Android Service Security: Victimized Android Device by Transferring Malicious Service via Near Field Communication Using Social Engineering

Mohammad Khosheghbal Ghamsari

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
February 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Gürcü ¨Oz
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Alexander Chefranov

2. Asst. Prof. Dr. Gürcü ¨Oz

3. Dr. Hüsnü Bayramoğlu

ABSTRACT

The purpose of this thesis is to investigate on vulnerabilities of the Android operating system and security breaches which would be harmful and could be misused for destructive purposes. In this thesis the whole idea of social engineering would be investigated and the relevance of implemented system would be explained. The implemented system includes client and server architecture which allows attacker to spy on the device's owner as a target and obtain full access of target device remotely. Example can be listed as listening to his or her voice calls, checking his or her messages, finding victim's location, accessing to all stored files and checking victim's call logs. The system would make the attacker able to get a kind of remote access from the target device. This system is implemented for spying on Android operating system and shows that how legitimated application can be turned to malicious software by adding services. The main benefit of this service is that service will run on the background of target's device and stick to device even target's device is restarted. The interface of this malicious application can be changed easily based on information that attacker is gathered from victim' interest. Transferring this malicious application to Android device can be done by using Near Field Communication (NFC). This application has been implemented on Eclipse environment and coded with Java programming language.

Keywords: Android, Malicious service, Security breach, Near Field Communication, Social Engineering, Hacking.

ÖZ

Bu tezin amacı Android işletim sisteminin güvenlik açıkları üzerinde araştırma yapmak, güvenliğin ihlal edilmesi durumunda sistemin nasıl zararlı, yıkıcı ve kötü amaçlar için kullanılabileceğini göstermektir. Bu çalışmada, sosyal mühendislik fikri araştırılıp, uygulanan sistemde kullanılmış olup önemi belirtilmiştir. Uygulanan sistem, bir istemci/sunucu mimarisi olup, saldırgana hedef cihazdan casusluk imkanı sağlamakta ve hedef cihaza uzaktan tam giriş vermektedir. Saldırgan kurbanın sesli aramalarını dinlerken mesajlarını da kontrol edebilmektedir. Ayrıca, kurbanın yerinin tespiti, saklanan bütün dosyalarına girişi, çağrı listelerinin takibi de mümkün olmaktadır. Tasarlanan sistemle saldırgan hedefi uzaktan kontrol etme imkanı elde edebilmektedir. Bu sistem, Android işletim sistemi üzerinden casusluk yapmak için yaratılmış olup, meşru uygulama hizmetleri eklenirken, meşru olmayan bir yazılım cihaza nasıl eklenebilir onu göstermektedir. Bu servisin en büyük avantajı , hedef cihaz kapatılıp açılsa bile arka planda yeniden çalışabilir olmasıdır. Bu kötü niyetli uygulamanın ara yüzü hedefin ilgi alanına göre kolayca değiştirilebilmektedir. Oluşturulan bu uygulamanın Android cihazına aktarımı Yakın Alan İletişimi (NFC) kullanılarak yapılabilir. Bu uygulama, Eclips ortamında oluşturulmuş olup, Java programlama dili kullanılarak kodlanmıştır.

Anahtar Kelimeler: Android, Kötü Amaçlı hizmet, Güvenlik ihlali, Yakın Alan İletişimi, Sosyal Mühendislik, Hacking.

To my lovely parents
For their devotion to me

ACKNOWLEDGMENT

I would like to express my special gratitude and thanks to my supervisor Asst. Prof. Dr. Gurcu Oz, you have been a marvelous mentor for me. I would like to thank you for encouraging my research and for allowing me to grow as a research scientist. I would also like to thank my committee members, Assoc. Prof. Dr. Alexander Chefranov and Dr. Husnu Bayramoglu for serving as my committee members even at hardship. I also want to thank you for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions, thanks to you.

A special thanks to my family. Words cannot express how grateful I am to my mother, my father for all of the sacrifices that you have made on my behalf. Your prayer for me was what sustained me thus far. I would also like to thank my brother, my sister and all of my friends who supported me in this period of time, and incented me to strive towards my goal. At the end I would like express appreciation to my girlfriend Shirin who was always my support in the moments when there was no one to answer my queries.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

API	Application Programming Interface
APK	Application Package File
AWT	Abstract Window Toolkit
DEX	Dalvik Executable
IDE	Integrated Development Environment
IMEI	International Mobile Equipment Identification
JAR	Java Archive
JPanel	Java Panel
LAN	Local Area Network
NFC	Near Field Communication
OS	Operating System
RFID	Radio Frequency Identification
SDK	Software Development Kit
VPN	Virtual Private Network
WAN	Wide Area Network
XML	Extensible Markup Language

Chapter 1

INTRODUCTION

1.1 Background

It is undoubtedly true that with development of technology people around the world prefer to do their daily tasks without wasting their time too much. Before invention of mobile phone, scientists started to think about mobility and how human would be able to communicate with each other when they are moving from one place to another place. However, after invention of mobile phone in 1973 [1], many companies such as Apple and Samsung tried to produce new cellphone with new technology. By development of technology, producers of mobile phones tried to improve the design of their products as well as adding more features to them. While the world of technology is growing up, scientists did many research in order to make mobile phones as smart devices. This was commencing idea of smartphones which are smart enough to handle processors and work with operating system. Nowadays, Android operating system (OS) is one of the most popular OS for smartphones based on its functionality [2, 3].

The word of Android in dictionary means a robot with a human appearance. Android OS has been produced by Google Company and firstly released in 2008. This OS is based on Linux kernel and it has been designed for smartphones. Based on Google report [4], there are more than hundreds of million devices running Android OS in the world. In addition, they mentioned that the number of users is increasing around

a million per day. Because of being open source project, Android provides an opportunity for developers to make their own OS and customize it.

Besides this, in Android OS, users have an opportunity to make their desire applications to be used on their own devices which needs some java coding knowledge. As regard, there is no restriction for developers to create legal or even illegal applications. It is obvious that anybody with enough knowledge in java programming languages can develop destructive applications and victimize any user who uses this application. According to Alcatel-Lucent's Q2 malware report [5] in 2013 the number of infected Android devices became more than 1% of total Android devices around the world and unfortunately this number is increasing.

As it has been explained before, convenient communication is human's desire and one of the latest technology which satisfies people is Near Field Communication (NFC) [6]. NFC is based on Radio Frequency Identification (RFID) communication and growing fast same as Bluetooth in the past [6]. Nowadays this sensor is built-into many smartphones and it is used for paying bills in some payment gateways which support NFC communication. Vulnerabilities of this new technology is discussed in the second chapter of this thesis.

Investigation on NFC shows that for establishing communication between two devices using this technology, there is no need for authentication or pairing devices. Although this technology has many advantages such as increasing speed of communication, but it has a dangerous disadvantage that device can be tampered easily. This disadvantage can be the cause of a serious security breach in device. One of the suitable methods for using this vulnerability is social engineering. Social

engineering is about how to trick people in any position to get some advantages from them. These advantages can be divided into two parts which are economic benefit and information benefit. The person who uses this science in vector attack can gather a lot of vital information easily.

1.2 Thesis Contribution

In this thesis, creating malicious Android service under legitimate application and transferring it by NFC using social engineering skill is proposed. In specific, by using client server architecture, getting full remote connection from the target and making it victimized is possible. Mobile security especially for Android OS, because of its huge number of users, becomes one of the biggest issues in mobile computing. By establishing client server communication and adding several features to this application, attacker can get full control of victim's device.

1.3 Thesis Outline

The outstanding body of this dissertation is as follows: In Chapter 2 an overview of Android architecture, application and service, NFC and social Engineering is provided. Chapter 3 offers system overview, system description, presentation of system architecture; different situation of target location which is inside of the LAN or is connecting from WAN. In Chapter 4, aims and features of the mentioned application and results of the proposed application graphically will be deliberated. As a final point, Chapter 5 delivers conclusion on the results and future works.

Chapter 2

LITERATURE REVIEW

2.1 Android

Nowadays, Android OS is running on more than hundreds of millions devices with different versions of Software Development Kit (SDK) [4]. Android started from 1.5 version named as Cupcake, 1.6 Donut, (2.0, 2.1) Éclair, 2.2 Froyo, 2.3 Gingerbread, (3.0, 3.1, 3.2) Honeycomb, 4.0 Ice Cream Sandwich, (4.1, 4.2, 4.3) Jelly Bean, 4.4 Kit Kat and the latest one 5 Lollipop. These versions have developed through years and made Android to be the most delicious OS in the world [7]. Android in all versions is multiprocessing and multithreaded OS which is not only limited to smartphones [8]. Although there are some differences between versions regarding their extra features but all of them obey same architecture.

2.1.1 Android Architecture

Android architecture or Android software stack is included of six layers, as it is shown in Figure 2.1. Layers are Linux kernel, hardware abstraction layer, libraries, Android runtime, application framework and application layer. [9-11].

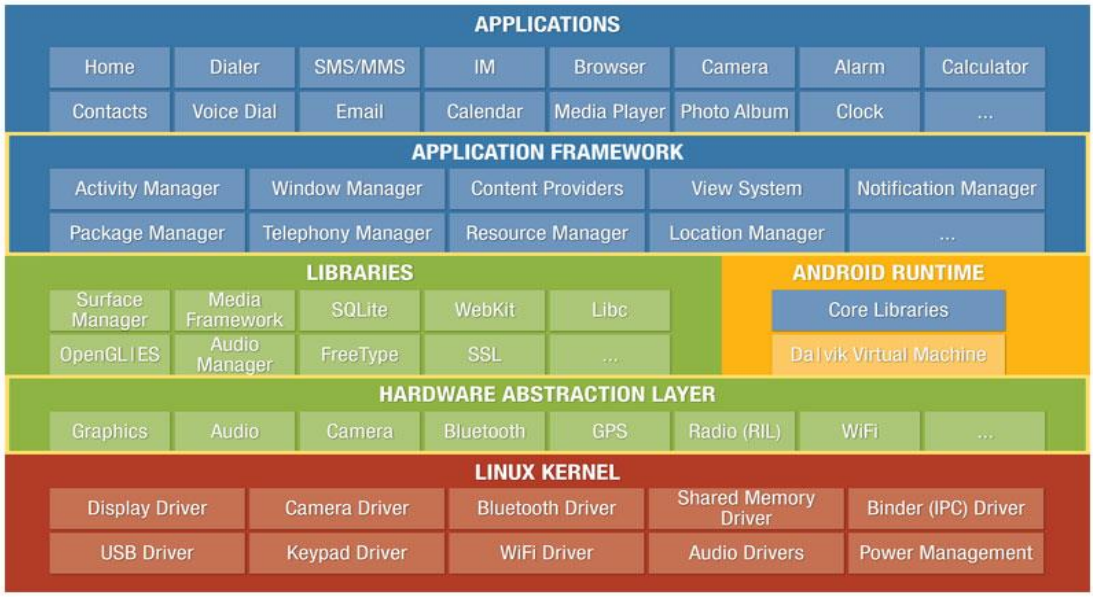

Figure 2.1. Android architecture [12]

Linux Kernel Layer

The base layer is improved Linux kernel [9] and all the hardware drivers is stored in this layer. Power and memory management process are handled by this layer as illustrated in Figure 2.1. The reason of using Linux kernel as the base of architecture for Android OS is that Linux kernel can manage the process and memory better than other kernels and also it can support shared library unlike the other kernels. Beside this, security in Linux kernel depends on the user permission which makes this OS secure enough.

