ANALYSIS OF IN-APPLICATION ADVERTISEMENTS FOR SAFEGUARDING ANDROID USERS FROM EMBEDDED ADWARE

By

Anum Javaid

A thesis submitted to the faculty of Information Security Department Military College of Signals, National University of Sciences and Technology, Pakistan in partial fulfillment of the requirements for the degree of MS in Information Security

AUGUST 2016

ABSTRACT

With hundreds of millions of Android phone users in around 190 countries, the global smartphone market has witnessed extraordinary and explosive growth of Android phone sales in recent years, which is escorted with the huge number of its applications. It has an open source code platform that encourages app developers to develop android applications and introduce them free of cost in the market. Applications are considered as the heart of Android phone that drive innovation, leisure, ease of availability and compatibility with the mobile devices. Embedded adware or mobile advertising is rapidly finding its ways into android applications in the form of banner ads, rich media or interstitial ads. The fluidity of application markets and the embedded adware thwart Android phone security. Availability of applications (apps) comes in two versions i.e. free or paid. Mostly in free version apps, developers embed advertisement libraries in his application code to generate the revenue. This practice may cause in-application advertisement attacks to steal unauthorized data of the user. One of the major reasons is the correlation model or permission-based model running between advertisements and the mobile applications, where permissions of app and ad-libraries are not separated. So eventually they can misuse each other's permissions and become the privacy compromise source for the Android users. Using the quantitative research technique in the current study, the research has found that there is a significant impact of embedded adware on Android user's security by causing advertisement attacks. However, certain limitations of the existing solutions can be highlighted by exploring the permissions of ad-blockers and the techniques used for the separation of ad-libraries and app permissions to reduce the permission bloat. To fulfill the requirements to protect Android users from in-application advertisement attacks, a recommendation model is provided for Security professionals and general public. Meanwhile, areas for future research have also been provided in this research thesis.

SUPERVISOR CERTIFICATE

It is to certify that final copy of thesis has been evaluated by me, found as per required format and error free.

Dated: _____ _____

Dr. Imran Rashid

DECLARATION

I hereby declare that no portion of work presented in this thesis has been submitted in support of another award or qualification either at this institution or elsewhere.

_____

Anum Javaid

DEDICATION

“In the name of Allah, the most Beneficent, the most Merciful"

I dedicate this thesis to all my teachers and family, for their never ending love, faith and motivation

ACKNOWLEDGEMENTS

Foremost, I am highly grateful to Allah for His blessings that continue to flow into my life, and because of Him, I made this through against all odds.

I would first like to express my deepest sense of Gratitude, immense respect and thankfulness to one of the most prestigious teachers I ever had**, Dr. Imran Rashid**, my thesis supervisor, who is always been a source of inspiration and motivation for me. I have been amazingly fortunate to have an advisor who gave me the freedom to explore on my own, and at the same time the guidance to recover when my steps faltered. Without his continuous support I would never been able to complete my thesis on time.

I would also like to thank my co-supervisor and committee members for their humble guidance, encouragement and support.

Finally, I must express my very profound gratitude to my parents, siblings and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

1. INTRODUCTION

This introductory chapter will help in giving a brief introduction of this research thesis. It begins with the initiation of advancements in Android based smartphones. It will put some light on embedded adware in Android applications and introduce the advertisement attacks caused by them. The last section of the chapter includes the significance of the study to the industry and academia.

1.1 Research Overview

The trend of smartphones with Android OS (Operating System) is increasing rapidly since its emergence in 2005. It has gained popularity due to its advanced computing capabilities and other useful features [1]. Android is developed by Google and it is considered as world's most dominant operating system [2]. According to the latest report it has been found that Android has 82.8 percent market share, compared to iOS 13.9 percent [3]. Android is a Linux based and open sourced mobile platform. The Android OS presents never before seen flexibility and support for third party applications (mostly referred as "apps"). Presence of Apps distinguish an ordinary phone and smartphone [4-5]. Applications make use of advanced hardware, software, local and served data exposed through the platform to bring modernizations and value to the customers [6]. Applications are offered to the potential customers in two versions i.e. free or paid. A study carried out over last year found that overtime the trend of using free apps is more increasing as compared to the paid apps [7]. Free applications are available in the market but most of the time developers embed advertisements in their apps to increase the revenue and to get paid for their work. The method of increasing revenue and still introducing the application to the user free of cost is done by incorporating the advertisement library in the application code. It is an alarming situation because it can lead to certain types of attacks known as "In-advertisement application attacks".

Applications are available in the market on certain app stores such as Google Playstore, Apple App store, Amazon App store, Windows Phone Store and Blackberry World. Android phone

users download the apps from Google Playstore or other third party app stores. Amazon AppStore for Android, Getjar, Slide me, Appszoom, CNET, APPS, Yandex, Get5, AppsLib, Mobango, 1MobileMarket, AppBrain, OperaMobile Store, LG Smartworld, Samsung Galaxy Apps and F-Droid are some existing third party app stores available for Android users [8]. Google Playstore is more secure, reliable and popular than other existing app stores. It contain more than 1.6 million apps either with free or paid version [9].

Mobile advertisements are considered as the primary business model to provide Android Application developers financial incentive to introduce free apps in the market [10]. There are different types of mobile advertisements such as banner ads, rich media and interstitial ads, click to call and SMS/Text ads. With over 38% of the Android application developers prefer to embed ad-libraries in their app code to generate maximum revenue [11].

Advertisements in applications need to be understand thoroughly in order to cognize its behavior. Example of In-application advertisements is shown in the Figure.1 [12].

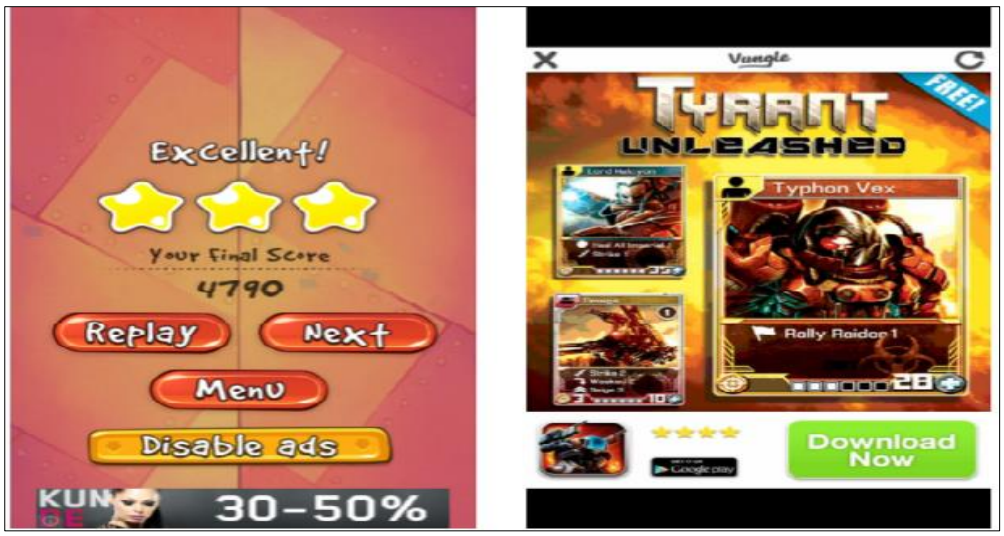


Figure 1. Screenshots of two examples of In-application Advertisements Showing Banner and Full Screen Ads [12]

The left side of the figure shows the example of banner ads that appear on the top or bottom of the screen at run time. Whereas right side of the figure shows the full screen advertisements who can stay at the screen for some seconds.

Applications of Android are mainly written in Java and at the time of its development the permissions of that particular application are declared in the (AndroidManifest.xml) file. This is simply the case of an ordinary application. When a developer incorporates the advertisement library in its application code, the permissions of advertisements are also declared in the same manifest file. This is particularly the description of Android permission mechanism.

Mobile based advertisement attacks are less studied as compared to web-based advertisement attacks. Open nature of Android OS makes it more vulnerable to attacks. It is sound attractive for the developers to develop the application of their own choice and introduce them to the Android users through different market places such as 'Google Play store' or 'Amazon Play store'. Attackers find different ways to exploit the vulnerabilities of the system to get user's personal data and to use it for their own purposes. "In-application advertisements attacks" are increasing promptly because attackers take advantage of vulnerabilities in the system rather than the security knowledge of the user.

Although the manufactures and developers put their efforts to secure the Android architecture from hardware to software flank, still there are lots of areas that need to be improved.

Till now, Android based OS are most vulnerable for smartphone attacks because the source code of Android is open. A strong security architecture and arduous security programs are required to justify the open source code of Android.

We will refer the (AndroidManifest.xml) file as 'Manifest', advertisement as 'ad' and we will call Android applications as "apps" for the sake of simplicity in the remainder chapters.

1.2 Motivation and Problem Statements

- Google play store for Android has more than 1,600,000 applications and around 1,000,000 are still "questionable" according to the security point of view [10]. Some of the applications include "advertisements" in their applications to increase the developers' revenue. There is a point arrives when users' privacy is compromised due to the presence of malicious advertisement libraries or malicious ads.
- Google has its own ways to find the system vulnerabilities but still there are some questions that need to be resolved. Problem arises from the basic point i.e. advertisement library is embedded into the host application so these require additional permissions.

- Android permission model has some design problems such as permissions of Android application and host application are not separated, when users accept the permissions they cannot differentiate between the permissions demanded from app or ad-libraries. Unnecessary permissions are requested in the form of contact list, SMS or GPS location information required by some ad-libraries (to target the user) while they actually don't need it for functioning. This increases the level of threat. So application and ad-libraries share permissions of each other because both run within same environment, and they have same UID (User Identifier). Eventually they both can misuse the permissions of each other.
- There is no method for granting permissions to some and denying others. Some of the solutions are available but they have certain limitations. Trusting that the application would not misuse the user's resources is not the solution to this problem. Letting an unknown to sneak through your data is a dangerous thing, so the permissions need to be accepted wisely.

1.3 Aims & Objectives

This research will be focused on review of advertisements in the Android applications. The objectives of the thesis are as under:

- Study of the process of embedding advertisements in an Android application
- Study of the work already done in this domain and identifying the problem
- Study and analysis of existing techniques which are useful in identifying some of the peculiarities of embedded advertisements and also see their limitations for further improvements
- Presenting a problem solution caused by In-application advertisements based on the findings of the research
- Propose a framework for efficient analysis of applications with embedded advertisements

1.4 Research Questions

This research will show that

RQ1. How can In-advertisement applications or embedded adware impact Android user's privacy?

RQ2. What is the overall scenario of user's privacy protection at developer, users and Android platform end?

RQ3. What are the existing solutions in solving the problem statements and finding their limitations?

RQ4. What steps or Preventive measures should be taken to avoid advertisement attacks caused by embedded adware?

1.5 Research Methodology

At first literature review will be conducted to find out the behavior of ad-libraries with their corresponding applications. The literature will also help in highlighting the existing mechanisms and techniques used to solve a problem. Survey will be conducted to gather the information for quantitative analysis. Based on findings of research a proposed framework will be presented.

1.6 Data Collection Methods

This study is based on two approaches i.e. Primary and Secondary data collection.

Secondary data collection comprised of research papers and articles from well-known journals and conferences. There were more than 100 papers examined for successful completion of this thesis. Primary data collection has been conducted by designing a questionnaire. Survey is conducted from 200 respondents belonging to different field or domains in order to analyze the results in an appropriate manner.

1.7 Relevance to the national needs

Android applications are used by every kind of mobile users. They allow the application permissions to access various resources of the mobile phone whereas they do not know that the embedded advertisements will also get the access to the mobile resources. These embedded adware can cause harm to the user and can compromise their privacy. This research work will

enable users to logically analyze various android applications for embedded adware. It will be a useful research based on a practical problem which a common user faces today.

1.8 Advantages

- It will provide extensive information on current problem caused by advertisements in Android applications
- It will help in analyzing the mobile application in different perspective
- It will give awareness to the users to understand how their personal data can be exploited by such applications having extra permissions than they actually require
- It will help in improving user experience

1.9 Area of Application

It will be helpful for the organizations that wish to build a secure mobile product and the Android users who wish to secure themselves from malicious activities on their smartphone.

1.10 Thesis Progression

In this dissertation there are 10 chapters in total. Chapter 1 of the thesis is "Introduction" that covers the background of the topic. In addition to this, first chapter is about problem statements, justification of the study, research aims & objectives, advantages and areas of application.

Chapter 2 of the thesis is "Literature Review". In this chapter, previous studies related to research topic are studied in order to find research gap. It also include History and importance of Android and its architecture, applications and their market distribution, advertisements, ad-libraries role and existing attack factors.

Chapter 3 is "In-advertisement applications & Attack Model". It highlight the attacks caused by the presence of embedded adware in an Android application and their impact on Android users' privacy. It also include the deep analysis of advertisements in an application and attacks caused by them by showing attack model and click fraud attempt on a practical ground.

Chapter 4 covers the existing solutions to the problem and their limitations. Existing techniques to solve the permissions' over privileged problem, reduce the permission bloat and stop displaying ads on the users' screen will also discussed and analyzed.

Application anatomy and analysis methodology with five different ways will be introduced in Chapter 5. It also includes the results and analysis of embedded ads' permissions and ad-blocker's permissions to explore the problem statements on a practical ground. Chapter 6 contains the theoretical framework, hypothesis formulation, data collection techniques, questionnaire design procedure and the analysis results to prove the authenticity of hypothesis. Chapter 7 includes importance of privacy and its policies. Methods of protecting location information by regulatory strategies, privacy policy, anonymity and obfuscation will also be discussed in this chapter. Chapter 8 includes the proposed model along with its significance in two different phases I & II by considering the ideal case and practical world separately. Conclusion, research limitations & future directions will be provided in chapter 9 & 10 respectively.

2 LITERATURE REVIEW

2.1 Introduction

Analyzing the behavior of in-advertisement applications has been a topic of interest in the domain of Android Application and user's security. Android organizes its application in the market place with a specified system's structure and architecture. A lot of research has been carried out to design and analyze the behavior or impact of the attacks caused by the advertisement services on the Android platform. Different techniques are presented by different researches to control the attacks caused by embedded adware. Some of them are designed to control the excessive use of the permissions where as some of them are designed to block those advertisements. Multiple approaches are introduced to cope up with the problem.

If a weather app demands for accessing your photos or contact list which it actually does not need for functioning, then users have no other option to deny such permissions in order to install that particular app. Advertisements may cause damage to the system because they demand extra permissions from users. Permissions of advertisements and applications are not separated so they can get access to each other's requested permissions and eventually it becomes the threat for the user privacy.

On the basis of boundaries defined in the previous chapter, this chapter will highlight the basic concept of Android user security and will put some light on various schools of thought. It will also cover the Android history &Architecture, Ad-Libraries and their role in protecting user's privacy.

2.2 History & Importance of Android

Android was emerged in 2005 and is available in different versions. Existing versions are CupCake 1.5, Donut 1.6, Éclair (2.0 – 2.1), Froyo 2.2, GingerBread (2.3.3 - 2.3.7), HoneyComb (3.0 – 3.2.6), Icrecream Sandwich (4.0.3 - 4.0.4), JellyBean (4.1.x - 4.3.x), KitKat 4.4, Lollipop 5.0 & 5.1 & Marshmallow 6.0. The first version of Android was alpha that was released in 2007 and the most recent update is Android 6.0 "Marshmallow" which was released in October, 2015.

According to the latest data collection report till Feb-2016, versions older than 2.2 are accounted for about 1% of Android devices. Figure.2 shows the market share of the versions 2.2 & above [13]. This data is gathered from Google Play app store. It also shows that KitKat has the maximum market share of 35.5% whereas Lollipop covers 34.1% of the Android phone selling market. All the above mentioned versions supports both smartphones and tablets except HoneyComb (3.0 – 3.2.6) which only supports tablets. As it doesn't support Android phone and its full source code is not released yet in the market so in our comparison graph in Figure.2 we excluded it.

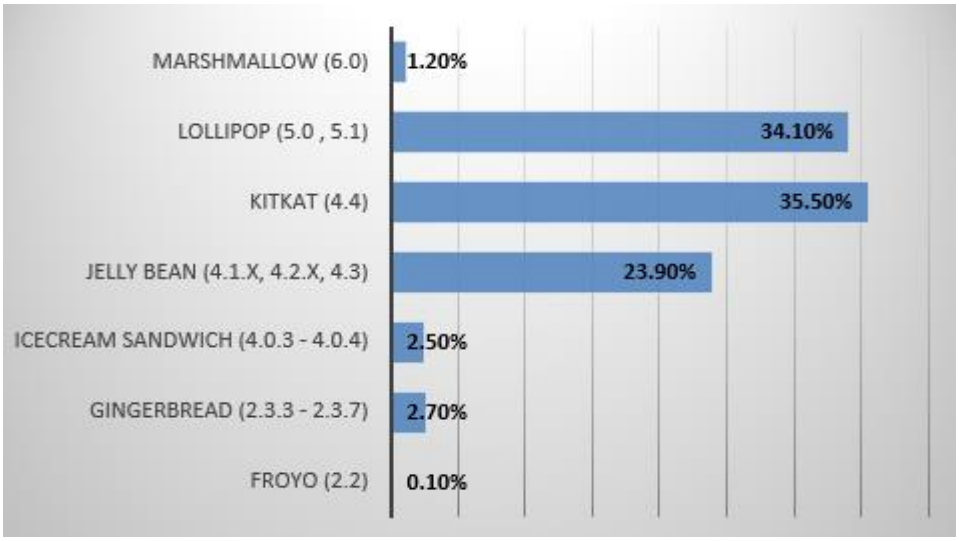


Figure 2 Android Platform Versions & Market Share [13]

Left Side of the figure shows the latest versions of Android. On the other hand, right side of the figure shows the market share of these versions with their uniqueness as explained above.

2.3 Android Architecture

Android has Linux based kernel. It has an open source code and its system architecture consists of four layers including**:**

- Kernel
- Libraries and Android runtime

- Application framework and
- Applications

Figure.3 below shows Android four layer architecture.

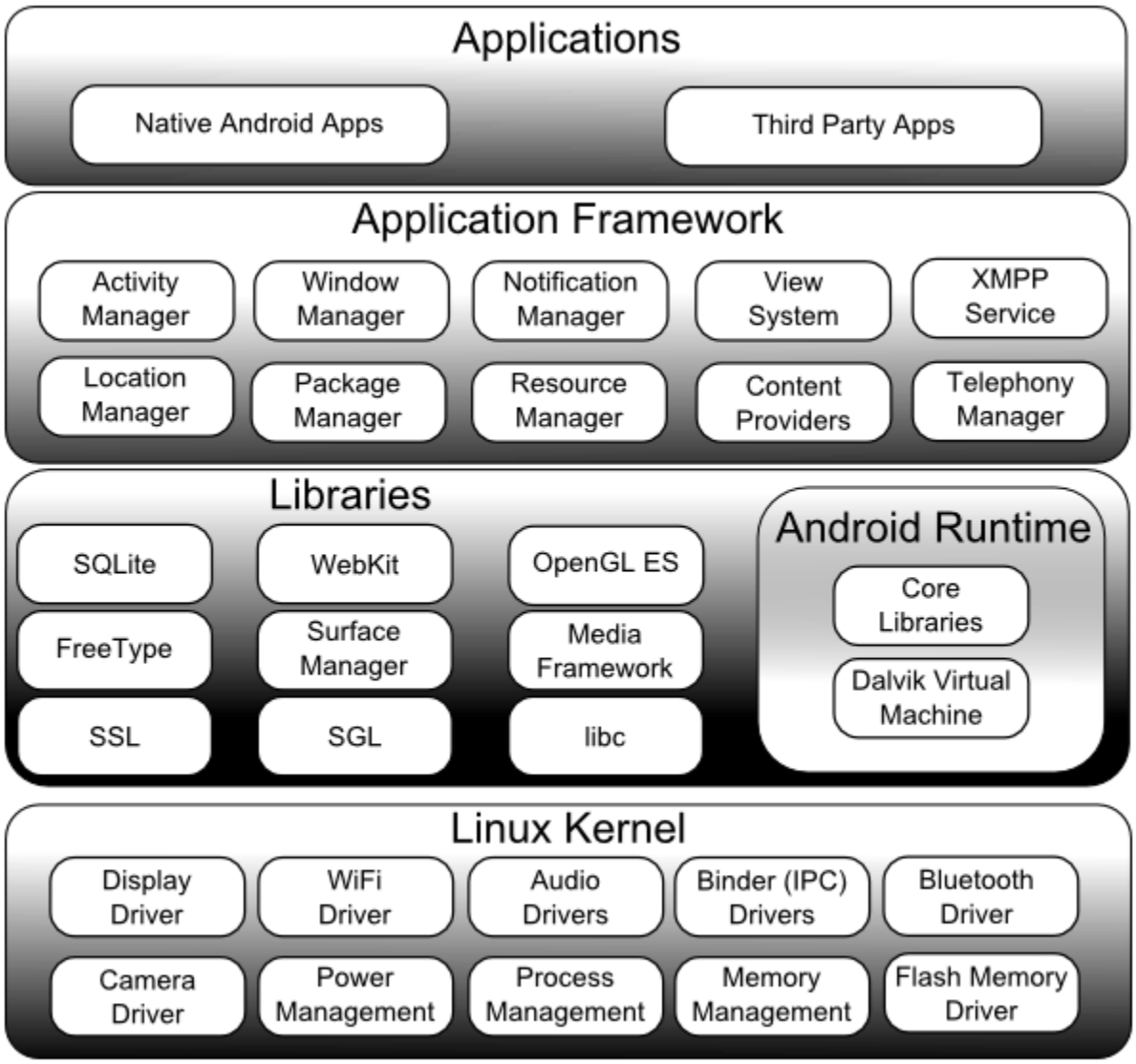


Figure 3 Android Four Layer System Architecture [14]

It is important to note that Android only uses Linux based kernel. It provides multitasking such as display, camera, keypad, audio, and memory, process and power management. On the other hand the second layer of Android (Libraries and Android runtime) contains libraries available for

applications. Android runtime consists of core libraries and the Dalvik Virtual machine (VM). Basic Java functionalities are provided by Core libraries whereas for running the applications the Dalvik virtual machine is designed. Android applications are written in Java but not executed by Java Virtual Machine (JVM). Instead, they run on a custom virtual machine called Dalvik. Applications are converted into Dalvik byte code by SDK (Software Development Kit) [15]. Compared to the typical JVM, Dalvik combines the Java class files and converts them into Dex format files before application executes on the Android platform. The purpose of the conversion is to decrease the space occupation of .jar files by reuse of the information from multiple class files. Hence, Dalvik enables the light weighed operation system on Android phones.

Third layer of the Android is Application framework that contains which the developer may need to build an application. Applications can directly interact with the help of Application framework layer. Its main services include activity manager, content providers, resource manager, notification manager, location & telephony manager as mentioned in Figure.3. View system service is used to create application user interface [14].

Lastly the fourth layer i.e. Applications. It is on the top of Android software stack. It has its own significance according to the security point of view. Average user can interact most with this layer. It is divided into two main categories i.e. Native android apps & Third party apps. Native apps are pre-installed apps before user purchase the phone such as email, Google Playstore or Facebook etc. Third party apps are those apps which are installed by users themselves after purchasing the phone.

2.3.1 Building Blocks of Android Platform

There are three platform building blocks of Android [16]. It includes:

Device Hardware: Android runs on smartphones, tablets and set-top-boxes. It can run on various hardware configuration.

Android OS: As described above, Android has Linux based operating system. OS give access to camera, telephony & Bluetooth functions, network connections and GPS data.

Android Application Runtime: Applications run in Dalvik VM and are written in JAVA programming language. Both Dalvik and native apps run within same security environment. Apps' Private data, including databases and raw files are written in a dedicated part of filesystem.

2.4 Applications

Smartphone is also termed as miniature PC with its installed applications. Android apps are written in Java and C/C++ and compiled to (DEX) Dalvik Executable Bytecode format. In Dalvik VM interpreter instance, each app is executed. In Linux platform subsystem app isolation can takes place via UNIX user identifier to execute each instance [17]. Apps are easily available in the market and users can take multiple advantages from them. For instance, users browse the web and manage their schedule. Some apps are used for personal finance whereas some of them can help travelers to move from one place to another. Other types of apps including games and leisure pursuits, some are related to health, entertainment, education, comics, music, calendar, appointment organizers, cameras, maps, managing Wi-Fi connections, ringtones, wallpaper images and other important setup information. Built in apps are usually for making and receiving calls and text messages. For privacy and security settings of the smartphone, multiple apps are developed and introduced on the existing app stores. Revenue of the developers is being generated in both cases. In case of paid version, the entity who pay the developers is the app user. On the other hand developers now a days have become wiser, they introduce the free version apps to increase the rating of the app and make it top priority for the users. But in reality they may generate more revenue as compared to their paid version by incorporating ad-libraries' SDKs in their applications. In short, developers are paid by the advertisement networks by displaying ads in their application on run time.

2.4.1 Building Blocks of Android Applications

App components are considered as the main building blocks of an application. They are four in number [18]:

Activities: It represents a user interface with a single screen. It performs actions on the screen. User interaction to the smartphone screen can be handled by activities.

Services: Background processing such as long running operations are done with the help of services. For instance, playing music in the background while using a different app.

Broadcast Receivers: Communication between Android OS and Android app can takes place with the help of broadcast receivers. It helps to communicate the broadcast message from system or from other apps. For example when any data is downloaded to the device, it informs the other apps about its availability to use them also. Each message of the broadcast is implemented as an **Intent** object.

Content Providers: Data management issues and data is handled by content providers. It supplies data from one app to another on request.

2.4.2 Primary Sources of Android Applications

Core Android OS are extended by Android apps. There are two primary sources of Android apps i.e. Pre-installed & User-installed applications [16].

Pre-installed Applications: Pre-installed applications include phone, email, calendar, contacts or web browsers. These apps can be developed by OEM for a specific device or open source Android platform.

User-Installed Applications: 3rd party applications are developed and thousands of them are introduced in the Android market i.e. Playstore. Users have multiple options to download and install applications of their own choice on their phone.

2.5 Market Distribution of App

There are different app stores from where users can download and install the applications. As discussed in the previous chapter Google Playstore is the most popular Playstore. It is considered as the leading app store as shown in the below figure.

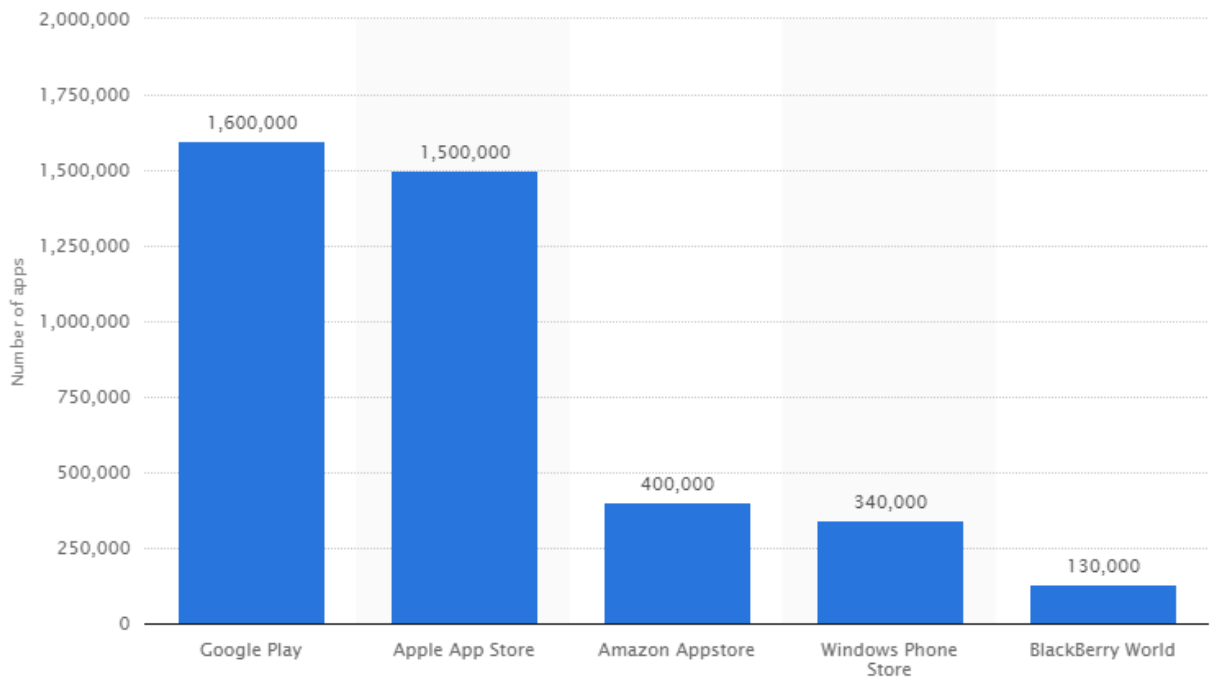


Figure 4 Leading App Stores [9]

This figure.4 shows that Google app store contains the maximum number of applications i.e. 1,600,000 as compared to other market places [9]. Most recently in 2015, the number exceeds from 1.8 million [19].

2.6 Permissions

Permissions are considered as the agreement between app developer and Android users. It is the mutual consent that causes user to accept it before downloading the app from Playstore and give access of the phone resources either to the app developer or to the ad-libraries. Permissions are necessary for the proper functioning of an application. Permissions can be in the form of internet permission, SMS/ call logs, phone contact information and in worst cases rooted phone to fully control the device. An example of demanding permissions at the time of installation is shown in Figure.5. Permissions are often termed as "declarations". It means users have no choice to uncheck some of them and permit others. Whereas users have to accept or decline all. A clear description of declarations are shown below.

