength, health and patience to complete this work truly without his grace nothing is achievable.
Greater appreciation and gratitude for the man who put full confidence in us to complete this research.
Our supervisor **T.MOHAMED OSAMA HWAIT ALLAH**
Also thanks and respect for those who did not hesitate to help us and made their efforts in order to achieve our desires:

- **T.ZEINAB YASSEN.**
- **T.ISMAIL ALI IBRAHIM.**
- **T .DALIA MAHMOUD ALSIR.**

Thanks to every teacher who has taught us throughout our college years and got us.
To where we are today.
Thanks to all our college in the eighth batch, we are honored to have studied with such a batch for everyone know our name.

Abstract

There is no doubt about the massive technology nowadays that affect our everyday life. One of these fronts is the rapid evolution within smart phones and their applications which simplifies our lives and our daily tasks.

By August 2013 there were 900,000 application in Google’s Play Store for android devices, which shows the wide spread of smart phone apps.
This wide spreading of apps leads to the creation of numerous stores that allow apps downloading, most of these stores don’t have the systems to test these apps which results in stores being filled with malicious software.
This System/App is providing an appropriate solution for this type of problems mentioned before, we illustrates the creation of a system that includes two parts, and the first is the client which runs on the android smart phone. The second is a system that tests apps and classifies them as safe, warning or dangerous. After the executing and testing of the system it becomes possible to download apps from all available stores based on results, therefore then the system users can decide whether to install the application or not.
.

المستخلص

ممـا لا شـك فيـه و مـا لا يخفـى علـى أي أحـد منـا التطـور التكنولـوجي المتسارع الذي يشهده العالم اليوم في شـتى منـاحي الحيـاة. و مـن هـذه المناحي، التطور السريع في الهواتـف الـذكيه و البرمجيـات الـتي تسـهل علينا انجاز مهمامنا اليوميه.
حتى اغسطس 2013 كان هنالـك حـوالي 900,000 تطـبيق فـي متجـر جوجل بلاي وهو متجر على الويب للبرامج تـديره جوجـل لأجهـزة أندرويـد ,مما يعني مدى انتشار تطبيقات الهواتف الذكيه.

هـذا الازديـاد المسـتمر فـي عـدد التطبيقـات ادى الـى ظهـور العديـد مـن المتاجر التي يمكن من خلالها تحميـل التطبيقـات .معظـم هـذا المتـاجر لا يوجـد فيهـا نظـم لاختبـار التطبيقـات ممـا يـؤدي الـى وجـود العديـد مـن التطبيقات الضاره بالمستخدمين .

اقترحنا لحل هذه المشكلة إنشاء نظام من جزئين الجزء الأول و هـو جـزء العميل و هو عبارة عن تطـبيق يعمـل علـى أجهـزة الهواتـف الذكيـة الـتي تستخدم نظام التشغيل أندرويد. والجزء الثاني و هو عبارة عن نظام يقوم بإختبار التطبيقات و من ثم تصنيفها الى ثلات قيم توضح حـالات التطـبيق وهي : آمنة وتحذيرية وخطرة .

بعد تطبيق النظام واختباره اصبح من الممكن للمستخدم ان يقوم بتحميل التطبيقات من جميع المتاجر المتاحـة والتأكـد مـن سـلامة التطـبيق قبـل تثبيته,واعتمـادا علـى النتيجـة المسـتخرجة مـن النظـام امكـن للمسـتخدم باتخاذ القرار بتثبيت التطبيق او عدم تثبيته.

TABLE OF TERMS:

Term	Description
API	Application program interface

APK	Android Package
Centos	Community Enterprise Operating System
DEX	Dalvik Executable
GPS	Global Positioning System
GID	Group Identity
IDE	Integrated development environment
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IOS	Internetwork Operating System
IPC	Inter-process communication

JAD	Java DE compiler
JAR	Java Archive
OEM	Original equipment manufacturer
PAE	Physical Address Extension
RHEL	Red Hat Enterprise Linux
RSH	Remote shell
SElinux	Security-Enhanced Linux
SSH	Secure Shell
UID	User Identity

TABLE OF TABLES:

TABLES OF FIGUERS:

TABLE OF CONTENT:

CHAPTER ONE

INTRODUCTION AND BACKGROUND TO THE STUDY

1.1 Introduction:

Android is a modern mobile platform that was designed to be truly open. Android Applications make use of advanced hardware and software, as well as local and served Data, exposed through the platform to bring innovation and value to consumers. To protect that value, the platform must offer an application environment that ensures the Security of users, data, applications, the device, and the network. Android has become an integral part of daily utilization with the presence of millions of users and Provides several sources for programs and products, the various Android stores. [1]

1.2 Background to the study problem:

Applications on the stores are leaking some users' information without their knowledge, there are many of these applications, including the well-known and used by most people. These applications take the higher powers to get the data and use this data in other processes. For example, some applications look for private bank accounts and sell these accounts on the black market "deep Web" and other sensitive information.

1.3 Study questions:

The study sought to find answers to the following questions:

Such as know the purpose of the use of the data required by the application? Determine if the application contains any vulnerabilities that can be exploited by adversaries? And detect malicious behavior in Android Applications be using static analysis.

1.4 Objectives of the study:

- The main objective of this study is to protect user's data from malicious software that could violate their privacy.

- Another objective of this project is to help users download applications from any android markets and check the applications for the presence of unauthorized information leaking.

1.5 Research Methodology:

The descriptive and analytical approach was used to describe the way that system works and the characterization and analysis of the system and then do the construction. The methodology that was used in the following manner: Develop an android application that scans applications that have been downloaded from the store before installed on devices using Java programming language, integrated development environment (Eclipse) and server conducted by testing and analysis operations.

1.6 Research Scope:

The main afford of this project scan applications that have been downloaded from the Android stores before installing it on the phone and tested whether its contain malicious code leads to the threat of user data, based on the number of operations in the way to determine whether the application is safe or not to the user through this application to ensure the confidentiality and integrity of its data from the threat of malicious codes on the Android stores.

1.7 Research Structure:

This research shows in chapter two theoretical background, chapter three illustrate the previous studies in first section and tools and techniques used in solving the problem of the research in the second section, chapter four addresses the description of the proposed system and its operations, chapter five deals with the implementation of the system, and displays screens embedded in the system along with a brief description of each, and chapter six reviews the results Conclusion and recommendations.

CHAPTER TWO
BACKROUND THEORY

2.1 Introduction:

This chapter illustrate backgrounds theory to the research , it discuss the android platform and explain the sandbox security mechanism for separating running programs , clarification the security risk and security architecture in mobile devices , static analysis concept, access permissions in Linux and Android .

2.2 android market:

It is an application that allows users to download application of the android OS, and it allows them firstly to search for application, then they can download and install it on their devices. Finally, they can update these applications. The official Android Market provided by Google is called (Google Play), there are also (Samsung Apps) by Samsung, and (Amazon app Store) and (Getiar) in America. Users can use the Android Market app that they want from all existing android Markets we choose (Google Play). which is a digital distribution platform operated by Google .It serves as the official app store for Android operating system , allowing users to browse and download applications developed with the Android SDK and published through Google. Google play also serves as a digital media store, offering music, magazines, books, movies, and television programs. It previously online hardware devices for purchase until the introduction of a separate online hardware retailer, Google Store, on March 11, 2015 Applications are available through Google Play either free of charge or at a cost. [2]

2.3 Android Platform

Apps for Android are developed in Java and executed in a virtual machine, called Dalvik VM. They are supported by the application framework, which provides frequently used functionality through a unified interface. Various libraries enable application to implement graphics, encrypted communication or databases easily. The Standard Library (“bionic”) is a BSD derived libc for embedded devices. The respective Android releases’ kernels are stripped down from Linux 2.6 versions. Basic services such as memory, process and user management are all provided by the Linux kernel in a mostly unmodified form. However, for several Android versions, the deployed kernel’s version

was already out of date at the time of release. This has led to a strong increase in vulnerability, as exploits were long publicly available before the respective Android version’s release. [3]

The following (figure 1) illustrates the architectural Android operating system:

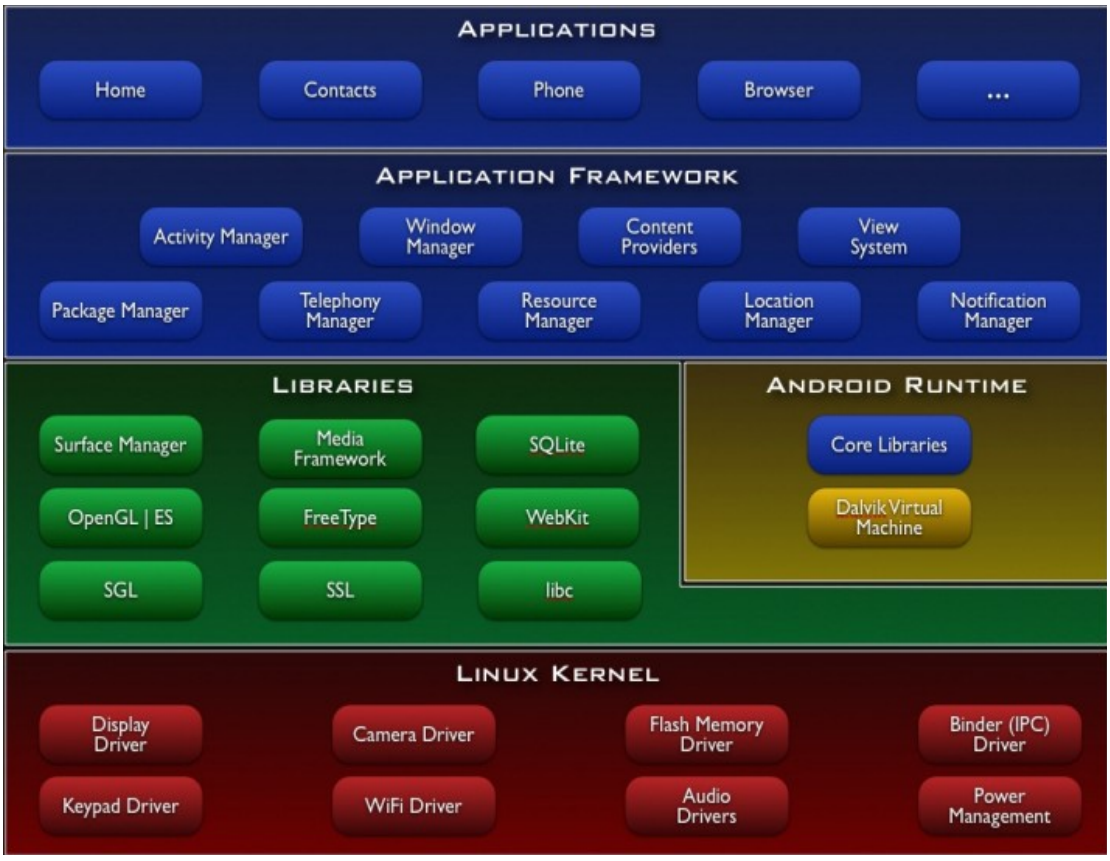

Figure (2.1): Android operating system architecture

2.4 Sandbox:

A sandbox is a security mechanism for separating running programs. It is often used to execute untested code, or untrusted programs from unverified third parties, suppliers, untrusted users and untrusted websites. [4] A sandbox typically provides a tightly controlled set of resources for guest programs to run in, such as scratch space on disk and memory. Network access, the ability to inspect the host system or read from input devices are usually disallowed or heavily restricted. In the sense of providing a highly controlled environment, sandboxes may be seen as a specific example of virtualization.

Sandboxing is frequently used to test unverified programs that may contain a virus or other malignant code, without allowing the software to harm the host device. [5]

2.5 Android application package (APK):

Android application package (APK) is the package file format used to distribute and install application software and middleware onto Google's Android operating system. Certain other operating systems and devices, such as BlackBerry devices with the operating system version 10 or higher, also support APK packages. APK files are analogous to other software packages such Dep packages in Debian-based operating systems like Ubuntu. To make an APK file, a program for Android is first compiled, and then all of its parts are packaged into one file. An APK file contains all of that program's code (such as .dex files), resources, assets, certificates, and manifest file. As is the case with many file formats, APK files can have any name needed, provided that the file name ends in ".apk". APK files are a type of archive file, specifically in zip format packages based on the JAR file format, with ".apk" as the filename extension. The MIME type associated with APK files is application/vnd.android.package-archive.[4]

2.6 Contents:

An APK file is an archive that usually contains the following file structure:

- **META-INF directory** :

- **MANIFEST.MF**: the Manifest file.
- **CERT.RSA**: The certificate of the application.
- **CERT.SF**: The list of resources and SHA-1 digest of the corresponding lines in the MANIFEST.MF file; for example:

```
Signature-Version: 1.0
Created-By: 1.0 (Android)
SHA1-Digest-Manifest: wxqnEAI0UA5nO5QJ8CGMwjkGGWE=
...
Name: res/layout/exchange_component_back_bottom.xml
SHA1-Digest: eACjMjESj7Zkf0cBFTZ0nqWrt7w=
...
Name: res/drawable-hdpi/icon.png
SHA1-Digest: DGEqylP8W0n0iV/ZzBx3MW0WGCA=
```

- **Lib** : the directory containing the compiled code that is specific to a software layer of a processor, the directory is split into more directories within it:
- **Armeabi**: compiled code for all ARM based processors only.
- **Armeabi-v7a**: compiled code for all ARMv7 and above based processors only.
- **X86**: compiled code for x86 processors only.
- **MIPS**: compiled code for MIPS processors only.

- **Res**: the directory containing resources not compiled into resources.arsc.
- **Assets**: a directory containing applications assets, which can be retrieved by AssetManager.
- AndroidManifest.xml: An additional Android manifest file, describing the name, version, access rights, referenced library files for the application.

- **classes.dex**: The classes compiled in the des file format understandable by the Dalvik virtual machine.
- **resources.arsc**: a file containing precompiled resources, such as binary XML for example. [6]

2.7 Security Risks:

The mobile device have some high risks rather than desktop computer why? First, the quality of data stored on a user's mobile device tends to be more personal. Apart from e-mail, there are instant messages, SMS/MMS, contacts, photos, and voice email. "So what?" you say. "Some of these things exist on a desktop computer." True, but consider this: The data on your mobile device is most likely going to be of higher value than that on your desktop because you carry it around with you all the time. It is a converged platform of both your computer and mobile phone that contains a richer collection of personal data.

Because the level of user interaction is higher on the smartphone, the data is always newer than on your desktop computer. Even if you have configured real-time sync to a remote location, that still only protects you from a loss of data and not a loss of privacy. [7]

2.8 Security Architecture:

Security Architecture to refer to a plan and set of principles that describe the security services that a system is required to provide to meet the needs of its users, the system elements required to implement the services, and also the performance levels required in the elements to deal with the threat environment. [8] The Figure summarizes the security components and considerations of the various levels of the Android software stack. Each component assumes that the components below are properly secured. With

the exception of a small amount of Android OS code running as root, all code above the Linux Kernel is restricted by the Application Sandbox.

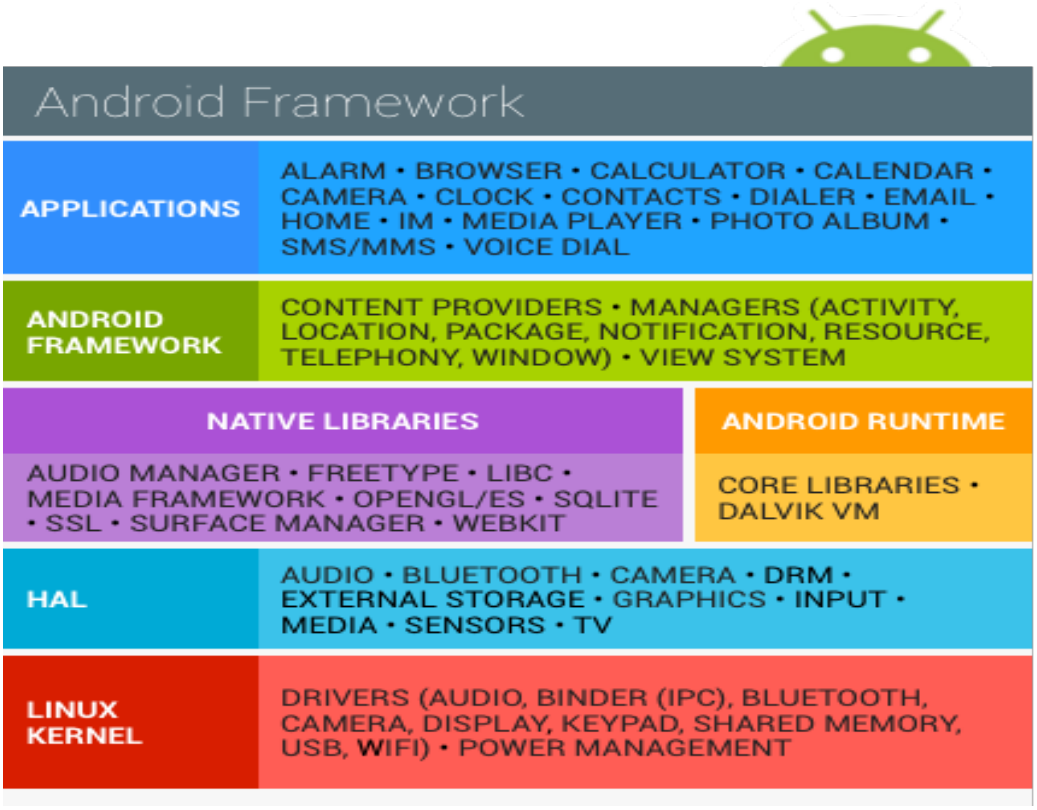

Figure (2.2): Android Security Architecture

2.8.1 Android Platform Security Architecture:

- Security at the OS through the Linux kernel
- Mandatory application sandbox
- Secure intercrosses communication
- Application signing
- Application-defined and user-granted permissions