Hardware Abstraction Layer

The second layer is known as HAL which is standing for Hardware Abstraction Layer. In many discussions in this field, some developers think that this layer can be added to the first layer but there are some differences [11]. This separate layer handle communication between Linux kernel layer and native libraries and because of this, it can be considered as a separate layer as it is shown in Figure 2.1. This layer consists of hardware of the device.

Libraries and Runtime Layer

The third layer is one of the most essential layers of Android architecture and it is included of two significant parts as shown in Figure 2.1. The first part of this layer is native libraries which is included of Bionic C libraries, function libraries and native servers. Some of these libraries can be used for specific processes where the others are generic, HAL can be used for all Android devices [9-11]. In the Figure 2.1 it can be seen that surface manager is supposed to show user interface window. The other part is Android runtime which is included of core libraries and Dalvik virtual machine. The core libraries allow programmers to develop their code using java programming language. Basically core libraries are in communicate with native libraries and Dalvik virtual machine. The developed code by programmer should be implemented in Dalvik virtual machine. Using this virtual machine can provide application portability. The format of file that can be used in this virtual machine is Dalvik Executable (DEX).

Application Framework Layer

Another layer is application framework which contains all Application Programming Interfaces (APIs) as it is shown in Figure 2.1. All services run in this layer and applications which have service can share their data or use other application resources. Activity manager is supposed to manage life cycle of applications. Beside this, notification manager notifies user by popping up the alerts.

Application Layer

The last layer is application layer. In this layer, all applications such as pre-installed and downloaded applications by user, work. All applications with their process, run on their own Dalvik virtual machine and they are isolated from each other and make

Android secured enough. The focus point of this thesis is on the last two layers and it has been tried to find vulnerabilities of these two layers.

2.1.2 Android Application and Service

Android application is a set of components which can be used to develop Android application. These components should be bounded with manifest of application to create file that can be installed on device. These components are activities, services, content providers, intents, broadcast receiver and notifications. In addition, each Android project has extensible markup language (XML) file, named as manifest file that describes the structure of application and components [13-14]. XML has some node tags such as application and user permission. In application tag, developer can define the content of the application which to have service or not. All the security explanations can be described in uses permission and permission tag. After developing Android application, platform makes installable file for device with Android Application Package File (APK) format. The icon of any installed Android applications can be seen in the menu of device but it is possible to hide this icon from user interface.

Moreover, google provides online market that developers can publish their application on the market for free or paid version. Regarding the terms and conditions of google play, developers are eligible and have right to develop Android application and upload them into market, if their developed applications do not have malicious activity.

Depend on the purpose of application, developer can provide service for application. Services always run on the background of device. Although application's service can be killed by user or memory management of the device, but there is possibility to

keep service working on device even if the user kills the process. The life time of Android application and service depends on their priority for the processor, which is shown in Figure 2.2. As it has been discussed about application, services can be malicious and novice users and even the common Android users do not care about them. Unfortunately if the service is malicious, it can control the device always.

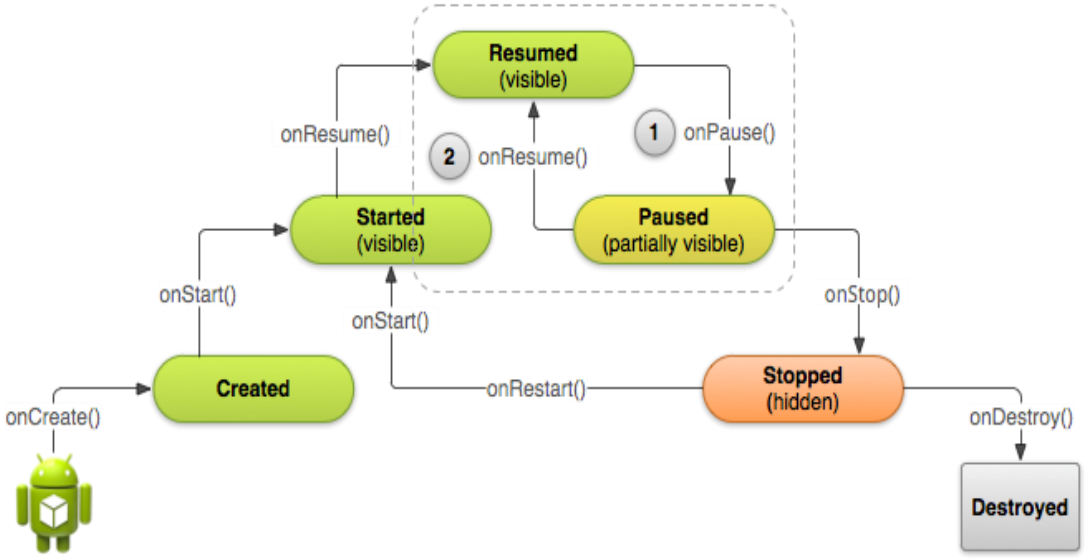

Figure 2.2. Android application lifecycle [15]

It has to be noticed that service without application could not run on the device, thus it is necessary for a service to be added to an application. Besides this, it is not possible to add a service to an application which is running on device.

2.2 Near Field Communication (NFC)

NFC is a new technology for transferring data between two devices, wirelessly within short range. Originally, NFC is based on Radio Frequency Identification (RFID) [6] and the speed of data transferring is 424 Kbit per second, shown in Figure 2.3.

Figure 2.3. NFC range comparison [16]

This technology is similar to Bluetooth in some way and becomes popular these days. The differences between NFC and Bluetooth is that expects of piconet, Bluetooth works just between two devices but NFC can work between two devices as well as device and tag. The NFC tag includes memory, power and antenna as it is shown in Figure 2.4. There are two type of tags based on their usage; writable tag and readable tag. In writable tag, NFC writer which can be device or application, can write and insert data into memory of tag but in readable tag, there is no possibility to write on tag again and just the previously written data can be read.

Figure 2.4. NFC Tag [17]

Moreover, NFC does not follow any authentication protocol and any pairing device same as Bluetooth and this can be measured as the weakness point of this technology. This weakness can be the cause of many attacks like eavesdropping, data modification, relay attack and so on. These attacks make NFC to be one of the most vulnerable communication technologies in the world.

2.3 Social Engineering

Social engineering [18] is not an engineering major or engineering course. It is all about art of human hacking. In other word, it is an action that makes people to do what might or might not be in their best interest. Social engineering can be done in three ways which are psychologically, physiologically and technologically. In addition, social engineering can be divided into many categories based on the purpose of use. Hackers and attackers, penetration testers, spies, identity thieves, information broker, scam artist, executive recruiters, sales people and government are some examples of social engineering.

Using social engineering for any attack can be helpful because there is no way to monitor people, for instance staffs of organization. In other words, it is the best way to trick people and gain desired information. Unfortunately, the easiest way to get access to the network of an organization is using social engineering for people who work that network. Based on cisco report [19], the number of social engineering attacks is increasing with the rate of 26% per month and this number can be good example of showing how social engineering works.

Ultimately, in this thesis, the process of victimizing Android device by using social engineering method and vulnerability of NFC for transferring malicious Android service is shown in next section.

Chapter 3

SYSTEM OVERVIEW

3.1 System Architecture

The proposed system is based on client server architecture. First of all, server is on waiting mode and client sends request to server. Then when server accepts, client automatically connects to the server. In this thesis, Android user becomes client and on the other side personal computer becomes the server. For establishing communication between client and server, request and respond functions are mandatory as it is displayed in Figure 3.1. By setting IP address and port number of server on client device, the connection between client and server can be established.

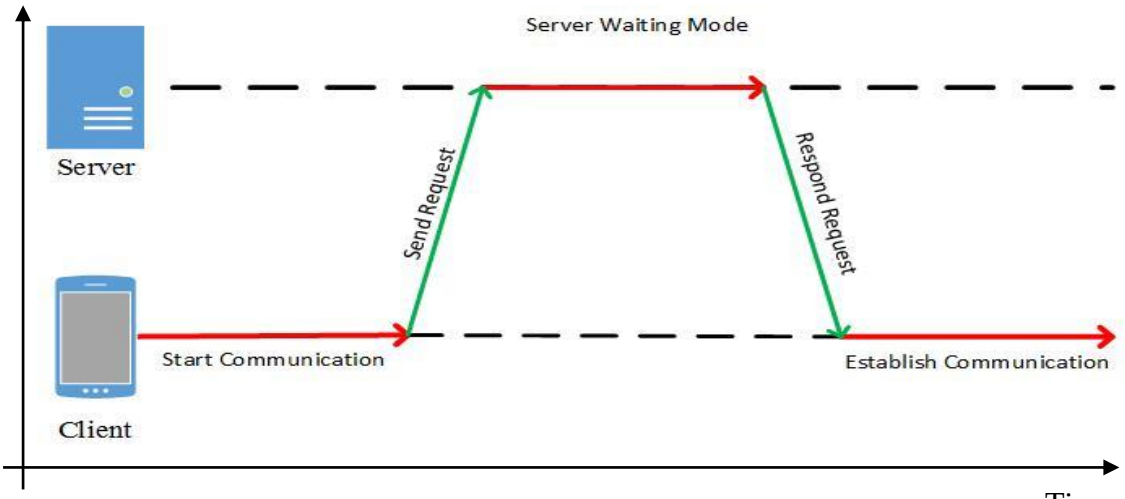

Figure 3.1. Client- Server communication

The proposed system can work on different types of networks, such as local area network (LAN) and wide area network (WAN). It depends on victim's location which can be in same network as an attacker is or in different network from attacker.

If the victim is in the same network as attacker is, just by setting IP address and port number, client server communication can start as it is shown in Figure 3.2.