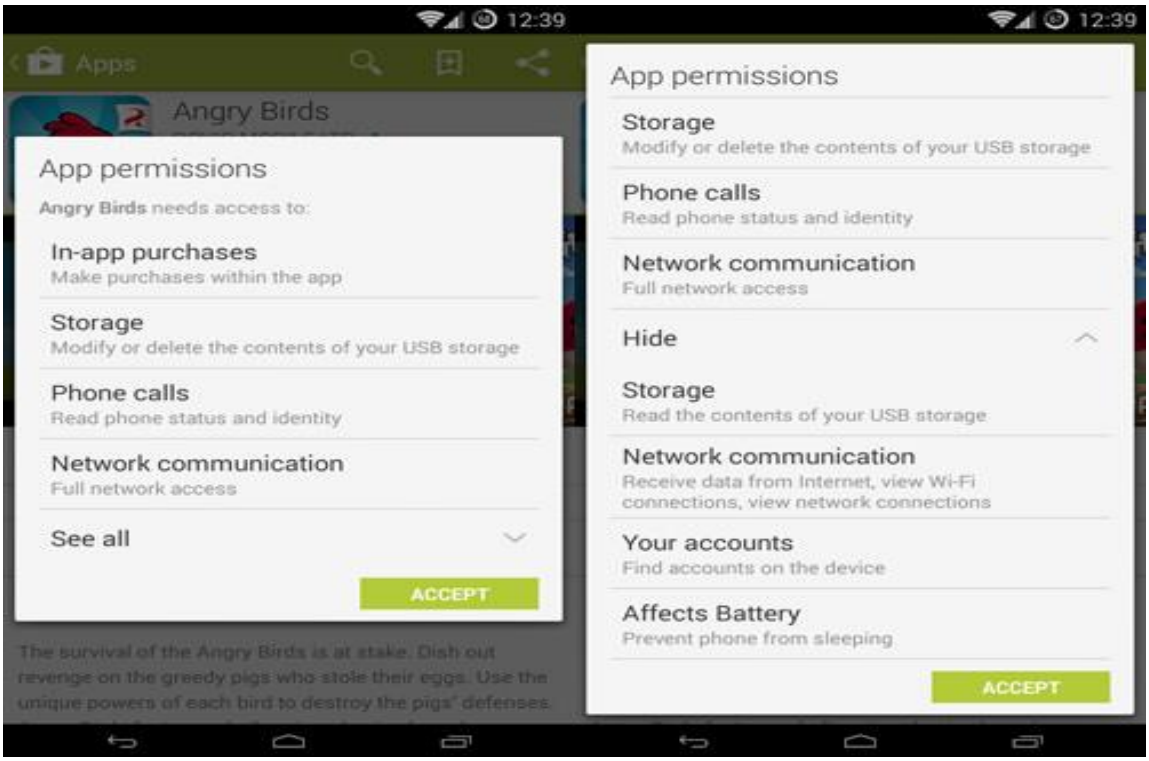


Figure 5 Permissions Required by an Application [4]

It is the screenshot captured at the time of installation of "Angry Bird" app on the Android device. It shows the best example of "declarations". All the app permissions are listed including In-app purchases, access to storage to modify or delete the contents, account information as well as network communication.

2.7 Android & Security

Till now, there are more than 67,550 unique registered developers on Android market [20].

While designing the Android architecture, two entities were taken into account i.e. Application developers & Users. Security protection of Android developer is as much important as the security of Android users. For this purpose, Android was designed to safeguard application developers who are less familiar with the security issues as well. Other entity who was kept in mind while designing the Android architecture was its users who frequently download and install the applications from the existing markets. Architecture is designed to reduce the probability of attacks on Android phones [16].

Distribution of the Applications can be done by Google through its particular play stores such as Google Play store or Amazon and other third party app stores. As far as security is concerned, when an application is uploaded on the Google play store there is a security check called "bouncer" that particularly checks the malicious activity in the given application. If it alarms for the malicious activity then the application is discarded but if it successfully passes the clearance then it becomes the part of Google play store [12].

Whenever an application has to use the system resources then a particular check based on the UID will be made to check the pre-defined permissions for the particular application in case of Android OS. This process helps to prevent defraud the network provider and implementation phase becomes easy. But the drawback for using this method gives rise to many possible ways of attack. Such as permissions shared by application and the particular advertisement library, so there are unnecessary user confidential data that might be shared with the network provider. Another threat caused by the advertisements is fetching and loading dynamic code. This comes in the category of threat for two reasons, first dynamic code cannot be analyzed and downloaded code can be changed at any time making it difficult to predict [21].

Till now, we have covered Android Applications contents, history of Android phone and Permissions declaration model to emphasize its importance and impact on user's privacy. In the next chapter, we will introduce in-advertisements applications or embedded adware, ad-libraries and their role and eventually the threat model. Process of embedding ads in the Android application will be shown with the help of practical example.

Chapter 3

3 IN-ADVERTISEMENT APPLICATIONS and ATTACK MODEL

3.1 Introduction & Importance of Advertisements in an Application

Emerging technology of Android based smartphones opened a new chapter in the domain of embedding advertisements in an Android Application. An Advertisement system is a combination of advertisers, publishing networks, advertisement libraries and application itself. It is a great source for the Android application developer for generating revenue. Advertising is one of the means to monetize (make money with) mobile applications. For embedding an advertisement in an application, publisher first creates its publishing account with the mobile advertisement networks. Types of advertisements are described in the previous chapter. Developers are paid based on Cost per Click (CPC), Cost per Mile (CPM), Cost per Action (CPA), downloads, installs and impressions generated via the application. Advertisements can be appeared on the top or bottom of the screen known as banner ads. They only take tiny space of the application. Full screen ads or interstitial ads also appear on the smartphone while using the application. Other types of ads are Rich Media, Click to Call and SMS/Text Ads [11].

Advertisers connect to different service providers for displaying their ads to the user. There are many advertisement networks such as AdMob, Millennial Media, AppLovin, AdFit, MdotM, LeadBolt, RevMob and Cauly Ads etc [10]. Later on these ad-providers are contacted by Publishers (App developers) who embed the ad-libraries of these ad-services to their application code. Finally, when the application is installed on the user's phone, ads are displayed to the users.

Android phone is not less than a portable computer and with its advancements people share many sort of information through different channels in different forms, so the security threat related to it is also increasing. Click fraud attempts have been extensively studied on web pages rather than the mobile platform [22, 23].

This research aims to provide extensive knowledge on the In-application advertisement attacks and its consequences on user's privacy.

As described in the previous chapter, a list of permissions is displayed to the users while installing an application from Playstore. Permissions are divided into two parts in which ad-libraries are embedded. One part of the permissions is demanded from the app developer for proper functioning of an application. On the other hand second part of the permission contains the permissions demanded from ad-libraries for displaying the advertisements. While installing an application, users are unable to determine whether those permissions are required by the application developer or those are demanded by the embedded ad libraries. Eventually users are bound to accept all of the listed permissions in order to install that application [24].

This chapter covers the detail overview of embedded ads in an Android application. It will put the light on significance and relationship between involved entities in the whole process. Threats and attacks caused by the advertisements in an application will be discussed here. Threat model will be presented to explore the damage or affect caused by embedded adware on Android users' privacy. Eventually the significance of the threat model in formulating the theoretical framework and proposed model will also be discussed.

3.2 Participants & Hierarchy of displaying ads

Advertisers contact to different service providers for the display of their advertisements.
Advertisers may be a common person or a company who want to advertise their product or services. The next phase begins with the involvement of publishing networks who receive ads from the advertisers and forward them to the ad-libraries. In the following section we will review the Android system Architecture to get the point of connectivity between ads and its required services.
Advertisement libraries are of two types i.e. mobile web & rich media library. Developers include the SDKs of ad-libraries in their application code. And finally those ads are displayed to the user with the help of the required permissions [12]. The overall process has been shown in Figure.6.

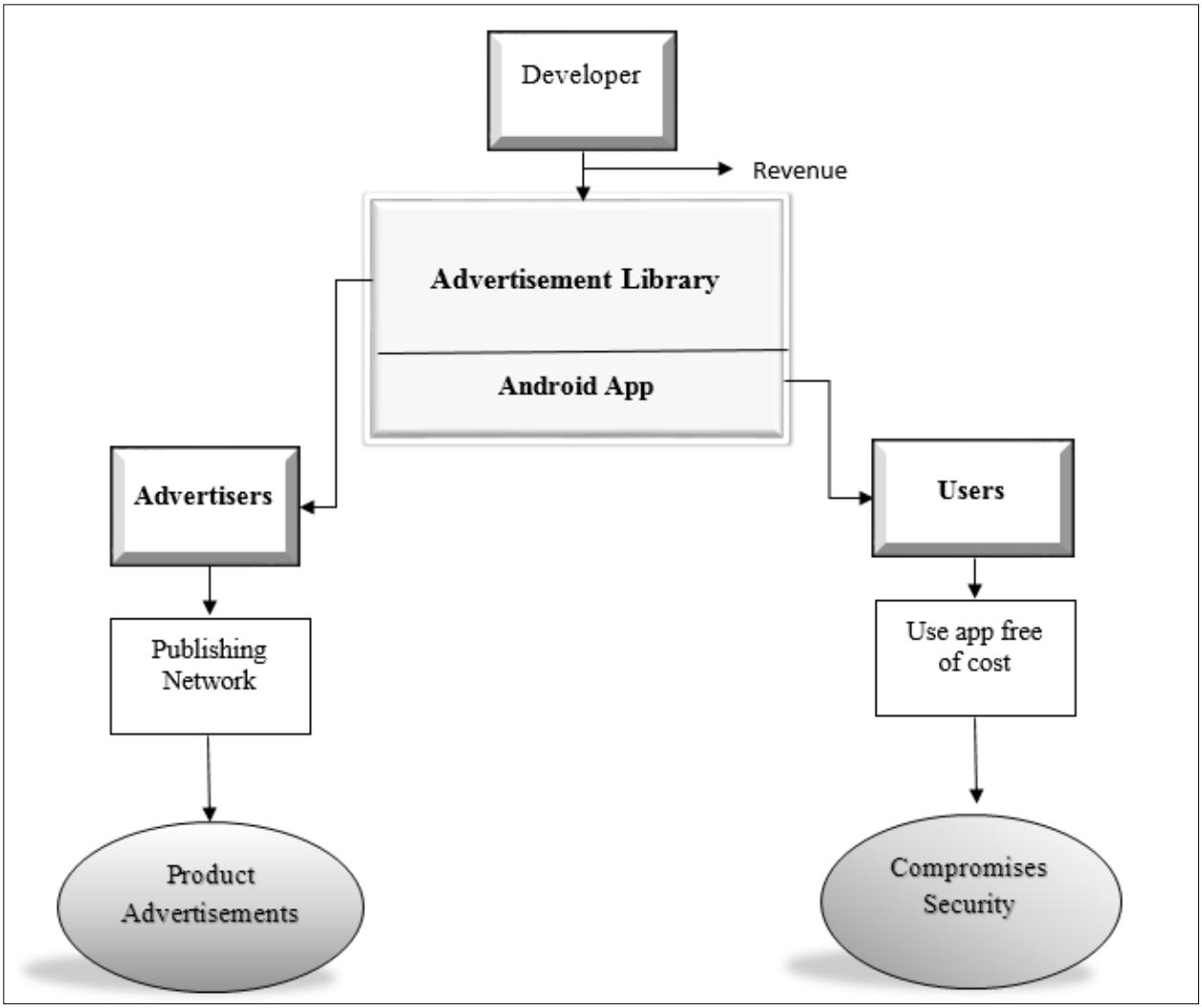


Figure 6. Relationship between Advertisers and Android Users

The above figure shows the relationship between the required entities involved in the mobile ad-display process. It shows that if an individual, company or any other marketing associate wants to advertise their products or services to the users, they need to contact the service providers i.e. publishing networks. Publishing networks introduce those ads to the app developer. The purpose of an app developer is to make money by incorporating ad-libraries. So the advertisement libraries are embedded in the developer's application code. The whole process revolves around the ease of use and money making process. It means developers introduce them in their app code to get paid for their work and ad-libraries wants to advertise their products or services. On the other hand users mostly prefer to install free apps while knowing they may probably face the advertisements at run time. Ads are displayed either in the form of interstitial or banner ads. In

this whole process the entity which suffers most is the android app user because those advertisements may lead them towards certain types of in-app advertisement attacks.

3.3 Ad-Libraries and their role

Android app developer tries to increase revenue for his free application. To accomplish this goal he contacts to different publishers to embed ads in his application. Ad-supported apps can be developed rapidly with the availability of ad-libraries [25]. Freely available apps on Google Playstore are more than 83% [26] and free app developers rely on advertisements for generating revenue [27].

Previous research shows that ad-libraries take extra advantage of the permissions demanded from their host application. There are dozens of ad-libraries with different market share [28]. For instance ad-mob library is widely used library now a days.

3.4 Connectivity between Advertisements and Android Applications

The process of embedding apps in an android application is necessarily to understand to reach the contact point between users and attackers. For this purpose we will use Admob library as an example and will focus on 'banner ads' for simplicity, as almost all the ad-libraries behave similarly in case of banner ads.

Android Studio is chosen as a tool for analysis purposes. To include ads in the services, developers first need to install play-services-ads through Google Repository in Android Studio.

Modification in the Manifest file is required in order to embed Advertisement library into Android Application. For this purpose required permissions of ads are declared in the same Manifest file. Most common and necessary permissions are "android.permission.INTERNET" to access the internet and "android.permission.ACCESS_NETWORK_STATE".

But somehow ad-libraries manage to get extra permissions while some of the permissions are hidden from the users when he install the application in his phone.

3.4.1 How ads are displayed

Advertisements are acquired from the advertisement server and mobile application displayed it at a fixed regular interval [29]. Figure.7 explains the hierarchy of displaying ads.

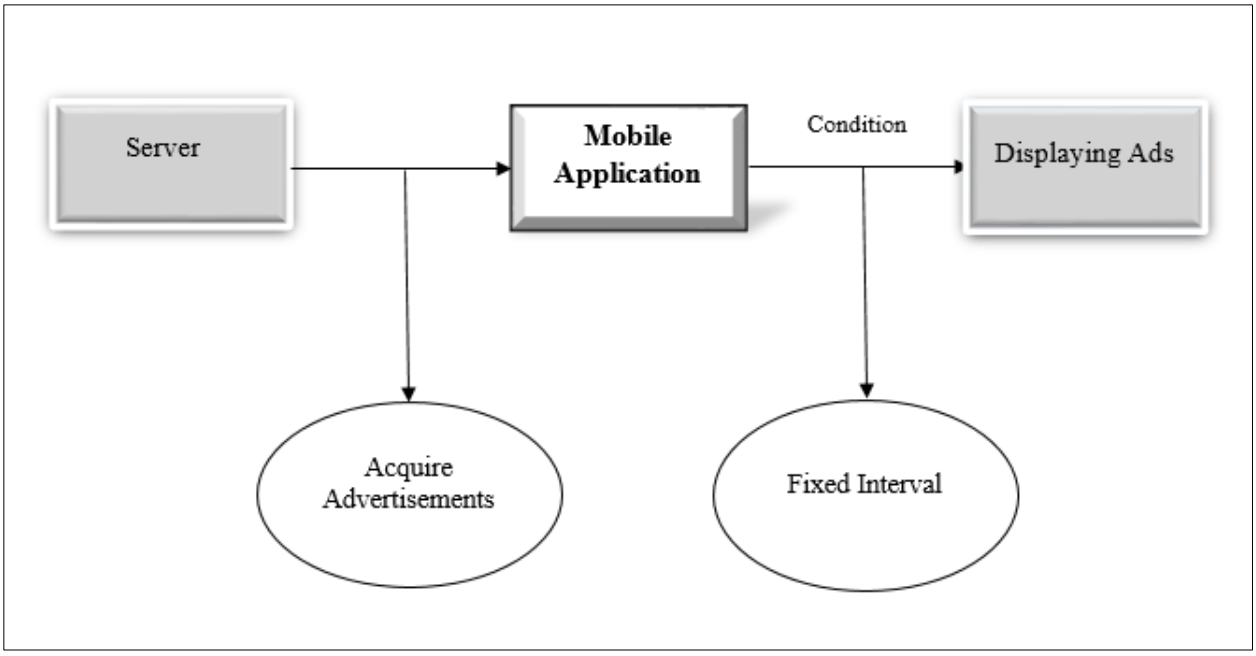


Figure 7. Ads Acquisition

Mobile application acquire advertisements from ad-providers or server indirectly. It means ads acquisition is possible with the involvement of developers. As described above, Mobile library SDK is embedded into the host application and when the required condition is fulfilled, ads are displayed at the regular interval on Android user's mobile screen.

3.5 Practical Example: Embedding AdMob Library

To understand the behavior of apps and their relationship with the advertisement networks, we developed a Hello World App and we named it as "In-Application Ads". We focused on banner ads and AdMob is chosen as a library. To embed the ads in our application, AdMob library is incorporated in our app code. After fulfilling the required conditions live ads started displaying on the bottom of the screen. Some of the screenshots of our developed application are shown in figure.8.

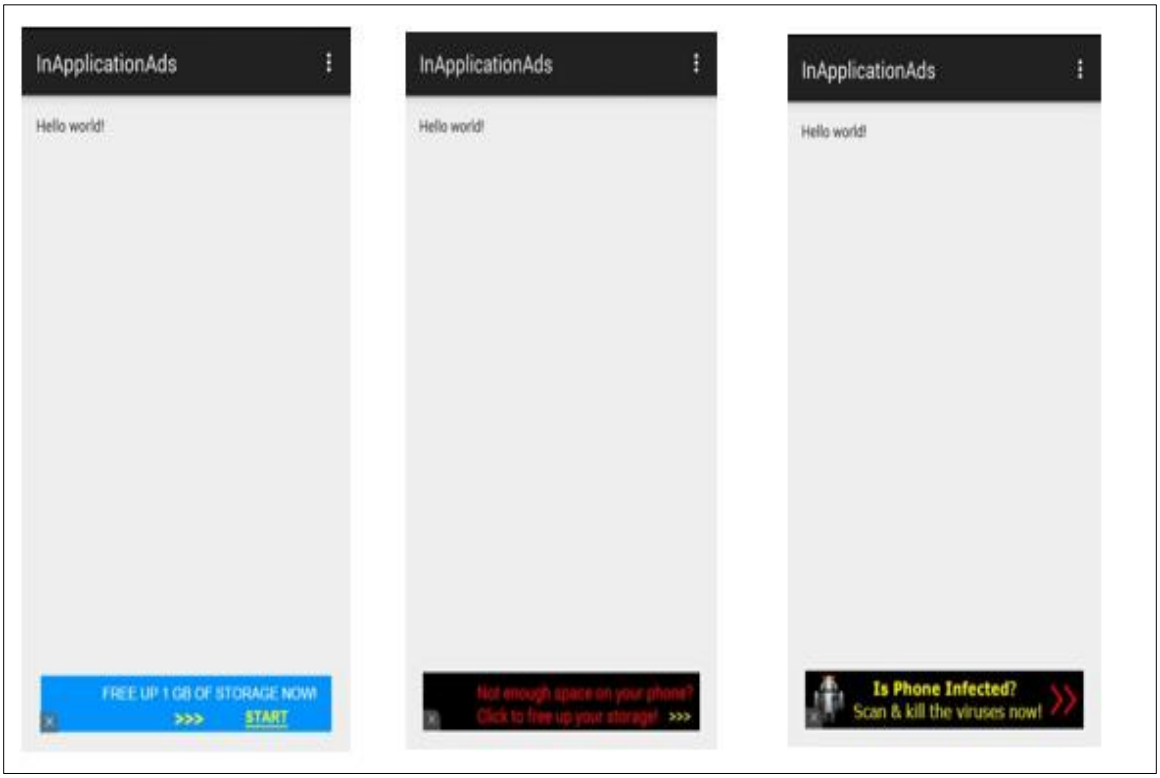


Figure 8. Displaying Banner Ads

Ads took the place at the bottom of the screen for some seconds. After a fixed or regular interval of time (in our case it is 30sec) ads started displaying on the screen.
Although test ads are used for experimental purposes, but in this scenario we highlighted the real case by incorporating and showing the real ads from AdMob library at runtime.

Next section shows the pattern and function of the application when users intentionally or unintentionally click on an advertisement.

3.6 In-Application Advertisement Attacks

Mobile based threats are divided into several categories including application based, web based, network based threats and physical threats. For the sake of simplicity, this section includes the application based attacks. Presence of ad-libraries SDKs in the developer's application code give rise to the generation of attacks.

Security related threats due to in-advertisement applications are divided in following categories:

- Attack caused by malicious advertisement libraries
- Attack caused by app developers

Advertisements in mobile applications can compromise user's privacy. Android's in-app billing service has gained rapid popularity [30] so as the incentive of attackers [31]. Web based or client server apps are different from mobile based applications. Mobile based attacks are more important because of their sensitive nature, they can access to SMS logs, Contact lists, Call history & location [32]. In our analysis we will focus on how user's privacy is affected due to presence of ads in mobile applications. This research will cover the area of In-application advertisements, its impact on user privacy and recommendations to improve the functionality & reliability of Android phones. The aim is to provide extensive information about the In-application advertisements' attacks and to meet the challenges at an appropriate level.

Some of the popular and common attacks faced by android users due the advertisements in mobile application will be covered in the following section.

3.7 Motivation & Introduction of the Threat Model

The threat model (as shown in figure. 9) will be used throughout this report to highlight the in-application advertisement attacks and their impact on user's privacy.

Attackers have different purposes for instance some have the goal to get money just as click fraud attack (attack on ad-libraries). On the other hand others have an intension to gather personal information of the users (attack on users' privacy). There are many other ways to exploit vulnerabilities of the system.

The motivation of formulating the threat model is CIA (Confidentiality, Integrity & Availability). There are several attacks that works on the confidentiality of the user such as privilege escalation, financial charges and leakage of private information. We will discuss and analyze the behavior of these attacks in our threat model. There are many other attacks that can affect the integrity of the user such as Man in the Middle attack. Availability of the user can be effected by botnets, usurpation or proposal delay.

We have divided the threat model in four different cases (1-4). Case 1 & 2 includes the threats/ attacks whereas case 3 & 4 will cover the security weakness at Users & Android platform respectively. Case-1 will cover the threats or attacks caused by the malicious libraries whereas case-2 highlights the attacks on advertisers or users by the app developer. Figure: 9 show the division and motivation of the threat model.

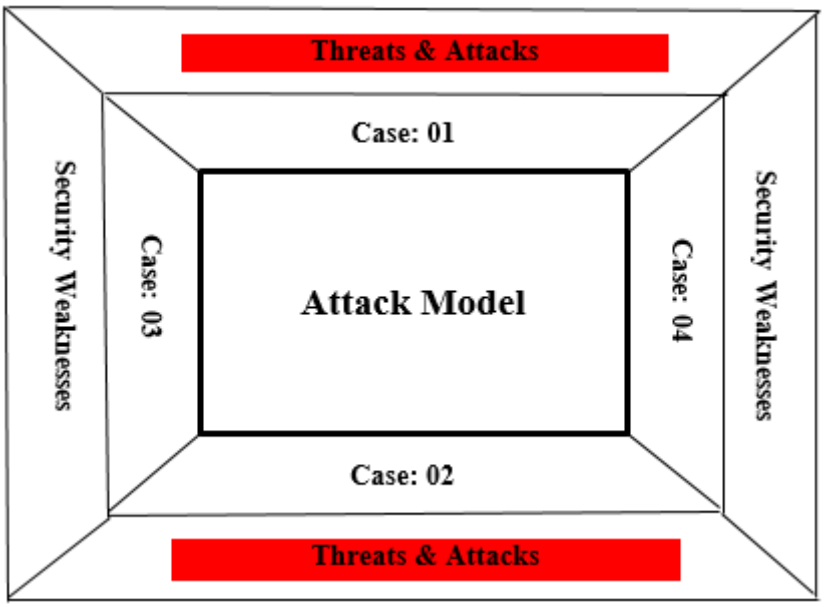


Figure 9. Motivation of the Proposed Model

Threats/ Attacks (Case 1 & 2)

Case: 1

Attack caused by malicious advertisement libraries or networks.

Ad-libraries act maliciously to take control user's phone, private data and money [31].

Case: 2

Attack caused by developers. It works in two directions either towards the advertisers or users side. Figure. 10 shows case: 2 in hierarchal form.

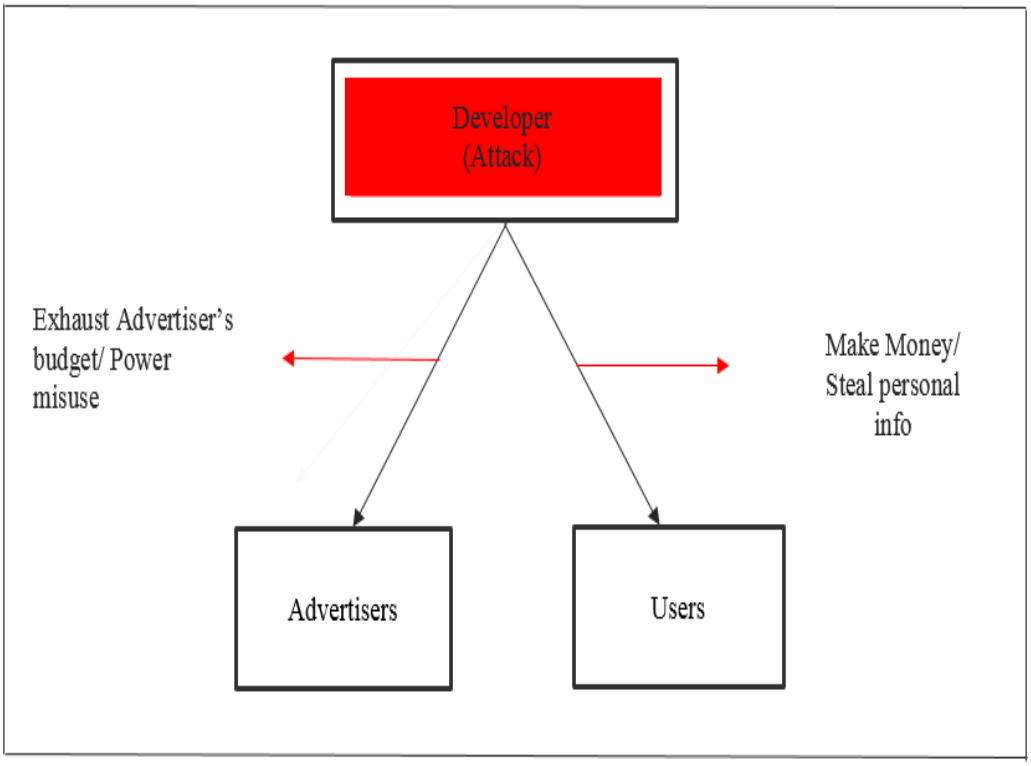


Figure 10. Attack caused by App Developer (Case-2)

Security Weaknesses (Case 3 & 4)

Case: 03

Users lack of security awareness knowledge and implementation of existing defensive measurements.

Case: 04

The most important factor in security control is the Android platform. It has some flaws such as it lacks the privilege escalation Advertisement SDKs and developer's application code permissions.

In the later section we will explain each case separately that will eventually lead us in the formulation of our proposed model.

3.7.1 Threat Model

Threats & Attacks highlighted in figure. 9 are divided in two cases 1 & 2. In the following section we are going to explain them one by one. Case (1 & 2) are interrelated and will be discussed side by side. The reason behind is that app developers and ad-libraries share each other's permissions and placed within a same code with same UID. The only reason for putting them in a different case is to understand their behavior separately.

Red Colour is assigned to the attacks and attack victims in the given model throughout this report.

Presence of advertisements in Applications is a major problem now a days. It can cause multiple threats. For the sake of simplicity and time management we will highlight the common threats here. Figure. 11 shows the hierarchy of the threats or attacks caused due to the presence of advertisements in Android applications. The structure itself explains the commonly used attacks by the advertisement networks.

Apart from those threats some of the legitimate ad-libraries are modified to introduce malicious libraries threats [33]. Some other types of attacks include aggressive library threat. Personal data collection from the users with the help of this attack is possible when legitimate libraries behave aggressively [34]. Some may include the privilege escalation, local or remote injection, and authentication bypass & session manipulation [32].

In the following section common advertisement attacks (as listed in figure: 11) will be explained with the help of threat model.

Assumption of Adversary Model:

In this research we assume that all the adversary attacks are caused by incoming advertisements in an Android application.