2.9 Application Signing:

All APKs (.apk files) must be signed with a certificate whose private key is held by their developer. This certificate identifies the author of the application. The certificate does not need to be signed by a certificate authority; it is perfectly allowable, and typical, for Android applications to use self-signed certificates. The purpose of certificates in Android is to distinguish application authors. This allows the system to grant or deny applications access to signature-level permissions and to grant or deny an application's request to be given the same Linux identity as another application. [9]

2.10 Signature level permissions

Android: protection Level

Characterizes the potential risk implied in the permission and indicates the procedure the system should follow when determining whether or not to grant the permission to an application requesting it. The value can be set to one of the following strings. [10]

The default value. A lower-risk permission that gives

"normal"	requesting applications access to isolated application-level features, with minimal risk to other applications, the system, or the user. The system automatically grants this type of permission to a requesting application at installation, without asking for the user's explicit approval (though the user always has the option to review these permissions before installing).
"dangerous"	A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user. Because this type of permission introduces potential risk, the system may not automatically grant it to the requesting application. For example, any dangerous permissions requested by an application may be displayed to the user and require confirmation before proceeding, or some other approach may be taken to avoid the user automatically allowing the use of such facilities.
"signature"	A permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission. If the certificates match, the system automatically grants the permission without notifying the user or asking for the user's explicit approval.
"signatureOrSyste	A permission that the system grants only to applications that are in the Android system image or that are signed with the same certificate as the application that declared the permission. Please avoid using this option, as the signature protection

m"	level should be sufficient for most needs and works regardless of exactly where applications are installed. The "signatureOrSystem" permission is used for certain special situations where multiple vendors have applications built into a system image and need to share specific features explicitly because they are being built together.

2.11 Mandatory access control (MAC)

Mandatory access control (MAC) refers to a type of access control by which the operating system constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target. In practice, a subject is usually a process or thread; objects are constructs such as files, directories, TCP/UDP ports, shared memory segments, IO devices etc. Subjects and objects each have a set of security attributes. Whenever a subject attempts to access an object, an authorization rule enforced by the operating system kernel examines these security attributes and decides whether the access can take place. Any operation by any subject on any object is tested against the set of authorization rules (aka policy) to determine if the operation is allowed. [11]

2.12 Static analysis:

Static analysis is the process of finding errors and security weaknesses in software through detailed analysis of source code. Static analysis requires the full Intel® Parallel Studio XE or Intel® C++ Studio XE product. The analysis itself is performed by the compiler.
Static analysis finds a wide range of errors that could potentially be exploited by an attacker to defeat application security or cause the application to malfunction. Detected errors include buffer overflow, misuse of pointers and heap storage, unsafe or incorrect use of C/C++ or FORTRAN language features and libraries. Static analysis requires that your code compile without serious errors using the Intel Compiler. [12]

How Static Analysis Works:

The compiler performs the requested analysis by operating in a special mode. In this mode, the compiler dedicates more time to analysis and bypasses the instruction generation process entirely. This allows it to find errors that go undetected during ordinary compilation. This means your code must compile without serious errors with an Intel compiler, but the results of the compilation cannot be executed. Therefore, you can use static analysis even if you do not plan to use an Intel compiler to build your production binaries. [12]

CHAPTER THREE

PREVIOUS STUDIES

&

TOOLS AND TECHNIQUES

3.1 Previous Studies:

3.1.1 Source Code Analysis for Software Vulnerabilities in Android based Mobile Device:

It is mainly focuses on identification of malwares in mobile phone application using search based malware detection algorithm namely (N-gram analysis).

Malware is a malicious code that aims to harm the devices .Malwares can cause system failure, decreasing battery charges, steals the information and corrupts data and go up the maintenance cost. If malware is present in any mobile application, it performs malicious activities like sending SMS to address book, etc. Static code analysis is used to reverse the code without executing the android application file (apk) and convert it into the source code of the file and extracting the features of the application and use the machine learning tool to create the learning model database which is based on malware and benign applications to classify the malwares. To check whether the given test application is having malware or not. For this, compare the N-gram signatures of the test application and signature which is already stored in the files. The result will be the malicious or benign code of the given test application.

The block diagram below illustrates the system architecture:

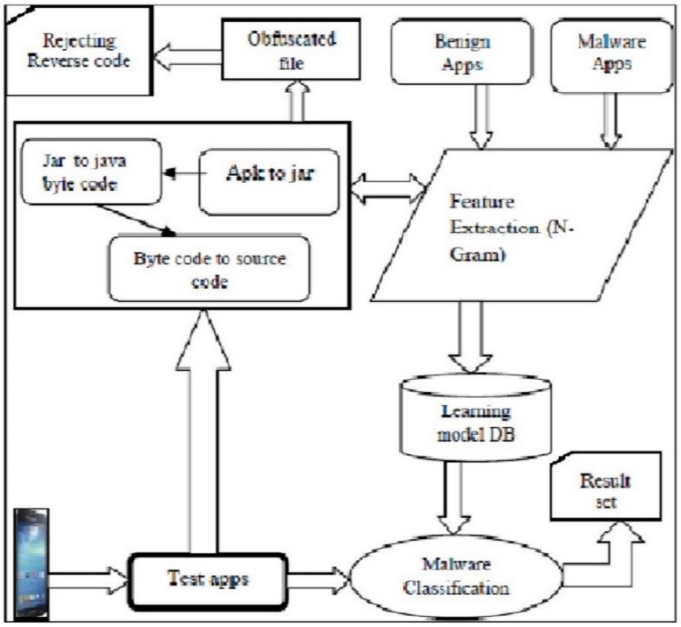

Figure (3.1): Architecture Diagram

The Similarities in our study:

- Both Applications used for identification of malware in mobile phone application.
- Both Applications based on android OS.
- Both Applications use static analysis for detection vulnerability.

The Differences in our study:

- Their application can update for new signature, in our application there is a specific number of signature.

3.1.2 (Static Detection of Dangerous Behaviors in Android Apps):

This paper presents a scheme to detect dangerous behaviors in Android apps. In order to identify different kinds of dangerous behaviors, they designed two analysis engines. On the one hand, taint analysis engine mainly detects privacy leak by tracking how user's sensitive data is used by an app; On the other hand, constant analysis engine focuses on the constant information in an app to identify other dangerous behaviors such as SP services ordering, phone bill consuming, and so on. They have implemented these two engines in a system called ApkRiskAnalyzer which identifies the dangerous behaviors by simulating the running process of an Android app statically. They were used **IDA pro** to disassemble the bytecode and employ **IDA Python** Plug to collect the required information to the database. **IDA pro** is a powerful Commercial disassemble software and it starts to support Dalvik bytecode from V6.1. They saved the disassembly information to the MySQL

database, so it does not require re-disassemble when they want to re-analyze an app. The step above is called **Pretreatment**. After the pretreatment, ApkRiskAnalyzer reads bytecode information from the database, and simulates the execution process of an app. The simulation execution includes two phases: intra-procedural analysis and inter-procedural analysis. In intra procedural Analysis, the execution sequence of instructions is calculated according to the control flow graph (CFG), and each instruction should be simulated based on its semantic. In inter-procedural analysis, function call process is simulated based on the call graph. The taint and constant analysis engines are implemented on the basis of simulation Execution to trace the data flow. Then based on the results of the two analysis

Engines and detection rules defined in the rule base, they detect dangerous behaviors at the call point of sensitive functions, and the results are shown to users finally. The figure below illustrated their System architecture.

