**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**

***IJCSMC, Vol. 3, Issue. 4, April 2014, pg.1204 – 1208***

**RESEARCH ARTICLE**

# Transfiguring of an Android App Using Reverse Engineering

**Taranjeet Kaur Chawla, Aditi Kajala**

M.Tech(CSE), Mody University of Science and Technology, Lakshmangarh, India

taran.chawla17@gmail.com, aditikajala.fet@modyuniversity.ac.in

*Abstract*— ***Today in this webbed world, smartphones have conquered a major part in human's life. An android application is an assemblage of Java which is edited from the source code, to .class file, further to .dex files to execute on the Dalvik virtual machine. Here, in this paper we have implemented the reverse engineering principle on an android app which is responsible for leaking the personal information from your smartphone such as IMEI, IMSI etc. Our research includes and emphasizes the misuse of personal/phone identifiers as well as the physical location. This paper seeks better to analyse over more than 1000 android applications which are free for the user. We have carried out the analysis i.e. static as well as dynamic.***

***Keywords— Android; IMEI; IMSI; Reverse engineering, Dalvik***

## I. INTRODUCTION

Android has refashioned the mobile world. It is a comprehensive as well as an open source platform. Along with the android becoming popular software, there also come a number of loopholes. The day-by-day increasing numbers of free applications, which are provided by the markets, such as Apple's App Store and Google Android Market, with just one click have led to a hindrance in security. The security model of android is unique. The open nature of the platform always welcomes the changes and countermeasures. These changes can even help or violate with the security, therefore, it's more necessary to focus on protecting the device from hackers. The loopholes and the bugs of the current application act as a favourable condition for the hackers to exploit. Phishing, usage of local data, unsafe http connections etc. are the most common today. [1]

It has been surveyed that around 50 applications in Google's Play Market leaked sensitive information as it migrated from handset to handset specifically on the Ice Cream Sandwich version of android and also on the other web services. Attributes of successful online communities are recognized and help in attaining and designing a security platform for the android applications. A good number of exploits were found as the handsets were connected to a local network. A number of available android apps have a necessity to pass the messages over the web and therefore are considered for securing the sensitive data during the transition. Further it has been studied that these apps make use of SSL/TLS protocols. It has been estimated that 1,074 (8.0%) of the apps studied and analysed have the SSL/TLS code which is prone to MITM attacks.[2] But still, the problems were tackled and were able to win over the SSL layer and the transport layer which were implemented. Particularly, the specific applications were not discoverable but still they were downloaded for around 39.5 million to 185 million times.

 

Here we mainly focus on security as recently a severe fault has been identified which sends the personal data through an SMS or acknowledge the hacker to review a URL. So therefore, a developer must be cautious of the hackers and must enhance the security measures.

*A. Exploitation of Phone Identifiers*

The remote networks servers receive the information related to the Phone Identifiers such as IMEI, IMSI, and ICCID. Here the detailed analysis is conducted to find their existence as well as the reason for it. [3]

*B. Exposure of Physical Location*

Hackers get the physical location of the users through a SMS. Sometimes the user desires the physical location of itself which is advantageous. So, therefore code held responsible for it is found and gone through. [4]

## II. PREVIOUS WORK

Various analysis and research has been carried out to enforce the security measures in android platform. Jess [5] explained that Android being a Linux platform has itself its own secure measures specifically for the mobile world, along with the programming of Java. Sascha [2] explains unnecessary use of SSL/TLS which can be a reason for MITM attacks. 13,500 apps are analysed and are presented. Online research is carried out to evaluate user's approach of certificate warnings, resulting in that about 754 users could not examine whether their surfing session was secured by SSL/TLS. William [4] quoted that in future mobile devices will be the operating systems in use. Due to the open nature of the android, there also comes a risk to its security which is to be kept in mind. Adrienne [6] informed the users about the risk of installing applications. A survey was conducted to find out whether or not Android permission system is capable of alerting the users. Two studies were also compared i.e. online research of 308 Android users and a laboratory study of 25 Android users.

## III. REVERSE ENGINEERING

Retrieving the technological details of a program or device, to assure the working of some operations, to remove an error or to enhance the efficiency of the software is referred to as reverse engineering. Twisting the applications and reforming the app into a malware can also be considered a part of reverse engineering. Here, the operation of the malware is understood, its variables and code structure are examined which has been modified by the malicious software. The tools used for reverse engineering are Dex2jar, Apktool, JD-GUI, 7 zip along with Eclipse.

*A. Apktool*

It is a 3rd party tool which implements and dissimulates the apps to the previous form and modifies them after applying certain changes. It enables the user to figure out the malware in the application. It can also be employed for addition of reinforcement for customers and localizing.

*B. 7-Zip*

Helps in archiving the files easily i.e. the apk file of the application. It is considered among the best one with the advantage of strong encryption and decryption, high as well as more compression as compared to other zippers.