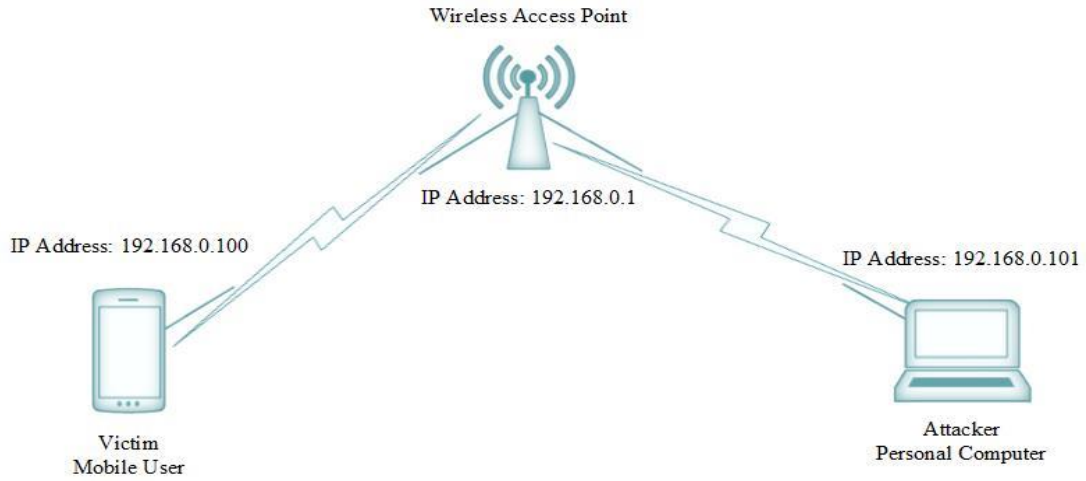

Figure 3.2. Attacker and Victim both are in the same LAN

In another scenario, if attacker and victim are not in the same network and attacker needs to open the port, getting access cannot be convenient as first scenario. To get success and overcome this problem, attacker must create a fake host in the Internet with static IP address.

To solve this problem, attacker has to register on no-ip website and create virtual host with static IP address. After registering for static IP address, the second step is to get virtual private network (VPN) server to open the port and connect to the virtual host. VPN server can be found in the Internet by paying 15 euro per month. However, as it is shown in Figure 3.3, after opening the port on server, attacker can get access to victim device without any obstacle.

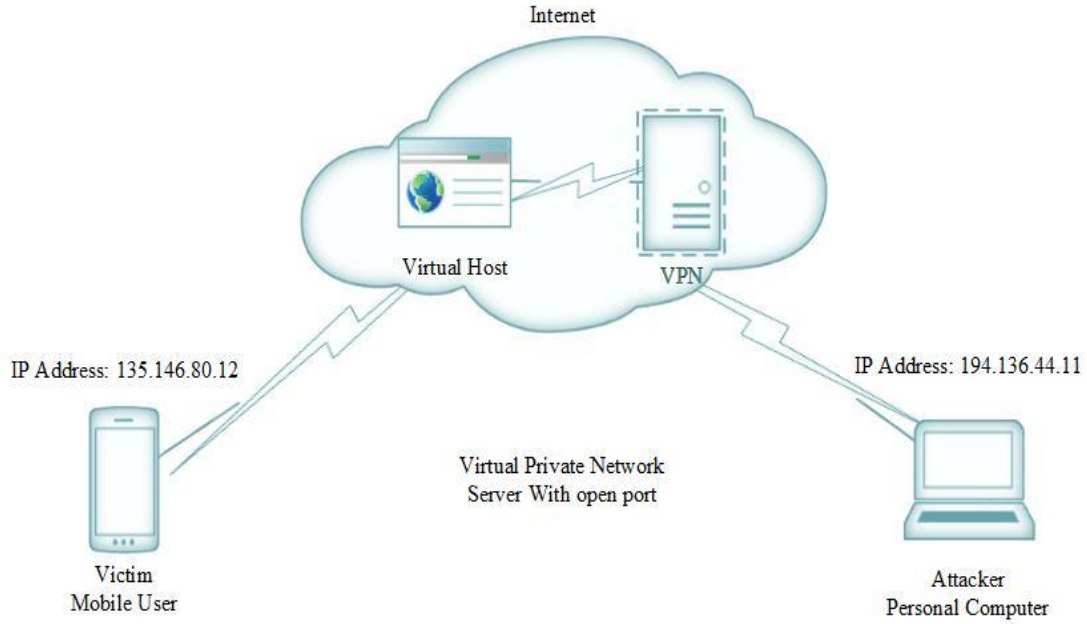

Figure 3.3. Attacker and victim in different LANs

3.2 System Description

In this section, explanation of proposed system is discussed in depth. This section can be categorized in three subsections which are attacker side, target side and communication library between attacker and target.

3.2.1 Communication Library

In each communication over TCP/IP there are two significant parts which are data and communication line between client and server. In this thesis, the client and server sides are called as target and attacker, respectively.

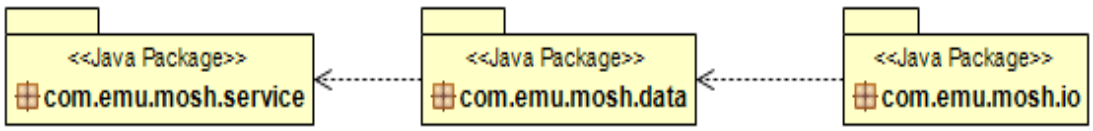

Figure 3.4. Library Packages

In this library, beside communication line and data there is another package for some of program features called as service, shown in Figure 3.4.

Data Package

Data package is the most important package in the communication library. The major step in this package is to prepare a list of desired information to be gained from targeted device. As it is shown in following figures, there are association between classes as well as relation. Associations are shown in diagrams with solid line plus the property which is transferring between classes and also dotted line is explained relation of classes. For each item of mentioned list, the related class is defined.

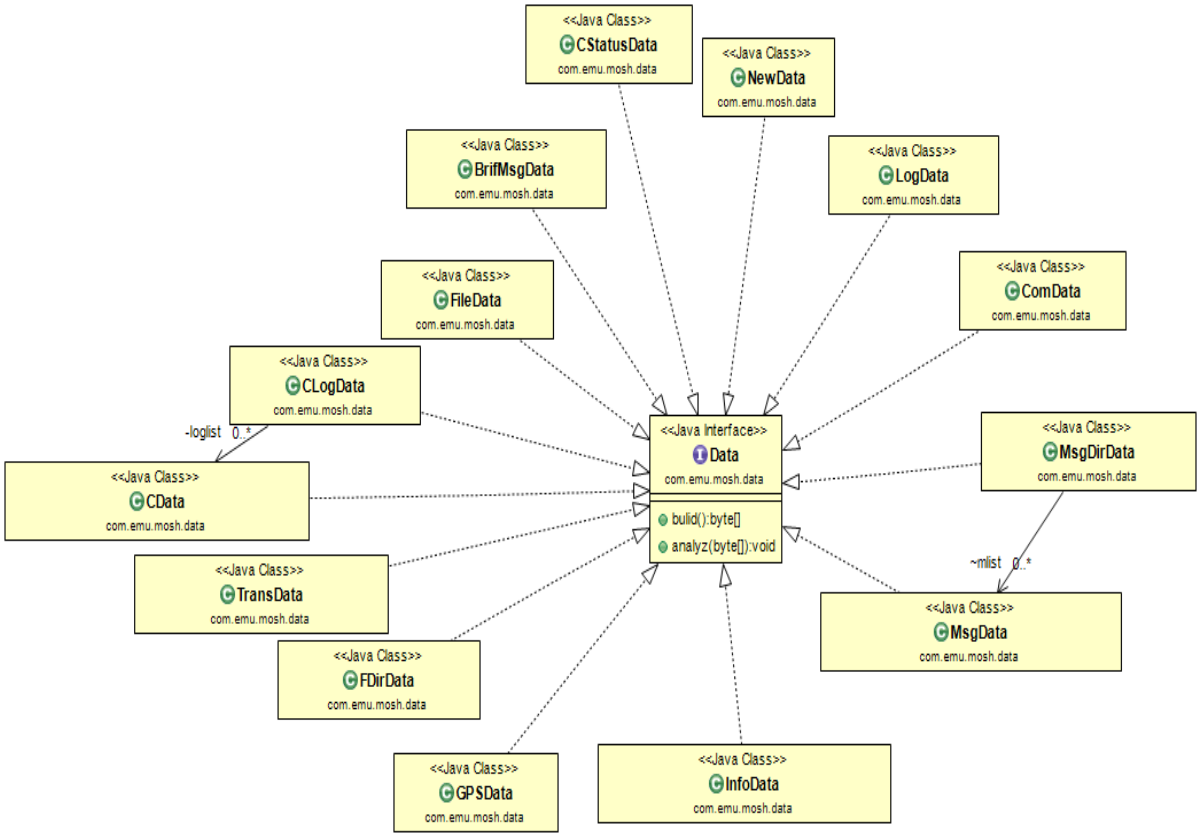

Figure 3.5. Data Package

After defining mentioned classes, buffering data is essential. In this package, interface class which is called as Data.java is contained two important methods, namely build and analyze. All other classes have been implemented on this interface. As shown in Figure 3.5 the first method is responsible for building byte and second

method parses and analyzes the data. By providing place for buffering and parsing data, this package of the library is completed.

Service Package

The other package in communication library is service. For some features in application such as voice, a converter is required for converting audio to byte and sending it over communication line, as it is shown in Figure 3.6. Beside this, for showing map and downloading file, converting is also required. So by using simple mentioned method reaching to purpose is not difficult.

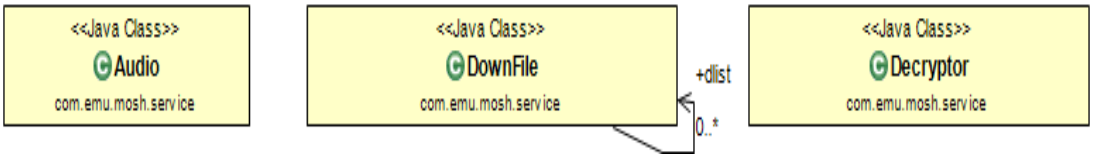

Figure 3.6. Service Package

IO Package

For establishing connection over Transmission Control Protocol / Internet Protocol TCP/IP, there is need to have some rules to setup the communication. These rules are explained in Setup.java class. In this class (Appendix) which works as a protocol for communication, all needed fields are mentioned for making structure of data packet which is transmitted over Internet Protocol. Input signal and output signal that works like multiplexer and de-multiplexer are necessary. Output signal is send by sender and input signal is received by receiver. Beside these classes, there is a manager class to manage connection as it is shown in Figure 3.7. In Connect.java class socket programming is used to build socket by identifying IP address and port.