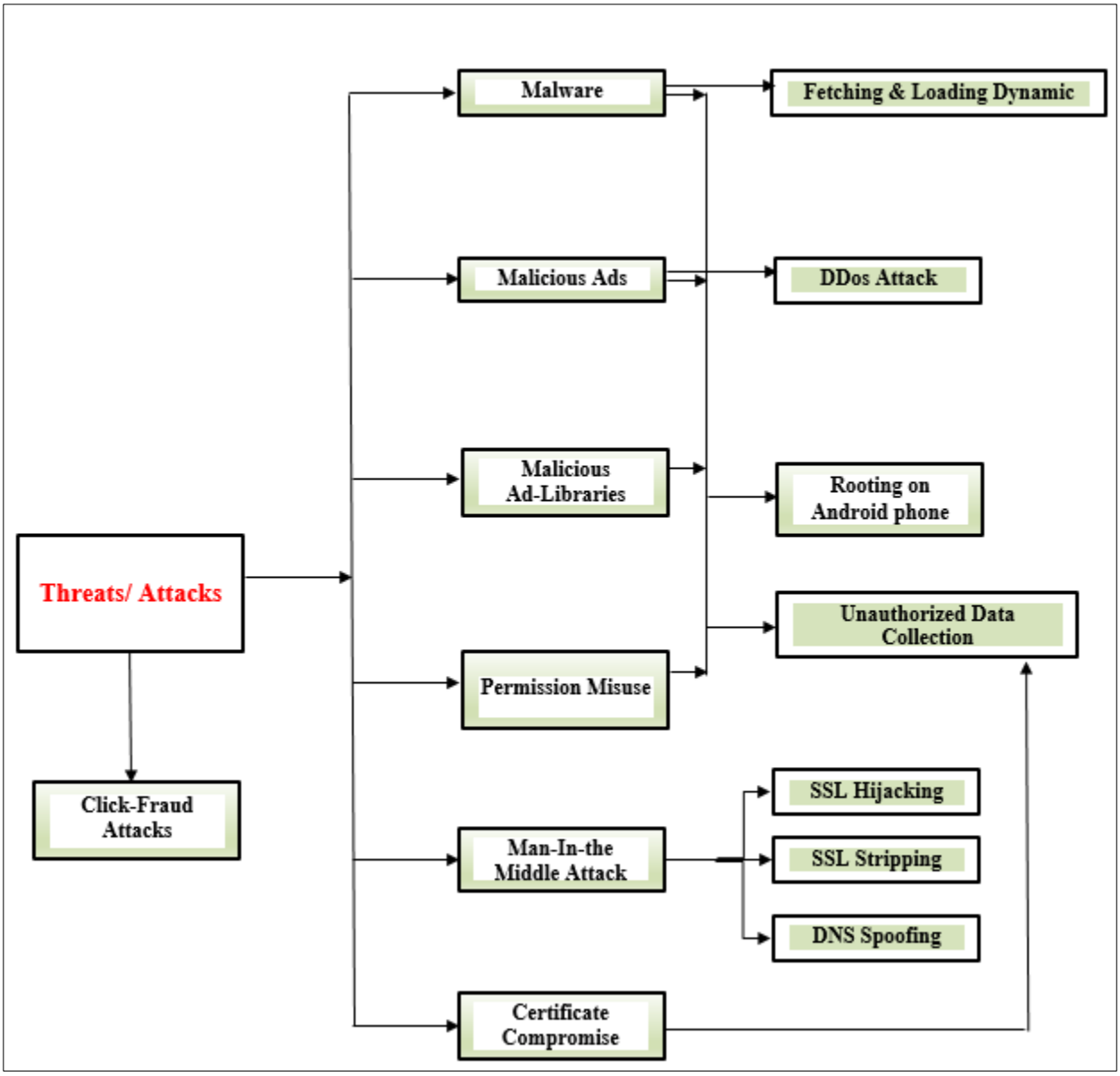


Figure 11. Common Advertisement Attacks

1. Malware

The term "Malvertising" is used for the online advertisements to spread malware [35]. Assuming that antivirus and other ad-blockers in our phone are enough for the protection against advertisement attacks is a false sense of security.

Malware can be inserted in the Android phone through different means such as through malicious software or through malicious ads. When user click on the ads that are displayed on

the screen while using the application they may direct us to another page to download any other software or other application. It may act malicious in nature. Most of the Android malware may come in the form of 'Trojans'. They may have the hidden malicious function as well as the ostensible useful function [36]. It can track the user where ever it goes without having its permission as well. Threat model is shown in figure.12.

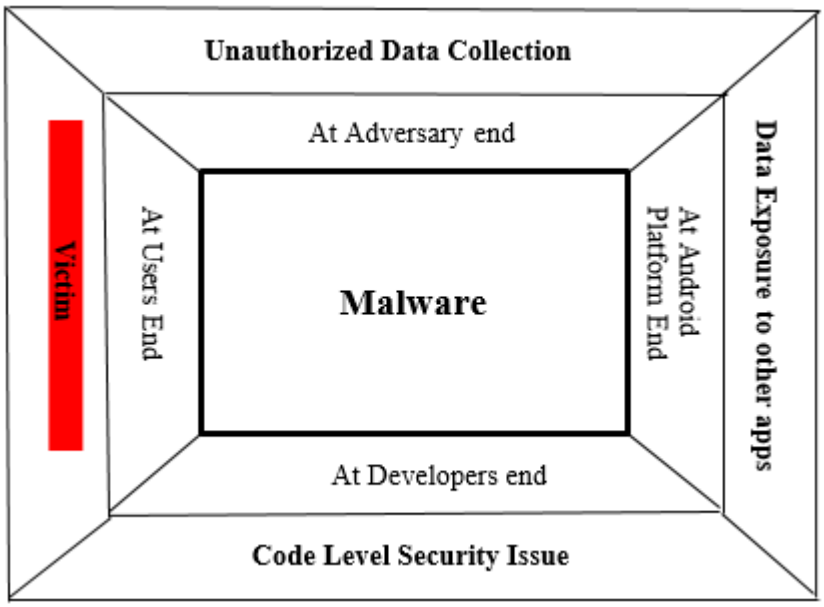


Figure 12. Threat Model 'Malware Attack'

At Adversary End: The above figure shows that malware can be inserted by the advertisers. The sole purpose of the adversary is to collect the unauthorized data.

For showing the behavior of the real applications towards "directed page" we have developed an application and clicked on the displayed advertisements at run time as shown in figure: 13. Directed page may ask the user to download any other software or perform other actions. For instance, after clicking on the advertisement it leads us to another page by providing a link to download 360 Security-Antivirus free and DU Speed Booster- Cache cleaner. This is simply a description of a common case.

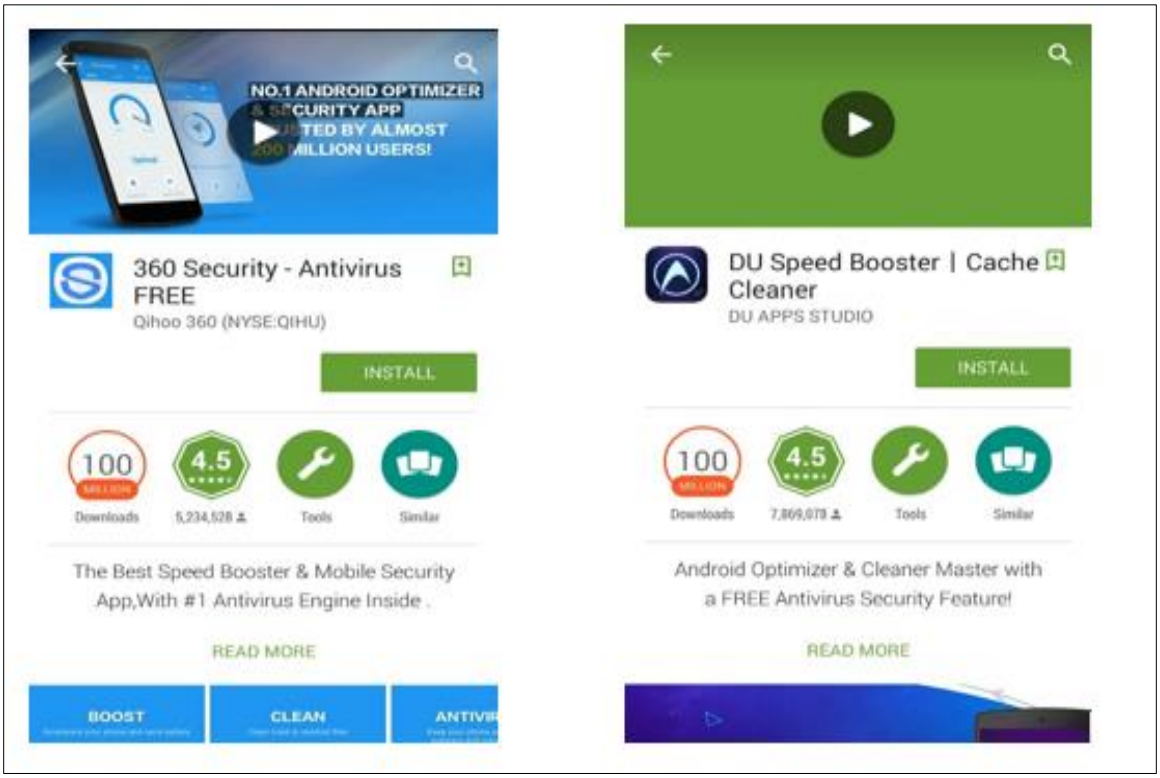


Figure 13. Directed Page

While some of the users may be directed to another page that shows a warning or indication to download another app or software. Sometimes when users click on the provided link and install other apps or software adversary can take control of the device fully or they may compromise the users' security. Attractive advertisements may be dangerous for user's security.

At User's End: Malicious software can record the keystrokes or track the location whenever user will be connected to the internet. Confidential information such as login IDs or passwords of the bank account can be recorded by such malicious software as well. So in any case the victim of the whole scenario is user (that is assigned red colour in the attack model Figure.12). Attackers can get hold to the Android device fully as part of the biggest android botnet. As an example Palo Alto discovered a malware caused by Ad-Libraries who could send and receive SMSs without user's knowledge. Such attacks can give command and control to the user's functionality [37]. Malware can make changes on the user's phone even without its knowledge

such as making changes to user's phone bill. Sending SMS to the user's contact is one of the serious threats now a days.

At Developer's End: Malware attacks are not only restricted to adversaries but these can also be generated by developers as well. In our attack model we have mentioned that "at developers end" there may be "code level security issues" which explains itself that developers may insert the malicious program in the app code while designing the app. These can also be considered as run time flaws. Code level issues such as coding in JAVA in case of Android can cause serious troubles for user's security. It can eventually damage the application as well.

At Android Platform: Official sites and Playstores need to check the security of the app, whether they contain the malware or not. But unfortunately, it has some flaws that will be discussed later.

2. **Malicious Ads**

It is somewhat related to malware injection. The purpose of separating it from the malware is to categories the malicious ads only whereas malware can be injected from developers' code or by malicious libraries as well. The Attack Model of malicious ads is shown in figure.14

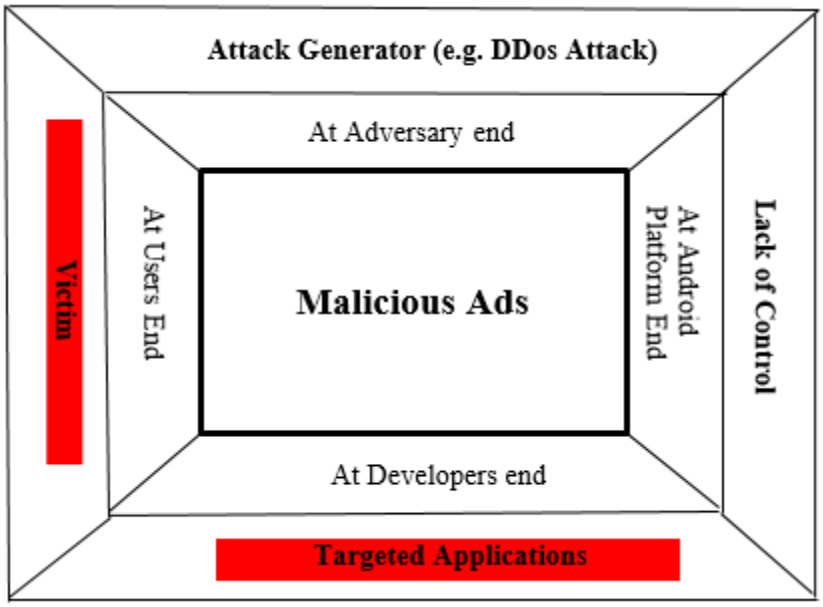


Figure 14. Attack Model 'Malicious Ads'

In the above figure Users and developers both are considered as victims and they are highlighted with red colour in the model.

At Adversary End: Malicious ads masquerade themselves as a legitimate one. They mainly use tried-and-true infection technique [38]. Different attacks can be generated to prompt users to install the recommended actions by the directed page of malicious ads. Users may download the applications from 3rd party unauthorized stores other than Google Playstore. Adversary is the main attack generator in this whole scenario. It can cause certain types of attacks such as DDos attacks.

CloudFare has reported the denial of service attack which involves the mobile ad-networks [39]. In this case ad-networks deliver the malicious ads containing the JavaScript code to target users.

At User's End: Users are victim of malicious ads because they directly deal with them. They unintentionally click on the malicious ads which results in the leakage of personal information or phone hacking. Attackers can fully control the device by using malicious ads.

At Developers End: Malicious ads can damage the application and it can become blacklisted after detection of continuous attack.

At Android Platform: It lacks to defend against hidden malicious JavaScript code.

3. Malicious Ad-Libraries

As described previously, Ad-libraries have access to all sensitive information as their containing application [40]. There are several ad-libraries who take advantage of the 'no privilege escalation' principle.

Ad-libraries can send targeted advertisements to the users and can collect data through the specified permission mechanism. At least 65 Ad-libraries can be used by the developer at a time. There are number of ad-libraries available in the market and some of them are malicious in nature. Developers embed them to make more money without prioritizing user's security. Attack model of Ad-libraries is shown in figure: 15.

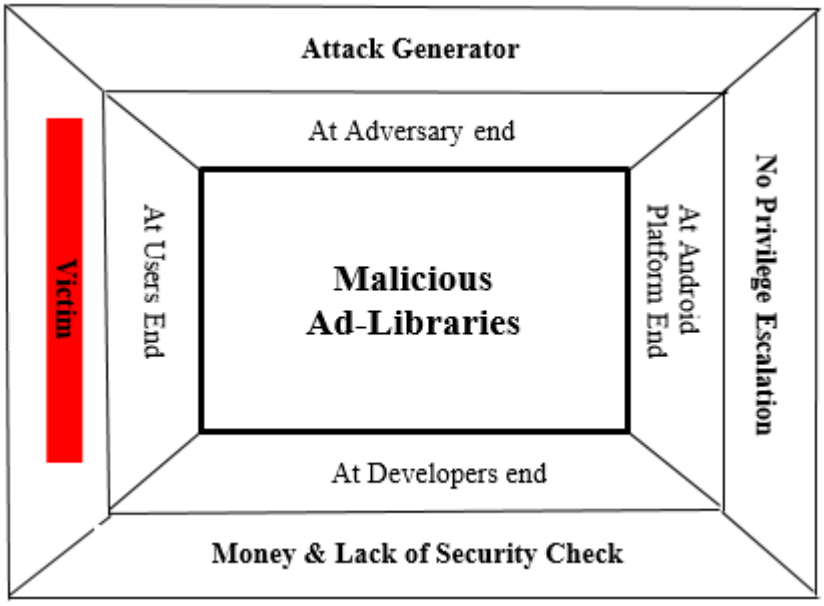


Figure 15. Attack Model 'Malicious Ad-Libraries'

At Adversary End: In the given attack model, we referred malicious ad-libraries as adversary which is the attack generator.

Study reveals the fact that malicious ad-libraries can display ads in SMS inbox as well [41]. They can send SMS after ad-click. This whole process can be done without user's knowledge because users never granted such permissions while the app installation. In the later chapters we will discuss about the hidden permissions to highlight this problem.

Mobile ad-network attacks can trick users to execute malicious attachments. Attackers can built backdoor into the application by targeting an ad-network SDKs.

All the ad-networks cannot be trusted, for instance BadNews has been detected by Marc Roger who masquerades itself as a legitimate network. It was the first attack via the advertisement networks that points the security of the Google Play. The reason behind is that till 2013 it was challenged to get malicious code into Google Play Store. One of the advertisement libraries "Plankton" has been found that can fetch and load the dynamic code. Threats introduced by malicious advertisement library are also increasing these days, where advertisement library masquerades as a legitimate network. BadNews was found in 32 applications across four different developer accounts on Google and it was downloaded between 2 and 9 million times [42].

At Users End: Ad-libraries can behave aggressively to collect the Device Information of user's phone. Leakage of location information, SMS Sim card & device phone number information and the list of installed applications can be obtained to the adversary by the malicious application.

It has also been noticed that most of the Ad-libraries collect IMEI number or other hardware related information of user's device.

At Developer's End: There is a problem at developer's end too. Their main incentive is to increase the revenue from their free application. For this purpose they contact to multiple ad-libraries without checking their ranking and providing any security check mechanism to confirm their reliability.

At Android Platform: We are repeatedly facing the problem of lack of security check at Android platform. It gives a thought that there are some security flaws at Platform end too.

Malicious ad-libraries are embedded in the host app before uploading on the Android market. Google Playstore is a well renowned app store but malicious apps containing malicious ad-libraries can bypass the security check. The reason behind is that their false behavior is shown after their installation on the phone. Not providing the privilege escalation is also considered as the root cause problem in this case.

4. Permission Misuse

The most important factor of the application security control are permissions. As described previously, permissions are demanded at the time of app installations. Study reveals that permissions can be misused by the ad-libraries or developers. Because permissions of both are presented in a same place with a same UID so they both have authority to use each other's permissions. Attack model of permission misuse is shown in figure: 16.

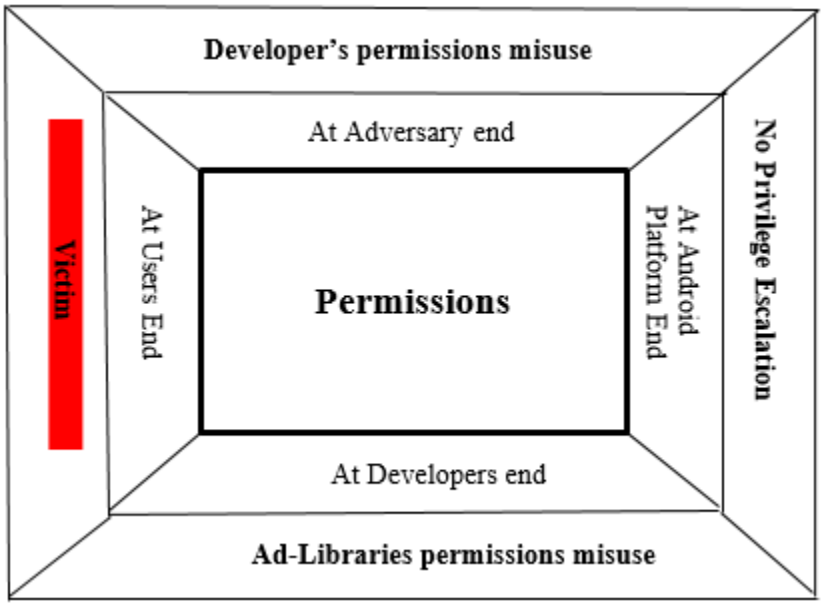


Figure 16. Attack Model 'Permissions Misuse'

At Adversary End: Ad-libraries can use malicious permissions, some hidden permissions and permissions demanded from the host app to exploit the user's security.

At Developer's End: Permissions demanded from ad-libraries can also be accessed by the developers. For instance, Location is mostly tracked by ad-libraries so eventually developers may also get this information on the behalf of their ad-libraries.

At User's End: Attackers can fully control your device and can get your device information fully with the help of permissions [43]. For instance, your phone contact information can be accessed by demanding the <CONTACT permission, as mentioned above, you have no choice to deny some permissions to install the app. The worst case can be described as permissions of Android app and Ad-Libraries can be misused by each other.
Attackers can target business users to know about the company trade secrets and their phone contact detail. They can even locate them and send SMS to their contacts even without the notice of the user. There are many other dangerous permissions used to compromise user's security.

At Android Platform: No Privilege Escalation between developers' & host application permissions.

Rooting of Android Phone: In the given structure (figure: 11) the main causes of rooting an Android phone are malware injection, malicious ads, malicious advertisement libraries and misuse of permissions by the authorities. Acquiring root access to the Android device is called rooting. There is another possible scenario of attack which is possible via the advertisement libraries by gaining complete access to the device. Some of the Android applications act so maliciously that they can breach the security by rooting the device.
Rooting is only good if it goes at the user end. It means if the user can get more and full access over the device. But it mostly inclined at the wrong direction, i.e. malicious software can get hold on the device completely. For the sake of security "super user access" must not be granted to the application even if one has any rooted phone. [44]

5. Man-In-the Middle Attack

It is also considered as “active eavesdropping attack (MiTM)”. In a Client/ Server flow of communication, attackers inserts himself [43]. It has been claimed by the FireEye Mobile Security Team that significant portion of Android applications are susceptible to MiTM attacks [45]. Attack model for this attack is shown in figure: 17.

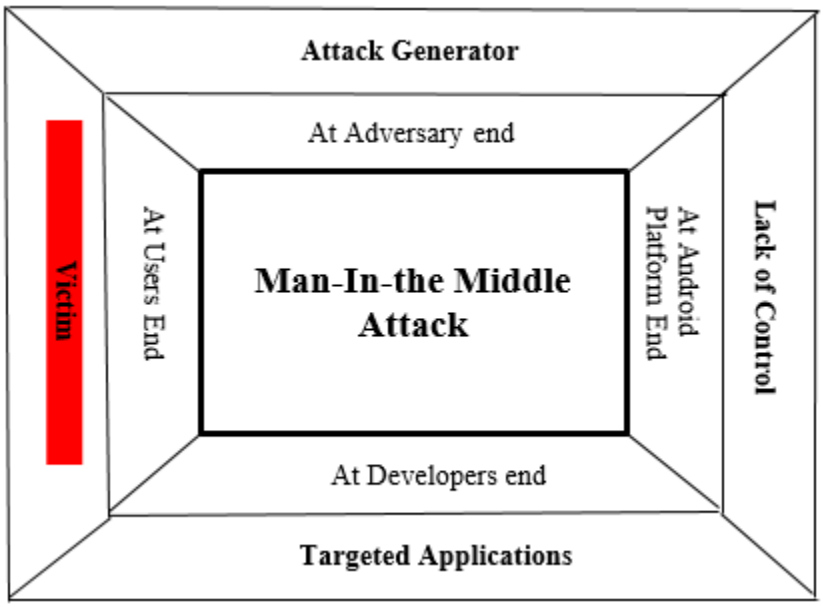


Figure 17. Attack Model 'MiTM'

At Adversary End: Smartphone users are targeted by attackers that comes in the category of MiTM. Commonly these are three 1) SSL Hijacking 2) SSL Stripping & 3) DNS Spoofing [46] are mentioed in figure: 11.

SSL Hijacking Attack: Purpose of SSL Hijacking attack is to capture and obtain the data from user’s phone. SSL (Secure Socket Layer) provide secure communication for insecure networks/channels with the help of encryption [47]. A complete picture of Hijacking is given in figure: 18. It is divided into two phases named as phase-1 & phase-2. In this scenario Eve wants to take control on Alice’s phone. He sets up social engineering attack to know about the interests

of Alice. Her interest may be in games, scientific apps, education or in daily update apps. Our survey analysis shows that majority of the people prefer to install games on their phone. Social Engineering techniques can easily trap the gaming interest of the user. For instance candy crush requests on Facebook etc. Once Attacker (Eve) knows about the interest of Victim (Alice) he uses different techniques to trick her to install the application. As described earlier, advertisements are one of the common ways to trick victims in such cases. First phase of the attack ends here when Alice download and install such apps on her phone.

Phase-2 shows the process after installation of the targeted application on victim's phone. Targeted app contains fake SSL to initiate session hijacking attack. Fake SSL is used to take control victim's communication over the complete network.

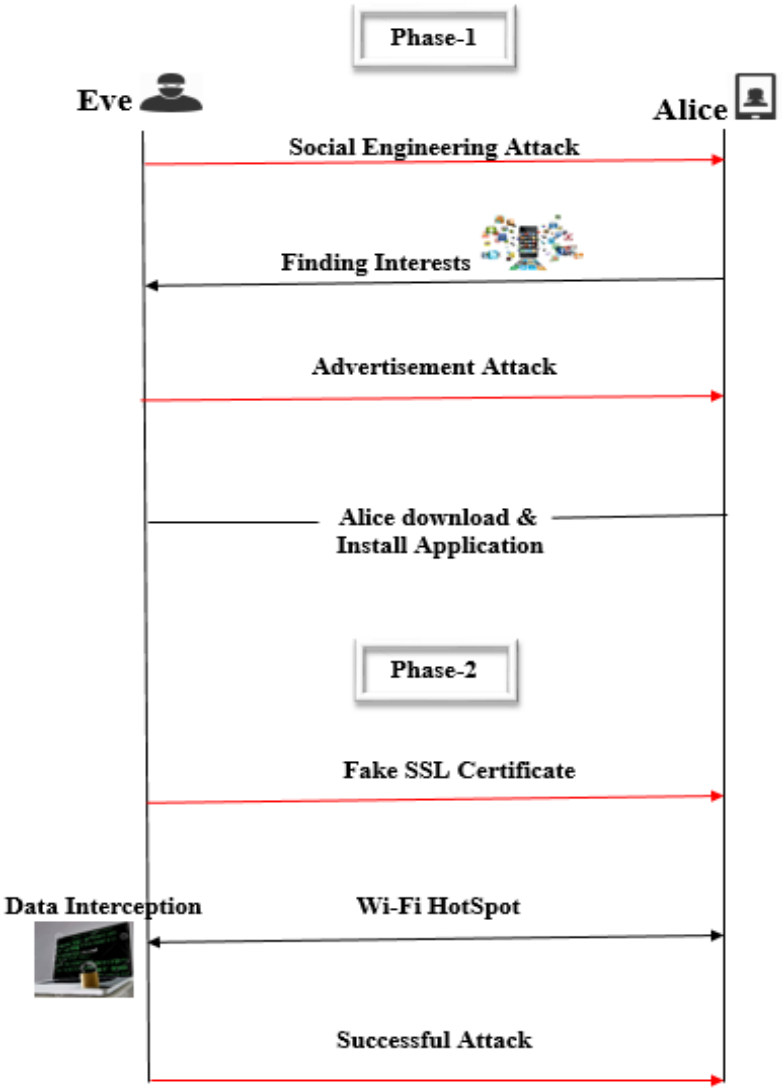


Figure 18. SSL Hijacking

Near the Alice's device, Eve sets up Wi-Fi hotspot. The purpose of this hotspot is to capture and intercepts all the data between a network and Alice's device as shown on the left side of figure.18. This process made possible for the Successful SSL Hijacking attack (MiTM) by Eve (Attacker) to obtain (Victim) Alice's credentials and personal data.

SSL Stripping: Malicious applications initiate the MiTM SSL Stripping attack to redirect HTTPS links to the HTTP connections [46]. Victim of this attack unknowingly connected to the HTTP server while attacker intercepts traffic between user and server by stripping off 's' in HTTPS. HTTPS is a secure version of Http [48] and helps to create a secure channel over insure

environment. Adversary purpose is to stop the secure communication between user and its communicating network.

DNS Spoofing: Logical web addresses into the corresponding IP addresses is done by DNS protocol [49]. In DNS Spoofing attack DNS query is intercepted by attacker and fake DNS response for the client is generated. It is another way to extract unique ID of the Android phone. DNS spoofing can cause serious damage to the user because it is hard to detect on smartphones [46].

At Developers & Android Platform end: Malicious ads and targeted applications have the major role in generating Man-In-the Middle attack.

At Users End: Android users are the victims of this attack, unique ID of their phone, communication with their Bank (in the form of Bank account etc.) can be traced and misused commonly by MiTM now a days.

6. Certificate Compromise

The authentication of digital certificates is done by private/ public key. It links the key (private/ public) to a subject identity such as an organization, an account, a person or a site. Certificate is being used within a given time period and a trusted CA (certificate authority) [46].

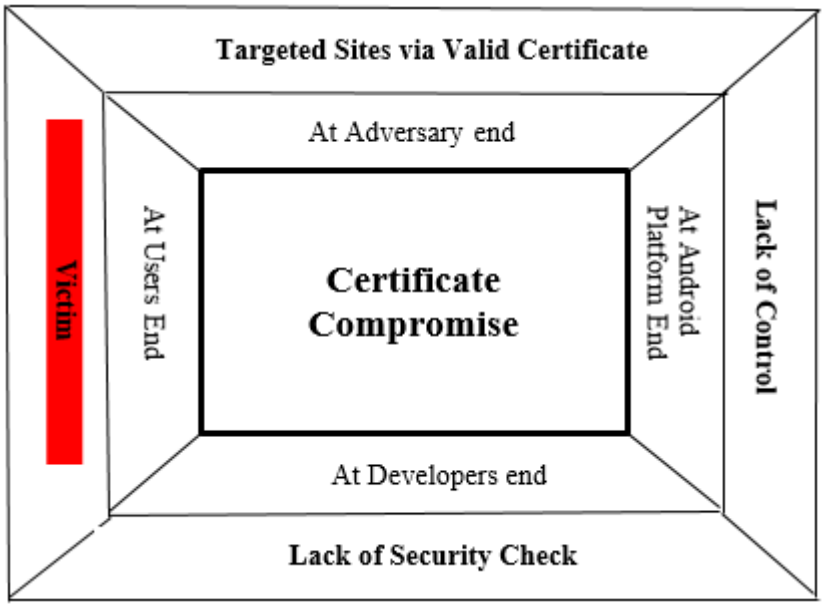


Figure 19. Attack Model 'Certificate Compromise'

At Adversary End: In the last couple of years, it was found that more than 200,000 malware binaries were discovered in the result of digital certificate compromise attack. First organization who suffered by the cyber-attack by compromising digital certificate is Comodo [50]. The authenticity of the web browsers can be achieved by certificate validation. Valid certificate is possessed by the attacker for a targeted site. Figure. 19 shows the attack model designed for certificate compromise attack [51].

At Developers & Platform End: Issues arises when digital certificates are not verified by public key & issuer signature of the certificate. Illegitimate digital certificates are installed by malwares transferred through advertisements in an applications.

At Users End: Digital certificates are used in electronic documents, signed emails and in HTTPS connections [52]. Compromised certificate attacks exploit and vulnerabilities and eventually let users to suffer in the form of personal data loss or being hacked by third parties.