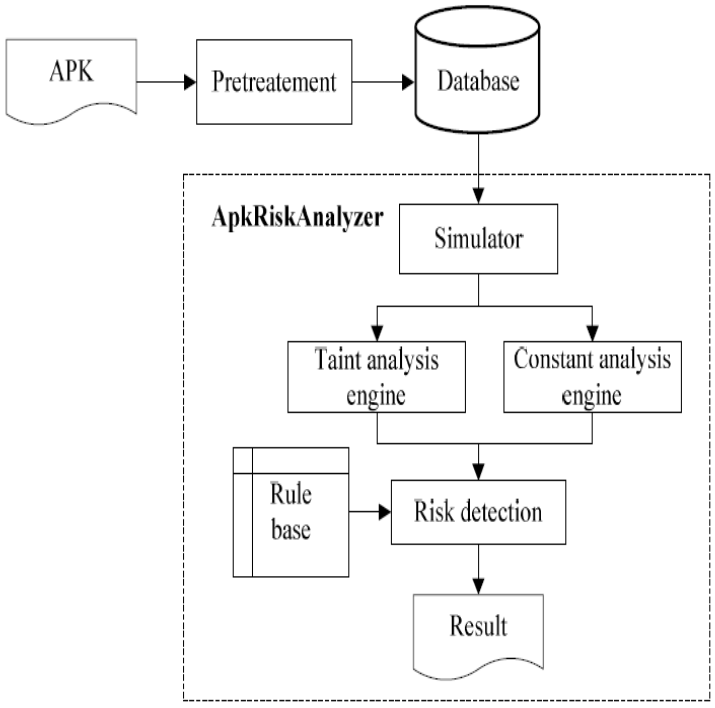

Figure (3.2): ApkRiskAnalyzer overall architecture

Advantages of their proposed system:

- Directly analyze the Dalvik bytecode and do not need to get the source code, nor decompile the Dalvik bytecode to Java source code.

- When you needs to reanalyze the application you don’t require to re-disassemble because after any disassemble process they save disassemble information in MySQL database.

- Do not run the apps and instead they complete the dangerous behavior’s detection directly on the computer.
- Treated the implicit invocation in Android systems problems.

The similarities in our study:

- Both Applications used for identification of malware in mobile phone application.
- Both applications based on android OS.
- Both applications used Rules to detect behavior for dangerous application.

The differences in our study:

- This application doesn’t need to get the source code, and no need to decompile bytecode to java source code.
- This application has Database for save bytecode information about application.
- Our application save information about apps tested it.
- ApkRiskAnalyzer identifies the dangerous behaviors by simulating the running process of an Android app statically and based on simulating two engines Which we have mentioned previously identifying in what the application have dangerous behaviors or not.

3.1.3 Apposcopy: Semantics-Based Detection of Android Malware through Static Analysis:

They present Apposcopy, a new semantics-based approach for identifying a prevalent class of Android malware that steals private user information. This study summarizes in incorporates points:

- High level language for specifying signatures that describe semantic characteristics of malware families.

- Static analysis for deciding if a given application matches a malware

Signature. The signature matching algorithm of Apposcopy uses a combination of static taint analysis and a new form of program representation called Inter-Component Call Graph to efficiently detect Android applications that have certain control- and data-flow properties.

Advantages of their proposed system:

- Apposcopy can detect malware with high accuracy and that its signatures are resilient to various program obfuscations.

- Apposcopy combine taint analysis with high-level malware signatures to effectively identify malicious code.

Limitation of their study:

- **Apposcopy** is not invincible, and it faces some obstructions such as:

- **Apposcopy** may be defeated by obfuscation techniques such as dynamic code loading and use of reflection in Combination with obfuscation of method or class names.
- **Apposcopy** performs deep static analysis to uncover semantic properties of an app, it may be unfit for scenarios that require instant detection of malware.

The similarities in our study:

- First, the similarities in this study in their use of the static analysis to detect malicious applications.

The differences in our study:

- As well as static taint analysis Apposcopy uses a new form of program representation called Inter-Component Call Graph to efficiently detect Android applications that have certain control- and data-flow properties.

3.2 Introduction:

This section illustrates the tools and techniques that are used to achieve the project objectives, and by taking technology or tool and explain the brief exposure to take the most important advantages or disadvantages, if any.

3.3 Tools:

3.3.1 Eclipse:

Is an integrated development environment (IDE) it contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages through the use of plugins, including: JavaScript, Perl, Php, python, C, C++, Fortran ... etc.

Why do we use Eclipse?

- Project management.
- Good debugger.

- Error checking features (e.g. error checking as you type, suggestions to fix compile errors).
- Auto complete feature (e.g. if you type "reference Variable." lets you choose a method/field from the variable's class from a drop down list). [13]

3.3.2 Java:

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any java virtual machine (JVM) regardless of computer architecture. As of 2015, Java is one of the most popular programming languages in use particularly for client-server web applications, with a reported 9 million developers Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

We were used Java due:

- **Easy to learn:**

  Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

- **Object-oriented:**

  This allows you to create modular programs and reusable code.

- **Platform-independent:**

  One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program

on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

- **Distributed:**

  Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it. Writing network programs in Java is like sending and receiving data to and from a file.

- **Security:**

  Java considers security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.

- **Robust:**

  Robust means reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detect many problems that would first show up during execution time in other languages.

- **Multithreaded:**

  Multithreaded is the capability for a program to perform several tasks simultaneously within a program. In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading. [14]

3.3.3 Dex2jar:

Dex2jar converts android DEX files to java CLASS file. This is useful because many tools are already available that can decompile Java bytecode back to source code. It is open source. It has grown from just a decompile into a tool suite that preforms many different task. However that focus in this section is on converting Android DEX files to java file. [15]

3.3.4 Jad:

It is computer program that performs the reverse operation to that of a compiler. That is, it translates program code at a relatively low level of abstraction (usually designed to be computer readable rather than human readable) into a form having a higher level of abstraction (usually designed to be human readable). Decompilers usually do not perfectly reconstruct the original source code, and can vary widely in the intelligibility of their outputs. Nonetheless, decompiles remain an important tool in software reverse engineering. A decompiler takes as input an executable file, and attempts to create a high level, compliable, possibly even maintainable source file that does the same thing. It is therefore the opposite of a compiler, which takes a source file and makes an executable. However, a general decompiler does not attempt to reverse every action of the compiler, rather it transforms the input program repeatedly until the result is high level source code.

3.3.4.1 Advantages of Jad:

1. Fast, written in c.
2. Available for most Operating Systems/platform.
3. Several GUIs and IDE plug-ins available.

3.3.4.2 Other Tools smeller Jad:

- Mocha.
- Jode.[16]

3.3.5 Shell:

A shell script is a computer program designed to be run by the UNIX shell, a command line interpreter .The various dialects of shell scripts are considered to be scripting languages. The typical Unix/Linux/Posix-compliant installation

includes the korn shell in several possible versions such as ksh88, Korn Shell '93 and others. the oldest shell still in common use is the bourn shell, Unix systems invariably include also the C shell , Bourne Again Shell , a remote shell , a secure shell for SSL telnet connections (Ssh), and a shell which is a main component of the Tcl/Tk installation usually called Tclsh; wish is a GUI-based Tcl/Tk shell. Related programmers such as shells based on python, Ruby, C, Java, Perl&c in various forms. Another somewhat common shell is Osh, whose manual page states it "is an enhanced, backward-compatible port of the standard command interpreter from Sixth Edition UNIX. [17]

3.3.6 Linux:

CentOS (abbreviated from Community Enterprise Operating System) is a Linux distribution th0at attempts to provide a free, enterprise, community-supported computing platform which aims to be functionally compatible with its upstream source, Red (RHEL). In January 2014, CentOS announced the official joining with Red Hat while staying independent from RHEL, the Linux was chased because it is open source operating system and the tools of reverse work more efficient in Linux [18].

3.3.7 Enterprise Architect:

Enterprise architects work with stakeholders, both leadership and subject matter experts, to build a holistic view of the organization's strategy, processes, information, and information technology assets. [19]

3.3.8 UML:

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering that is intended to provide a standard way to visualize the design of a system. [20]

3.3.8.1 We were used UML due to:

- The communication of the desired structure and behavior of a system between analysts, architects, developers, stakeholders and users.
- The visualization and control of system architecture.
- Promote a deeper understanding of the system, exposing opportunities for simplification and re-use.
- Manage risk.

3.3.8.2 UML Diagrams:

- Use case diagram.
- Sequence diagram.
- Activity diagram.[21]

CHAPTER FOUR
SYSTEM ANALYSIS

4.1 Introduction:

This section illustrate the general description of the system and its functions. It’s also clarify the software components and software, and the detailed analysis of the operations of the system were showed using the schemes UML.