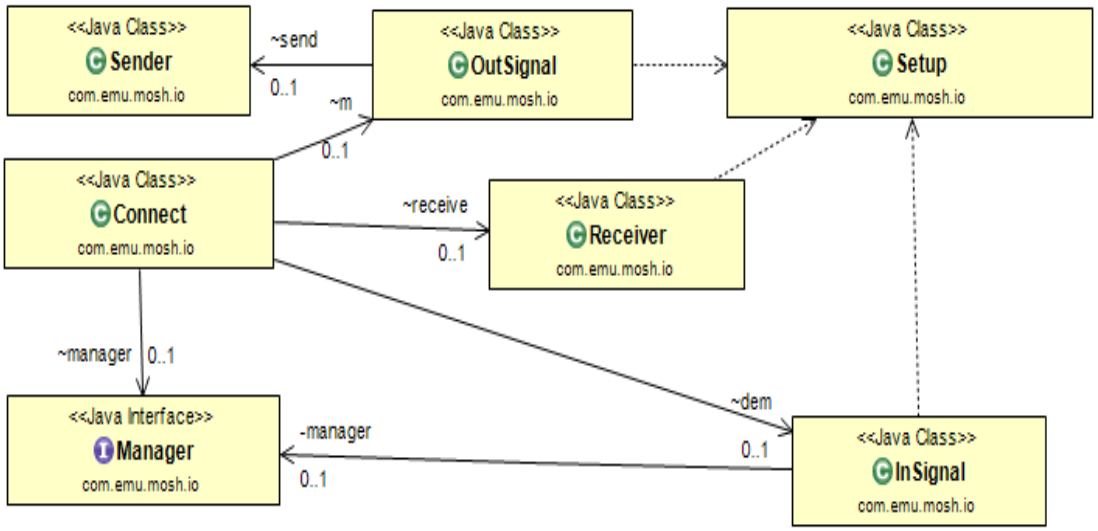

Figure 3.7. IO Package

3.2.2 Server or Attacker Side

As it has been mentioned in previous part, in client server architecture there are three important parts called as client, server and communication between them. After providing communication, server or attacker side has to be implemented. In the sever side of this application, there are four packages which are board package, manager package, user interface package and server package, as it is presented in Figure 3.8.

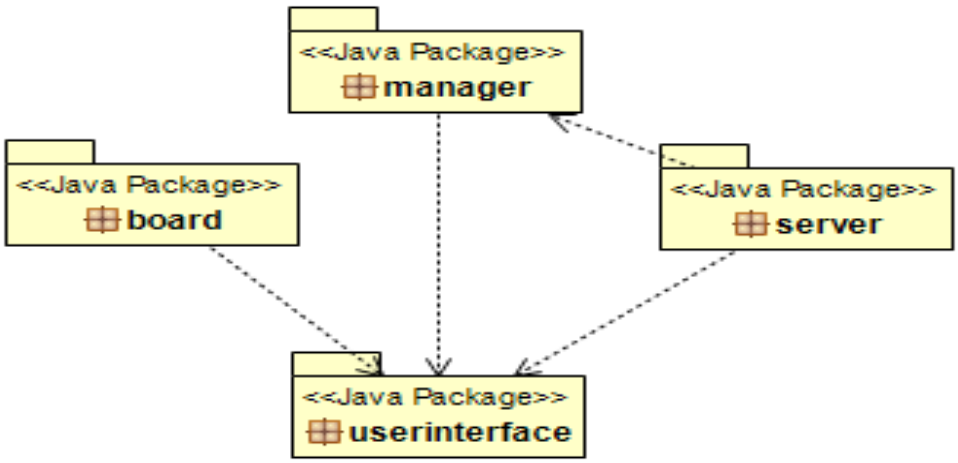

Figure 3.8. Server or attacker side

Board Package

The server side of proposed application requires a user interface for each task, since this part of application is an executable Java Archive (JAR) file. As it is shown in Figure 3.9, for each task, there is a graphical user interface. These interfaces has been made graphically and Eclipse Integrated Development Environment (IDE) which generates the required code for each one of them automatically by using two libraries called as Abstract Window Toolkit (AWT) and Java Panel (JPanel). For adding more tasks for each class, the required code can be implemented easily.

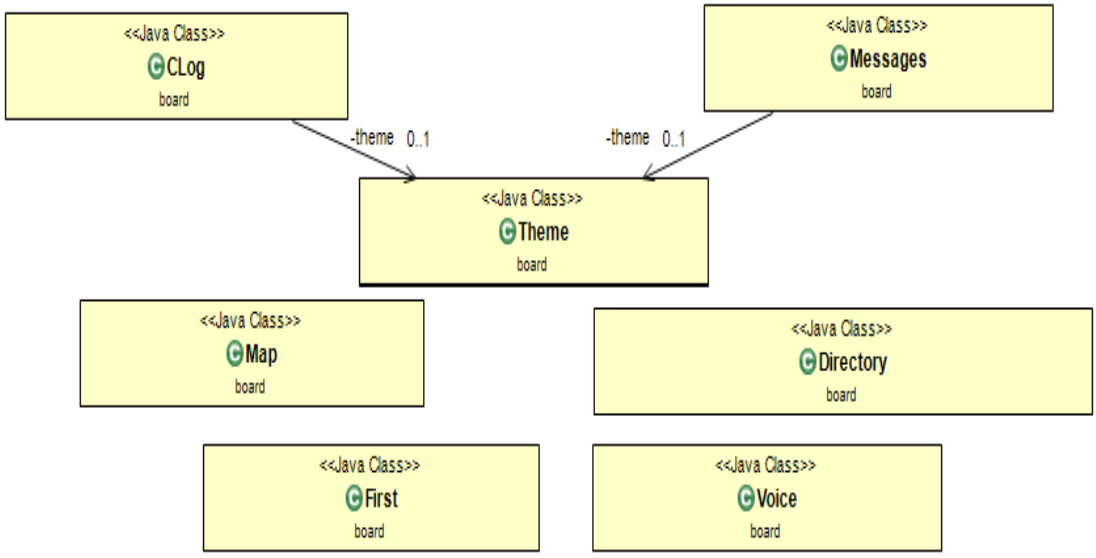

Figure 3.9. Board Package

Server Package

There are two essential classes called as server and target manager in this package. The task of target manager class is to handle multiple targets connected to the server, by channelizing the connection for each. This class is connection between library and server. Server class is the main part of application which is related to IO package of application and the manager package. The main task of server class is to run the

server as well as handling all exceptions which might be occurred in application, as it is illustrated in Figure 3.10.

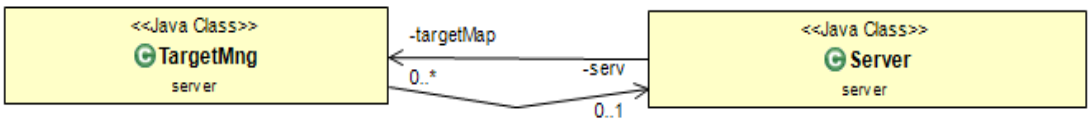

Figure 3.10. Server Package

Manager Package

The interface of this class is called as data manager which is the interface for all other classes. In the mentioned interface, there are two main methods which are responsible for different tasks as it can be seen in Figure 3.11. First method, which is receive method, tries to get data based on the International Mobile Equipment Identifier (IMEI) of target device. Data manager is the second method and this method is responsible for sending prepared data to the server class for further processing. The manager package is responsible for handling data, based on IMEI of target device. Also, manager package is responsible for writing command for each activity which has been done in the application. The written command is displayed in a predefined place in the user interface.

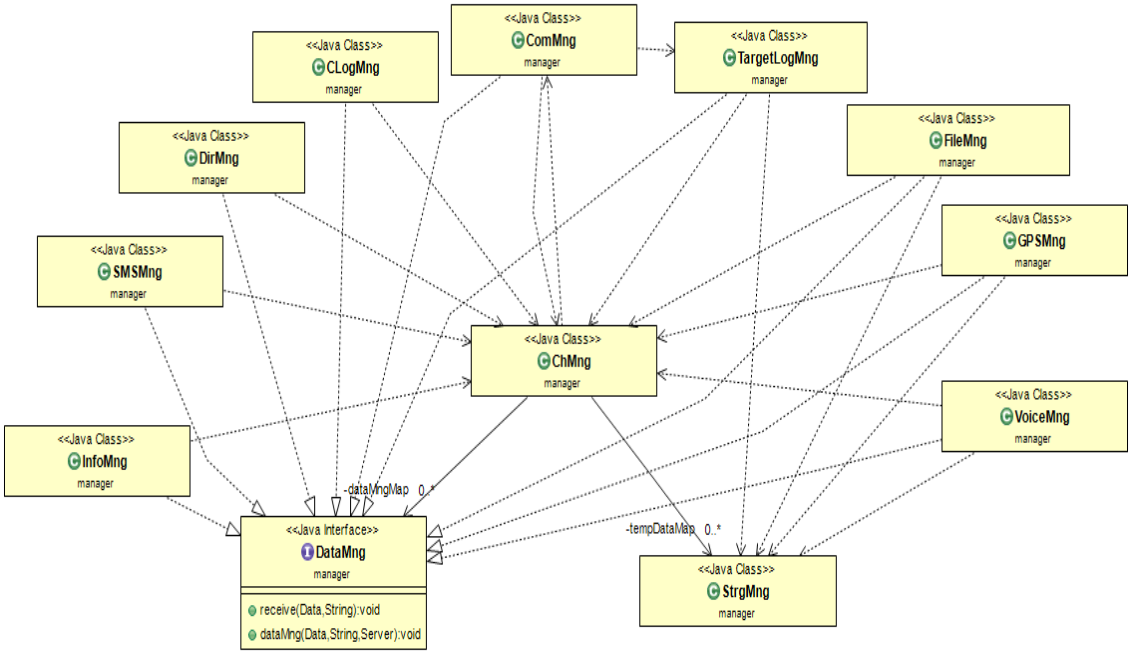

Figure 3.11. Manager Package

User Interface Package

Before creating user interface for application and managing different targets, a target model must be created. After providing the main user interface and adding required functions, attacker will be allowed to run different functions on target device. This package is connected to all panels which have been designed in board package. As it is shown in Figure 3.12, the connection of two main classes of this package is interface of different targets. Making call and sending message are extra features of this application. These extra features have different user interfaces and they are not inherited from board package, as it is shown in Figure 3.12.

In about us section, the usage of this application is explained and it is mentioned that the demonstrated application is only for educational purpose. This is because of the functionality of this application, which can be used for destructive purposes.

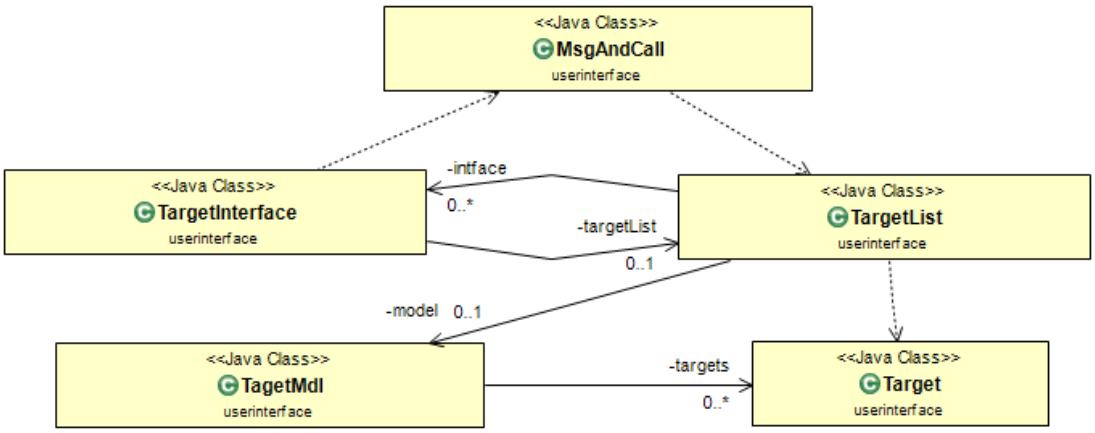

Figure 3.12. User Interface Package

3.2.3 Client or Target Side

The target side of this application is written for Android OS by using java programming language. The minimum Software Development Kit (SDK), which is used for this part of application is 8. So, this target side would be run on device with minimum SDK 8 and above. Since there is a bug in importing JAR library to the Android application, the library must be directly imported to the coding part. In this part of application, there are two vital packages which are target and connection packages, as it is shown in Figure 3.13.