7. Click-Fraud Attack

Android users are not the only the victims of embedded adware, rather attacks can also take place on ad networks as well. The purpose of its prominence is to find out the security flaws at Android Platform. Attack Model highlights click-fraud attack in figure.20.

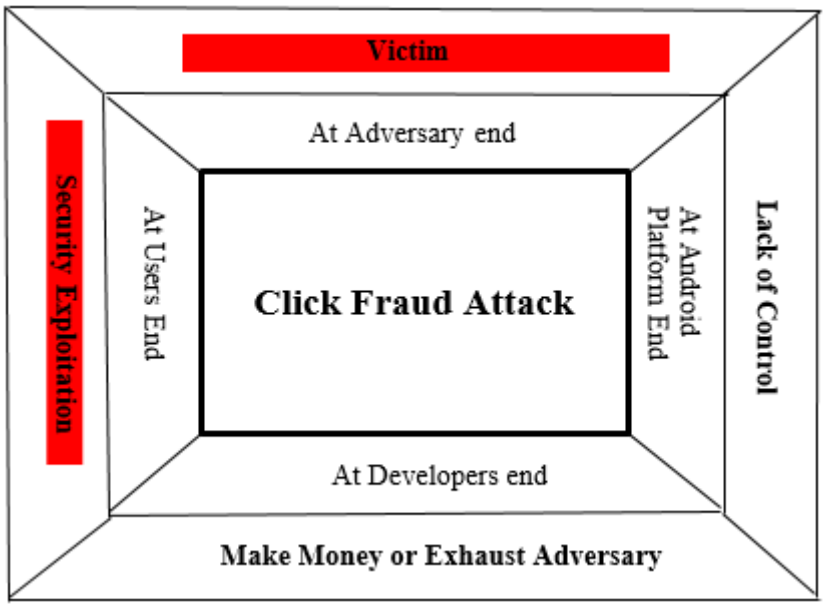


Figure 20. Attack Model 'Click Fraud Attack'

- **Fraudulent Click Events**

Cyber-criminal activities are increasing in new ways such as “click frauds”. The tenacity of this attack is to enhance the revenue of the developers or to exhaust the advertiser’s budget. The more the number of clicks or impressions, more will be the revenue of app developer. Click events can only be detected by viewing the device identifier or IP address. If there are high number of clicks in a short time interval advertisement networks can identify it. But in reality, attackers can modify the device identifiers and generate more revenue [10].

Click fraud attacks may be of three types:

- Program generated touch events
- Click fraud attacks with fake advertisement banners
- Anomalous behavior generated by click fraud attacks

Normal situation of click events and fraudulent behavior of the device due to the presence of bot program is shown in figure.21. It explains that in normal situations ad-network can accommodate many devices at a time who attempt to click on the ads appear on their mobile screen. Left side of the figure shows multiple devices (from 1 to N) and they are recognized and differentiated by their respectable device IDs.

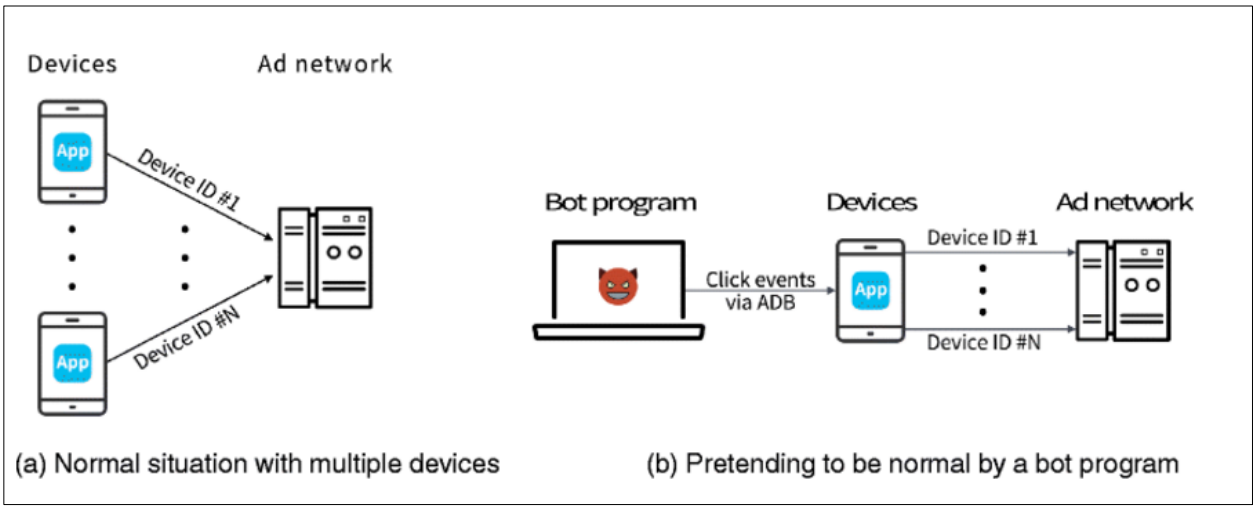


Figure 21. Bot Program Generate Click Events [10]

On the other hand, right side of the figure.21 shows the click fraud attack caused by bot program. It pretends to be a normal program with multiple devices. For breaking the security this attack occur in two forms. Click frauds are not only be generated by Bot programs but it also prevent detection of the devices by their device IDs. It can be done by introducing fake identifiers so that ad-networks may not be able to detect the exceed limit of clicks from a single device. ADB (Android Debug Bridge) is used for the communication purpose between host and android

devices [10]. Ad-clicks may direct the users to download and install different software and other applications. Advertisement networks can also send the location based targeted ads to the users. Ads may act positively or maliciously to the user's privacy. For example they may direct the users to the fake website to insert the account details, or they can make them to install a malicious program on their phone by generating security alarms.

Experiment & Analysis: We have launched Click Fraud attack to explore its impact on ad-networks. There are two major purposes of click fraud attacks. First is to increase third party revenue and other is to exhaust the advertiser's budget [53]. The experiment results reveal that security problem can occur on each side of the picture i.e. either at developer, mobile user or Android Architecture as well. In our scenario we embeded AdMob library in our application code and did not upload it to the Playstore intentionally. We have generated an account on the official site of AdMob.

For the app testing we preferred to get real ads on the phone screen which appear after every 30 seconds. In a normal case, developers upload their app on the Playstore. Users download & install them. They often click on the advertisements that appear on the phone screen while using the application. On the other hand developers get the incentive in the form of money. The total amount of money (as per click) starts appearing on ad-libraries account. In this case our account on AdMob library shows the amount of money as a result of user clicks as shown in figure.22.

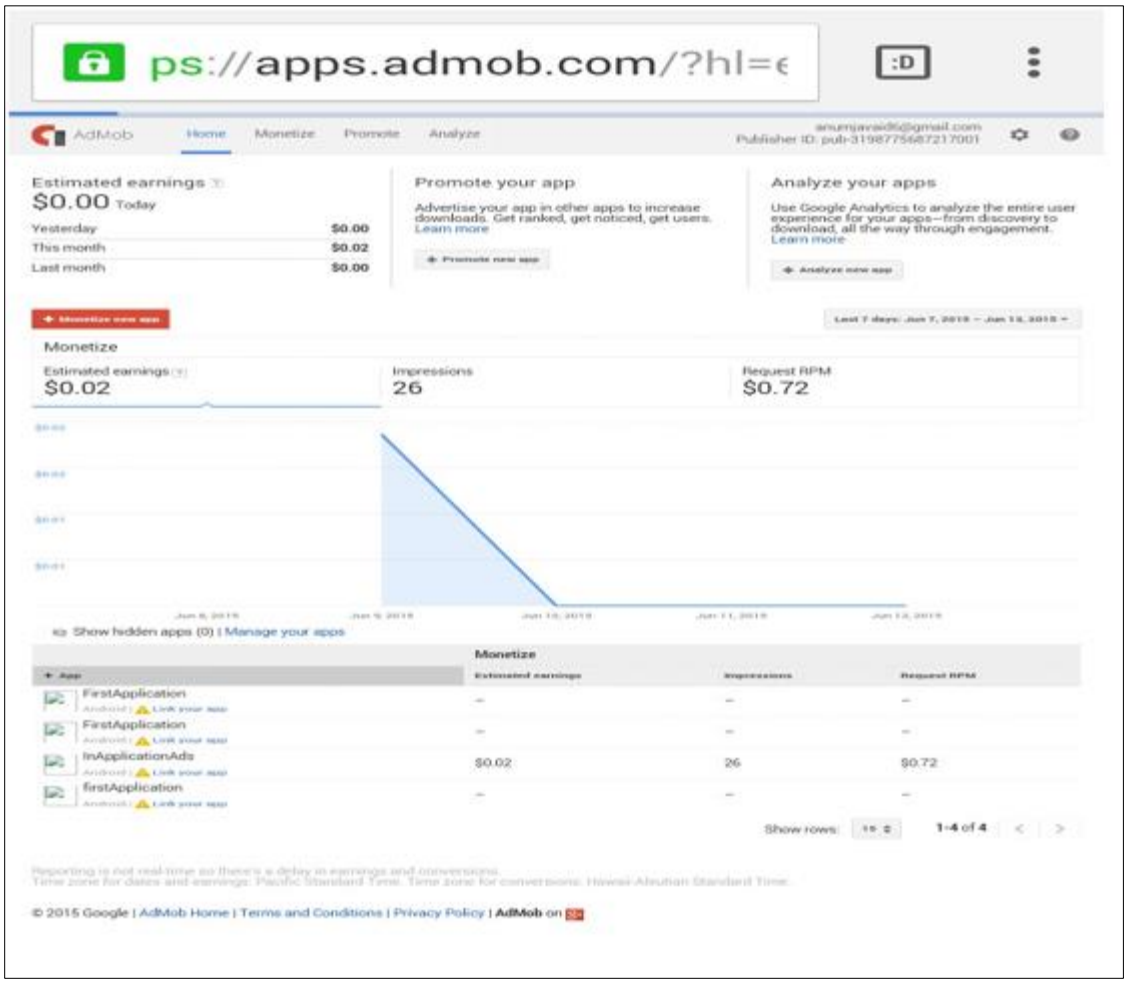


Figure 22. Fraudulent Activity

It should be noted that we as a developer installed this application on our own phones and all the ad-clicks are done by ourselves rather than actual users. So it proves that it is a fraudulent activity. It also prove that there are some flaws at the developer, user and Android Architecture end in order to protect the user's privacy. Publishing networks can attack the users and it can be attacked by the application developers themselves as found in the above mentioned case.

In the next chapter existing solutions to the problem statements mentioned in the introductory chapter will be presented. It will also introduce the existing techniques used to control the attacks highlighted in the current chapter.

4 EXISTING SOLUTIONS and LIMITIATIONS

4.1 Problems & Solutions

In April 2015, 100 best Android apps were selected from Google Playstore on the basis of users' reviews [54]. Later on in January 2016, 100 best Android apps were also been selected based on their popularity. They belong to different categories such as books, business, communication, education, entertainment, finance, food & drink, home & family, lifestyle, games, comics, social, productivity and personalization [55]. Although these apps were considered as top listed Android applications but they doesn't fulfilled the requirements for protecting user's security. Some of them offer password based encryption which is found to be insecure. These popular apps can act maliciously to compromise user's security.

Our aim is to highlight the impact of malicious apps on the basis of users' privacy. The problem comes from the point when such apps installed from 3rd party or unknown sources that disturb the security flow [56]. For example people install free apps that work as an alternative of typical SMS service facility. The reason is to save money and send SMS along with Audio and Video. Privacy is neglected at each step for the sake of money. Malicious nature apps can steal user's data and use them for their own purposes.

This chapter includes the existing solution of the problem statements given in chapter: 01. Existing solutions include "ad-blockers" and existing techniques which are useful in identifying some of the peculiarities of embedded adware. It also covers the limitations of the existing solutions for further improvements.

4.2 Existing Techniques

There are some techniques proposed by many researchers for protection of user's data and system resources from In-advertisement application attacks.

We analyzed many techniques and merge them at a point where they be used collaboratively.

Our task is to obtain the following results:

- Detection and prevention of Ad-frauds
- Isolation or separation of advertisements and application permissions
- Reduction of permission bloat
- Collection of Smartphone apps those are malicious in nature

After literature review and analysis we have chosen some techniques on the basis of their efficiency and performance. These include traditional approaches, Ad-Droid, Ad-Split, A-Frame, PEMO, Ad-Attester and Ad-HoneyDroid. The approach of these techniques to protect user's security is explained below.

4.2.1 Traditional Approaches

As described in the previous chapters advertisements and applications' permissions are not separated in a typical Android architecture. This can cause serious damage to the user's privacy as described in the threat model. Privilege escalation is one of the solution of this problem.
There are two traditional approaches of privilege escalation. The reason is to provide separate domain to ads and apps by granting them different permissions. Purpose of traditional approach is:

- Privilege separation of ads and apps.
- Give the user empowerment to reduce the permissions of an application.

Privilege separation in traditional approaches can be done in two different ways [58]:

i- Privilege Separation within a process

It is purely based on the "permission model" where multiple protection domains run in the same virtual machine and process. Java class is assigned to its own protection domain. Java runtime exist in the rest of the application whereas advertisement library exist in a same process.

Advantages:

- Same virtual machine
- No extra methods need to call for running the process of application or advertisement library.
- Both have their own permissions

Disadvantages:

- Native libraries are included by different applications which can violate the integrity of virtual machine running in a same process.
- Android development is based on the "Dalvik Virtual Machine". Isolation between different parts of the code within a same virtual machine is not supported by the default process of "Dalvik Virtual Machine".

ii- Privilege separation between process

Users can install advertisement free applications by uninstalling and controlling the original application (ideal assumption).

Advertisement libraries run as a separate application in privilege separation between processes of traditional approaches.

Advantages:

It is the advantage for the user's because they can uninstall the advertisement applications from the original application.

Disadvantages:

Advertisement networks that paid the developers for inserting the ad-libraries SDKs in the application code can suffer if users uninstall the advertisement applications.

Limitations of Traditional Approaches

- Some of the applications have native libraries that can violate the integrity of the virtual machine.
- Dalvik virtual machine do not provide and support such privilege separation.

- Periodic Audits of the advertising networks are necessary.
- Android platform would need to be extended to support install time dependencies

4.2.2 AdDroid

Purpose of using AdDroid: There is another privilege separation method called 'AdDroid'. It is a framework designed for advertisement privilege separation. The main task of AdDroid is the separation of application and advertisement functionality from each other. They state that this methodology will lessen the requested permissions i.e. application will even work without requesting privacy sensitive permissions [18].

Methodology of AdDroid

For the Android platform, new advertisements APIs are introduced by AdDroid along with the corresponding advertising permissions. [18]

Limitations of AdDroid:

Two drawbacks have been found for the AdDroid model.

- AdDroid is need to be implemented by the development team of Android. Existing advertising libraries' functionality should be incorporated into the Android API.
- There are some limitations for advertisers and developers in case of using AdDroid. Apart from the Android release cycle, APIs of the advertising networks are declared and updated in their own pace. While in case of AdDroid they will have to depend on it even if some updates are required. This makes the entities unhappy for using this model.
- By using Ad-Droid even for small changes such as adding new advertising networks, signed configuration updated files can be disseminated by the Android market that describes ad-web service APIs.
- Platform updates might be required in case of larger updates.

4.2.3 AdSplit

Purpose:

- Separation of smartphone advertisements from application
- Protects advertisers against click fraud and ad-blocking
- In smartphone applications containing the advertisements, Ad-split has the specialty to reduce the permissions bloat.

Methodology:

In two different processes and activities, advertisement application and host applications are separated. AdSplit uses the technique known as 'transparency technique'.

Through the transparent region and in the application activity users are able to the see the advertisements [59].

Limitations:

- AdSplit uses the transparency technique that can cause different problems. For example Click Jacking and its variations can be caused by it.
- Overheads due to the transparency technique because multiple layers are combined.
- Current mobile advertisement architecture is changed by using this technique because stub library is required inside each application.
- This technique is lacking the commercial testing.
- Original ad-library is needed to be replaced with the stub library by the developers.
- For the communication between the stub library and the advertisement services of the Ad-Split, developers need to construct the standard IPC message.
- These changes need to be done in order to isolate other third party components. This is the reason Ad-split is difficult to extend even though this process can be automated to some extent.

4.2.4 PEMO

Ashmeet Kaur et.al [60] introduced this technique to empower users to reduce the permission bloat. Users can only select those permissions that are required for installing an application.

PEMO is an enforcement framework that modifies application's permissions. It works with the assistance of package manager functionality. Process of installation, upgradation and application removal has been done by the main API of the android known as Package Manager. Package manager and package installer collectively perform this activity.

The author of the paper has proposed and developed an application known as "Permission Modificator". They modified the concepts of the App "Advanced Permission Manager".

The main advantage of using PEMO is that it does not require the modifications in the Android System to work properly. They state that this framework is well-matched with the current security mechanism and it is tested on Android OS.

Limitations of PEMO

All the users do not understand the meaning of the permissions that are displayed on the permission list. They can uncheck the required permissions needed to perform the function as well, such as INTERNET ACCESS permission. The proposed application can be stopped working by blocking necessary permissions. Or sometimes system may also be crashed.

4.2.5 Ad-Attester

Purpose: It is designed to prevent and detect the widely spreading problem of well-known ad-frauds in Android based systems.

Methodology: Based on ARM's trust Zone technology the author proposed a framework called Ad Attester. Authors of this paper targeted two security issues known as 'unforgeable clicks' and 'verifiable display'. There is a requirement of Ad Attester utilization, one has to install the attestation application in a secure world and on the other hand ad attestation APIs will be provided by Ad Attester for ad providers to adapt to their ad SDKs.

Secure online Mobile ad-attestation is done with the help of Ad-Attester to detect ad frauds in mobile applications. Two types of ad-fraud like 'bot-driven' and 'interaction-driven frauds' are detected by ad-Attester by providing verifiable attestation [61].

Limitations of Ad-Attester:

- Ad-Attester needs per-device key in order to work properly and that is the reason it is still not widely spread. If they solve this problem by putting the TZAttester to normal OS and relaxing their threat model as well as trusting the normal OS then it can be acceptable.
- Ad-Attester can detect the ad frauds by the botnets and developer but they cannot detect ad-impressions and ad-clicks attack by the developer.
- Another limitation of Ad-Attester is that it cannot detect the ads that violate the ad policy of the ad provider.

4.2.6 Ad-HoneyDroid

Purpose

Collection of Smartphone applications those are malicious in nature. It identifies incoming malicious advertisement attacks by using signatures [62].

Methodology:

Smartphone apps can be attacked by various malicious advertisement libraries. To capture such advertisements there is a method proposed by the author of this paper known as "Ad-HoneyDroid". A crawler has been designed to gather apps APK on the Android marketplace [63]. They manually collect ad libraries and their vulnerabilities. After collecting such information Ad-HoneyDroid detects malicious advertisements by executing the application. To detect the attack event they opted the methodology used in API sandbox and TraintDroid [64]. Figure.23 shows its scenario.

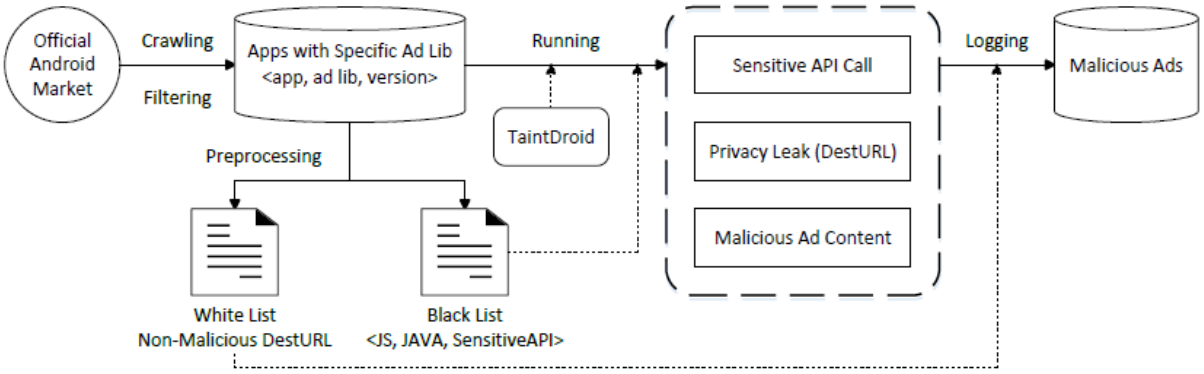


Figure 23. Workflow of Ad-HoneyDroid [62]

As shown above the malicious nature apps can be collected by the four steps of Ad-HoneyDroid such as Crawling, Preprocessing, Running and Logging.

4.2.7 A-Frame

Purpose

- It is designed to solve the over-privileged problem in android applications containing advertisements.
- It's a developer friendly method used to isolate the permissions as well as it provides display and input isolation. It is used to isolate un-trusted third party code.

Methodology:

Implementation of A-Frame has done by minimal changes in the code-base of Android. Both Applications and advertisements are placed in the same drawing surface but executed in different processes. The idea of A-Frame is adopted from browser's i-frame. It is a generic solution used to restrict the un-trusted third-party code [65].

4.3 Summary & Comparison between Existing Techniques

Researchers find different ways to secure Android users via different means by detecting the malicious nature apps or finding the security problem at developers end. Some of them are used to separate the permissions of ads and app. Three of the techniques Ad-Droid, Ad-Split and A-Frame achieved this task. Table.1 gives the brief summary and comparison between all selected techniques.

Table 1. Comparison between Existing Techniques

Techniques	Permissions separation of App and Ad	Users can select the permissions of their own choice	Prevent and detect ad-frauds	Collection of apps those are malicious in nature	Reduces Permissions bloat	Provide Input and display isolation
Ad-Droid	Yes					
Ad-Split	Yes		Yes		Yes	
PEMO		Yes			Yes	
Ad-Attester			Yes			
Ad-HoneyDroid				Yes		
A-Frame	Yes					Yes

While comparing the other techniques we concluded that only PEMO is able to grant user empowerment. Ad-Split and Ad-Attester helped to prevent and detect ad-frauds. Whereas permission bloat has been reduced by Ad-Split and PEMO, which is the biggest problem in the domain of protecting Android user security. Lastly, display and Input isolation is provided by A-Frame that proved to give less overheads as compared to the overhead caused by Ad-Split while privilege separation between ads and applications.

4.4 Existing Ad-Blockers

There is another method to stop facing ads in the application. We called it Ad-blockers. Online ad revenue model can be undercut by the ad-blockers [66]. We opted this solution because of its ease of use. It is easily available on the Android app stores. Our research and results of survey shows that most of the users dislike appearance of ads while using the application. Its graphical results will be given in the later chapters. The purpose of ad-blockers is to eliminate annoying ads, reduce risk of viruses and screen clutter. Amongst the existing ad-blockers "ad-blocker plus" is found to be the most popular one [67, 68]. As it blocks the incoming ads, ad-blockers are considered as the growing threat of the business models of ad-publishers and ad-industry [69, 70]. In the research work of Enric Pujol et al [71] it is proved that most of the users are interested in blocking advertisements rather focusing to protect their own privacy. Most commonly ad-

blockers are Ad-Blocker Plus [72], Ad-Block [73], AD Block, Ghostery [74], Disconnect [75] and AdMuncher [76]. Some of the Ad blockers are available free of cost in the market such as AD Block whereas most of them are also available in their paid version such as AdMuncher [77]. Android phone is categorized in two ways i.e. rooted & unrooted devices. Some of the Ad-blockers are available for rooted and unrooted Android devices. For rooted devices AdAway is considered as the good way to block to block ads because it does not required many resources and it is easy to use. On the other hand it has some ability to white/blacklist the stuff that is not included in the standard list. Apart from all these functionalities, AdAway has a biggest disadvantages such as it cannot be downloaded from Google Playstore and in worst cases it requires root access [78]. For unrooted devices "Adguard" is considered as one of the good ad blocking services. It has many advantages such as it is available free of cost and its ease of deployment. It can block trackers and provide protection against phishing attacks. Apart from all the available functionalities Adguard is not recommended for those who wish to secure their data from attacks. Adguard cannot provide protection against https site, it can't block them. It is based on VPN and uses more battery. Browser compression services cannot be used together with Adguard [78, 79].

4.4.1 Limitations and Security Flaws of Ad blockers

Existing Ad-blockers have some security flaws at their end. Figure: 24 show some categories of ad-blockers flaws.

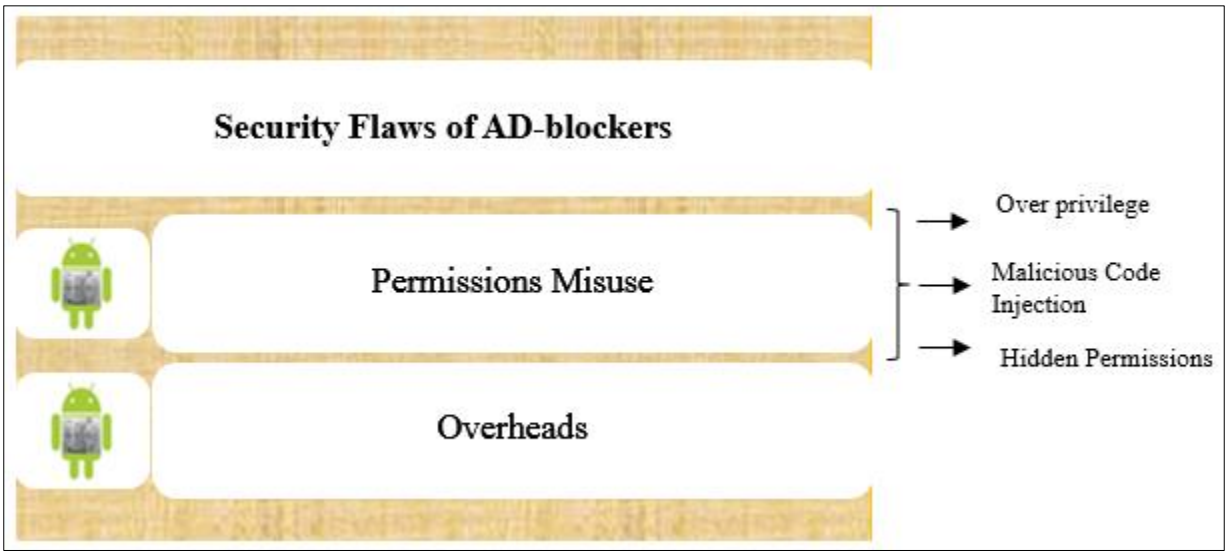


Figure 24. Ad-Blockers Security Flaws

Ad blockers can misuse the permissions for example they can act maliciously by incorporating malicious codes in their ad blocker code. Ad blockers installed from 3rd party stores have more chances of malware or malicious code injection.

Most of the Ad-blockers demand more permissions that can disturb user's privacy rather than protecting them from malicious ads. Some of the permissions are hidden and they can get hold the whole device even without user's consents.

Apart from INTERNET permission ad blockers require the ability to automatically set a device's server to 'localhost'. It is considered as the serious security flaw in Android issue section of Google [80]. Over privilege of these permissions will be given in the next chapter by exploring the ad blocker at a real time.

Ad blockers can cause negative impact on memory usage [81]. By using the ad blockers memory overheads of Android phone increases. This is again one of the flaws of using Ad blockers with excessive permissions. Most of the ad blockers do not block those advertisements but they only hide them from users. Users may exploit by the excessive permissions of applications, ad-libraries and ad blockers who can act maliciously in compromising user's security.

In the next chapter we will introduce privacy protection methodologies, analysis of android application permissions via different means.

5 PERMISSIONS ANALYSIS RESULTS

5.1 Analysis Methods

Mobile users can face various threats and vulnerabilities due to the presence of applications in their phone. Analysis methodology of Android applications is necessary to be selected. Usually there are two analysis methods in practice i.e. Static and Dynamic Analysis. Malicious patterns of Android Apps are scanned by using the analysis techniques as given in table. 2.

Table 2. Analysis Techniques [82]

Static Analysis	Dynamic Analysis
Scanning of malicious patterns	Check the effectiveness of Android Apps at run time by executing them in controlled virtual environment
Dex files analysis	
App de-compilation	
Pattern search	
Manifest file analysis	

As described in the above table static analysis scans malicious patterns of the application. It helps to maintain quality of the code [83]. Malware is not executed in the static analysis technique rather it process or inspect a sample. On the other hand behavior of the android applications can be analyzed by executing the samples of malware [84].

Dynamic analysis of android applications can be done by executing them at run time.

5.2 Analysis Strategy

Static Analysis means “Reverse Engineering”. This chapter includes the static analysis of Android applications, it examines the code without executing the program.

5.2.1 Application Anatomy

AndroidManifest.xml is obtained via Apktool to extract permissions, intents, activities, services and broadcast receivers. This process gives us the list of permissions at the AD Block developer end. Figure. 25 shows the code extraction procedure via Apktool.