*C. Dex2jar*

Every Android application runs with an instance of Dalvik Virtual Machine. Now, the DVM runs the files in .dex format which is acknowledged as a very efficient binary format of machine rules. The main app logic is present in the classes.dex file which is impossible for the user to understand. Therefore, this tool has been developed to convert the dex code to *.jar Java file [1].

*D. JD-GUI*

It is a Java decompiler to look through the *.jar files. It acts as a user interface which uploads all the entities encapsulated in the jar file and enlists the *.java code. It also displays the log files and allows the user to decode file in the Java Stack traces.

## IV. METHODOLOGY

The proposed system has the following phases:

*Phase 1:* With the a help of a large number of developers and their intelligence, a number of new applications are being launched and uploaded to the Google Play Store. The android's developers aimlessly prolong web services to smartphones, which therefore becomes one of the Android's apps major selling features. Surprisingly, the unbound integration of the Gmail, Contacts and Calendar with the web services is the most apparent point available. When the applications are downloaded from Google Play Store and installed on the smartphones, code is in such a way that automatically hackers get information through the content provider which includes the

 

messages and phone identifiers. Some other information is also mailed to the hacker on their mail address which includes:

- IP address of the device.
- Mobile network service and the code.
- Unique id which includes IMEI, IMSI or ICCID
- Version of the device
- Physical Location of the user.

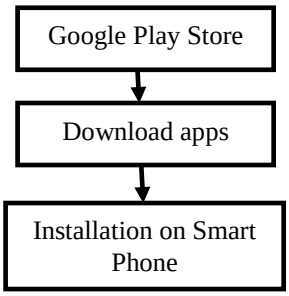


Fig1.Installation of .apk file

*Phase 2:* Decompilation of apk file is done by the reverse engineering. Android Manifest.XML contains the user permissions and allows the app to perform certain functions. This important file is extracted using the tools such as Apktool, JD-GUI, dex2jar etc.

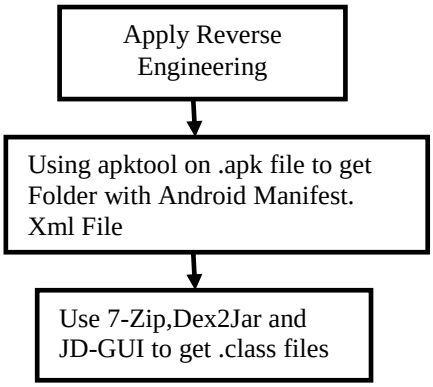


Fig 2. Reverse Engineering (Decompilation of .apk file)

*Phase 3:* After the review of the java code extracted using the Reverse Engineering, analysis is carried out. Analysis is carried out at two levels i.e. Static as well as Dynamic Analysis. Static Analysis refers to learning the program code at grammatical or lexical level. Dynamic Analysis refers to a number of methods that run an application in a designated ambience and supervise its behaviour. In dynamic analysis security is applied and application with malicious code is recognized.

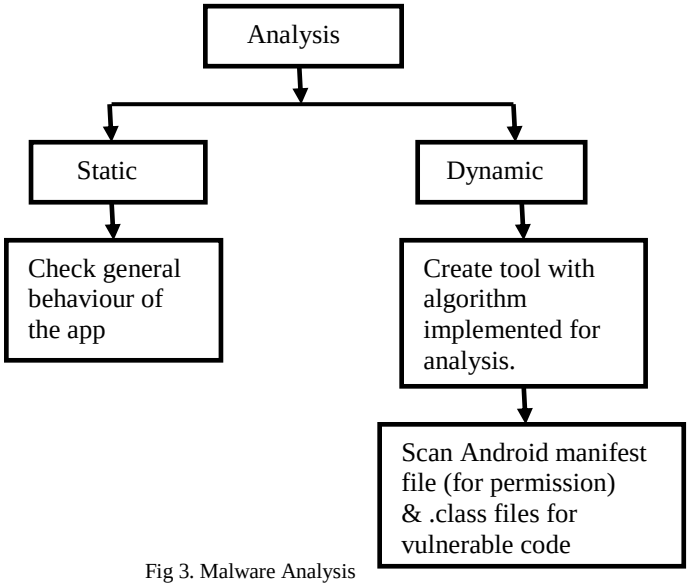


Fig 3. Malware Analysis

 

## V. RESULTS

In android to run an application, an Android Virtual Device (AVD) is created. So therefore, an application which stores the database of employees is created and is run on AVD with Configuration Android 4.1-API Level 16.

In Emulator Database Demo is launched. In this Screen Add Employee Tab is to add Employees Details and Employees tab is to display employee's details.