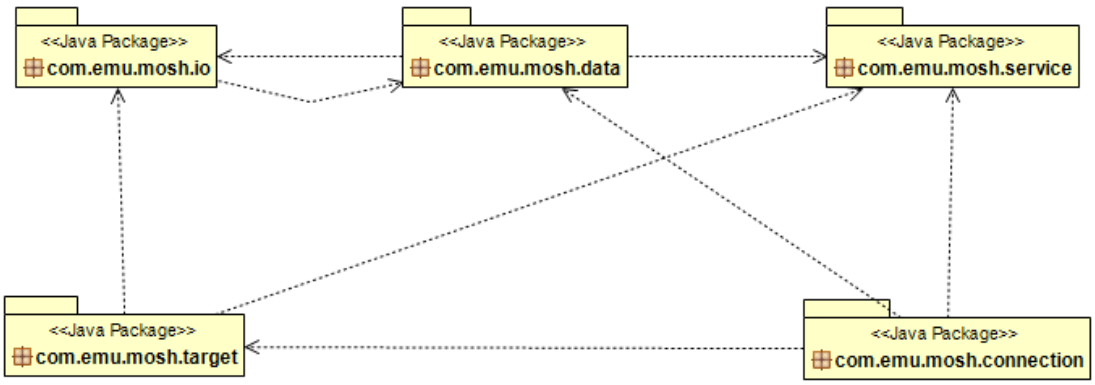

Figure 3.13. Target Side

Connection Package

All functionalities that attacker is looking for, have been explained in this package. This package does not include any Android activities or interface and its task is to gather required data and send it to the target, by using intent activity in target package. Besides, this package is connected to proposed library which has been imported to application, as it is illustrated in Figure 3.14.

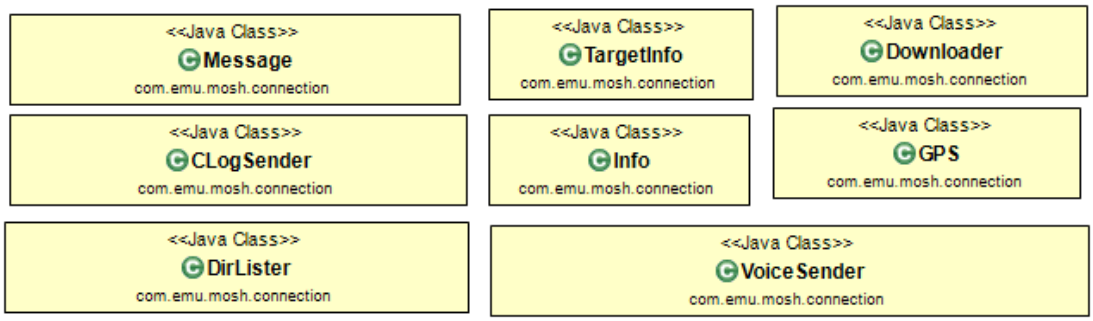

Figure 3.14. Connection Package

Target Package

All tasks in this application would be managed by target package. One of the classes in target package is listener class, which is responsible for streaming data provided by connection package, to the attacker device. The listener class works as a service and this service provides ability for the application to be rerun, if the device restarts. Server side of application is supposed to be on waiting mode to get signal from target device (Appendix). These signals are sent by target class, which is in cooperation with command class and listener class. For the first time, the application must be started manually by pressing provided button. After this step, the service of this application runs at the background without any human interaction. The provided button for this purpose, has been defined in activity class and the graphical part has

been designed in Extensible Markup Language (XML) file which has communication with activity class, as it is shown in Figure 3.15.

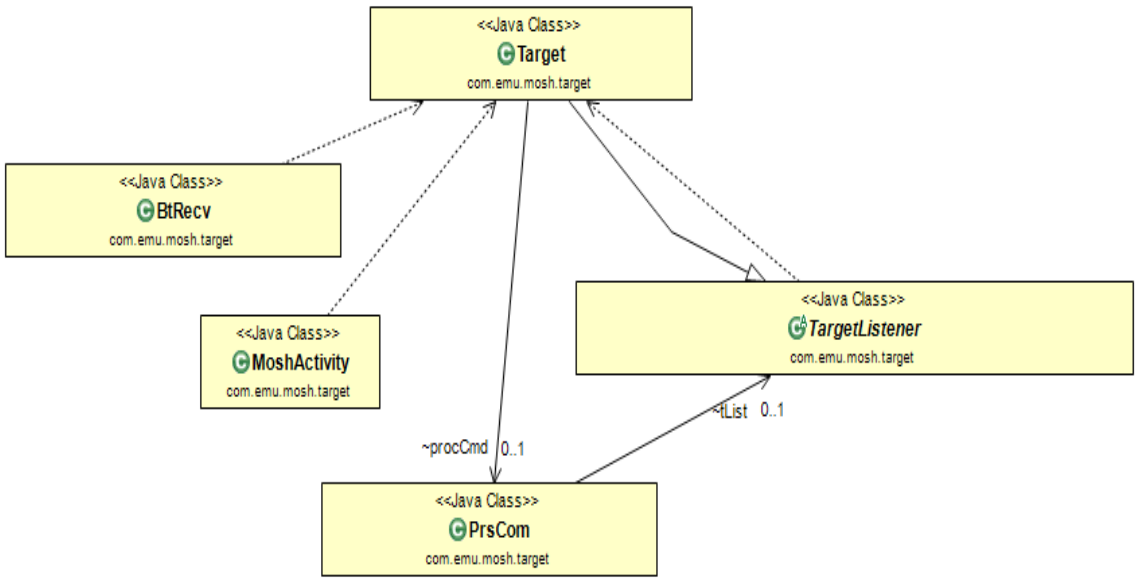

Figure 3.15. Target Package

3.3 System Overview (Attack Vector)

In this section of thesis, description of system is illustrated step by step.

Step 1

First of all target must be identified and the information gathering phase starts. In information gathering, attacker has to find any valuable information about victim. The required information can be gathered by searching on Internet, having conversation with victim and checking social networks' profile of victim. Information which is related to victim's desire and interests can be so much valuable for attacker.

Step 2

After finding valuable information about victim, changing the interface of application based on target's need and interests is the second step. The great advantage of this application is having customizable interface and attacker would be

able to use this application for any purposes. Thus by the end of this step, attacker has provided suitable user interface for target. The process of creating target's desired application and leading target to use mentioned application is called as social engineering.

Step 3

Generally Android users trust authorized stores such as google market, which provides various Android applications. As Android users trust such these markets, they prefer to download their applications from these places. Google investigates the provided applications before publishing in the market, from security point of view.

In case of being legal, the application can be published in the market to be available for users to download it. Besides this fact, each application before downloading asks the user for some permissions for specific features provided by application. These permissions allow user to be aware of accessibilities of the application.

Regarding the mentioned facts, this application cannot be uploaded on the authorized stores and also the permissions will let users know about the dangers that this application can cause for them. So for solving this problem, this application will be uploaded as a simple application which provides the target's desires. After downloading the application from trustable stores the device will be leaded to another source for updating the downloaded application. This process will be done by tapping NFC tag.

Step 4

After updating mentioned application on victim's device, by pressing the provided button, device will be automatically connected to attacker device and the hacking phase will be started.

Step 5

As it has been explained in previous steps after establishing connection between victim's device and attacker, all the required information can be collected. So the attacker can have access to message boxes, contact logs, files and directory of device which is included of images, videos, music and other formats of files.

Chapter 4

IMPLEMENTATION AND RESUALT

4.1 Features

As it has been mentioned in previous chapters this application uses client server architecture and it works over TCP/IP connection. By connecting target's device to the server, the following features can be done.

- Sending screen message
- Device vibration
- Retrieving general information of device like SDK number, IMEI and operator name
- Getting file directory of device
- Download any file from device
- Getting call log based on incoming, outgoing and missed calls
- Search in call log based on number
- Getting stored messages from inbox or outbox messages
- Search in messages based on number
- Turning device's microphone on or off
- Recording voice from phone calls or device's microphone
- Finding location of target on map based on the network or GPS
- Making call
- Sending Message

4.2 System Result

Result of discussed system is illustrated graphically in following parts of this chapter. All screenshots have been taken from the implemented application. Both server side and client or target side are shown separately.

4.2.1 Target or Client Site

As it has been mentioned in previous chapter, after transferring the implemented application to target device and installing, application automatically will be run. The following user interface, for instance, has been prepared for victims to make them interested in running the application. The mentioned user interface can be seen in Figure 4.1.

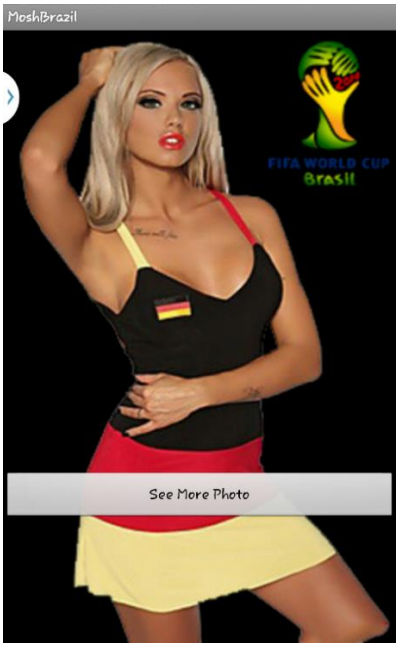

Figure 4.1. Android application interface

In this example, by pressing provided button as it is shown in Figure 4.1, application will be connected to server and the Android device becomes victimized. The major feature of this application is having a service which can run on the background of device and novice users cannot understand about its existence.

4.2.2 Attacker or Server Side

In this part, as it is shown in Figure 4.2, server's functionality has been illustrated. The first screenshot shows the connected victim to the server. For each device which has been connected to this server, new row will be popped up in the list. Each row contains some information such as country of victim, IMEI of victim's device and operator. Besides, each activity which has been done will be saved and displayed as a command.