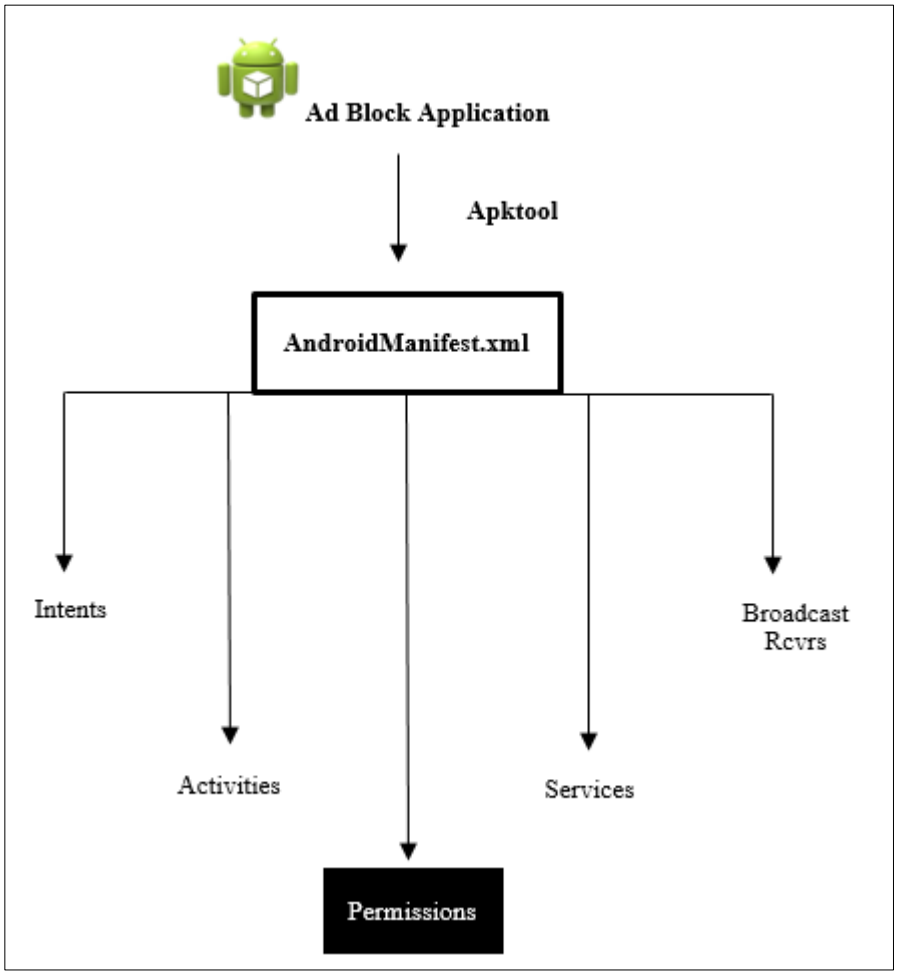


Figure 25. Manifest File Extraction

The above figure shows the anatomy of an application. The main building blocks of Android application are called “components”. They are further divided into four categories such as activities, services, content providers and broadcast receivers. Introduction and explanation of the entities used in figure.25 are given in the following section [85].

Activities: A single screen with a user interface has been shown by activity. We can describe activities with the help email example. Here, a single activity is used to describe a single function for example: 1. to list new emails, 2. compose email, 3. read emails etc. Although these activities combine together to give complete output.

Services: Long running operations in the background has been run or controlled by "services". As compared to activities it does not provide the user interface. For example to play music in the background while user is running a different app.

Content Providers: A shared set of App data is managed by "content providers". Storage of App data in the file system, on the web, other accessible locations of the App and SQLite database storage can takes place with the help of "Content Providers".

Broadcast Receivers: It responds to the system wide broadcast announcements. System can originate the broadcasts for example screen capturing and low battery.

Note: This whole process has been done with the help of "intent".
Intent: An asynchronous message has been activated by "**intent**". Intents are used as a messenger that binds individual components to each other at run time. We can pass intent by either starting a new activity or service, to initiate a broadcast message or to perform a query with the help of content provider.

5.3 Our Contribution

We explored permissions of the in-advertisement applications by five different ways.
In the following section we will describe the main structure and anatomy of Android Application to extract the components that need to be analyzed in our research.

Analysis of Permissions

To highlight the over privileged problem of permissions and exploring the hidden permissions following methods are adopted in our analysis:

1. Permissions at the time of installation
2. Display of permissions in phone after App installation
3. Explore hidden permissions (by using existing apps)
4. Permission Extraction from online tool during Static analysis
5. APK decompilation to extract Source code and analyze them in Android Studio

Our analysis is divided in two directions:

1. Analysis of In-advertisement applications (Problem Analysis)
2. Analysis of Ad-Blockers (Existing Solution Analysis)

Figure.26 shows the hierarchy or stages of our analysis methods. In the first stage in-advertisement applications will be analyzed by using five different methods as mentioned above.

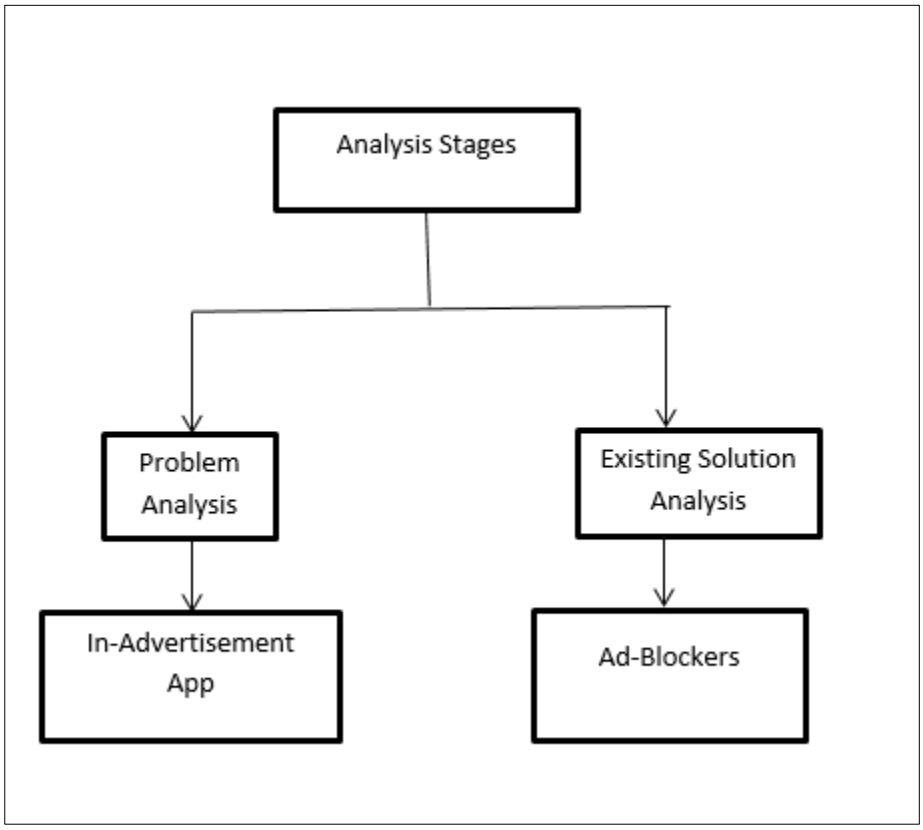


Figure 26. Analysis Categories

5.3.1 Problem Analysis (In-Advertisement Applications)

Case Study: 01

Most of the Android users use the free version applications. Therefore we have chosen the commonly used free application for our analysis. In the first case study, we considered the "Horoscope" application. It is selected for the experimental purposes and it is downloaded from Google Playstore. This app is an open source written in Java so we were able to download all of its Java files and examined through Android Studio. We installed it on the Android phone with 4.4.2 KitKat version. This app have advertisement libraries embedded in it and displays advertisements on runtime. As shown in the figure 27.

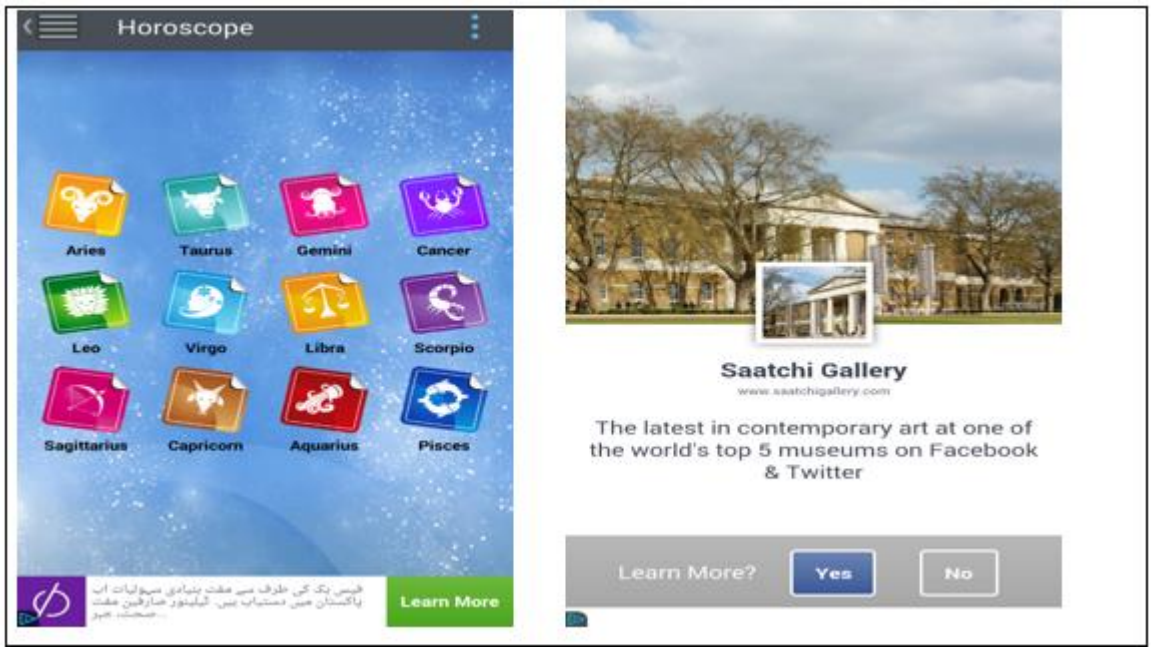


Figure 27. Horoscope App Displaying Ads at run time

The above figure shows that banner ads are displayed at run time.

Now we will extract and analyze its behavior according to the predefined categorization. The main goal of the analysis is to determine the information flow through the application to highlight the excessive private information demanded beyond what is needed for its functioning.

1. **Display of Permissions at the time of installation**

This process is displayed at user end. It means that user can see the permissions at the time of installation. Research shows that mostly users accept the permissions without paying attention to them.

The following snapshot (Figure 28) is taken at the time of installing Horoscope app. It shows that Horoscope app demands the permissions from In-app purchases till the device ID and call information.

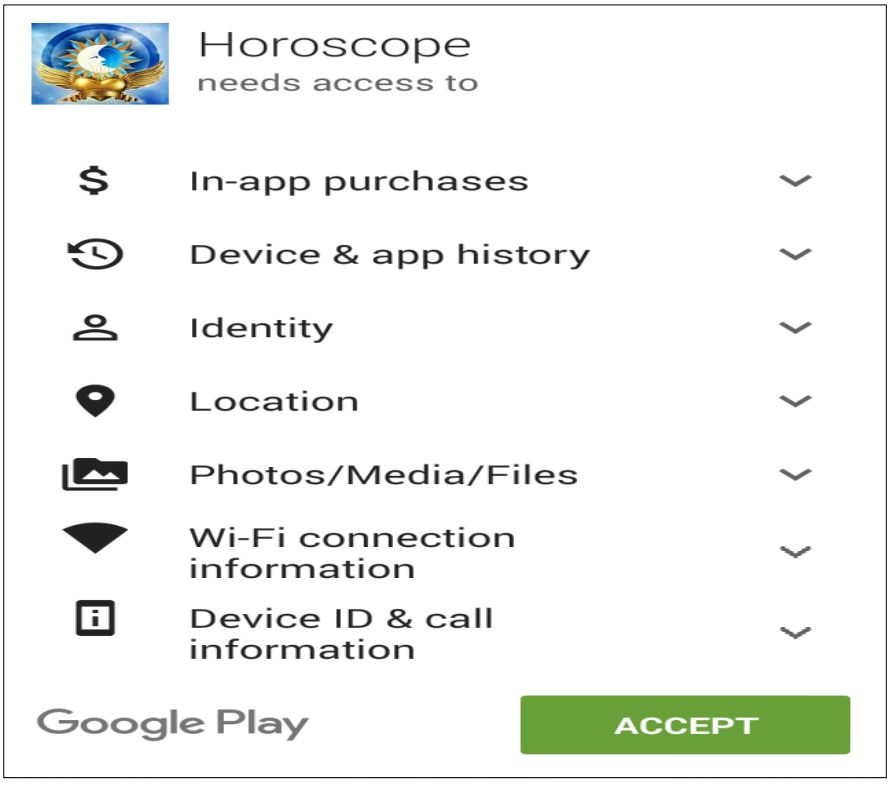


Figure 28. Permissions at the time of Installation

These permissions are driven by the careless approach of the developers who are more concerned about generating more revenue. Developers demand them by declaring them in the manifest file. “Horoscope” permissions can be declared in the manifest file by the following method.

<manifest>

```
<uses-permission android:name="com.android.vending.BILLING" />
```

In-app purchases are used to make purchases inside an application.

Device and app history permission is used to read sensitive log files/ data, can retrieve system internal state and running apps. Device and app history can read bookmarks and history of the phone [86]. To find out the accounts stored on the device “identity” permission is used [87].

These permission systems not only access the Device ID but it can also control the activity timeline of the user.

2. Display of permissions in phone after App installation

This process is also taken at the user end. It means user can easily access this information by simply opening the “App info” in its phone. The purpose of this step is to understand the actual meaning of the permission terminology used in the first category. Mostly users do not know the detailed meaning of the titled permission. For instance from the first method user can take the “Wi-Fi connection information” meaning is to collect the information about whether the Wi-Fi is on or off. Whereas it may means to get Wi-Fi connection along with writing the bookmarks and viewing history as well.

We explored the “horoscope” application by the most negligible information available on our devices i.e. “App info”.

We explored them that give the more meaningful information as compared to those who are available at the installation time. Screenshot of the results is shown in the figure 29.

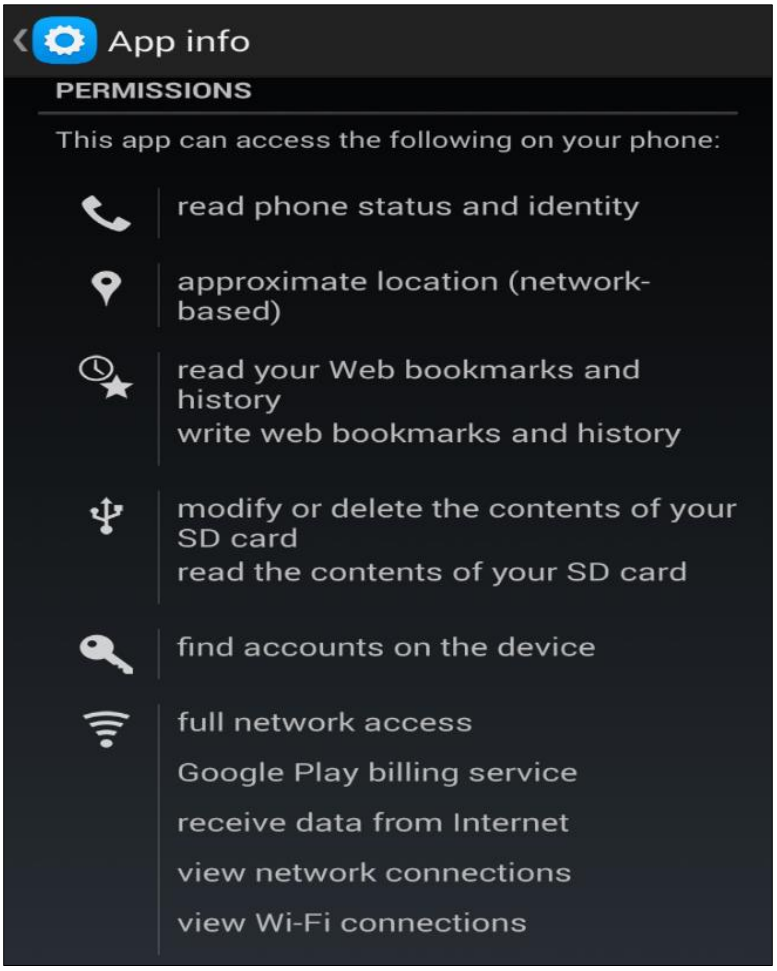


Figure 29. Permission Display in "App Info"

The above figure shows that “WiFi Connection information” in the method 1 is explained in method 2. It shows that it can fully access the network and can access the google play billing services. WiFi and network connections can be viewed by the permission system.

On the other permission display at the time of installation was unable to show the permissions who can read, modify and delete the contents of the SD card as shown in the method 2.

The above two methods shows the permissions analysis at the users level. In the following method we will explore the permissions via different approach.

3. Explore hidden permissions

We put it at ‘lower expert level’. It means that we explored the hidden permissions of the installed applications with the help of existing app named as “Permission Explorer”. This method helped to highlight the “horoscope” hidden permissions.

At the time of installation and even after installation of the application user is unaware of the hidden app permissions. Our analysis on those permissions gave us the following results as shown in the figure 30. They have been generated with the “permission explorer”.

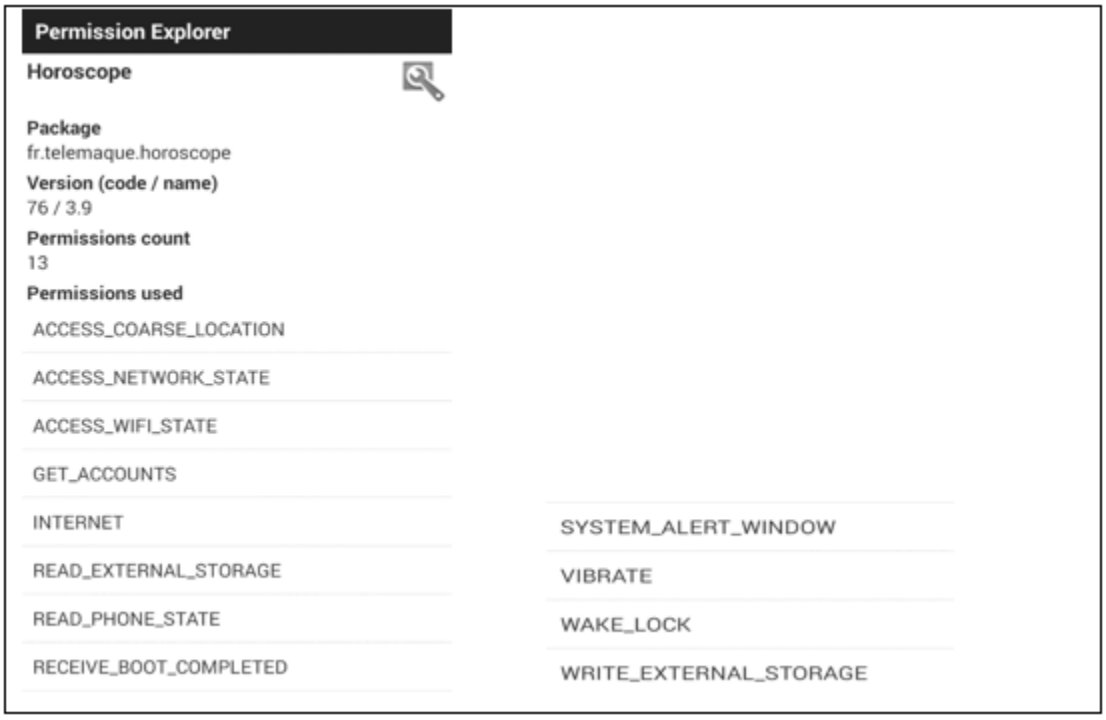


Figure 30. Hidden Permissions

While comparing it with the other two methods, this approach gave us the more meaningful results. System_Alert_Window, Vibrate, Wake_Lock, Receive_Boot_Completed are the additional permissions who were not displayed at the time of installation.

Some of the hidden permissions may cause the insertion of the malware in the phone.

4. **Permission Extraction from online tool/ A5/ During static analysis:**

This is at medium expert level. This step is included to analyze is there any malware present in the app or not. The results will display the necessary permissions required for this application to function properly and discard the extra permissions. We selected an online available tool called A5 and submitted our APK file of Horoscope and got the results shown in the figure 31.

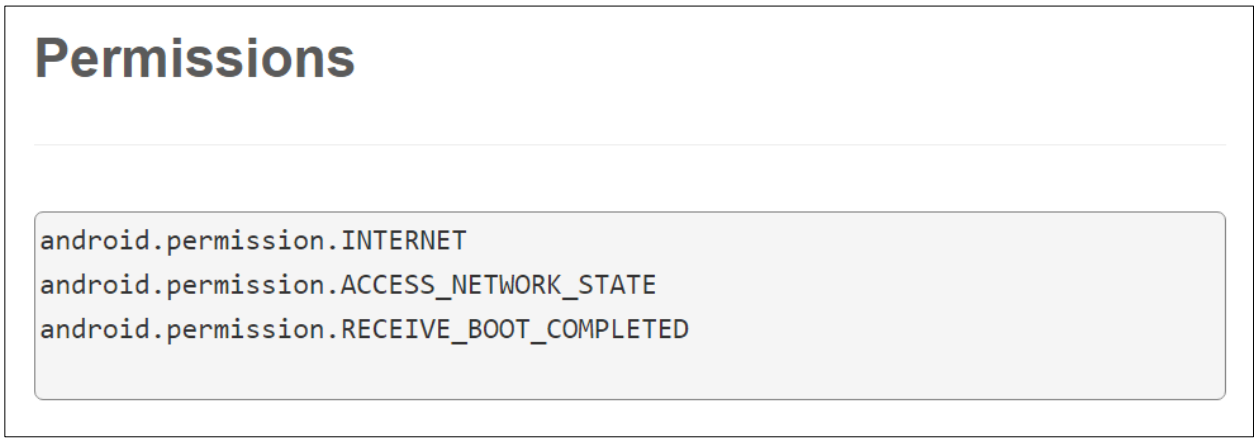


Figure 31. Permissions Extracted from A5 tool

The above results shows that INTERNET and ACCESS_NETWOK_STATE are the necessary permissions required for the functioning of HoroScope app. It also explores another hidden permission RECEIVE_BOOT_COMPLETED that was not listed at the time of installation.

5. **APK decompilation to extract Source code and analyze them in Android Studio**

The distribution of Android apps are in the form of .apk files which are compressed and packaged. In simple terms they are similar to .jar or .zip files [88]. Their source code can be easily obtained with the freely available tools.

We extracted the APK file of the application from dex2jar tool and displayed the extracted source code in the “Android Studio” to extract the source code for analysis purpose.

We will only display here the actual permissions of the code at the developer end to analyze the permissions demanded by the advertisement networks as well. Figure 32 shows the extracted code.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-feature android:name="android.hardware.wifi" android:required="false"/>
<uses-permission android:name="android.permission.READ_PHONE_DATA"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="com.android.vending.BILLING"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-feature android:name="android.hardware.location" android:required="false"/>
<uses-feature android:name="android.hardware.location.network" android:required="false"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
<permission android:name="fr.telemaque.horoscope.permission.C2D_MESSAGE"
android:protectionLevel="signature"/>
<uses-permission android:name="fr.telemaque.horoscope.permission.C2D_MESSAGE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS"/>
<uses-permission
android:name="com.android.browser.permission.WRITE_HISTORY_BOOKMARKS"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
</manifest>
```

Figure 32. Permissions Declaration in Manifest File

The above results shows that there is a prominent difference between the permissions displayed at the time of installation and permissions extracted from the source code.

Summary of Results:

There are three main sources of private data collection such as default Android hardware and software, its applications and data provided to the applications via 3rd-party applications. The first two of these three sources are supposed to be protected by Android API, and its corresponding Android permissions. We have explored those permissions to know their behavior on a practical ground. We have determined, specifically how horoscope collects user's sensitive data. It is found that it has some hidden permissions that are not displayed at the time of app installation which can cause serious damage to Android user's privacy.

5.3.2 Existing Solution Analysis (Ad-blockers)

Our task is to analyze existing ad-blockers permissions and their functionality. In our experiment we have chosen "AD Blocker" for the analysis of its behavior towards protecting user's privacy. "AD Blocker" app is available on the Google Playstore. Figure: 33 shows the screenshot of the permissions demanded at the time of installation. Some of the Ad-blockers demand more permissions as compared to the permissions demanded from the applications. We extracted the APK of the AD Blocker and reversed engineered its code in order to fully explored it.

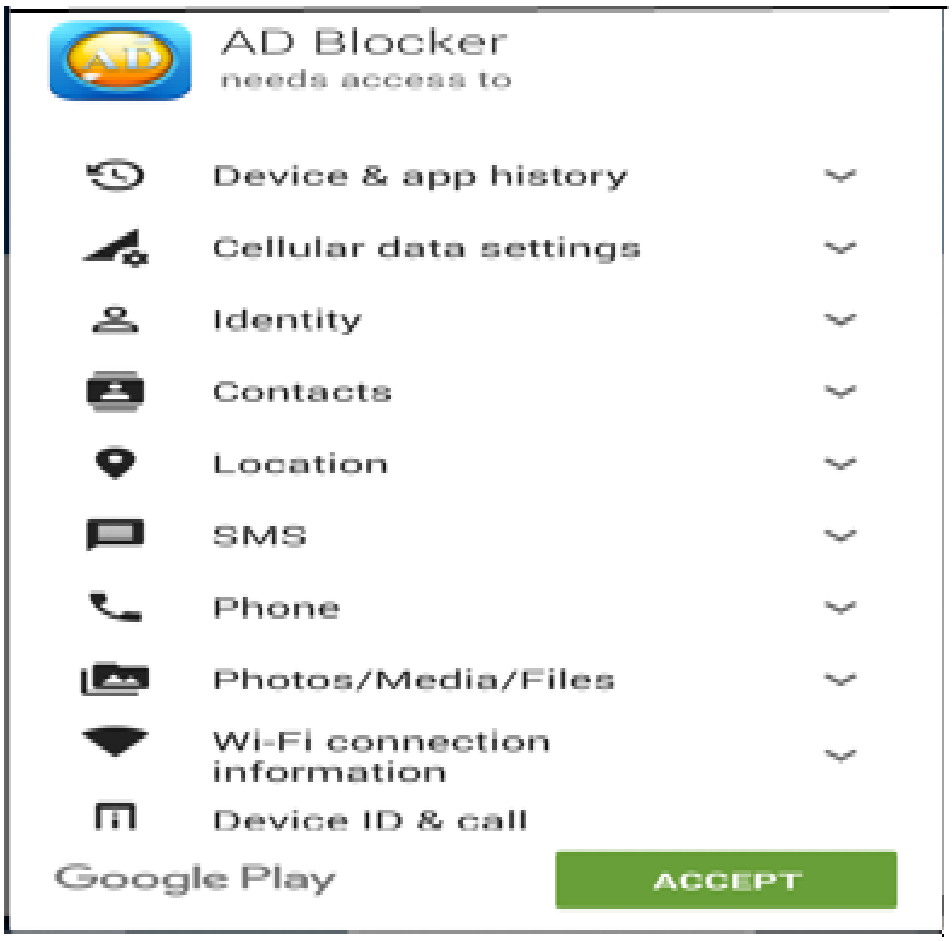


Figure 33. List of Ad-Blocker Permissions

It can be seen from the list that AD Blocker demands the permissions starting from Deice & App history to the Device ID & call. It has the ability to access SMS, Phone, Location, Contacts and phone identity. Traditional ad-blocker software relies upon hand-crafted filter expressions that can cause more overheads [89].

It is also proved that Ad Blockers can also be attacked by phishing attacks. On the other hand Ad Blockers can act maliciously. They can insert script into the browser and can fully control the device [90]. Ad Blockers may have the ability to take the data of the users and in worst cases the flaws in its software may also breach user's security [91]. Ad Blockers can demand extra permissions as compared to the permissions demanded from applications. Some of the permissions are hidden and users don't understand the meaning of all the permissions listed at the time of Ad Blocker installation. For the extraction of source code of AD Blocker we used Dex2Jar and Apktool. After extracting the APK of this particular ad-blocker and reverse

engineer its code we got some of the extra permissions which are not mentioned at the time of installation. Figure 34 shows the permissions of the existing Ad-Blockers.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RESTART_PACKAGES"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.WRITE_APN_SETTINGS"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.WRITE_SMS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.MODIFY_PHONE_STATE"/>
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.GET_TASKS"/>
<uses-permission android:name="android.permission.REORDER_TASKS"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.USE_CREDENTIALS"/>
<uses-permission android:name="android.permission.MANAGE_ACCOUNTS"/>
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES"/>
<uses-permission android:name="android.permission.DISABLE_KEYGUARD"/>
<uses-permission android:name="com.android.vending.CHECK_LICENSE"/>
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
```

```
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.CHANGE_CONFIGURATION"/>
<uses-permission android:name="android.permission.UPDATE_DEVICE_STATS"/>
<uses-permission android:name="android.permission.RUN_INSTRUMENTATION"/>
<uses-permission android:name="android.permission.WRITE_SECURE_SETTINGS"/>
<uses-permission android:name="android.permission.SIGNAL_PERSISTENT_PROCESSES"/>
<uses-permission android:name="android.permission.DEVICE_POWER"/>
<uses-permission android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"/>
<uses-permission android:name="com.motorola.dlauncher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.motorola.dlauncher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.motorola.mmsp.motoswitch.permission.READ_SETTINGS"/>
<uses-permission android:name="com.motorola.mmsp.motoswitch.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.htc.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.htc.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.aspire.mm.permission.READ_SETTINGS"/>
<uses-permission android:name="com.aspire.mm.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.qihoo360.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.qihoo360.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.ty.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.ty.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.sonyericsson.homescreen.permission.READ_SETTINGS"/>
<uses-permission android:name="com.sonyericsson.homescreen.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.oppo.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.oppo.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.mediatek.launcherplus.permission.READ_SETTINGS"/>
<uses-permission android:name="com.mediatek.launcherplus.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.huawei.launcher2.permission.READ_SETTINGS"/>
<uses-permission android:name="com.huawei.launcher2.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.huawei.launcher3.permission.READ_SETTINGS"/>
<uses-permission android:name="com.huawei.launcher3.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.baiqi.weather.permission.READ_SETTINGS"/>
<uses-permission android:name="com.baiqi.weather.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.fede.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.fede.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="mobi.SyndicateApps.ICS.launcher.permission.READ_SETTINGS"/>
```

```
<uses-permission android:name="mobi.SyndicateApps.ICS.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.motorola.dock.DesktopDock.permission.READ_SETTINGS"/>
<uses-permission android:name="com.motorola.dock.DesktopDock.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.lge.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.lge.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.thunderst.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.thunderst.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="org.adw.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="org.adw.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.fede.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.fede.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="net.qihoo.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="net.qihoo.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.lge.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.lge.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.sec.android.app.twlauncher.settings.READ_SETTINGS"/>
<uses-permission android:name="com.sec.android.app.twlauncher.settings.WRITE_SETTINGS"/>
<uses-permission android:name="org.adwfreak.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="org.adwfreak.launcher.permission.WRITE_SETTINGS"/>
<uses-permission
android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURV
EY"/>
<uses-permission android:name="com.android.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.UNINSTALL_SHORTCUT"/>
<uses-permission android:name="com.android.launcher.permission.WRITE_SETTINGS"/>
```

Figure 34. Permissions Extracted from the Source Code

We can see that ad-blockers already have so many permissions which can disturb user's privacy itself. Some of the permissions are necessary to block the advertisements but others provide over privilege to ad-blockers. While installing the ad-blocker an ordinary user is unable to understand the meaning of each permission it is demanded for. It is also freely available on the Google play store and we cannot trust such applications who can damage the user more than the actual application does.