4.2 System Description:

This system detects risky behaviors Android applications using static analysis and make telephony identifiers exfiltration and knowledge statements that the application uses to threaten user data Privacy and sent over the network

4.3 System Environment:

Integrated development environment (Eclipse) was used to develop the application. Database was created for the results of tested application, because if we want to make reversing operation for the application in another time we will not need to replay the operation again.

4.4 System functions:

Client-server model was used in this project and both client and server have specific functions:

In client side: user selects the package of the application which they need it to know the application state before installing it on the device and sends it to the server, and wait until the result of the package state.

In server side: it will make search operation in the database for the key which it received from the client to determine if the application was stored in the database or not, if the key was found the server will return the result of the application state as a report form, and if it was not found the server will notify the client that the key was

not be found then the client will send the apk file to it to make the reversing operation to get the java files.

4.5 Analysis using UML diagrams:

4.5.1 Use case diagram

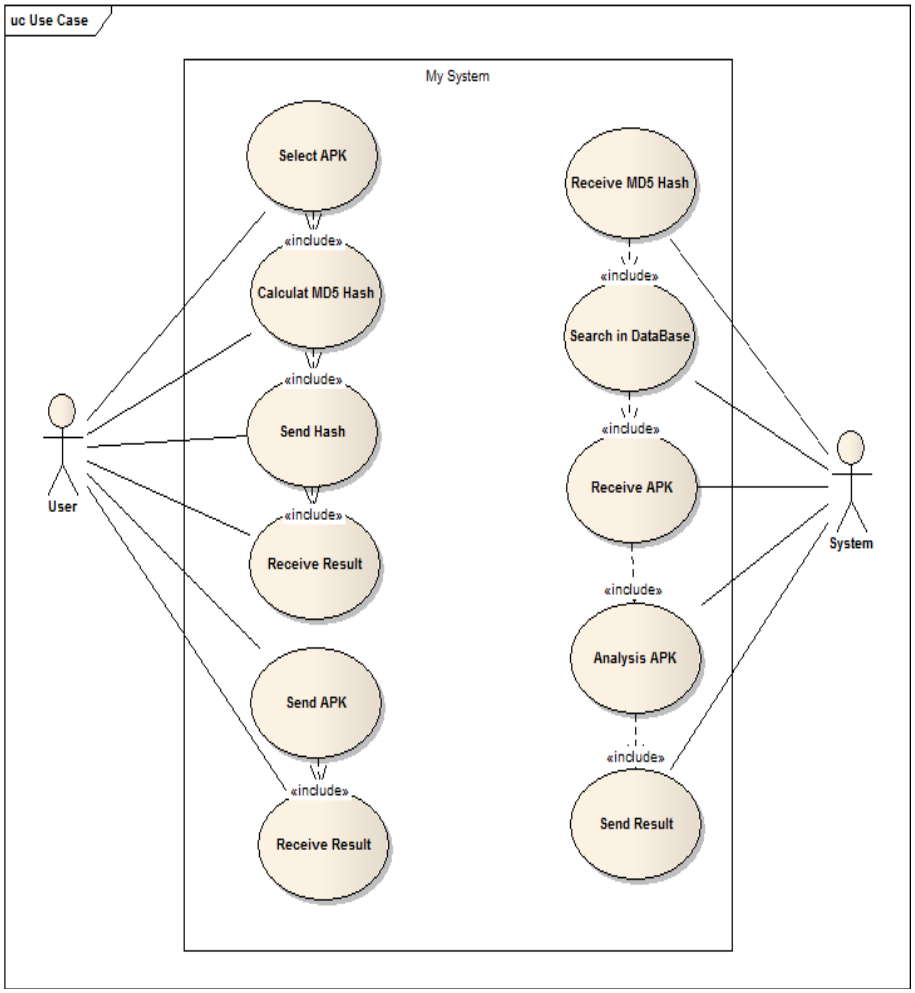

Figure 4.1: Use case

This diagram is used to illustrate and describe how the proposed system works and how to use it.

4.5.2 Activity Diagram

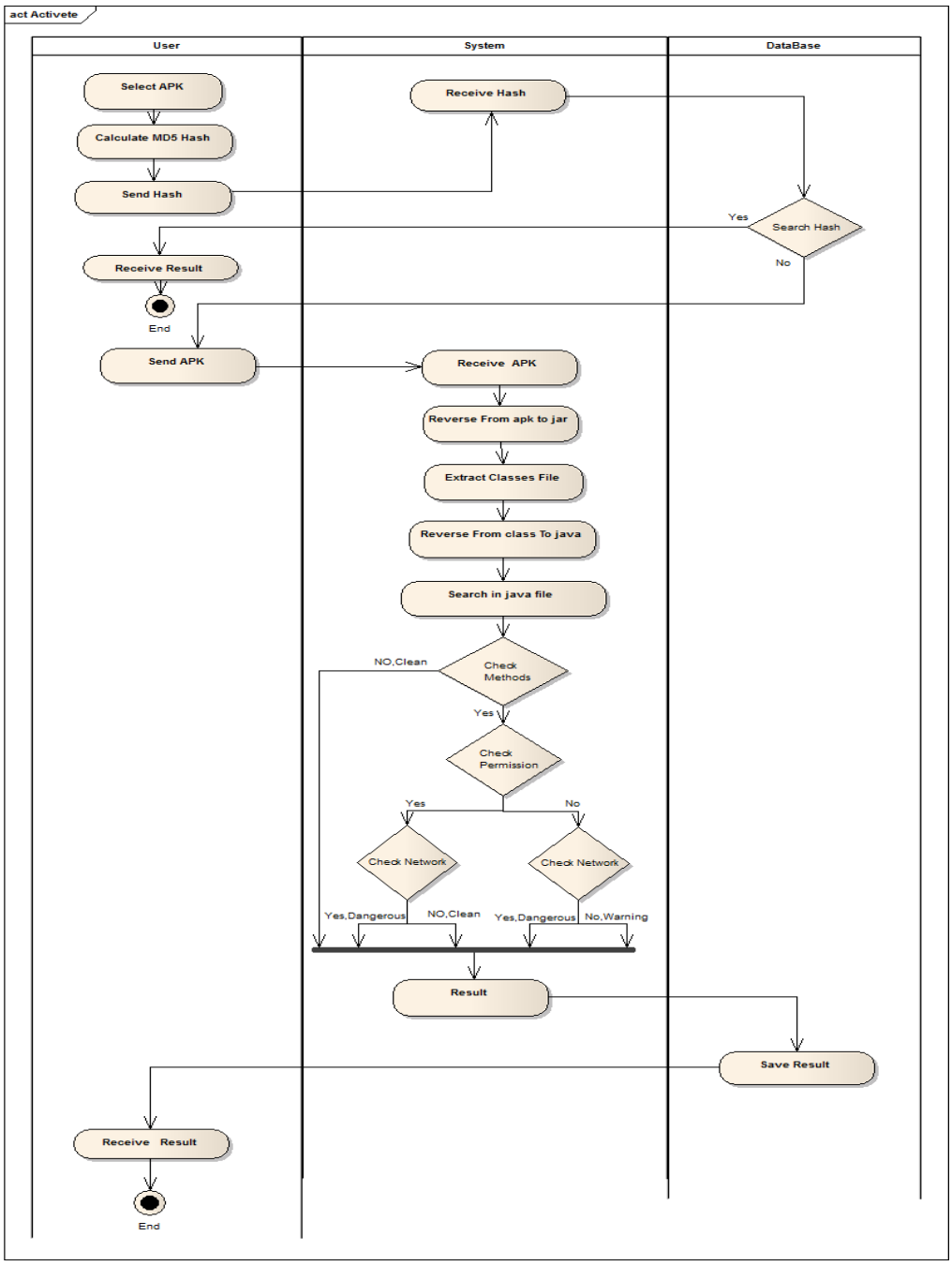

Figure 4.2: Activity diagram

4.5.3 Sequence diagram

Used to show the flow of data and messages between objects and the various system components.
Horizontal components in Figure describe the shared objects in the system.

Vertical components describe the ordered exchanged messages depend on the order that is in the system.