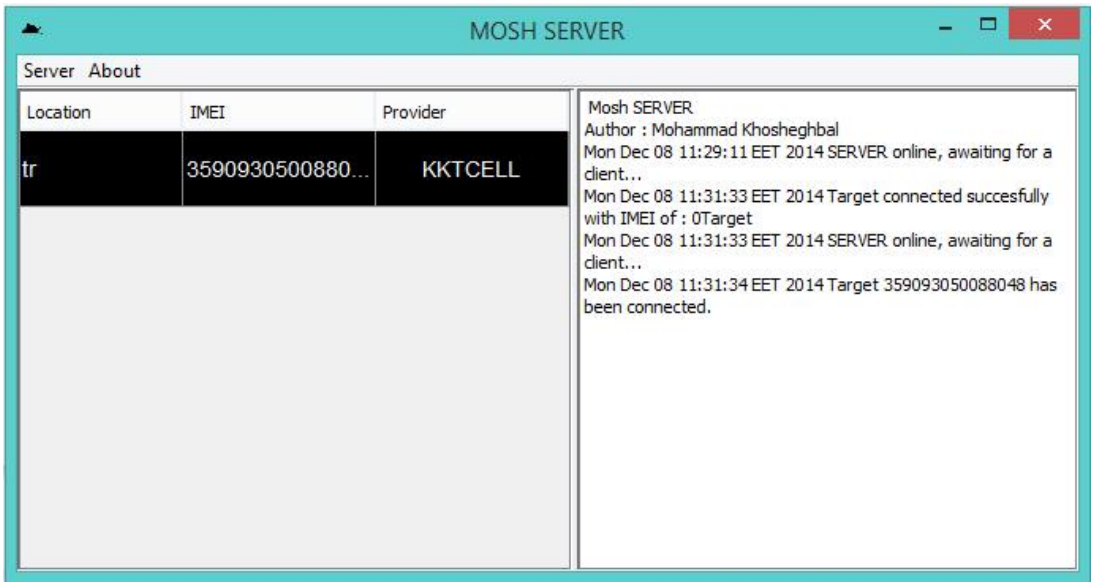

Figure 4.2. Main Page of Server Side

In addition to mentioned features, before connecting target to the server, attacker can change the port of connection. This feature can be useful in some networks with specific firewall policies which can block the desired port. With this action, always there will be a way for establishing connection with target, as it is shown in Figure 4.3.

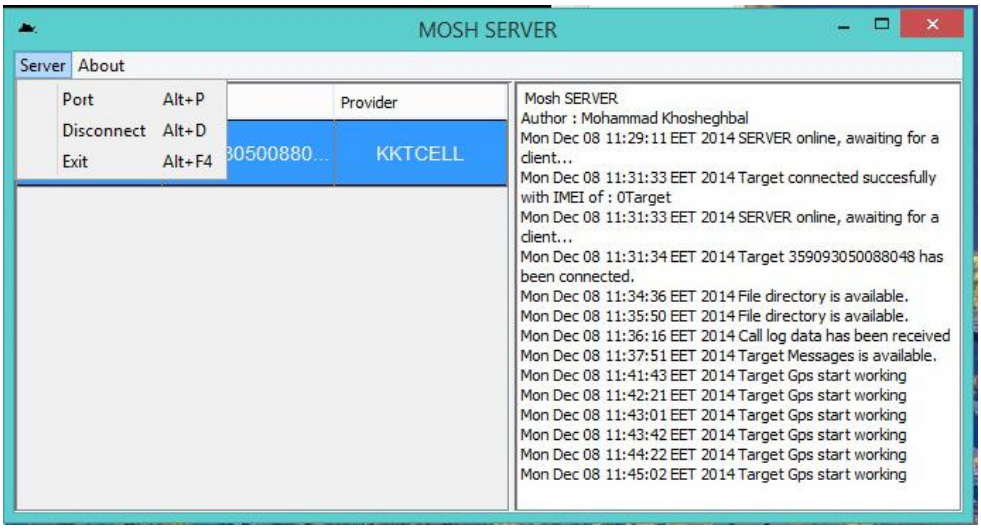

Figure 4.3. Server Menu Bar

By double clicking on each connected target in the list, new menu will be popped up which allow attacker to see some device information like IMEI code, country, operator, Android version and version of SDK.

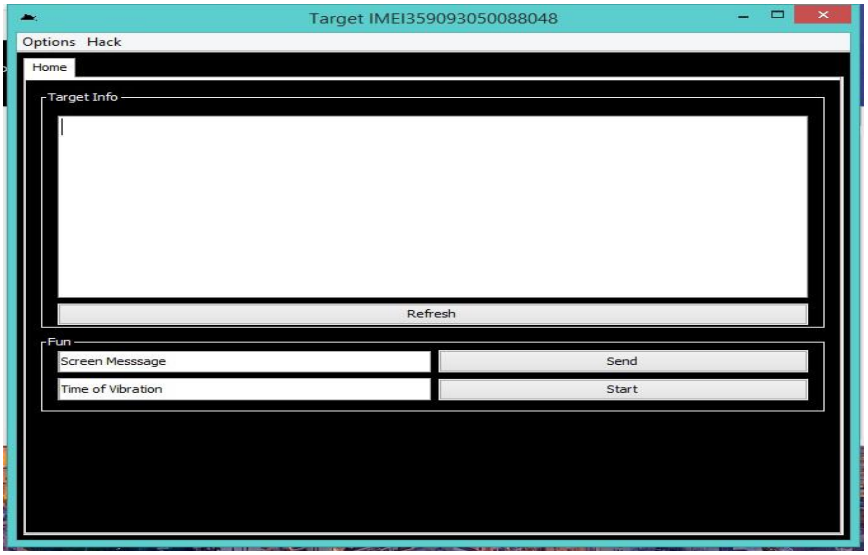

Figure 4.4. Target Information Panel

In addition to these information, in another panel, called as fun panel, there are some functionalities such as sending toast message to target or set time for vibrating device as it is shown in Figure 4.4.

Before this stage, attacker gained some information about device but the most important part of this attack is left, as it is presented in Figure 4.5. On the top left side of information panel, in the menu bar another button has been prepared and called as Hack. By pressing this button, another menu will be opened which shows the most vital functionalities of this application.

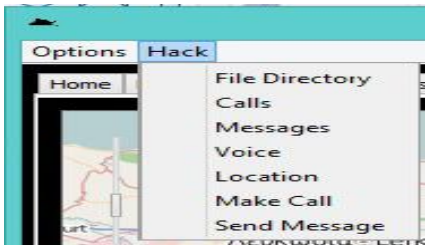

Figure 4.5. Hack Menu

The first option of this menu, as it is presented in Figure 4.6, is file directory. This function is responsible for displaying all stored files on victim's device. These files can be downloaded from victim's device to attacker's device.

Figure 4.6. File Directory Panel

Although getting view of stored information on victim's device could be time consuming, but it can be much valuable information for further attacks.

The second option is getting all call logs from victim's device. In the search box, there is an option for choosing type of calls which has been shown specifically in Figure 4.7.

Figure 4.7. Call Type

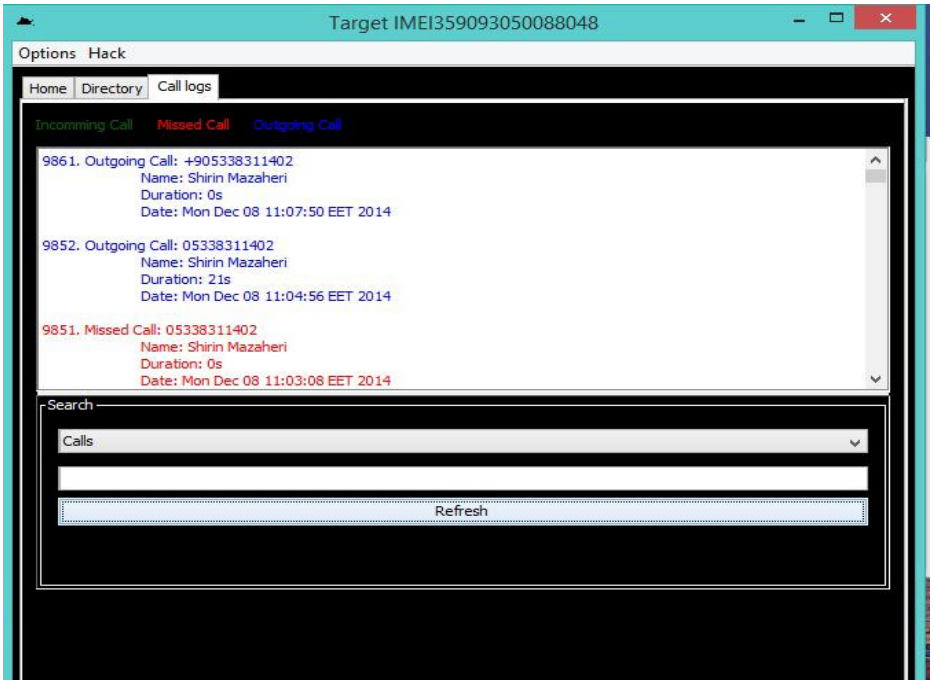

Figure 4.8. Call Log Panel

By choosing each type of call, all the information based on the chosen type will be displayed in the panel, shown in Figure 4.8. The provided information includes name, duration of call and date. Also, the displayed call logs in the panel have been categorized based on the type of call. Missed calls will be displayed in red, outgoing calls in blue and incoming call will be shown in green.

The third option of this list is related to retrieving all messages from victim's device. For this option, attacker can have all messages either sent or received. In the search box, there is a dropdown list which provides different type of message boxes such as inbox or outbox. Attacker will be able to choose the desired message box from the mentioned dropdown list. By choosing each one, the message in the chosen box will be displayed in provided panel, shown in Figure 4.9. Each record will be shown with some specifications such as body of message, recipient number and date of message, Information from each message box is planned to be shown in different color for ease of use. In this panel, another ability has been provided for the attacker which is ability to search. So, the attacker would be able to search for desired message based on phone number.