We analyzed the existing applications permissions and the problems caused by them with the permissions hazards of the existing solutions such as Ad-blockers. We conclude that none of the existing solution is reliable and secure who guarantee the protection of user's privacy and provide ultimate solution to the existing problem.

Next chapter includes the research framework with hypothesis formulation from the information gathered in the first five chapters. It will be proceeded by the survey conduction from different respondents and hypothesis will be chosen for the proposed model based on analysis results.

6. DATA COLLECTION AND QUANTITATIVE ANALYSIS

6.1 Theoretical Framework

Based on the literature review and the research conducted in the previous chapters, the following theoretical framework has been developed which help to study the impact of embedded adware on Android user's privacy in the form of advertisement attacks.

This chapter will contribute towards formulation of theoretical framework, hypothesis development and will analyze the results obtained from collected data.

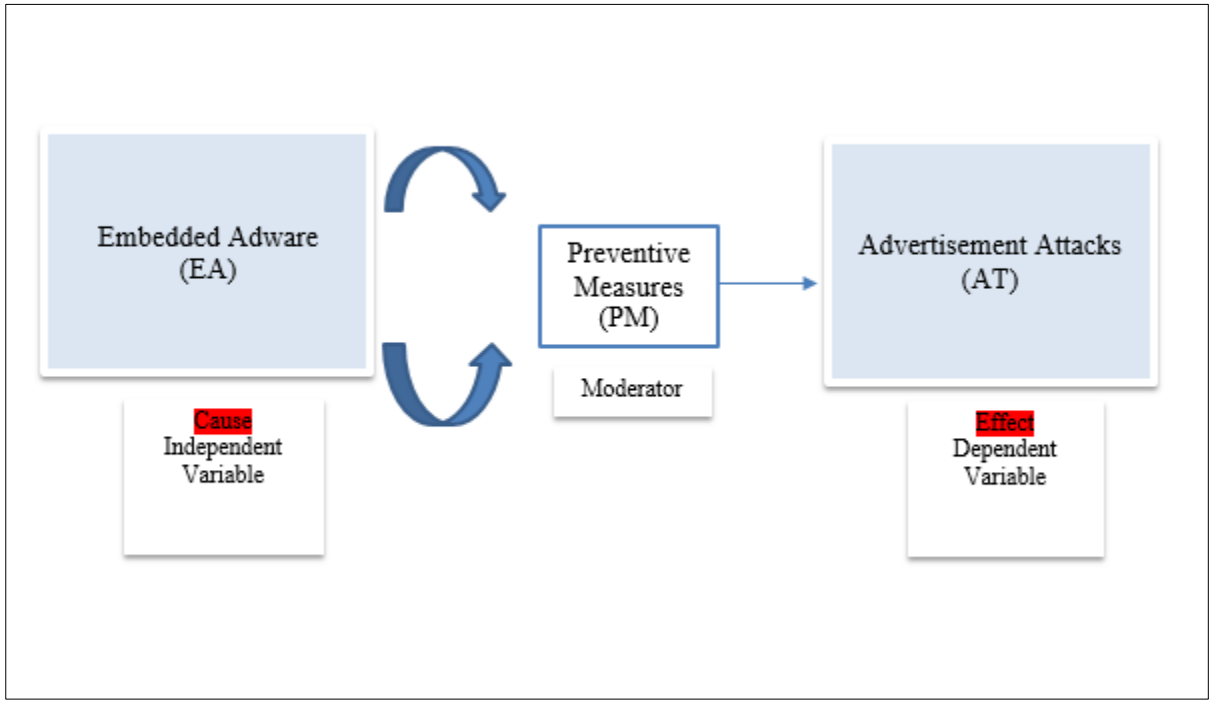


Figure 35. Theoretical Framework

In this model (Figure.35) the independent variable is embedded adware that is considered as "cause". Whereas dependent variable is advertisement attacks that are considered as the effect of embedded adware.

The mediator of the theoretical framework is "preventive measure" because it can lessen the impact of effect on Android user's privacy. Mediator is controlling the significance of the model.

6.2 Hypothesis Formulation

The research has been started with two possible hypothesis.

Embedded Adware (Dependent Variable)

H1: Embedded Adware can cause in-application advertisement attacks.

H_a: Embedded Adware cannot cause in-application advertisement attacks.

Preventive Measures (Moderator)

H2: Preventive Measures can help in reduction of advertisement attacks caused by embedded adware.

H_b: Preventive measures cannot help in reduction of advertisement attacks caused by embedded adware.

6.3 Data Collection

Correct data gathering/ collection is an important task to maintain integrity of research.

In this research thesis quantitative analysis techniques have been used. Quantitative analysis is a study of data that can be measured and provides the framework of the study [92]. Collection of data for a quantitative analysis is important because of its validity and reliability. A questionnaire has been designed for the analysis of data in our quantitative analysis approach. It helps to analyze the behavior and approaches of Android user's towards advertisement attacks.

Different samples have been collected from 200 respondents working at different levels in different domains.

6.3.1 Unit of Analysis

Since perception of protection from Advertisement attacks and its impact on user's privacy, data is collected from students, unemployed and employed people at different hierarchical levels in several organizations.

6.3.2 Time Horizon

Research has been conducted at a single point of time. In short, we can call it a cross sectional study.

6.3.3 Study Setting

The study has been carried out in a natural environment without any arrangement of artificial or controlled environment. Along with some of the limitations, the research is based on unbiased responses of the respondents.

6.3.4 Tools/ Software

SPSS, Interaction Software and Google docs is used for the Quantitative analysis and validation of data.

6.4 Data Collection Instrument & Questionnaire Design

For collecting the primary data through survey, questionnaire is selected as an instrument of the present study.

On the basis of literature review and development of theoretical/ conceptual framework, questionnaire has been designed to show the overall trend of the embedded adware and usage of Android phone.

6.4.1 Validity & Reliability of an Instrument

Reliability test is performed to check the validity and reliability of the questionnaire.

The reliability of this scale is reported as 0.984, which makes the scale highly predictable and reliable in the current research. The Cronbach Alpha value for all the variables is shown in the below table.

Reliability Statistics

Cronbach's Alpha	N of Items
.984	3

Tested items chosen for reliability test are embedded adware (EA), Preventive Measures (PM) and Advertisement Attacks (AT).

6.5 Data Analysis from Questionnaire Responses (Quantitative Analysis)

In our theoretical framework there are three variables involved i.e. Independent variable "Embedded Adware", Dependent variable "Advertisement Attacks" and moderator "Preventive measures". For convenience in the rest of the chapter, we will use the terms **EA**, **AT** and **PM** for embedded adware, Advertisement attacks and preventive measures respectively.

6.5.1 Demographic Analysis

Gender of Respondents

Popularity of Android based smartphone is justified because of its useful and new features. Below table and chart shows that 200 respondents participated in the survey. Where 99 of them are found to be females and 101 of them are males. This shows the almost equal frequency of males and females (i.e. 50.5% and 49.5% respectively) in the sample selected for data collection.

Gender

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Male	101	50.5	50.5	50.5
	Female	99	49.5	49.5	100.0
	Total	200	100.0	100.0	

In addition to gender, respondents were asked about their age to know about the trends of using Android phone in different age groups. The data shows that among 200 Mobile users there were 10% who lie in the category of age group (15-20) years, 45% of them are in (21-25) years category. Similarly, there were 33% and 24% users who belong to the category (26-30) and (31-Above) respectively. It also proves the fact that most of the Android users are youngsters and are mostly between the ages of 20-30. They are more interested in downloading and using new applications introduced in the Android market as compared to the senior citizens. So eventually they become more vulnerable to In-application advertisement attacks.

Age

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	15-20	20	10.0	10.0	10.0
	21-25	90	45.0	45.0	55.0
	26-30	66	33.0	33.0	88.0
	31-Above	24	12.0	12.0	100.0
	Total	200	100.0	100.0	

Respondents were asked about the Operating System of their phone. Figure 36 shows that most of the respondents' phone is based on Android. The demographic analysis in the following chart shows that among 200 respondents 83% have Android phone while rest of them are using Windows, iOS and only 5% don't have smartphone. This shows the popularity of Android phones among people due to its ease of availability and enhanced features.

MobileOS

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Android	166	83.0	83.0	83.0
	Windows	5	2.5	2.5	85.5
	iOS	19	9.5	9.5	95.0
	Others	5	2.5	2.5	97.5
	Not Applicable	5	2.5	2.5	100.0
	Total	200	100.0	100.0	

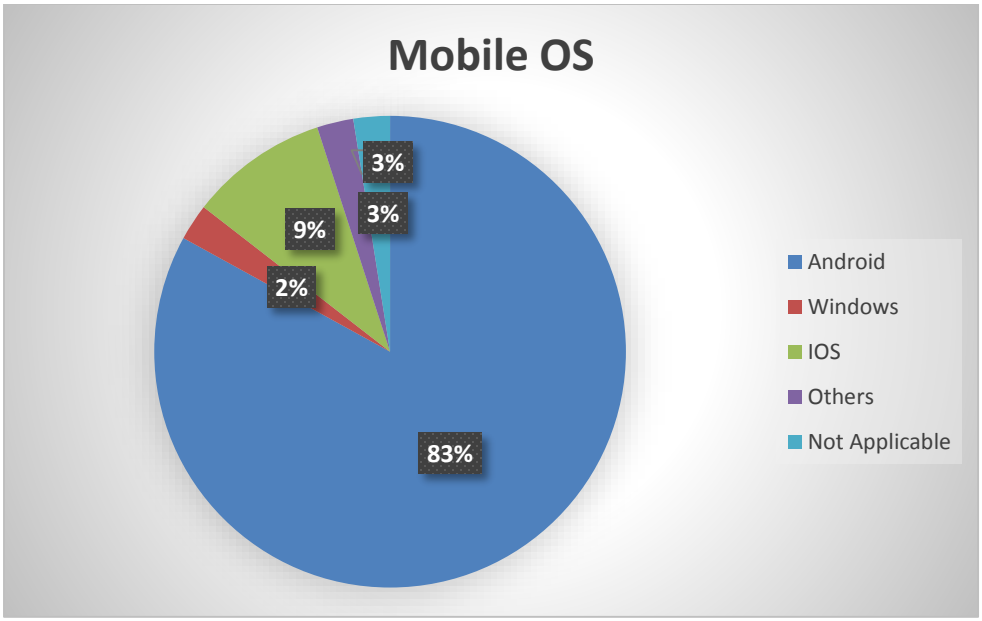


Figure 36. Popularity of Android OS

This survey has been carried out from different people belonging to different field/ domain. Most of them are found to be students.

The reason behind is that people belonging to the age group 20-30 are mostly students or employed. We have a big percentage of people in the category between 20-30 age groups. Hence those are mostly students or employed as shown in the demographic chart below.

Profession

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	Employed	86	43.0	43.0	43.0
	Student	102	51.0	51.0	94.0
	Others	12	6.0	6.0	100.0
	Total	200	100.0	100.0	

Privacy of employed personnel and students both are important. Students use games or other ads embedded applications which can compromise their security. Those Android users who are employed are more vulnerable of attacks they can become a source of leakage of sensitive information or trade secrets of their companies or workplace by using Android applications.

6.5.2 Embedded Adware (H1 & H_a)

Firstly we will apply tests to approve or disapprove hypothesis H1 and H_a. For this purpose we have chosen AT as a dependent variable and EA as an independent variable.

Dependent Variable: AT (Advertisement Attacks**)**
Independent Variable: EA (Embedded Adware**)**

H1: Embedded Adware can cause in-application advertisement attacks.

H_a: Embedded Adware cannot cause in-application advertisement attacks

Correlation Analysis

To determine the strength of the variable involved in the framework, correlation is applied. Pearson correlation method is used with the variables EA and AT. Significance of both variables with respect to each other is found to be .000 that shows a higher degree of significance with 0.965 correlation value. Following table shows the correlation among the selected variables.

Correlations

		EA	AT
EA	Pearson Correlation	1	.965**
	Sig. (2-tailed)		.000
	N	200	200
AT	Pearson Correlation	.965**	1
	Sig. (2-tailed)	.000	
	N	200	200

**. Correlation is significant at the 0.01 level (2-tailed).

It is found that correlation among embedded adware and Advertisement attacks are strong and positive.

Linear Regression Analysis

It is a model based technique. We have applied linear regression with residual model "Durbin-Watson". Zero-order correlation among existing variables can be found by linear regression. As suggested by Barren and Kenney (1986), it is important to establish zero order correlation among variables and independent variables before doing further analysis.

Model summary of linear regression is shown below.

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics					Durbin-Watson
					R Square Change	F Change	df1	df2	Sig. F Change	
1	.965[a]	.930	.930	.3062	.930	2645.489	1	198	.000	.113

a. Predictors: (Constant), EA

b. Dependent Variable: AT

The above table provides the R and R^2 value. Correlation among the variables is provided by the value of R whereas R^2 value indicates how much total variation in the dependent variable. Both values are given in the above table in a separate column. The model summary tells that R value is 0.965 which indicates the higher degree of correlation between embedded adware (EA) and advertisement attacks (AT). On the other hand R^2 value is 0.930, analysis shows that with independent variable (EA) there are 93% chances of Advertisement attacks. Hence, prove that our H1 is proved right as compared to the H_a. It will be further proved by checking the significance of the model.

Now, the significance of the model is checked by ANOVA table, which tells how well the regression equation fits the data (i.e. predicts the dependent variable) and is shown below:

ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	248.072	1	248.072	2645.489	.000[a]
	Residual	18.567	198	.094		
	Total	266.639	199			

a. Predictors: (Constant), EA

b. Dependent Variable: AT

The above table indicates that the regression model predicts the dependent variable significantly well. In the Sig. column of the ANOVA table it is found that sig value is 0.000, which indicates the statistical significance of the regression model was run. We have found that $p < 0.000$, which is less than 0.05 which proves that it is good fit for data and model is statistically significant that predicts the outcome variable.

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Correlations			Collinearity Statistics	
		B	Std. Error	Beta			Zero-order	Partial	Part	Tolerance	VIF
1	(Constant)	-.523	.077		-6.757	.000					
	EA	1.257	.024	.965	51.434	.000	.965	.965	.965	1.000	1.000

a. Dependent Variable: AT

The above coefficient table provides the necessary information to predict the possibility of advertisement attacks (AT) from embedded adware (ED). It also helps to determine whether embedded adware (EA) contributes significantly to the model (by finding the value from the "**Sig**." column). It is significant as the value of sig. is less than 0.05 (p=0.000).

The above table reports the coefficient of constants and independent variable. The negative value of coefficient indicates that when there will not be presence of embedded adware (EA) there will be less chances of advertisement attacks (AT).

On the other hand coefficient of Embedded Adware (EA) is 0.965 with a positive sign which tells that with one unit change in EA, there will be 1.257 unit change in AT.

For accepting the hypothesis p (.Sig) must be less than 0.05, t and Beta values should be acceptable as proved in the above table.

Hence, it is proved that hypothesis H1 is supported and H_a is not supported.

6.5.3 Moderator (H2 & H_b)

Now we will apply tests to approve or disapprove hypothesis H2 and H_b for checking the impact of moderator in the framework. For this purpose we have chosen AT as a dependent variable and EA as an independent variable and PM as a moderator.

H2: Preventive Measures can help in reduction of advertisement attacks caused by embedded adware.

H_b: Preventive measures cannot help in reduction of advertisement attacks caused by embedded adware.

Dependent Variable: AT (Advertisement Attacks**)**
Independent Variable: EA (Embedded Adware**)**
Moderator: PM (Preventive Measures**)**

We will apply the same tests to prove the above mentioned hypothesis. Here, we need to check if the moderator "Preventive Measures" (PM) are helpful in reducing the advertisement attacks (AT) caused by embedded adware (EA) or not.

Regression Analysis

The model summary is given below.

Model Summary[b]

Model	R	R Square	Adjusted R Square	Std. Error of the Estimate	Change Statistics					Durbin-Watson
					R Square Change	F Change	df1	df2	Sig. F Change	
1	.979[a]	.958	.958	.2377	.958	1508.377	3	196	.000	.152

a. Predictors: (Constant), Moderator, Zscore(EA), Zscore(PM)

b. Dependent Variable: AT

To find the moderator, we have taken the mean values of EA and PM named as ZScore (EA) and ZScore (PM) respectively and run the regression test.

It is given in the model summary that value of R is 0.979 which shows the higher degree of correlation amongst the selected variables EA, PM and AT. Value of R^2 is 0.958 which shows that by applying the moderator there are 95% chances to minimize the effect of advertisement attacks caused by embedded adware. (.Sig =0.000) which also indicates the significance of the model.

The model significance is given below in the ANOVA table.

ANOVA[b]

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	255.570	3	85.190	1508.377	.000[a]
	Residual	11.070	196	.056		
	Total	266.639	199			

a. Predictors: (Constant), Moderator, Zscore(EA), Zscore(PM)

b. Dependent Variable: AT

Coefficients[a]

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Collinearity Statistics	
		B	Std. Error	Beta			Tolerance	VIF
1	(Constant)	3.216	.022		147.431	.000		
	Zscore(EA)	.144	.091	.125	1.585	.115	.034	29.206
	Zscore(PM)	1.024	.092	.885	11.081	.000	.033	30.114
	Moderator	.084	.014	.094	5.928	.000	.842	1.187

a. Dependent Variable: AT

In the coefficient table analysis it is indicated that t value is greater than 2 and Moderator significant value is less than 0.000 which proves the hypothesis H2 as successful.

Correlation Analysis

Correlation between the involved variables in a framework is found to be significant, where each variable is significant with respect to each other (.Sig= 0.000). Results are shown below in the correlation table.

Correlations

		AT	Zscore(EA)	Zscore(PM)	Moderator
Pearson Correlation	AT	1.000	.965	.974	-.256
	Zscore(EA)	.965	1.000	.982	-.311
	Zscore(PM)	.974	.982	1.000	-.352
	Moderator	-.256	-.311	-.352	1.000
Sig. (1-tailed)	AT	.	.000	.000	.000
	Zscore(EA)	.000	.	.000	.000
	Zscore(PM)	.000	.000	.	.000
	Moderator	.000	.000	.000	.
N	AT	200	200	200	200
	Zscore(EA)	200	200	200	200
	Zscore(PM)	200	200	200	200
	Moderator	200	200	200	200

.Sig = 0.000

Limitations of hypothesis H2:

At certain points H2 fails and H_b is successful by indicating the values in the above results. The reason behind is that H2 fails when preventive measures are not applied properly or there is a lack of users' awareness in protecting their security by in-application advertisement attacks from embedded adware as well as limitations of existing solutions. To understand the behavior of moderator for not giving the 100% results of H2 we have categorized the moderator in three

parts by giving low, medium or high effects for controlling Advertisement attacks. This categorization is explained in the following section.

6.6 Moderator

It can changes the relationship between two related variables. In our case moderator is preventive measures labelled as PM in the analysis. PM can lessen the happing of Advertisement attacks (AT) caused by Embedded Adware (ED). So moderator can moderate the relationship between independent and dependent variables.

6.6.1 Moderator Behaviour Analysis

Here, we have divided our moderator Preventive measures (PM) into three groups to check the impact of embedded adware on users' privacy by causing advertisement attacks. We checked whether Preventive measures have low, medium or High effect in preventing advertisement attacks. Preventive measure low, meduim and high is labelled as PM_Low, PM_Mod and PM_High respectively.

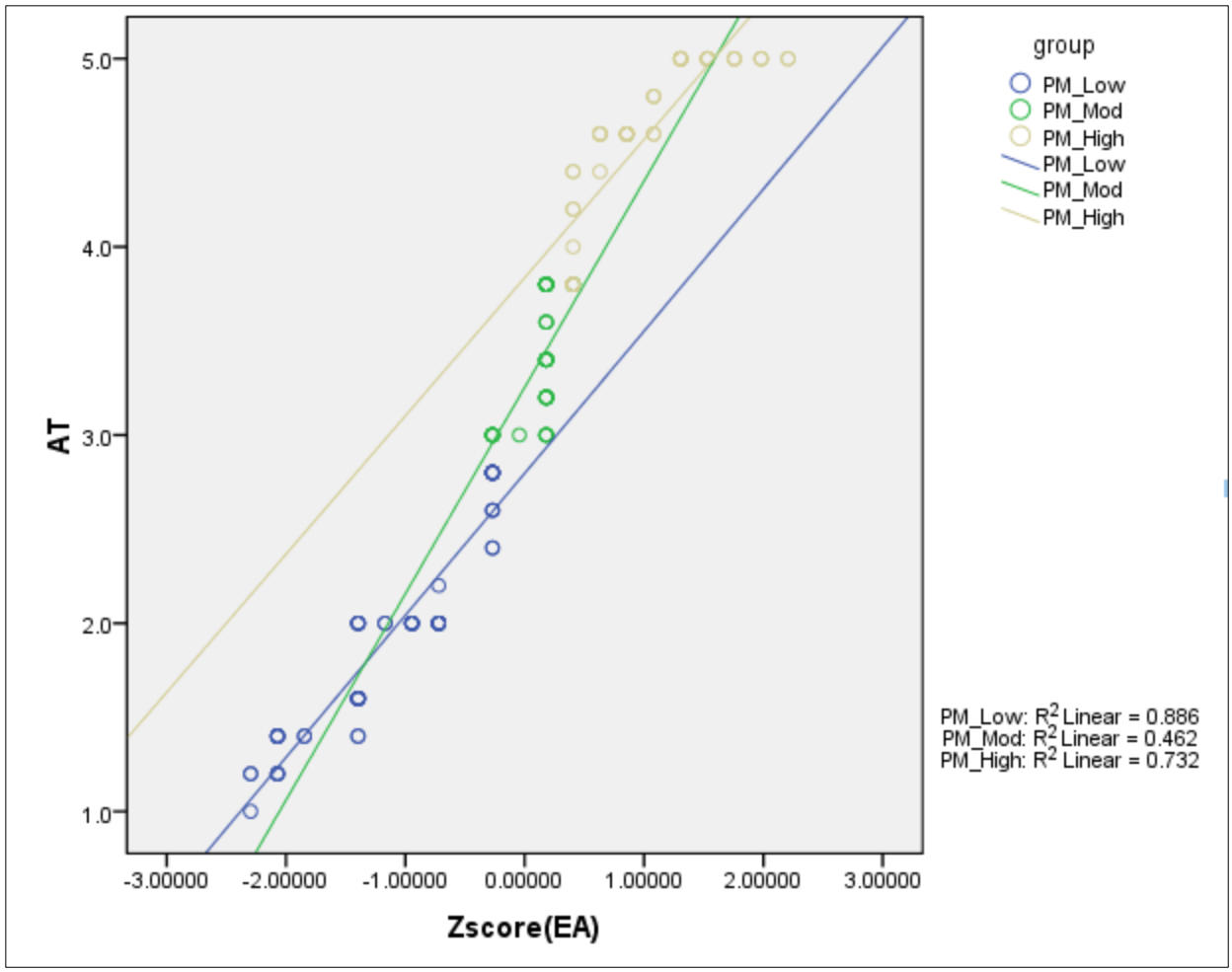


Figure 37. Moderator effects

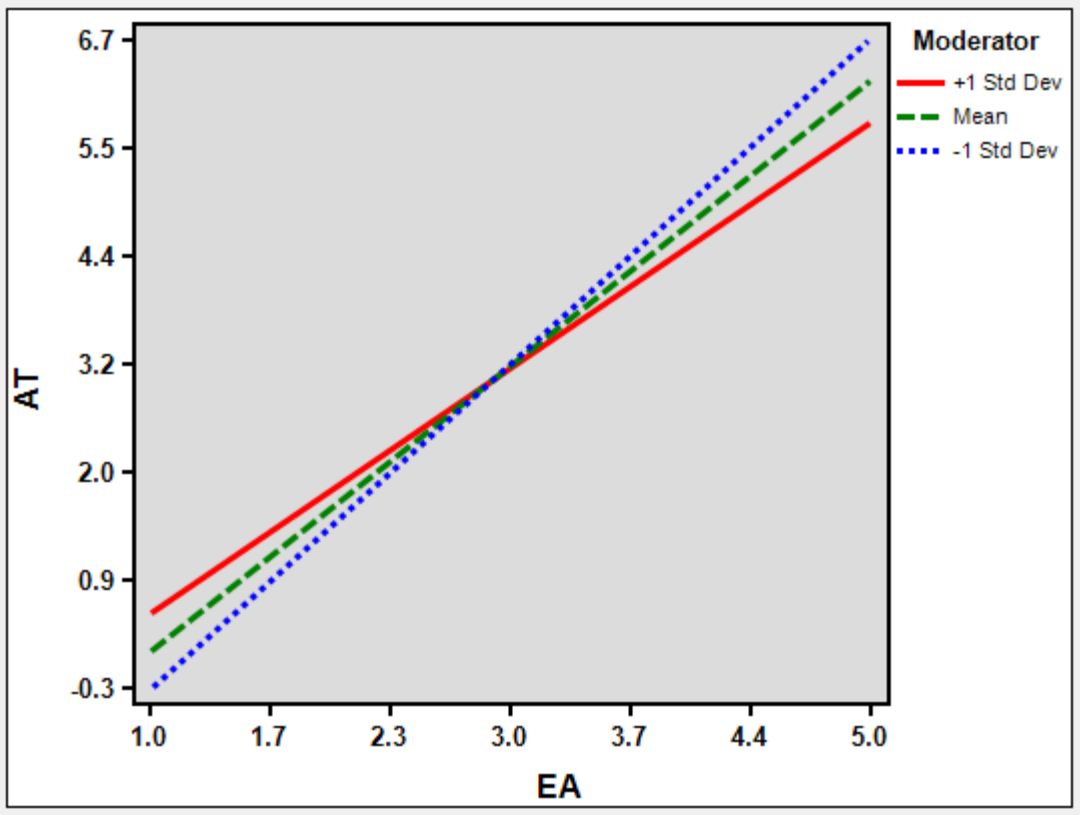
In the above figure.37 it is shown that PM_Low has the really strong regression effect. It can be seen that R^2 Linear is 0.886. So the correlation between Advertisements attacks and embedded adware is 0.93 for the group of Low preventive measures. It means if the preventive measures will not be applied, there will be greater chances of attacks due to the presence of embedded adware. Defined and suggested solutions in the questionnaire give the higher results by protecting users' security via exploring hidden permissions, advertisement and applications permissions separation, Appchoices to manage relevant advertising preferences , hiding exact location from 3rd parties and introducing more effective ad-blockers that demand less permissions and function properly. By applying such preventive measures, ratio of advertisement attacks can be lessen and eventually it gives the highest correlation as shown in the above figure.

PM_High has R^2 linear value which 0.732 and less correlation between embedded adware and advertisement attacks which is 0.855 as compared to PM_Low. It shows the limitations of existing solutions such as existing ad-blockers can damage user's privacy more aggressively as compared to the application itself. Some of the limitations are at the end of separating the permissions by previously defined or existing methods. The existing solutions has some certain limitations that is why the resultant security is still questionable. But still there is a great possibility to control in-application advertisement attacks due the presence of embedded adware if it is used properly in a hierarchal way as given in phase II of our proposed model.

PM_Mod has the medium regression effect. Its R^2 Linear is 0.493, hence correlation between Advertisements attacks and embedded adware drops to 0.68. The reason behind is the lack of user's awareness in understanding and applying solutions to protect against advertisement attacks. When respondents were asked about their preferences of convenience over privacy about 56% of the total respondents give preference the convenience over privacy and 18% were neutral. It means there is need to increase users' awareness to protect them against advertisement attacks due to embedded adware.

6.6.2 Relationship of Moderator

In this section, we will analyze the relationship of moderator Preventive measures (PM) with Dependent variable Advertisement Attacks (AT) and Independent variable Embedded Adware (ED). The analysis is carried out by "Interaction" Software and its results are shown below.



Green line in above result shows that it is at mean level.

Blue line is at -1 standard deviation below. It shows that when PM is low then there is more impact of embedded advertisements and advertisement attacks. There is a linear relationship between EA and AT or it increases in a linear way.

Red line shows that it is less than the mean value. It is +1 standard deviation where PM is high. If PM is high then there will be less impact of embedded adware on user's privacy by minimizing advertisement attacks.