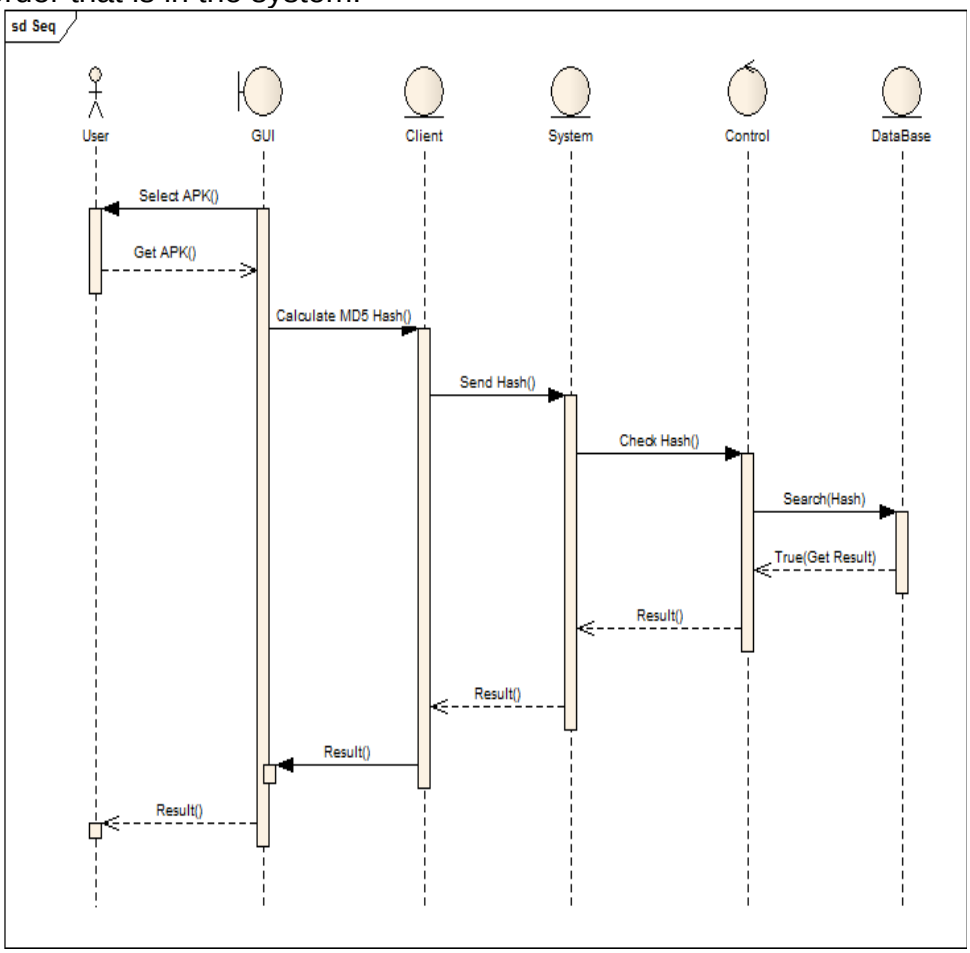

Figure 4.3: Result Not Exist in Database

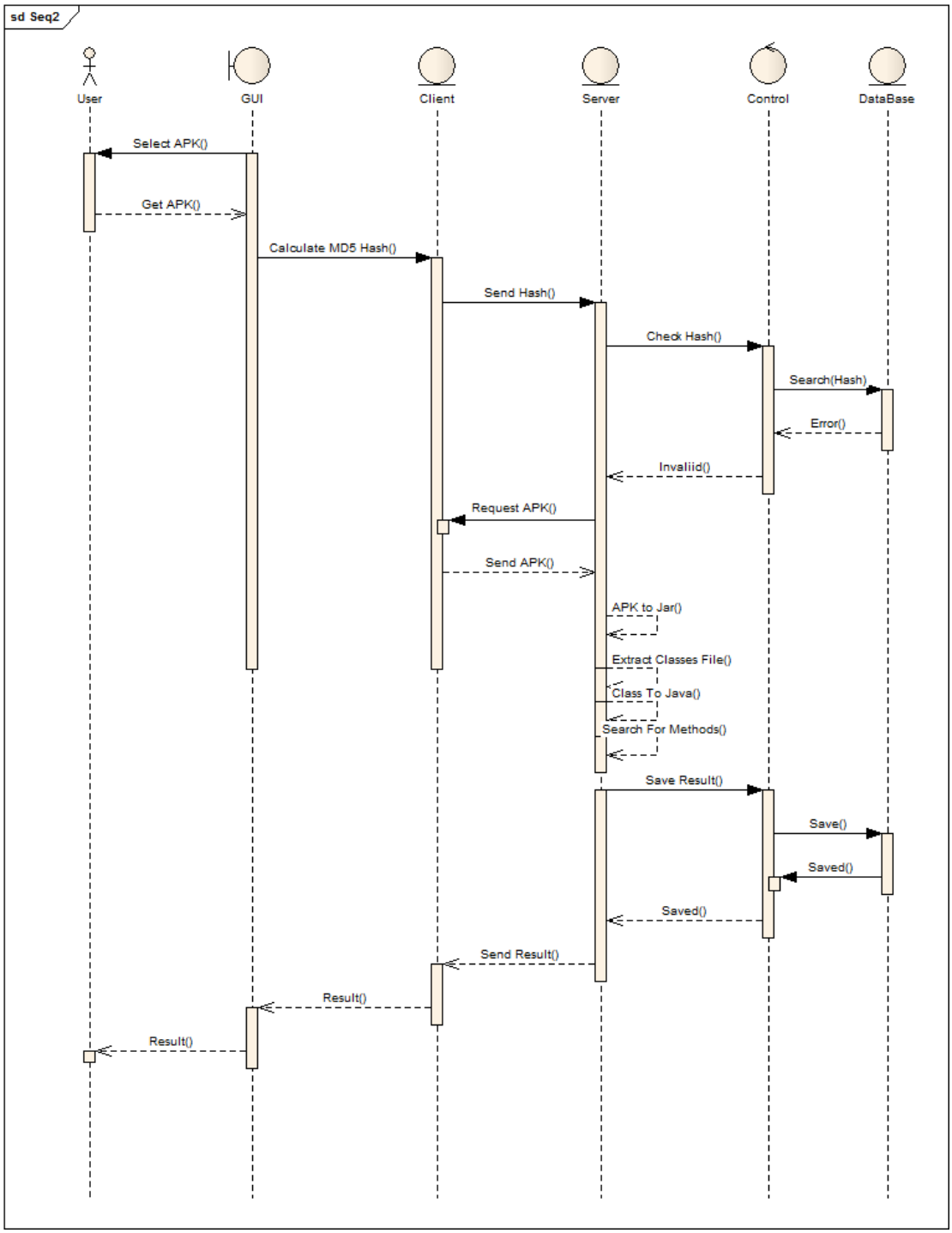

Figure 4.4: Result Exist in Database

CHAPTER FIVE

IMPLIMINTATION

5.1 Introduction:

In this chapter we will talk about the Graphical interface and how the user will interact with the software.

5.2 Graphic User Interface:

In the beginning the user must have an Android OS then install the application on this phone and he must be connected to the Internet. After that the user opens the application, and is shown the first window figure (5.1):

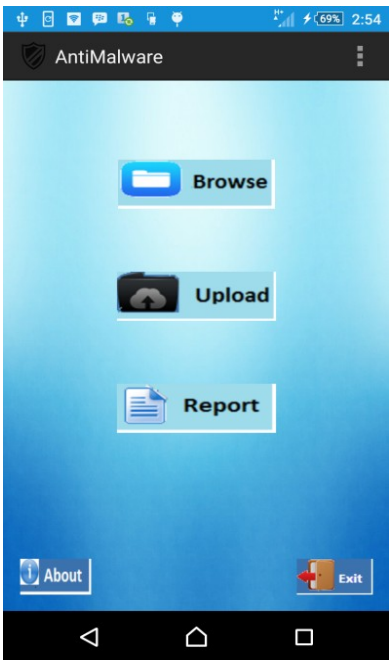

Figure (5.1): Main Screen

5.2.1 Select Application:

The first button (browse) when pressed will allow the use to search for the file that he wants to be tested, the following figure (5.2) illustrates this:

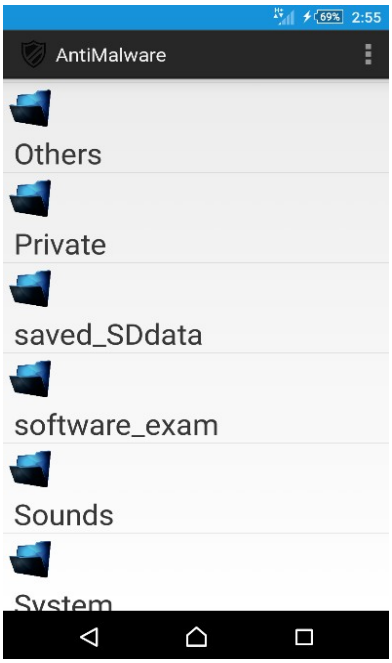

Figure (5.2): Select Application

5.2.2Upload file:

The second button (Upload) when pressed will send the selected file to the server following figure (5.3) illustrates this:

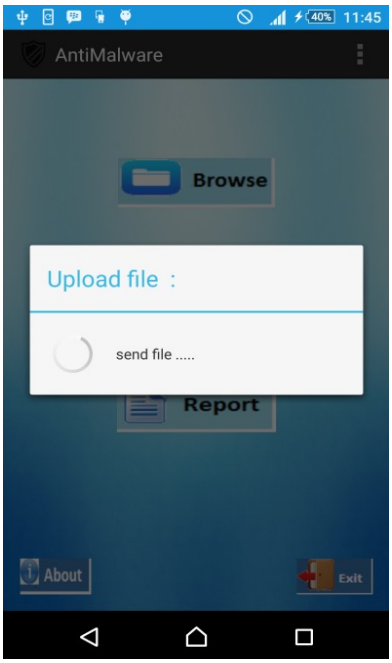

Figure (5.3): Upload File

5.2.3 Show Report:

The third button (Report) when pressed will show a report about file which was chosen, file has one of three condition: Safe, warning, dangerous following figure illustrates this:

5.2.3.1 Safe mode:

Application is Safe from malware method and get permission to access method. As show in figure (5.4) below:

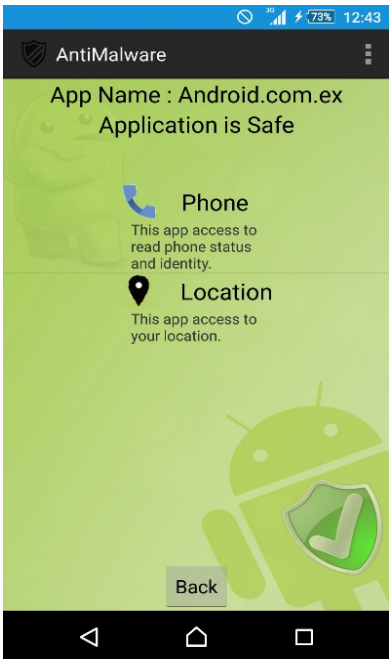

Figure (5.4): Safe Mode

a) Method used phone:

When you pressed will show a description about method which was chosen.

As show in figure (5.5) below.

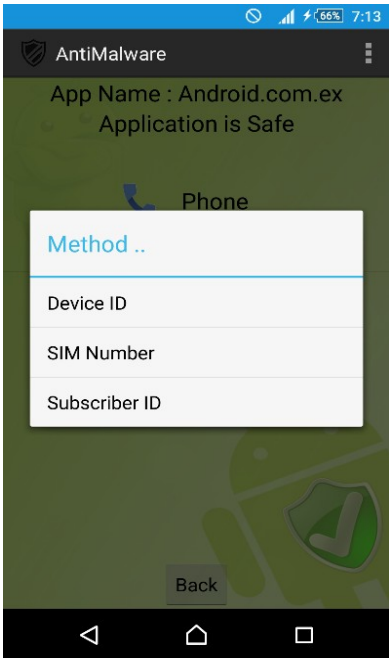

Figure (5.5): Method Used Phone

b) Method used in location:

(The Longitude, Latitude methods are used to determine the location of device)*.When pressed will show a description about method which was chosen. As show in figure (5.6) below:

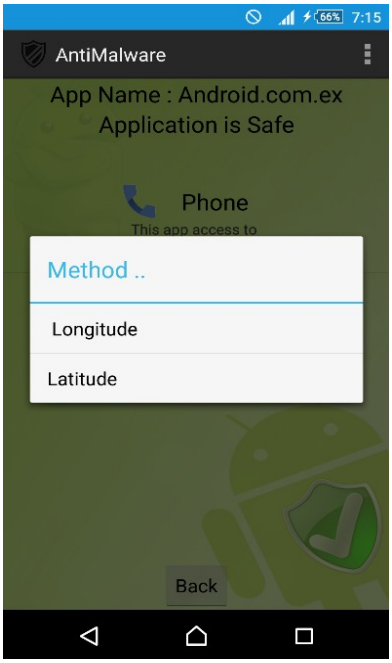

Figure (5.6): Method Used In Location

5.2.3.2 Warning mode:

The application takes phone or location‘s information without the using permissions. As show in figure (5.7) below:

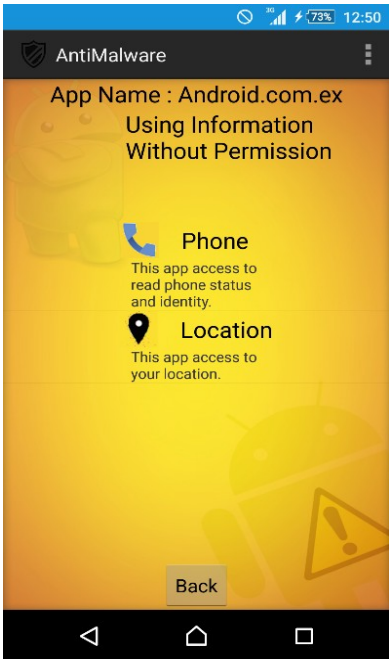

Figure (5.7): Warning Mode

a) **Method in phone:**

When pressed will show a description about method which was chosen. As show in figure (5.8) below

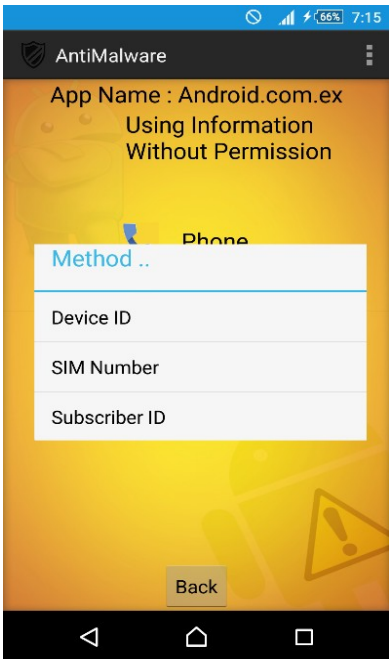

Figure (5.8): Method used in phone

5.2.3.3 Dangerous mode:

The Application takes phone or location's information and send it to internet with the using of the permission or not. As show in figure (5.9) below:

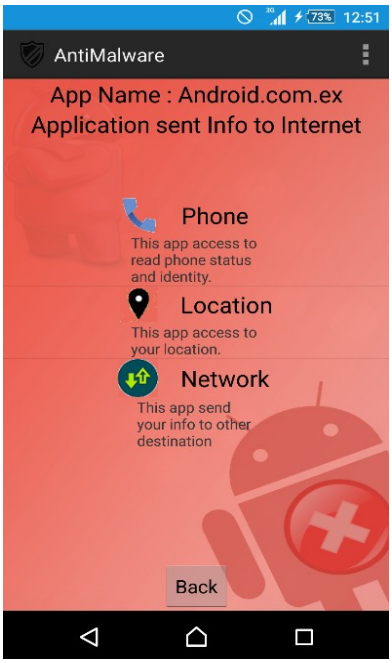

Figure (5.9): Dangerous mode

a) Method in phone:

When pressed will show a description about method which was chosen. As show in figure (5.10) below:

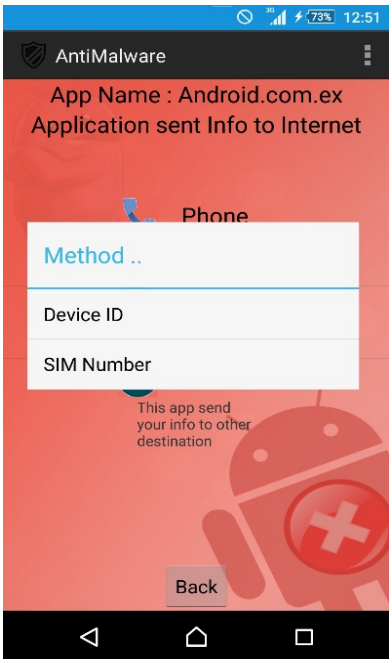

Figure (5.10): Method used in dangerous

b) Method in Location:

When pressed will show a description about method which was chosen. As show in figure (5.11) below:

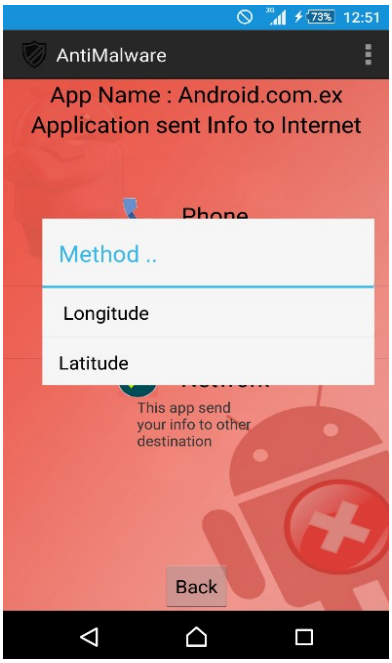

Figure (5.11): Method used in location

When the testing application used both this method that's mean it sends the geographic location to another destination, thus whenever you been you will be tracked, there are another methods used to detect the location but those are common in used.