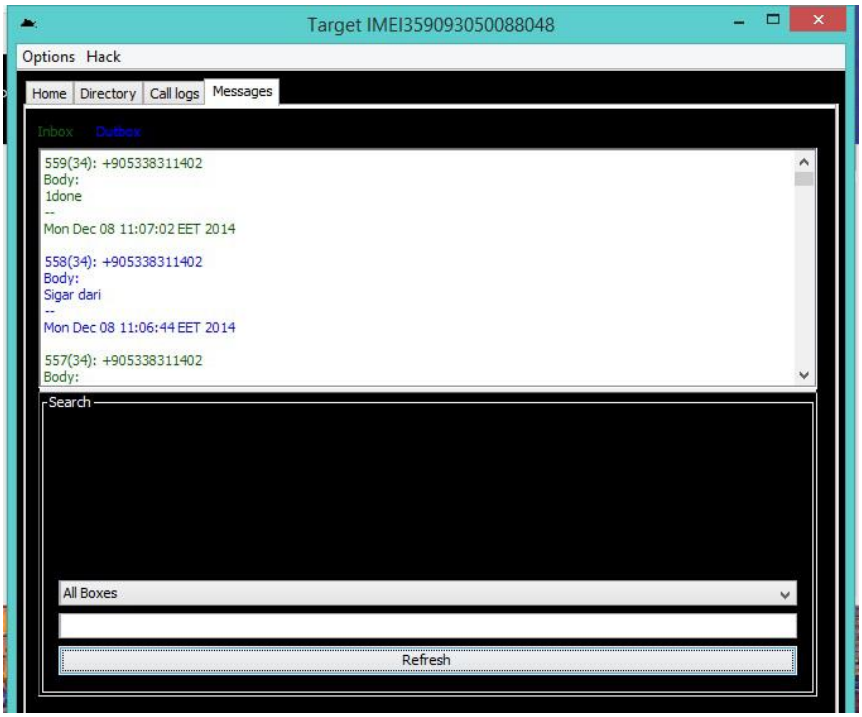

Figure 4.9. Message Panel

The fourth section of this panel is related to voice which can be the most interesting part of this application. It provides the ability of recording each conversation that

goes through the microphone of victim's device. Besides recording incoming or outgoing calls, the microphone of target device can be turned on and attacker will be able to listen to all voices around the victim's device. (Figure 4.10).

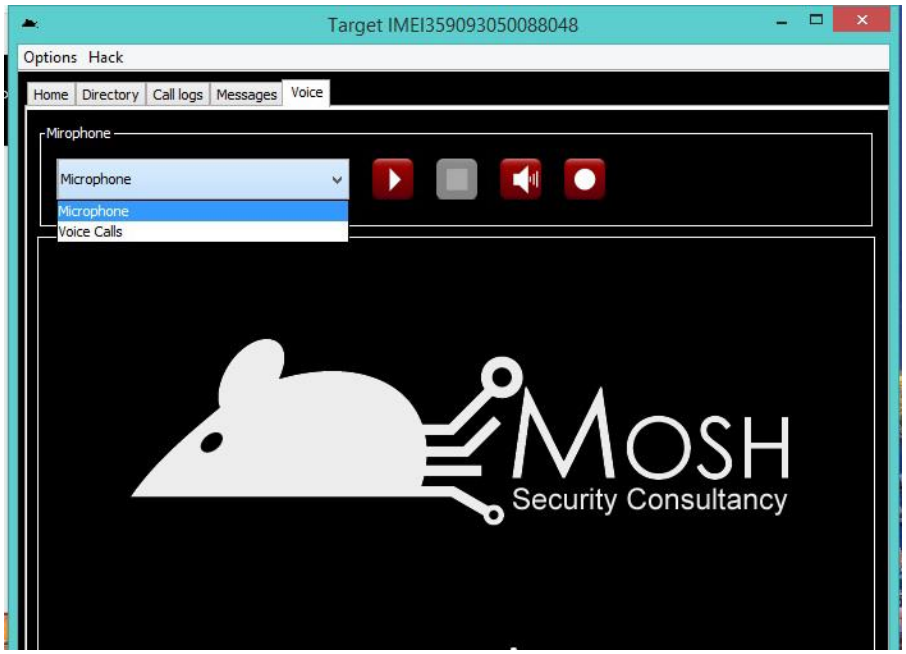

Figure 4.10. Voice Panel

In the fifth section of this panel, finding the location of device is the major functionality of this part, as it is shown in Figure 4.11. Location of device can be found in two ways. The first way is tracking device based on GPS coordinates such as longitude and latitude of the location and the second way is to trace the route of device based on the network, which the device is connected to.

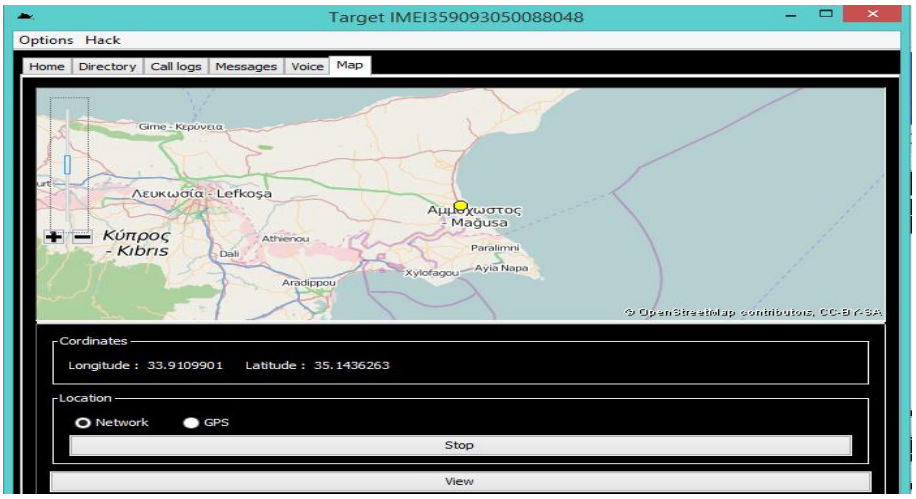

Figure 4.11. Location Panel

Two last sections of this panel, have been provided for using victim's device to make phone calls and send messages. So one of these sections, as it is shown in Figure 4.12, will be a new form which allows attacker to enter desired phone number and by pressing ok button, the call will be made using victim's device.

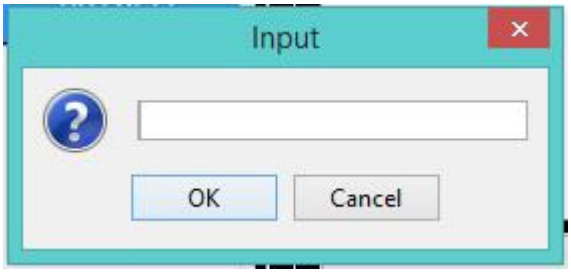

Figure 4.12. Call Panel

The other section, shown in Figure 4.13, will provide ability of sending text messages using victim's device. In this form, attacker will be able to write text message and enter desired phone number, the by pressing send button the message would be sent from victim's device.

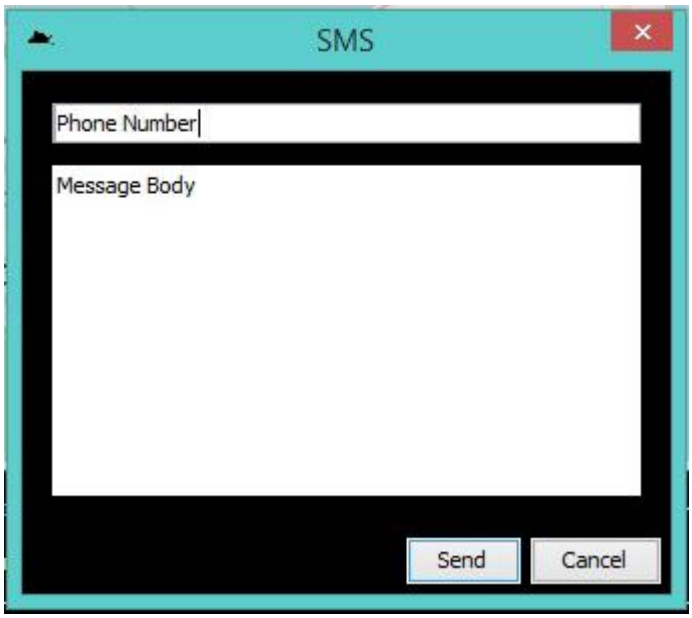

Figure 4.13. Sending Message Panel

4.3 Similar Works

By the end of 2013, the number of applications that work maliciously on Android devices reached to one million, as it is shown in Figure 4.14. Near twenty percent of these applications are hidden behind a legal application and work on target device without owner's awareness. These applications work with different purposes such as making expenses for target, adware, data stealing, remote controlling, malicious downloading and hacking tools.

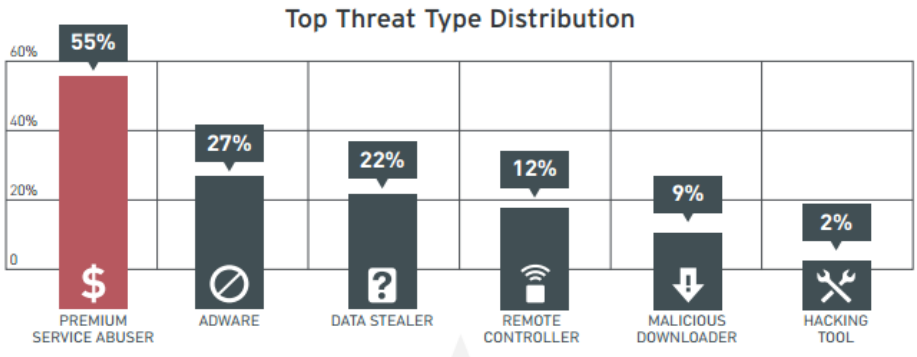

Figure 4.14. Purpose of similar applications [20]

Some of the most famous malicious applications, which in some functionalities are similar to this application, based on their popularity of usage are shown in Figure 4.15. Some of these application are described after following graph [21].

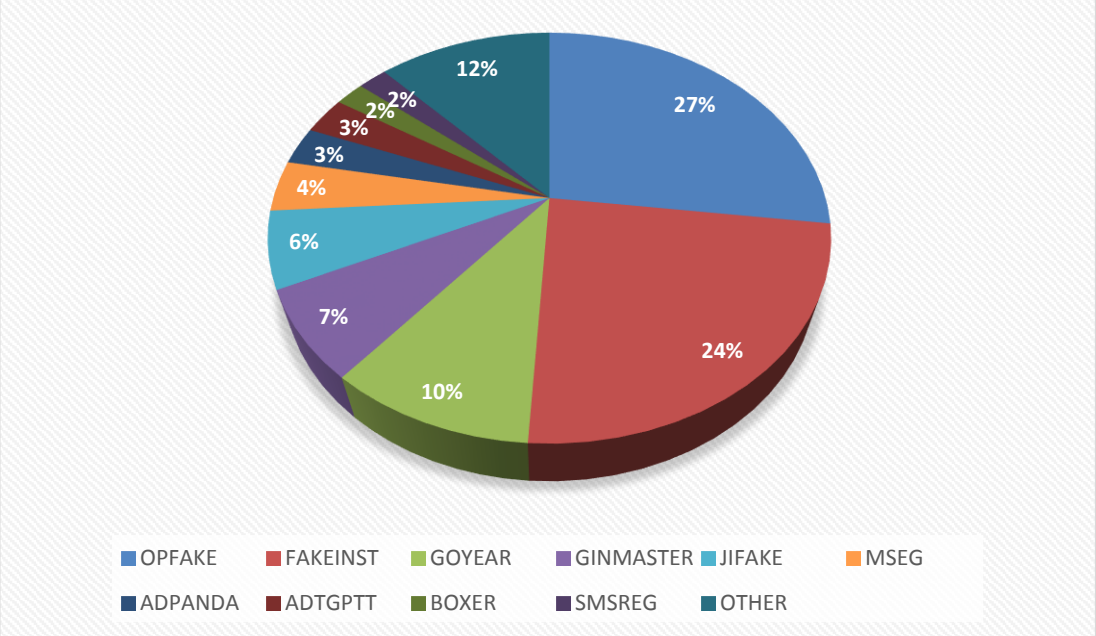

Figure 4.15. Similar Applications

OPFAKE

By allocating twenty seven percent of total malicious Android applications, presents itself as one of the dangerous applications in the world. OPFAKE is malicious application that works as a downloader on Opera and Google Chrome browsers. This malicious application works as a Trojan by sending messages to other devices. This application was found in 2012 [21] and its developers start to make this trojan for Symbian and IOS platforms as well. This application can be distributed in different ways such as using social engineering for tricking people to make them interested for having this application.