6.7 Linear Behavior

The following graph in Figure.38 shows the linear behavior between embedded adware and advertisement attacks. It also shows that with the increase in embedded libraries in an android application, there is a greater chances of advertisement attacks.

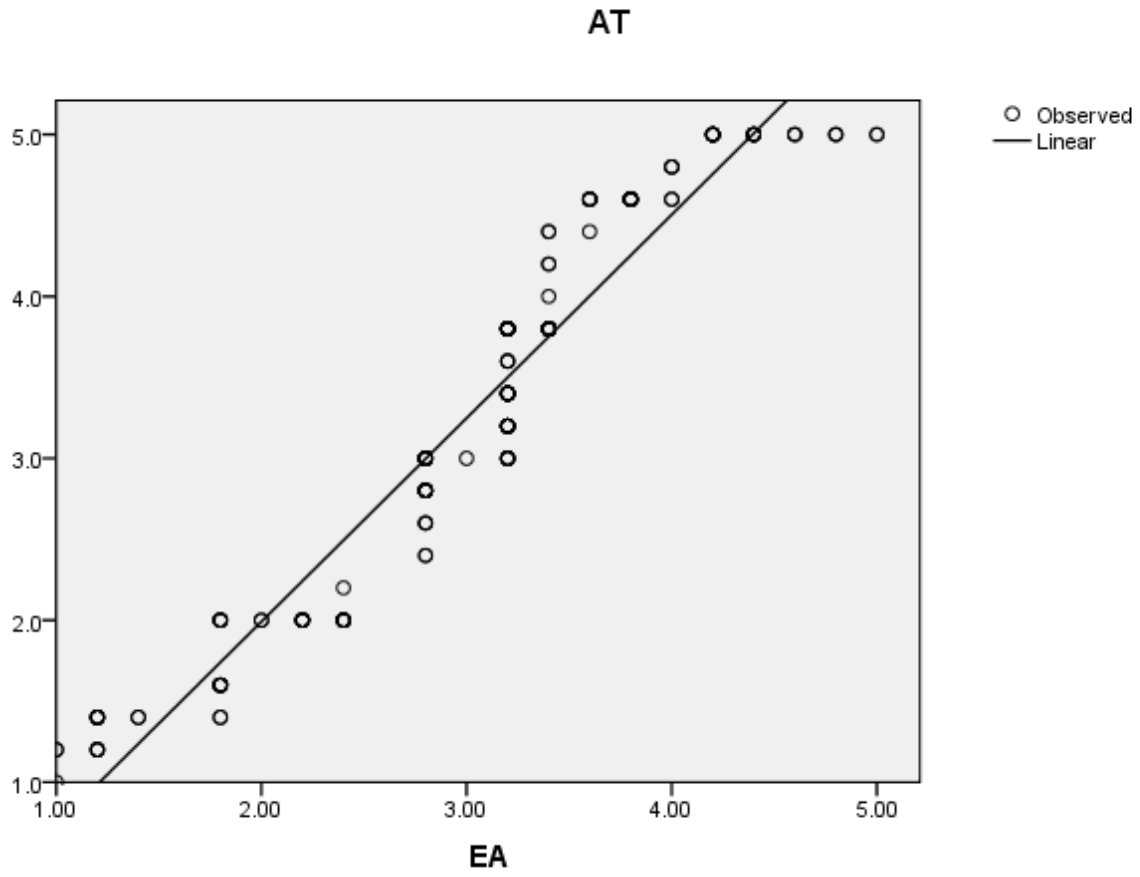


Figure 38. Linear Relationship between EA & AT

6.7.1 Residuals

The below figure. 39 shows the residuals of the above results. It depicts that residuals are not random. This also shows the other factors involved that can cause advertisement attacks.

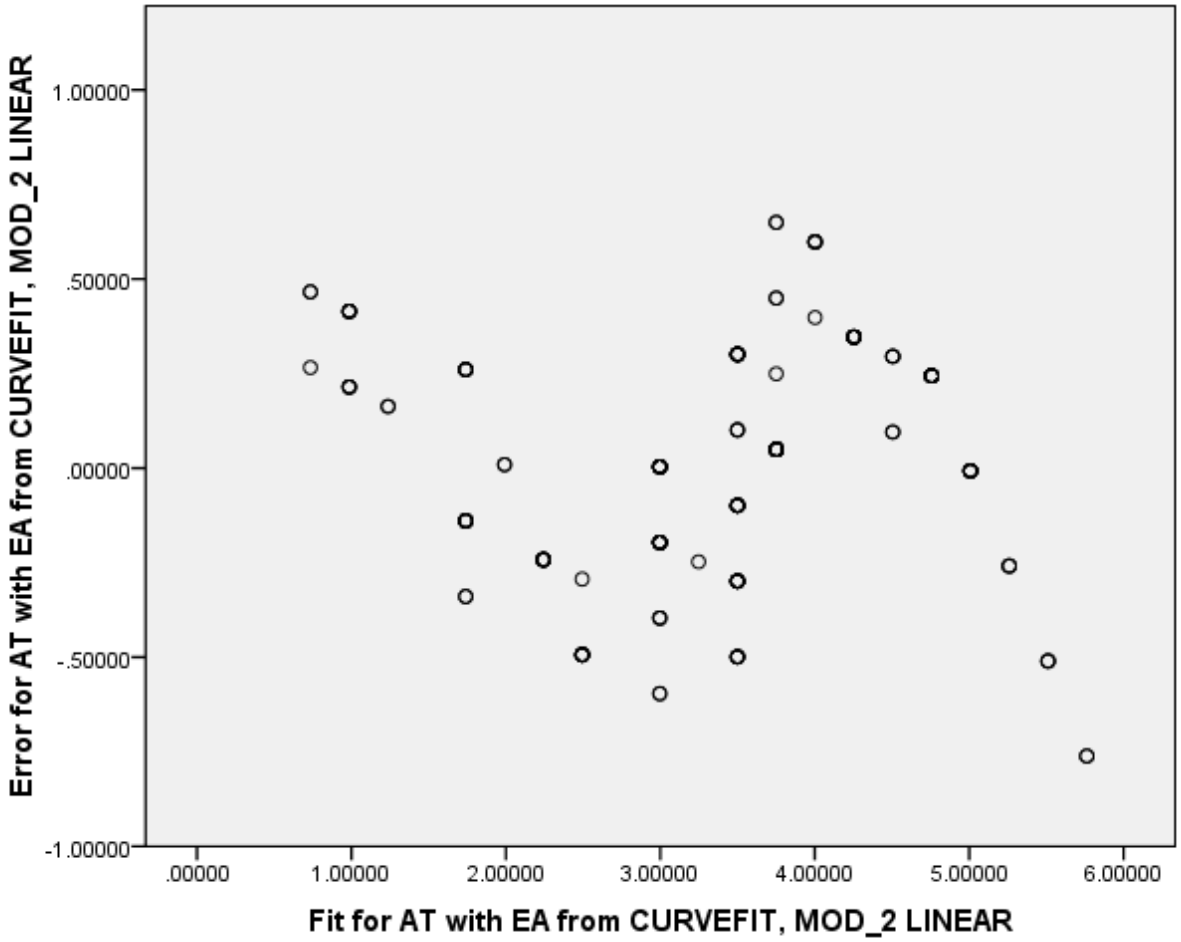


Figure 39. Residuals

6.8 Summary of Results

Table 3. Hypothesis Approval/ Disapproval

Hypothesis	**Results**
H1: Embedded Adware can cause in-application advertisement attacks.	**Supported**
H_a: Embedded Adware cannot cause in-application advertisement attacks	**Not Supported**
H2: Preventive Measures can help in reduction of advertisement attacks caused by embedded adware.	**Supported** (Conditions Applied)
H_b: Preventive measures cannot help in reduction of advertisement attacks caused by embedded adware.	**Not Supported**

6.9 Conclusion

The survey results conclude that trend of using Android based OS is increasing because of its unique features. The main source of its popularity are its applications. Some of them are freely available on the Android market where as some of them can be accessible only by paying some amount of money. This survey is conducted from number of people of certain age groups and belonging to different professions. The reason of knowing the category is to know the target group of the attackers. It has been found that students are more vulnerable to attack as compared to the professional people. It has also been found from the survey results that people also download apps from 3rd party playstores instead of google playstore. This is one of the reasons of advertisement attacks. Users intentionally or unintentionally click on the advertisements at the run time that also causes advertisement attacks. There is a security mechanism or framework needed that help users to protect themselves against such attacks.

Next chapter includes the privacy protection methodology to highlight the importance of privacy and the strategies to maintain the CIA (Confidentiality, Integrity and Availability) of the user. And later on we will present our proposed model which will help in solving current problem.

7 LOCATION PROTECTION METHODOLOGY

7.1 Privacy & its Importance

This chapter focuses on categorizes of privacy and its importance. Privacy has different roles in different domains. Privacy is categorized in certain domains such as bodily privacy, communication, and territorial and information privacy. Table 4 shows gives the brief introduction of privacy.

Table 4. Categorization of Privacy [82]

S. No	Privacy Category	Effects
1	Bodily Privacy	It involves the physical invasion protection
2	Communication Privacy	This category will hold the part of the mobile phone we used for communication such as email etc.
3	Territorial Privacy	Physical space like homes and workplaces are concerned to it.
4	Information Privacy	It completely involves the personal information of the user.

Information privacy includes every type of information of the user. Our research is purely based on the leakage of user's information due to advertisement attacks. Advertisement libraries are most interested in finding the location of the smartphone users as compared to the other one.

We have noticed in the previous chapters that "location" permission is acquired by all advertisement libraries. The reason behind is to send the targeted ads to the user. Almost all of

the ad-libraries demand location permission that is why our focus will be on Location privacy in this chapter.

In the previous chapters by analyzing the code of the various Android applications we determined that most of the applications require permission of location from the users. Our selected domain of "location" is important because it can become a serious threat for the user.

We call location privacy as a distinct type of privacy. Location information control is necessary to understand in order to protect users because technological advances make it easy to provide approximate and even precise location to the attacker. Location privacy and awareness is significant itself because it can control future threats. It is also the fact that location related information is important because no one wants public to know about his current location [22]. Many of the Android applications request the location information on the behalf of its embedded libraries. So this act can become annoying for the user because it can reveal the precise location either in the form of his current home address or the places he visits daily. In short, the generic profile of a person can be exposed by allowing such permissions in your phone.

7.2 Location Based Threat Model

Location is accessed by accessing "Location permission" from the user. Location based threat model is given in figure 40.

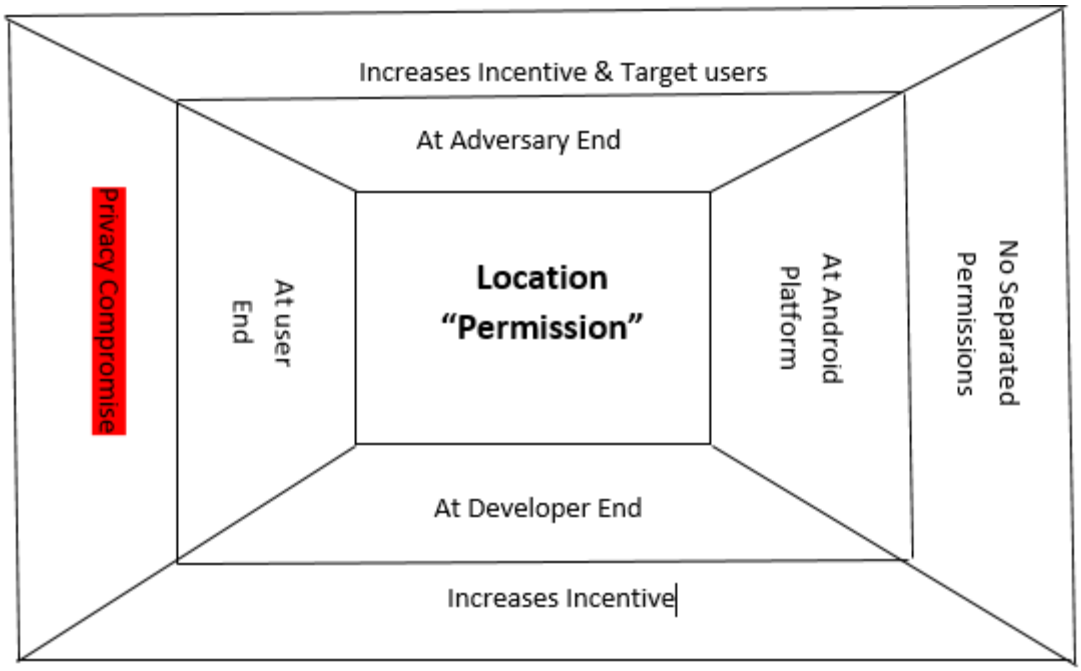


Figure 40. Location Based Threat Model

At Android Platform End: Both the advertisement libraries and developers app permissions are not separated.

At Developer End: Developers embed those permissions in their application code.

At Adversary End: It can target users for personal or marketing purposes.

At User's End: Users' privacy can be effected by their presence.

There are three major effects that can be occurred by not following the preventive measures of location. It include marketing & targeting users, personal and physical attacks and unpleasant implication [93, 94, 95].

1. **Marketing & Targeting Users**

Location can be used for the marketing purposes as well. Someone can use location in order to target the individual and send him the targeted advertisements. They can actually force him to

click on those advertisements and follow the steps mentioned there. Those steps can lead the user to download malicious applications or malwares.

2. **Personal or physical attacks**

On a very critical note, if someone tries to locate a person he can adopt this worst scenario i.e. to target users by taking help from advertisement libraries. It can lead to harm and stalk users. Business users can come in this category. Their peers or competitors can exploit their privacy by using single ad-embedded application.

3. **Unpleasant implication**

Location can be used to locate our residence, it can be helpful in crime scenes. It can also reveal the information about user's political affiliation, health state, medical facilities, personal preferences or meetings' location etc.

So privacy of location is a key issue in the emerging location-computing technologies. Figure. 41 shows the common threats caused by the location permission i.e. sending targeted ads to the users, stealing personal information and interests.

Figure 41. Location Based Threats

7.3 Methods of Protecting Location Privacy

Location privacy can be protected by implementing privacy policies and applying regulatory strategies. The other ways are Anonymity or choosing Obfuscation strategies. Table 5 shows them in a tabular form [82].

Table 5. Comparison of Information Protection Strategies

S. No	Strategies	Description
1	Regulatory	Fair Information practices
2	Privacy Policies	Proscribed certain uses of location information
3	Anonymity	Dissociation of information from actual identity
4	Obfuscation	Provide confusing information

7.3.1 Regulatory Strategies

In the regulatory strategies practices, users must be aware of the purposes of the regulation along with who is collecting their information. It covers transparency technique. Limitations and access to the private data has also been covered in the regulatory strategies. It also includes security, integrity and accountability of the collectors who try to access private information of the user.
Mostly regulatory strategies cover the personal information instead of location information.

7.3.2 Privacy Policies

Implementation of privacy policies are always good but privacy cannot be enforced to implement by only introducing privacy policies. The reason of its failure on the practical ground is that it relies on social and regulatory pressure.
So eventually privacy policies are vulnerable to the disclosure of sensitive information [92, 93].

7.3.3 Anonymity

Detachment of information from the actual identity of the user comes under the category of the Anonymity.

It is based on the trusted anonymity that retains the personal information of the user but do not provide it to anyone else. Authentication and Personalization are required in multiple applications but sometimes anonymity is a barrier to fulfill this approach. So we consider it as another flaw of using the anonymity for protection against location information.

7.3.4 Obfuscation

Quality of location information is degraded in order to protect user's privacy, this definition comes under the category of obfuscation. Obscurity means making the message confusing or ambiguous. It is planted such a way to make the actual meaning harder to understand to the public.

Till now, in our research privacy protection methodologies have been covered. We will use Obfuscation to protect Location information of the user in our proposed model explained in the next chapter.

8 PROPOSED MODEL & ITS SIGNIFICANCE

8.1 Significance of the Model

Researchers have find many ways to protect Android user security. Attack scenario is changing day by day and attackers find multiple ways to exploit the vulnerabilities of the system.

We introduced here the prosed model by keeping in mind three entities in a loop. These entities are Android application developers, Android platform and Android users. Our main variable of the scope are advertisers who embed the ads in the Android application code.

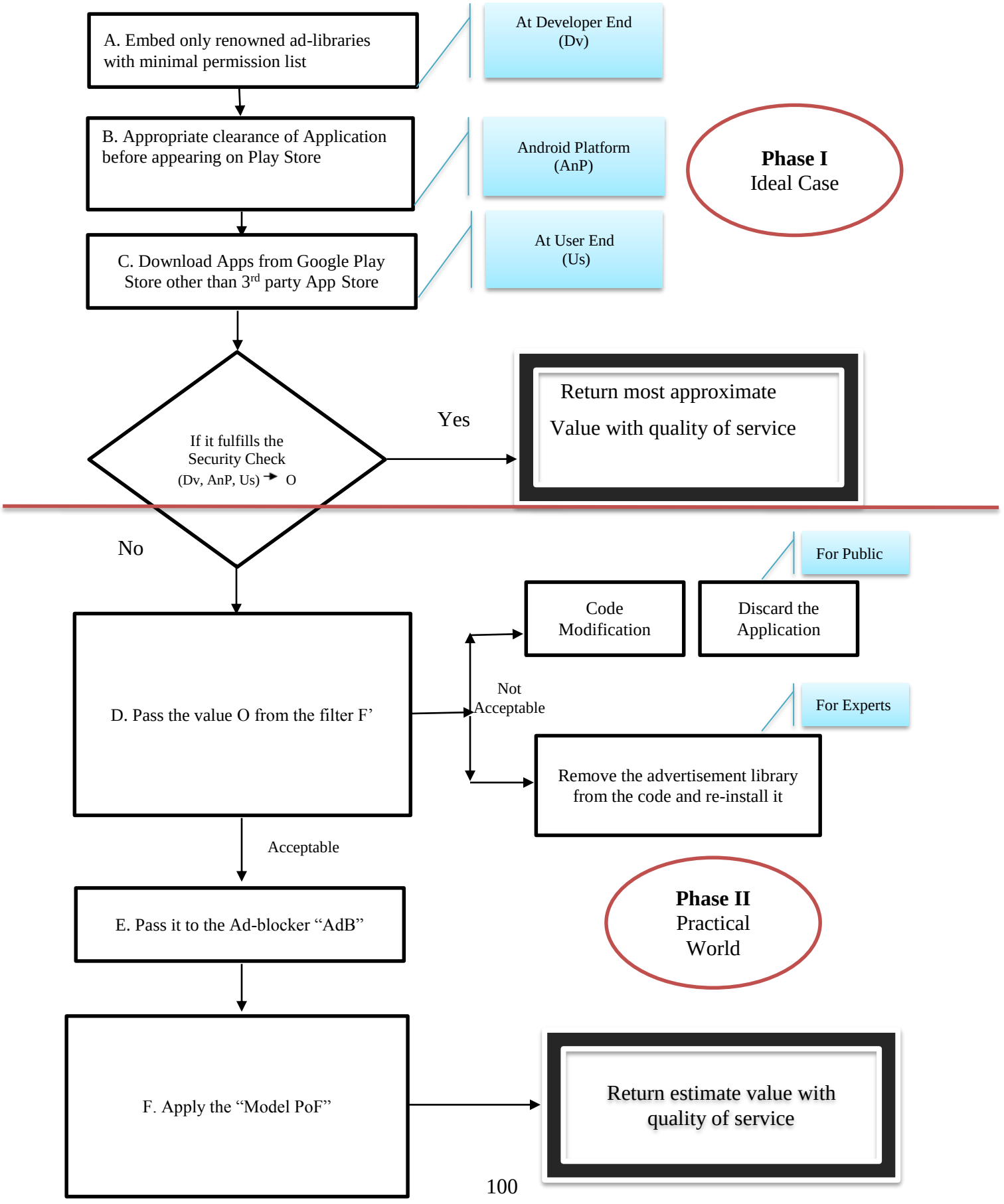
The significance of adopting this model is to protect the user's privacy at a very basic level. Its purpose is to give awareness about protecting user's security and highlight the upcoming threats he may face in the future.

Proposed model categorizes the behavior of the users in the form of general public and security experts and give recommendations accordingly.

8.2 Proposed Model

This Model covers the protection of user's private information from the Advertising libraries in an easy way. Practical world is different from the ideal scenario because number of factors are on the way to disturb the ideal case solution. Here, in this model we have divided the solution in two phases. Phase I covers the ideal case in which all the involved entities including developers, users and Android platform follow the instructions to protect user's security. On the other hand Phase II shows the practical or actual case in which user's security is compromised by the advertisement attacks.

Following figure shows the proposed model framework.


A. Embed only renowned ad-libraries with minimal permission list
At Developer End (Dv)
B. Appropriate clearance of Application before appearing on Play Store
Android Platform (AnP)
Phase I
Ideal Case
C. Download Apps from Google Play Store other than 3rd party App Store
At User End (Us)
If it fulfills the Security Check
(Dv, AnP, Us) → O
Yes
Return most approximate
Value with quality of service
No
For Public
Code Modification
Discard the Application
D. Pass the value O from the filter F'
Not Acceptable
For Experts
Remove the advertisement library from the code and re-install it
Acceptable
Phase II
Practical World
E. Pass it to the Ad-blocker "AdB"
F. Apply the "Model PoF"
Return estimate value with quality of service

8.2.1 Phase I (Ideal Case)

Phase I covers the **Ideal case** which is divided into three steps:

1. **Security Check at Developer End (Dv)**
2. **Security Check at Android Platform (AnP)**
3. **Security Check at Android User End (Us)**

Each one has its own significance because resultant security can be obtained only when each entity perform its tasks according to their privacy policies and mutual cooperation.

In Phase I, the ideal situation has been discussed where all the objects Us (User) , Dv (Developer) and AnP (Android Platform) try to secure the users from the attacks caused by the advertisement libraries. Phase I (security check) is shown in figure 42.

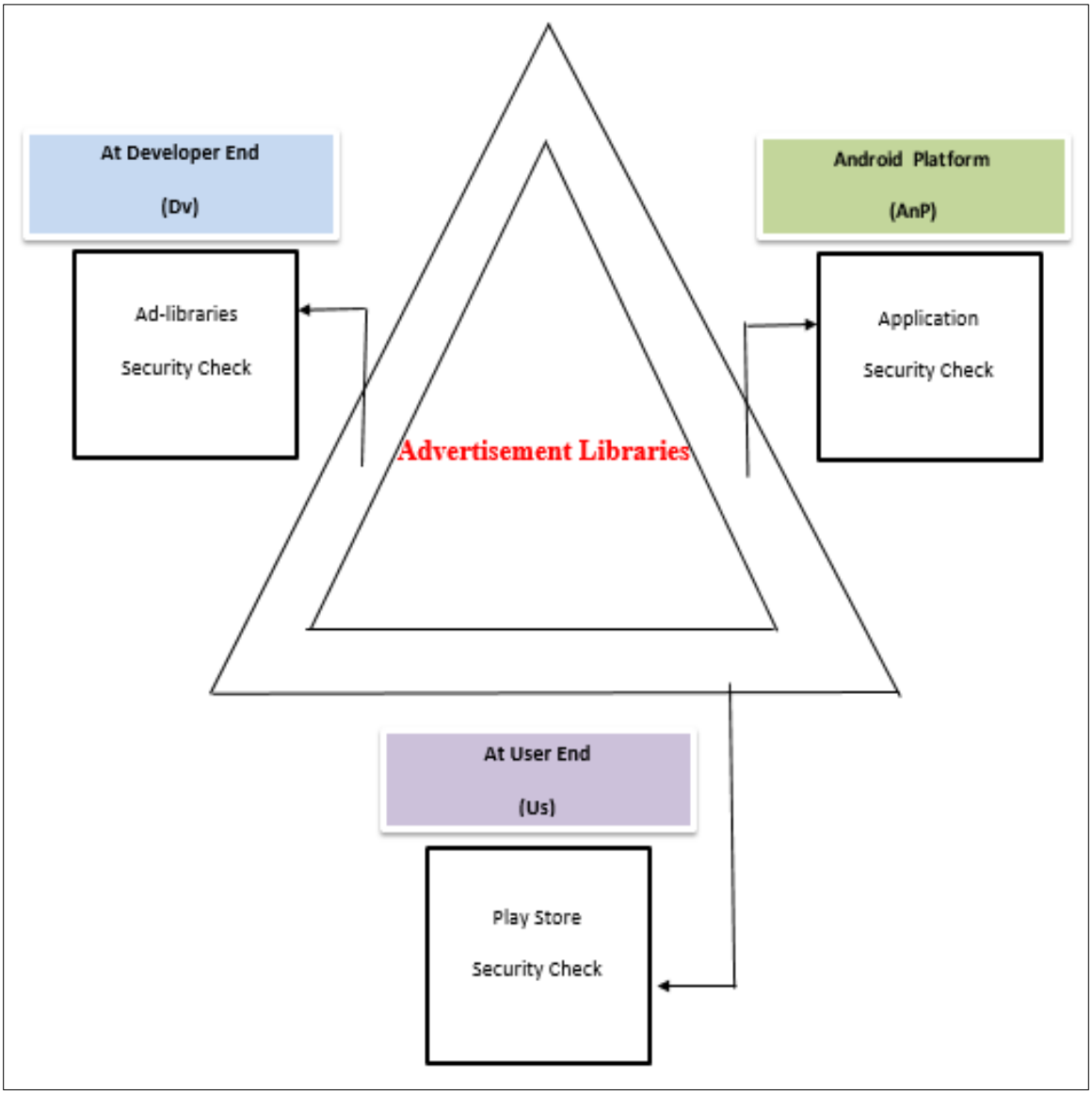


Figure 42. Phase I (Security Check)

As discussed above this phase has been divided into three categories. In figure 42 Developer's end is shown with blue colour, Android platform is shown with green and purple colour is given to the End users. These colours have nothing to do except these are used to differentiate them all. Now in the following section we are going to cover each step in detail.

1. **Security Check at Developer End**

The game begins at developer's end. Because developer embed the advertisement libraries in his application to generate the revenue. In the model, developer end is mentioned as "Dv".

Sometimes they embed more than one libraries to generate more revenue. This may cause a problem. Because in this way applications demand more permissions on the behalf of their embedded libraries.

In our model, we take the ideal case if the developer only embed those libraries in his code that are renowned and demand less permissions. On the other hand if the developer embed minimum ad-libraries in his code to decrease the effect of the attack as much as possible.

Another thing is to protect user's security is to define sets of policies [96]. If developers strictly follow the predefined policies security check gives the most appropriate values "O" as shown in the proposed model.

2. Security Check at Android Platform

We have assigned it the name "AnP" in the proposed model. This is the most important step because Android can control the attack factor as much as anyone else can. Android is an open so the chances of attack are always greater. There is a security check call "bouncer" that scan the Android applications and check if there are any bugs present or they are malicious in nature or not. If they cannot fulfill the security requirements then those applications should discarded by the Google Play store.

But it also has some flaws such as some applications execute their malicious code after installation on the phone by the user.

Our model focuses on the point that Playstore should adopt different methods to scan the applications or they should enhance the functionality of the "Google Bouncer". So that scanning and detection of the applications can be done properly. Common security functionalities such as cryptography or isolation of permissions are need to be implemented to maintain the security at Android platform.

3. Security Check at Android User End

Android users are more vulnerable to attacks so they should be careful in installing the applications.

Google Play Store is more reliable than other third party stores so users should install the applications from the reliable Play store. We mentioned it with "AnP" in our model.

Users' security is necessary in order to protect its own privacy. Now a days biggest security threats of Android in the enterprise is lack of users' control over app security [97]. User's control can be achieved if they update themselves about new security threats and implement existing solutions.

Conclusion of Phase-I

We obtain the value O from Dv, AnP & Us as mentioned in the above model. Now at this point if we achieve the value "O" we will be able to get our desired results. Value "O" is obtained if the all the involved entities independently perform their tasks well. Phase I shows ideal case where no malicious behavior, no buggy app, no dependency on malicious party store and a trustworthy relationship between Android Platform, developers and users is established. If all of the above mentioned requirements are fulfilled then this phase ends up with the most wanted results with maximum provided security.

We pass our application in above mentioned scenario and obtained the most approximate value with more quality of service.

Decisive Point

As shown in the proposed model, if the desired values obtained by finding the perfect security at the end of each entity, then there is no need to pass the application from phase-II. But practically this is not possible because there is a lack of security control at developer (Dv), User (U) and Android Platform (AnP) end. Developers embed more than one advertisement libraries in their application code to raise their revenue. On the other hand most of the android users don't know how to control their privacy and security. It shows the lack of security awareness. Survey results (given in chapter) also shows that most of the users use free version apps as compared to paid apps so eventually they may face more ads when they use the application. It also reveals that users click on the advertisements and follow the instructions directed by the ads in the form of either downloading some other malicious app or giving login credentials to the unknown sources. Users' data that is potentially at risk includes user names, authentications, passwords, stored application data, transaction histories, UDID/ EMEI, social address and credit card number etc. [97]. We have seen in the previous chapters that leakage of such information is caused by the presence of advertisements in android applications.

In phase-I we obtained the desired results by assuming the ideal scenario where there is no security breach or incident occurred because all the parties such as developers, users and android platform work together to make and function the app in a secure manner.
Now if the desired results are not obtained (which are expected in the real world scenario) then the problems need to be solved in a real phase i.e. phase-II.