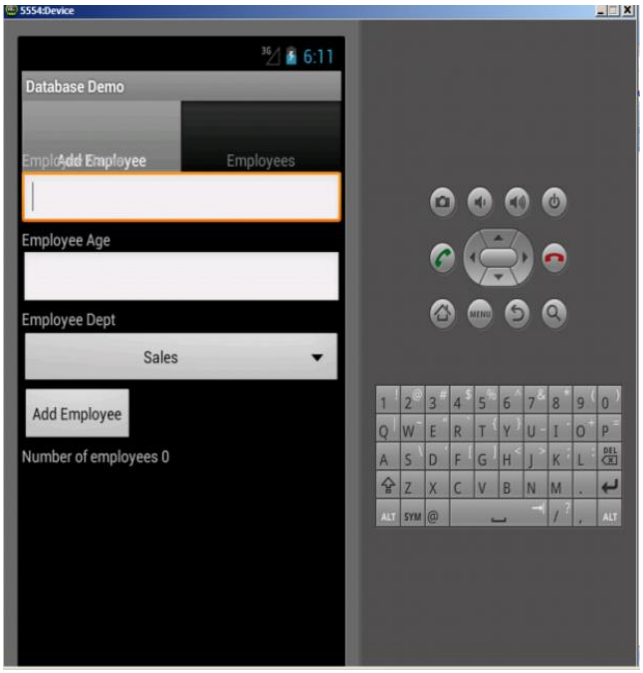


Fig 4: Database Main Screen

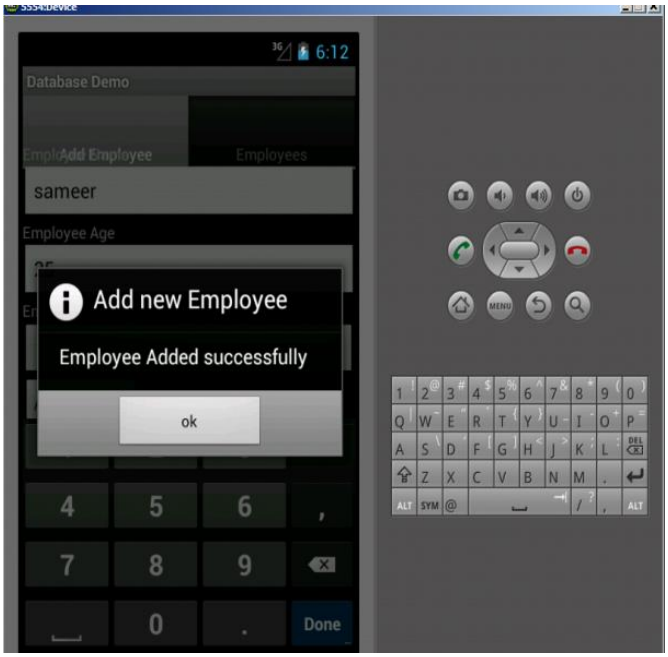


Fig 5.Employee Addition

 

When we run this application .apk file is created in the bin folder. This file can be copied in the real android device to run this application. But whenever we run this application on real device, it will send the mobile identifiers and location information of the device through mail .This is a malware or Vulnerable Android Application. It is somewhere violating the Android Securities.

*Real-time Device*


Fig 6. SMS Received

## VI. CONCLUSIONS

Mobile application security today has been a major issue which cannot be ignored. The day –by-day increasing number of users and their personal information and transactions on the smartphones has forced the developers to give a second thought to authenticate their information. We have broadly characterized the security of applications in the Android Market using the concept of Reverse Engineering. This project seeks to better understand smartphone application security by studying many popular free Android applications. One important point which should never be ignored that an important source of malware is through the web. Using the specialized apps, the whole security system can be improvised. Further, our future work will consider that all the private and confidential data will be authenticated by encryption and decryption and widen our approach to some other security vulnerabilities such as phishing, application installing etc.

## REFERENCES

[1]. Vibha Manjunath, "Reverse Engineering of Malware on Android", SANS Institute InfoSec Reading Room, August 31st, 2011
[2]. Fahl, Sascha, et al. "Why Eve and Mallory love Android: An analysis of Android SSL (in) security." Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012.
[3]. Enck, William, et al. "A Study of Android Application Security." USENIX security symposium. 2011.
[4].Enck, William, Machigar Ongtang, and Patrick McDaniel. "Understanding android security." Security & Privacy, IEEE 7.1 (2009): 50-57.
[5]. Jess Burns, "Mobile Application security on Android", ISEC Partners, Black Hat USA, July 30th,2009
[6]. Felt, Adrienne Porter, et al. "Android permissions: User attention, comprehension, and behavior." Proceedings of the Eighth Symposium on Usable Privacy and Security. ACM, 2012.
[7]. Pinola, M. 2011, 'Android data vulnerability: How to protect yourself', LifeHacker
[8].Desnos, Anthony, and Geoffroy Gueguen. "Android: From reversing to decompilation." Proc. of Black Hat Abu Dhabi (2011).
[9].Pacatilu, Paul. "Android Applications Security." Informatica Economica 15.3 (2011).

 