FAKEINST

FAKEINST gets twenty four percent of the malicious Android application market. This application sends a huge number of Short Message Services (SMS) to different Android devices. It is hidden in many applications offered in Android markets and the most of these applications are popular games. This malicious application has been found in Eastern Europe, Asia and Russia.

BOXER

Boxer is malicious Android application which has been designed to send many SMS in order to decrease the charges. This malicious software can be hidden in legal applications which are offered in Android markets. This malicious applications have been detected in many parts of the world such as Europe, Latin America and Asia.

GINMASTER

GinMaster or GingerMaster malicious application has been found in 2011 by university of North Carolina [21]. This application by allocating seven percent of total malicious Android applications, presents itself as one of the dangerous applications in the world. This application also hides under cover of another application that seems to be legitimate by using social engineering technique and works in the background of victim device. This trojan with rootkit functionality, will be installed on the root of device for transferring some valuable information like call logs, stored system files and other important data.

4.4 Aims

Each system can be used either in destructive purpose or ethical purpose. Some of these applications' beneficial purposes are listed below.

- For investigating suspect person in case of need, by authorized agency
- For monitoring children

Also as it has been mentioned above, it can be used for harmful purpose which is the main goal of this thesis. If this system being installed on an Android device, attacker can get full access from that device. Getting full access of device can be contained of all information such as contacts, SMS, call log and the entire stored files. Beside this, there are some functionalities in this system that help attacker even to make call, send SMS, find the victim's location and monitoring voice calls by turning the device's microphone on.

Chapter 5

CONCLUSION

The implemented application has been tested and the results showed that this system would allow the user to spy on desired target only by transferring the application on his or her smartphone which is running Android operating system. This system shows that Android operating system's security breaches could potentially be misused. Although the proposed system can be used for helpful purposes, as it has been mentioned before, it can also be used in malicious purposes as well. Thus, the vulnerability of Android operating system had been investigated to obtain the desired result. The similar systems exist on different operating systems but not on Android OS, so the implemented system is a new idea for Android operating system. To sum up, the malicious service of this application will be transferred to identified victim by using social engineering method based on target's desires. By running application and its sticky service, target will be always available for attacker for further attack and victim can be used as a zombie for reaching to other devices as well.

REFERENCES

[1] Tom Farley, Mobile Telephone History URL: http://www.privateline.com/archive/TelenorPage_022-034.pdf

[2] L. Darcey & S. Conder, Android Wireless application Development, Vol. 2, Third Edition, 2012

[3] R. Meier, Professional Android Application Development, Vol. 2, First Edition, 2010.

[4] Android Developer URL: http://developer.Android.com/about/index.html

[5] Alcatel Lucent Annual Report URL: http://www.kindsight.net/sites/default/files/Kindsight-Q3-2013-Malware-Report-final.pdf

[6] Near Field Communication (NFC) URL: http://www.nearfieldcommunication.org/

[7] MD. Zaber Tauhid Abir, New Version of Android URL: http://www.academia.edu/4431998/All_ANDROID_Versions_and_Kitkat_4.4

[8] Daniel Switkin, Android Application Development, URL: http://moss.csc.ncsu.edu/~mueller/g1/lecture3.pdf

[9] Dominique A. Heger, Mobile Devices - An Introduction to the Android Operating Environment - Design, Architecture, and Performance Implications, URL: http://www.dhtusa.com/media/AndroidInternals.pdf

[10] Anatomy Physiology of an Android, URL: https://sites.google.com/site/io/anatomy--physiology-of-an-Android

[11] Introduction to Android Development, URL: http://www.appdevtutorials.com/introduction-to-Android/

[12] Android Development URL: https://source.Android.com/devices/

[13] Android Application Life Cycle, Android Developer, URL: http://developer.Android.com/training/basics/activity-lifecycle/pausing.html

[14] Rodolfo Gomes, Introduction to Near Field Communication, 16th IST Mobile and Wireless Communication summit, July 2007, URL: http://www.stolpan.com/uploadfiles/1_Mobile_Summit_Budapest_NFC_TechnicalIntroduction.pdf

[15] Jan Kremer, Near Field Communication, URL: http://jkremer.com/White%20Papers/Near%20Field%20Communication%20White%20Paper%20JKCS.pdf

[16] Young-Sik Jeong, Young Ho Park & James J., Ubiquitous Information Technologies and Applications, Lecture Notes in Electrical Engineering, Springer, Volume 280, 2014, pp 363-371

[17] Christopher Hadnagy, Social Engineering Art of human hacking, URL: http://www.it-docs.net/ddata/121.pdf

[18] Cisco Report, Protect against Social Engineering, URL: http://social-engineer.org/wiki/archives/AttackersMightUse/Ciscosocial-engineering.html

[19] Ebrahim Al Alkeem, Chan YEob Yeun & Joonsang Baek, Secure NFC Authentication Protocol Based on LTE Network, Ubiquitous Information Technologies and Applications, Lecture Notes in Electrical Engineering, Springer, Volume 280, 2014, pp 363-371

[20] Android Development URL: https://source.Android.com/devices/

[21] Wireless Communication Standard URL: http://www.ampmpr.com/nfc-tap-to-change-the-world-and-your-wallet/

[22] SmartBot URL: http://www.overdriverobotics.com/SmartBot/smartbot-development-section/nfc-tag-in-the-robot/

[23] Dave Lee. (May 9, 2011). BBC Mobile. “Is ‘Open’ Killing the Android?” http://www.bbc.co.uk/news/uk-13284156

APPENDIX

Service source code

```
        @Override
        public int onStartCommand(Intent intent, int flags, int
    startId) {
                if(intent == null)
                        return START_STICKY;
                String who = intent.getAction();
                Log.i(TAG, "onStartCommand by: "+ who);

                if (intent.hasExtra("IP"))
                        this.ip = intent.getExtras().getString("IP");
                if (intent.hasExtra("PORT"))
                        this.port = intent.getExtras().getInt("PORT");

        }
```

Protocol source code

```
public class Setup {

    public final static int HeadLen = 15;
    public final static int DataLen = 2048;
    public final static int DataLost = 0 ;
    public final static int NoData = 1;
    public final static int ErrLen = 2;
    public final static int CompCom = 3 ;
    public final static int CompData = 4 ;
    public final static short Correct = 0;
    public final static short Wrong = 1;
    public final static short Connection = 2;
    public final static short ECmd = 3;
    public final static short Information = 4;
    public final static short Disconnection = 5;
    public final static short PrefSet   = (short)  20 ;
    public final static short PrefGet   = (short)  21 ;
    private static short DataSetup = 100;
    public final static short GpsGet = (short) (DataSetup + 0);
    public final static short GpsStart = (short) (DataSetup + 1);
    public final static short GpsStop = (short) (DataSetup + 2);
    public final static short VoiceStart = (short) (DataSetup + 4);
    public final static short VoiceStop = (short) (DataSetup + 5);
    public final static short InfoGet = (short) (DataSetup + 8);
    public final static short ScreenMsg  = (short) (DataSetup + 9);
    public final static short ContGet = (short) (DataSetup + 12);
    public final static short MsgGet = (short) (DataSetup + 13);
    public final static short FileDir = (short) (DataSetup + 14);
    public final static short FileGet = (short) (DataSetup + 15);
    public final static short CallMake = (short) (DataSetup + 16);
    public final static short MsgSend = (short) (DataSetup + 17);
    public final static short CLogGet = (short) (DataSetup + 18);
    public final static short AdInfoGet = (short) (DataSetup + 21);
```

```
public final static short Vib = (short) (DataSetup + 23);
private static short DataRes = 200;
public final static short GpsData = (short) (DataRes + 0);
public final static short DGpsStart = (short) (DataRes + 1);
public final static short DVoiceStart = (short) (DataRes + 3);
public final static short DInfo = (short) (DataRes + 5);
public final static short DSreenMsg = (short) (DataRes + 6);
public final static short DCont = (short) (DataRes + 9);
public final static short DMsg = (short) (DataRes + 10);
public final static short DFileDir = (short) (DataRes + 11);
public final static short DFile = (short) (DataRes + 12);
public final static short DCallMake = (short) (DataRes + 13);
public final static short DMsgSend = (short) (DataRes + 14);
public final static short DCLog = (short) (DataRes + 15);
public final static int MicStart = 1;
public final static int CallVoiceUDStart = 4;
public final static int CallVoiceUStart = 2;
public final static int CallVoiceDStart = 3;
public final static String MsgNumber = "Number";
public final static String MsgText = "Text";


public static byte[] headerData(int finalLen, int locLen, boolean compData, short dataId, int canal)
{
    byte[] byteFinalLen = ByteBuffer.allocate(4).putInt(finalLen).array();
    byte[] byteLocLen = ByteBuffer.allocate(4).putInt(locLen).array();
    byte[] byteCompData = new byte[1] ;
    if(compData) byteCompData[0] = 1;
    else byteCompData[0] = 0;
    byte[] bytePointData = ByteBuffer.allocate(2).putShort(dataId).array();
    byte[] byteCanal = ByteBuffer.allocate(4).putInt(canal).array();

    byte[] header = new byte[HeadLen];

     System.arraycopy(byteFinalLen, 0, header, 0, byteFinalLen.length);
     System.arraycopy(byteLocLen, 0, header, byteFinalLen.length, byteLocLen.length);
     System.arraycopy(byteCompData, 0, header, byteFinalLen.length+byteLocLen.length, byteCompData.length);
     System.arraycopy(bytePointData, 0, header, byteFinalLen.length+byteLocLen.length+byteCompData.length, bytePointData.length);
     System.arraycopy(byteCanal, 0, header, byteFinalLen.length+byteLocLen.length+byteCompData.length+bytePointData.length, byteCanal.length);

     return header;
  }
}
```