5.2.4 Information about application:

The fourth button (about) when pressed a window is opened, which displays about the application as shown in Figure (5.12).

When you press the button (cancel) are returned to the main menu of the application.

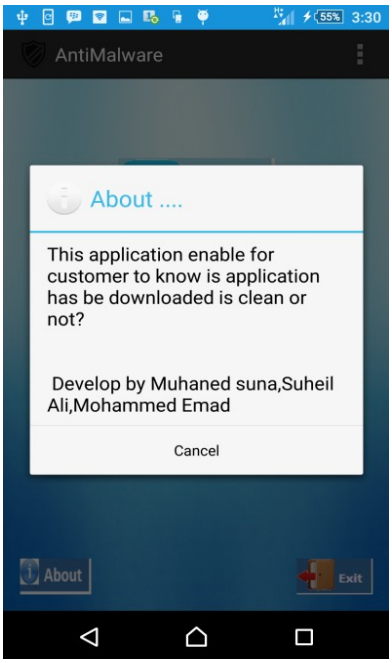

Figure (5.12): Information about application

5.2.5 Exit from application:

The fifth button (Exit) and when pressed will exit the program after the confirmation message asking the user whether he was sure to get out of the application as shown in Figure(5.13):

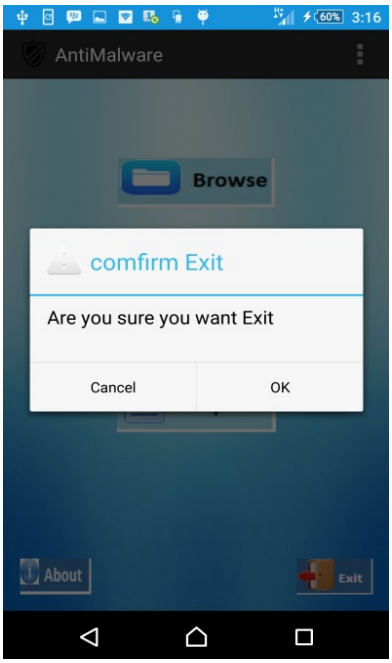

Figure (5.13): Exit from application

5.3 Server functionality:

server get apk file from user and search for malware function in the file by check the grant permission and compare the sources code with specific pattern determine level of risk (safe, warning ,dangers) and provide this information to the user.

5.4 Test System:

#	Name	Status	Location	Phone	Network	permission
1	Whatsapp	Safe				
2	Line	Safe				
3	Number book	Safe				

4	1mobileMarket	Safe				
5	Subway	Safe				
6	Messenger	Safe				
7	Viber	Safe				

This was in partial test number of application on the system and explains its effectiveness and efficiency in term of status of application risk this table (5.1) explain that.

Table 5.1: System Test

5.5 conclusion:

In this chapter the system was implemented and review of the screens for each part of the server and the client. It has also been testing the system illustrate the system's effectiveness and efficiency, and will be addressed in the next section the results that have been reached and recommendations for future studies and research.

CHAPTER 6

RESULT AND RECOMMUNDATION

6.1 introduction:

This section contains the results of the system and the recommendations that will work in the future.

6.2 Results:

At the end of this project we had been testing 20 applications from different market and get this information in tables:

In total, 13 applications included code to obtain a phone identifier; however, only 12 of these applications have the READ_PHONE_STATE permission required to obtain access. We found that all four identifier types: 9 applications have IMEI; 6 applications have phone number; 5 applications have IMSI; and 8 applications have ICC-ID. Location information is accessed with two way (getLastKnownLocation and LocationListener). In total, 18 applications attempt to access location, and all have permission to do.

Identifier	# Apps	# w/ Permission*
Phone Number	12	12
IMEI	9	9
IMSI	5	5
ICC-ID	8	8

* **Defined as having the READ_PHONE_STATE permission.**

Table 6.1: PHONE IDENTIFEIR

6.3 RECOMMENDATIONS

- The application need as well as exfiltration phone by data flow analysis to cover misuse of telephony services such as voice call and sms services by looked for hard-coded phone numbers in the sms API using semantic analysis.
- As an approximation of recording audio or video in background without user knowledge look for use of API specified for audio and video recording in code not accessible to an android activity component.
- Add dynamic analysis to application, because it detects the behavior of the application on runtime of it.
- Use static analysis in android device without server site.

6.4CONCLUSION:

We did overview of existing type of analysis in application based on android operating system (static, dynamic), and then use the flow analysis approach to explain the safety of the application from the misuse of user information (phone identification, location information).

After implementing the system the user can specify the level of risk (phone identification, location information) for all his application before install it.

Finally we discussed the additional features that can be added in the system.

REFRENCES

[1]: http://source.android.com/devices/tech/security/

[2] https://www.cs.rice.edu/~sc40/pubs/enck-sec11.pdf

[3]: Fedler, R., Banse, C., Krauss, C., Fusenig, V., Android OS Security: Risks and Limitations: A Practical Evaluation. Fraunhofer Research Institution for Applied and Integrated Security, 2012. [Online].

[4]: National Security Agency, “SELinux,” January 2009. https://www.nsa.gov/research/selinux/.

[5]: Geier, Eric (2012-01-16). "How to Keep Your PC Safe with Sandboxing" . TechHive. Retrieved 2014-07-03.

[6]: Bovet, D. P., Cesati, M. Understanding the Linux Kernel. 3rd ed. California: O’Reilly Media Inc., 2005.

[7]: Android Security Team, “Android Security Overview,” December 2011. http://source.android.com/tech/security/index.html.

[8]:http://www.webopedia.com/TERM/S/security_architecture.html.

[9]: https://developer.android.com/guide/topics/security/permissions.html.
[11]: https://en.wikipedia.org/wiki/Mandatory_access_control.
[10]: https://developer.android.com/guide/topics/manifest/permission-
[12]:https://software.intel.com/sites/products/documentation/doclib/iss/2013/compiler/cpp-lin/GUID-9B795CE3-8341-47C1-83A9-614AB60110B3.htm.

[13]: https://www.eclipse.org/eclipse/news/4.6/QFjADahUKEwi1oazv6-LHAhWnFtsKHRNWDBo&url=http%3A%2F%2Fwww.blackhat.com%2Fpresentations%2Fbh-usa-03%2Fbh-us-03-spett.pdf&usg=AFQjCNFy5P8tKPVTb2J1xAizJz2Kmbfjkg&cad=rja.
[14]:http://www.streetdirectory.com/travel_guide/114362/programming/most_significant_advantages_of_java_language.html.
[15]:https://books.google.com/books?id=UgVhBgAAQBAJ&pg=PA752&dq=documents+about+dex2jar+tool&hl=en&sa=X&ved=0CC8Q6AEwA2oVChMIfKLouHzxwIVwTkaCh1AfwQc#v=onepage&q=documents%20about%20dex2jar%20tool&f=false.
[16]: Hamilton, James; Danicic, Sebastian (2009). An Evaluation of Current Java Bytecode Decompilers. Source Code Analysis and Manipulation, 2009. SCAM '09. Ninth IEEE International Working Conference on. pp. 129–136.
[17]: Kernighan, Brian W.; Pike, Rob (1984), "3. Using the Shell", The UNIX Programming Environment, Prentice Hall, Inc., p. 94, ISBN 0-13-937699-2, The shell is actually a programming language: it has variables, loops, decision-making, and so on.
[18]: Perrin, Jim (2015-08-04)." [CentOS-announce] Release for CentOS 7 on AArch64". Retrieved 2015-11-01.
[19]: https://en.wikipedia.org/wiki/Enterprise_architect.
[20]: OMG (February 2009). "OMG Unified Modeling Language (OMG UML), Superstructure Version 2.2".
[21]: http://dthomas-software.co.uk/resources/frequently-asked-questions/why-use-uml-2/.