8.2.2 PHASE II (Practical World)

If the desired results do not obtained in phase-I, we pass our application from phase-II.
We have divided the next phase in the following parts:

- Apply Filter (F')
- Pass it to the Ad-blocker (AdB)
- Apply the model (PoF)
- Keep on monitoring and get the estimated desired results

1. Apply Filter F' (Step D in Proposed Model)

As mentioned in the proposed model, Filter F' is applied to obtain the better results in a practical scenario. At a very basic level users should download and install and application by checking its reviews, star ratings and download counts.
We have divided filtration process in two domains. These are listed as general public and security oriented organization. Figure 43 shows the breakdown and summary of the filtration process.

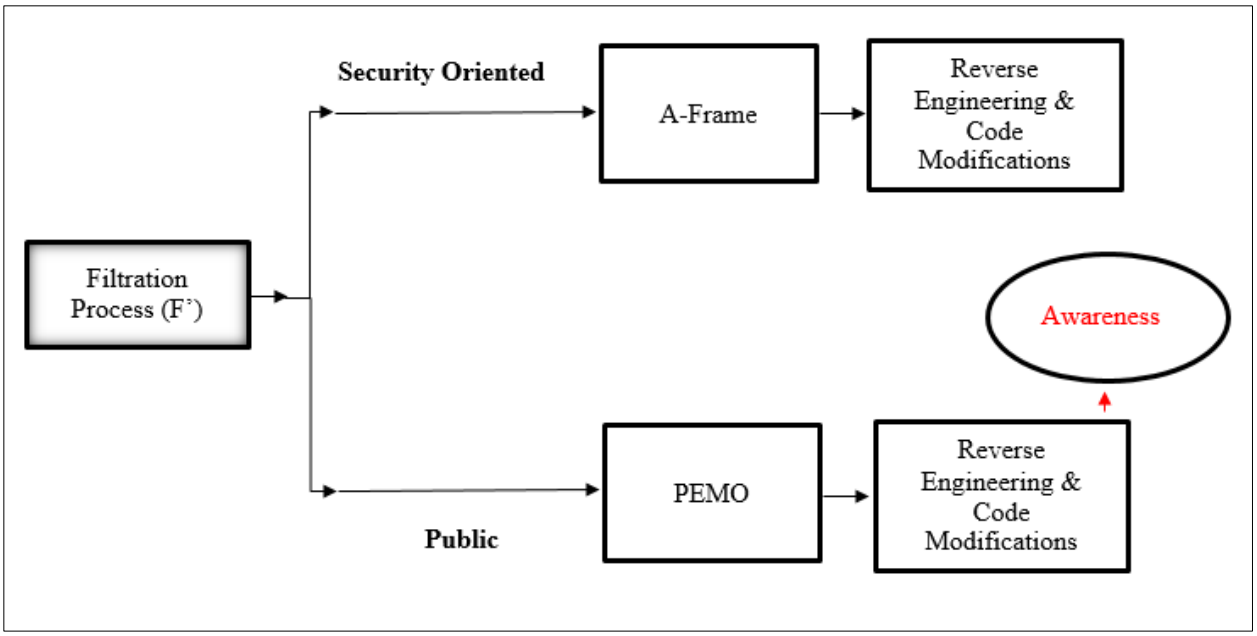


Figure 43. Filtration Process (F')

We start our filtration process from choosing the most suitable technique for general public and then for security oriented organizations or people.

For General Public:

PEMO: In **Section 4.2.4** we have discussed PEMO technique. It is used to reduce the permission bloat. Users can choose the permissions of their own choice and discard others. Figure. 44 shows the display of PEMO screen. It is freely available on Google Playstore. Once users remove the permissions from the provided list, PEMO removes it permanently via package manager functionality.

While comparing all the existing techniques, we choose PEMO as a best suitable method to reduce permissions for general public.

Figure 44. PEMO Display Screen

Code Modification:

Sometimes PEMO fails to function properly such as in "Pics Art" application when location and personal information access permissions were removed, that application stopped working. In such situations we need an alternative of it. That is why we introduced code modification step via reverse engineering of the code for public.

Reverse engineering can be done easily with the help of online available tools such as dex2jar and .Apk extraction.

Ad embedded applications demand more permissions. By keeping in mind their malicious nature, those extra permissions can be extracted from the code. It can be possible by reverse engineer the application code, reinstall the application after removing the advertisement permissions from the code.

Restriction:

PEMO is easy to use, it can be done by downloading the PEMO app from Google Playstore. But sometimes it stopped working or create more overheads. As an alternative, reverse engineering method can be applied but it needs users' awareness that need to be improved itself.

For Security Oriented Organizations or Individuals:

A-Frame

All the existing and high rated techniques in solving a problem caused by embedded ads have been discussed and analyzed in chapter: 04.

Study and comparison between all techniques in section 4.3 shows that A-Frame can provide better results by separating ads and application permissions as well as input and display isolation. We have chosen this technique in our filtration process for security oriented individuals. We put this methodology at expert's level. Its process and permission isolation made it attractive for adding it in our framework.

Code Modification:

Memory, Time-to-Start and Event- Dispatching overhead caused by A-Frame at a certain level also ends up in code modification.

2. Pass it to the Ad-Blocker (Step E in Proposed Model)

Condition applied in the proposed model shows that if the filtered vales are not acceptable, by passing them from PEMO or A-Frame then code modification method is applied. If the results are not acceptable then those apps are discarded. On the other hand, if the filtration process F' values are acceptable we pass our application to the next step i.e. AdB.

Ad-Blockers are used to prevent ads from being displayed on the users' screen. We have analyzed the behavior of existing ad-blockers and concluded that they can compromise the security of the Android users by demanding more permissions as compared to the actual permission does.

Our contribution of this study is to introduce the ad-blockers that demand less permissions as compared to the exiting ad-blockers in the Android market.

Practical implementation and experimental results will be provided in the future research by developing the Ad-Blockers who demand less permissions and will be least harmful for the Android users' security as compared to the existing ones.

3. Apply the Model "PoF" (Step F in Proposed Model)

We termed PoF as Model of obfuscation. It includes inaccuracy, imprecision and vagueness. We need an approach to minimize the threat caused by advertisements by demanding the location information. Most of the free version apps demand location information on the behalf of their embedded advertisement libraries. Leakage of location information may become the leakage of professional information. Now a days privacy threats due to location information are common. Future moments of the victim can be tracked by adversary. Location based threats may include political affiliation, medical problems or alternative lifestyle of the user [99]. To protect the user from leakage of location information we applied following three concepts in PoF.

i. Principle of Minimal Collection
ii. Need-to-know
iii. Obfuscation

Principle of Minimal collection

It comes under the category of data protection act (Principle 3). It requires that personal information can only be collected for the specified purposes [100]. In our case advertisement libraries demand more permissions as compared to their specified purposes. To mitigate this effect we have used need-to-know and Obfuscation methods in our proposed model.

Need-to-Know Principle

It is used to hide the actual location of our mobile station and let the advertisement libraries know only those services that are needed to function properly. It means we can give the approximate location, or the vague location to hide the actual location information. It is the most fundamental principle of security. Applications that require the locations can provide the advertisements according to our area. For example it can tell about the newly opened coffee shop nearby your home or about the other shopping stores.

Need-to-know only aim to protect users from releasing irrelevant and unnecessary information to the service providers. Extra permissions of the advertisement libraries such as to access contact list or to send SMSs to the contacts can be controlled by Need-to-Know principle.

We summarized need-to-know principle in figure.45. It shows that fundamental fair information practice and principle of obfuscation collectively provide Need-to-Know principle.

It is based on two conditions:

i. Clearance of Information
ii. Verification of Need-to-Know

Need to know principle says that half of the issue is information clearance. We can have half of our half done if we provide access to the authorized users only.

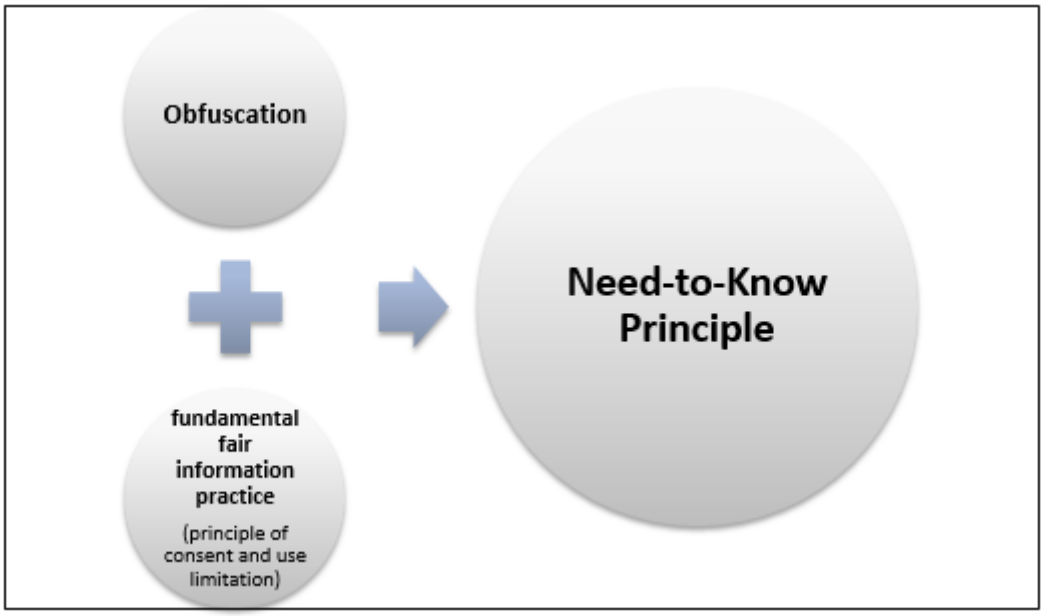


Figure 45. Need-to-Know Principle Components

Principle of Obfuscation

Purpose of obfuscation is to give true information at some extent and hide remaining [101]. Sometimes showing location information is useful for us. Our purpose is to control this information in order to minimize the effect of the attack caused by an attacker who consciously target the users for multiple purposes. By adopting the obfuscation all of the desired goals cannot be meet for example at one side we want our colleagues to track our location and on the other hand we don't want attackers to locate us. So this ambiguous.

Our mechanism of obfuscation is be based on three different perspectives i.e. Inaccuracy, Imprecision and vagueness [101].

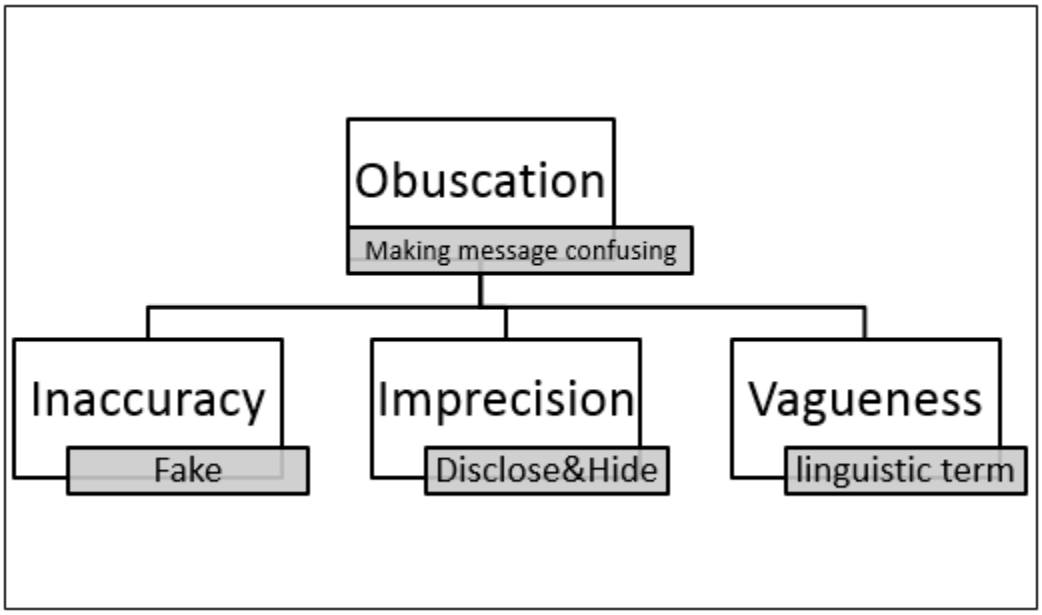


Figure 46. Types of Obfuscation

Figure. 46 shows the brief description of obfuscation. In the following section we are going to discuss them in detail.

I. Inaccuracy

The accurate information is replaced with the fake information. The purpose of adopting this behavior is to provide the inaccurate information to the service providers in order to avoid attacks. This behavior let the service providers receive the totally inaccurate information.

It is good for the security perspective but in real world it would be inconvenient for the user. For example as mentioned above if someone wants his colleagues or friends to get to know about his location.

II. **Imprecision**

It can be released information that will be partially accurate and will hide the actual information. For example if we talk about the “location” then only the “user’s region” will be disclosed and exact current position will be hide by the ad-libraries.

So it can overcome the flaws of the first part i.e. inaccuracy.

III. Vagueness

Linguistic terms has been involved here. For instance if the disclosed location is “far away” from the actual location.

By adopting the proposed model or recommendation model, in-advertisement application attacks can be controlled. It not only helps to reduce the permission bloat and leakage of personal information to third parties but to help in protection of location information targeted by advertisement libraries as well.

It is highly recommended model in today’s need to prevent Android users from In-application advertisement attacks.

9 CONCLUSION & RECOMMENDATIONS

9.1 Conclusion

The research has been conducted on embedded adware in Android applications in order to analyze its impact on android user's privacy.

The literature review was based on explaining the various schools of thoughts that focuses on Android security from where we also narrowed down the concept of Advertisements in Android applications. It further proceeded and helped in explaining various approaches towards solving the in-advertisement applications attacks that impact the users' privacy. After carrying out quantitative research it is hereby concluded that attackers exploit the vulnerabilities of Android phone in many ways. Analysis of data has shown that there is a significant impact of embedded adware on android users' privacy. This research was purely based on advertisements in an Android application. Complete process of embedding the advertisements in an android application has been explored in this research. Existing solutions such as existing ad-blockers are analyzed by exploring its dependencies and limitations. Study of existing techniques that are used to reduce the permission bloat, privilege separation of advertisements and applications and giving user's empowerment to discard the unnecessary permissions had also been the part of this research. Limitations of existing methods were found to put the light on the research. Survey was conducted from Android users to know the awareness of the users towards protecting their security. It was found that people mostly prefer to use freely available applications and click on advertisements and follows the directions asked by advertisement services. It results in leakage of private data. The reason behind is that developers introduce ad-libraries SDKs in their freely available apps to get paid for their work. Other results and their analysis also revealed the fact that Android platform has also some defects while protecting and securing user privacy. Such as permissions of advertisements and applications are not separated which is the root cause of the problem. In the end we introduced the model to solve the limitations of the existing techniques and solutions to give new and effective ways of protecting Android users (both general public and security oriented individual or organizations) from embedded adware.

Chapter 10

10 RESEARCH LIMITATIONS & FUTURE DIRECTIONS

This chapter shed some light on the limitations of this research. It will help in giving the future researchers a sense of direction that which area is under studied in this research that can be covered in future.

10.1 Limitation of Research

This research is carried out by considering only Android users. Effects of in-application advertisement attacks on Windows OS and iOS are need to be explored as well.

10.2 Future Directions

As currently only quantitative research method is used, so in future mixed method (including both qualitative and quantitate) approaches must be used.

Development of proposed Ad-Blocker application with minimal permissions and Practical implementation of proposed model at Android runtime will be logically the next step.

BIBLIOGRAPHY

[1] Snugg, "A brief History of Smartphones" available at http://www.thesnugg.com/a-brief-history-of-smartphones.aspx

[2] Androidcentral "Android History" available on "http://www.androidcentral.com/android-history

[3] Voctoria Wollaston "Worldwide Smartphone OS Market Share" available at http://www.dailymail.co.uk/sciencetech/article-3293254/The-decline-Android-Record-number-users-abandoning-mobile-software-favour-Apple.html

[4] Ashmeet Kaur, Divya Upadhyay (2014) PeMo: Modifying Application's Permissions and Preventing Information Stealing on Smartphones Available at: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6949282

[5] Paul Pearce et al. (2012) ,"AdDroid: Privilege Separation for Applications and Advertisers in Android". Available at. https://www.google.com.pk/?gws_rd=ssl#q=AdDroid:+Privilege+Separation+for+Applications+and+Advertisers+in+Android

[6] Voctoria Wollaston "Worldwide Smartphone OS Market Share" available at http://www.dailymail.co.uk/sciencetech/article-3293254/The-decline-Android-Record-number-users-abandoning-mobile-software-favour-Apple.html

[7] APPCASE "Free Vs Paid Apps" available at http://getappcase.com/blog/beginners-tips/free-vs-paid-apps

[8] Billa,2014 "16 Android App Store Alternative for those who hate Google Play""available at http://joyofandroid.com/android-app-store-alternatives/

[9] Statista "Number of apps available in leading app stores" available at http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[10] Geumhwan Cho et al. ,2016 "Combating online fraud attacks in mobile-based advertising"available at http://www.jis.eurasipjournals.com/content/pdf/s13635-015-0027-7.pdf

[11] AppsGeyser, "How to create the perfect app" available at http://www.appsgeyser.com/blog/make-money-from-a-free-android-app-mobile-ad-types/

[12] Fredrik Bugge Lyche & Jørgen Haukedal Lytskjold (2014)." Improving security solutions in modern smartphones". Available at: http://www.diva portal.org/smash/get/diva2:749357/FULLTEXT01.pdf

[13] Android Developers,2016 "Platform Versions" available at http://developer.android.com/about/dashboards/index.html

[14] Techotopia,"An Overview of the Android Architecture"available at http://www.techotopia.com/index.php/An_Overview_of_the_Android_Architecture

[15] Jason Rouse Neil Bergman, Mike Stan?eld and Joel Scambray (July 9, 2013) Hacking Exposed Mobile. Available at. http://www.amazon.com/Hacking-Exposed-Mobile-Security-Solutions/dp/0071817018

[16] "android security" available at https://source.android.com/security/

[17] Lynn Batten et al. 2013, " The risk of advertizing libraries on Android mobile devices" "available at http://iml.univ-mrs.fr/ati/crypto_puces/2013/slide/Batten-1.pdf

[18] Tutorialspoint, "Android- Application Components" available at http://www.tutorialspoint.com/android/android_application_components.htm

[19] Statista, 2015 "Number of available applications in the Google Playstore from December 2009 to November 2015" available at http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

[20] Philipp Berner, "Android Market" available at https://www.quora.com/How-many-developers-companies-have-submitted-an-app-to-the-iOS-App-Store-or-Android-Market

[21] Android security overview (May 6,2014) Available at: https://source.android.com/

[22] H Xu, D Liu, A Koehl, H Wang, A Stavrou, in Proceedings of 19th European Sysposium on Reseach in Computer Security. Click fraud detection on the advertiser side (Springer International Publishing, 2014)

[23] P Pearce, AP Felt, G Nunez, D Wagner, in Proceedings of the 7th Symposium on Information, Computer and Communications Security. AdDroid: privilege separation for applications and advertisers in android (ACM, 2012)

[24] Daniel W.K, x.Liu et al. (2014) Strategies In Improving Android Security. Available at http://www.pacis-net.org/file/2014/1953.pdf

[25] Bin Liu, May 2015, "Efficient Privilege De-Escalation for Ad Libraries in Mobile Apps" available at http://www-scf.usc.edu/~binliu/papers/mobisys_15.pdf

[26] 2014, "Distribution of Free vs. Paid Android Apps." available at http://www.appbrain.com/stats/

[27] N. Viennot, E. Garcia, and J. Nieh., 2014," A Measurement Study of Google Play. In Proc. ACM SIGMETRICS"

[28] Theodore Book, Adam Pridgen, Dan S. Wallach, 2013 "Longitudinal Analysis of Android Ad Library Permissions

" available at http://arxiv.org/pdf/1303.0857.pdf

[29] Rupali Sharma, "Developing for Android - An Introduction" available at http://www.cprogramming.com/android/android_getting_started.html

[30] Collin Mulliner et al, June 2014 "VirtualSwindle: An Automated Attack Against In-App Billing on Android

" available at https://www.mulliner.org/collin/publications/asia226-mulliner.pdf

[31] Lookout, 2011 "Mobile Threat Report 2011" available at https://www.lookout.com/resources/reports/mobile-threat-report-2011

[32] DataTheorem, 2013 "Threat Model for Mobile Applications Security & Privacy"available at https://datatheorem.github.io/resources/DataTheorem.MobileAppThreatModel.pdf

[33] Li Li, Daoyuan Li, Tegawende F Bissyand ´ e, David Lo, Jacques Klein, and Yves Le Traon. Ungrafting Malicious Code from Piggybacked Android Apps. In Technique Report, 2015.

[34] Li Li, Tegawende F. Bissyand ´ e et.al, Nov 2015 "An Investigation into the Use of Common Libraries in Android Apps

"available at http://arxiv.org/pdf/1511.06554.pdf

[35] How-To Greek "What is Malvertising and how do you protect yourself"available at http://www.howtogeek.com/227205/what-is-malvertising-and-how-do-you-protect-yourself/

[36] Vaibhav Rastogi.et al. February,2016 "Are these Ads Safe: Detecting Hidden Attacks through the Mobile App-Web Interfaces" available at http://pages.cs.wisc.edu/~vrastogi/static/papers/rscpzr16.pdf

[37] Mathew J. Schwartz, 2013, "Android Malware Being Delivered Via Ad Networks" available at http://www.darkreading.com/attacks-and-breaches/android-malware-being-delivered-via-ad-networks/d/d-id/1111145?

[38] Dennis O'Reilly, April 10,2014 "Protect your device from malicious ads"available at http://www.cnet.com/how-to/protect-your-mobile-device-from-malicious-ads/

[39] Liviu Arsene, September, 2015 "New Type of DDoS Attack Uses Mobile Ad Networks" available at http://www.hotforsecurity.com/blog/new-type-of-ddos-attack-uses-mobile-ad-networks-12761.html

[40] Ryan Stevens et al. "Investigating User Privacy in Android Ad Libraries" available at http://mostconf.org/2012/papers/27.pdf

[41] Bartlomiej Uscilowski, October 2013, "Mobile Adware and Malware Analysis"available at http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/madware_and_malware_analysis.pdf

[42] Lookout. The bearer of badnews. (October 14,2013) Available at: https://blog.lookout.com/blog/ 2013/04/19/the-bearer-of-badnews-malware-google-play/,April2013.

[43] Theodore Book, November 2013, "Privacy Concerns in Android Advertising Libraries " available at https://scholarship.rice.edu/bitstream/handle/1911/76347/MS-thesis1.pdf?sequence=1

[44] AVG. Android/dgen plankton a. (April 8,2014) Available at: http://www.avgthreatlabs.com/virus-and-malware-information/info/android-dgen-plankton-a/.

[45] Adrian Mettler, August 20 2014, "SSL VULNERABILITIES: WHO LISTENS WHEN ANDROID APPLICATIONS TALK?" available at https://www.fireeye.com/blog/threat-research/2014/08/ssl-vulnerabilities-who-listens-when-android-applications-talk.html

[46] Lynn Margaret Batten,2015 "Mallory-athreat to your mobile device?" available at http://summerschool-croatia15.cs.ru.nl/Mallory.pdf

[47] Chris Sanders, 9 June 2010, "Understanding Man-In-The-Middle Attacks - Part 4: SSL Hijacking"available at http://www.windowsecurity.com/articles-tutorials/authentication_and_encryption/Understanding-Man-in-the-Middle-Attacks-ARP-Part4.html

[48] Mozilla Blog, "Deprecating Non-Secure HTTP" available at https://blog.mozilla.org/security/files/2015/05/HTTPS-FAQ.pdf

[49] Olaf Kolkman, October, 2005 "An Introduction to the Domain Name System"available at https://www.nlnetlabs.nl/downloads/DNSforPolicyMakers.pdf

[50] July 28, 2014, InfoSec, "How CyberCrime Exploits Digital Certificates"available at http://resources.infosecinstitute.com/cybercrime-exploits-digital-certificates/

[51] July, 2012, "The Information Assurance Mission at NSA"available at https://www.nsa.gov/ia/_files/factsheets/adf-2012-1202_defending_against_compromised_certificates.pdf

[52] Jorn Lapon, "SecureApps"available at https://www.msec.be/secureapps/seminarie/msec_x509_android.pdf

[53] KC Wilbur, Y Zhu,Sci. 28(2), 293–308 (2009), "Click fraud" available at http://pubsonline.informs.org/doi/abs/10.1287/mksc.1080.0397

[54] BY PC MAG ME TEAM APRIL 5, 2015, "The 100 Best Android Apps of 2015" available at http://me.pcmag.com/mobile-app/1639/feature/the-100-best-android-apps-of-2015

[55] 31 Dec 2016, Phandroid Editors, "The 100 Best Android Apps of 2016" available at http://phandroid.com/best-android-apps/

[56] MAX EDDY, February 11,2014, "The 5 Best Android SMS Replacements"available at http://www.pcmag.com/article2/0,2817,2452987,00.asp

[58] Paul Pearce, Adrienne Porter Felt et.al, 2012, "Ad-Droid" available at https://www.eecs.berkeley.edu/~daw/papers/addroid-asiaccs12.pdf

[59] Shashi Shekhar et.al, "AdSplit" available at https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final101.pdf

[60] Ashmeet Kaur, Divya Upadhyay, 2014 "Advanced Permission Manager"available at https://play.google.com/store/apps/details?id=com.gmail.heagoo.pmaster&hl=en

[61] Wenhao Li et.al, 2015 "Ad Attester" available at http://ipads.se.sjtu.edu.cn/_media/publications/adattester-mobisys15.pdf

[62] Dongqi Wang et al, 2014, "AdHoneyDroid" available at http://accepted.100871.net/adhoneydroid.pdf

[63] Google Inc. Google play.https://play.google.com/store

[64] William Enck, Peter Gilbert, Byung-Gon Chun,Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth. Taintdroid: An information flowtracking system for real-time privacy monitoring on smartp

[65] Xiao Zhang et.al, 2013 "AFrame" available at http://www.cis.syr.edu/~wedu/Research/paper/aframe_acsac2013.pdf

[66]2011, J. Christopher Sandvig et. al "USAGE AND PERCEPTIONS OF INTERNET AD BLOCKERS:AN EXPLORATORY STUDY "http://iacis.org/iis/2011/59-69_AL2011_1603.pdf

[67]Google Chrome Store AdBlock Plus Statistics. available at https://chrome.google.com/webstore/detail/adblock-plus/cfhdojbkjhnklbpkdaibdccddilifddb

[68]Mozilla AdBlock Plus Statistics.https://addons.mozilla.org/en-US/firefox/addon/adblock-plus/statistics/usage/?last=30

[69] Amazon, Google and Microsoft Escape Adblock Plus, for a Price.http://www.engadget.com/2015/02/03/amazongoogle-microsoft-adblock-plus/

[70]Publishers Watch Closely as Adoption of Ad Blocking Tech Grows.http://adage.com/article/digital/adoption-adblocking-tech-grows/297101/.

[71] October 2015, Enric Pujol et al. "Annoyed Users: Ads and Ad-Block Usage in the Wild"available at http://conferences.sigcomm.org/imc/2015/papers/p93.pdf

[72]2014, Palant, "W. Adblock Plus" available at https://adblockplus.org

[73]2014, Gundlach, "M. Adblock" available at https://getadblock.com/,2014.

[74]2014, Signanini, J. M., and McDermott, B. Ghostery. available at https://www.ghostery.com/en/

[75]2014, Disconnect: "Online privacy and security". available at https://disconnect.me/

[76]2012, Hurps, M. Ad blocker: Ad muncher.available at http://www.admuncher.com/

[77]Digiex, "The Dangers of Ad Blockers on the Internet (AdMuncher / Adblock" available at http://digiex.net/guides-reviews/reviews/147-review-dangers-ad-blockers-internet-admuncher-adblock.html

[78]March 4, 2016, Philipp Greitsch , "Best Android Ad Blocker for Rooted & Unrooted Devices" available at http://trendblog.net/best-android-adblocker-for-rooted-unrooted-devices/

[79]"Aguard AdBlocker", available at https://chrome.google.com/webstore/detail/adguard-adblocker/bgnkhhnnamicmpeenaelnjfhikgbkllg?hl=en

[80]Feb 2013, David Ruddock, "Google Has Effectively Killed Adblock Plus In Android 4.2.2" available at http://www.androidpolice.com/2013/02/13/google-has-effectively-killed-adblock-plus-in-android-4-2-2/

[81]2014, Eric Ravenscraft, "This Just In: AdBlock Plus Still Uses a Lot of Memory"available at http://lifehacker.com/adblock-plus-once-again-found-to-dramatically-increase-1576341872

[82] J. Hightower and G. Boriello. Location systems for ubiquitous computing IEEE Computer, 34(8):57–66, 2001

[83]2009, Stack overflow, "What is static code analysis?" available at http://stackoverflow.com/questions/49716/what-is-static-code-analysis

[84]2013, Savan Gadhiya, "International Journal of Advanced Research in Computer Science and Software Engineering" available at http://www.ijarcsse.com/docs/papers/Volume_3/4_April2013/V3I4-0371.pdf

[85]https://www.google.com.pk/search?q=Static+Analysis+Techniques+used+in+Android+application+Security+Analysis&rlz=1C1CHWA_enPK607PK607&oq=Static+Analysis+Techniques+used+in+Android+application+Security+Analysis&aqs=chrome..69i57.4174j0j7&sourceid=chrome&es_sm=122&

[86]November 7, 2014, Sruti Bhagavatula et al. "Leveraging Machine Learning to Improve Unwanted Resource Filtering

" available at https://www.cs.cmu.edu/~sbhagava/papers/ml.adblock.pdf

[87]October 17, 2014, ALLAN BENNETTO, http://fruitful.io/2014/10/17/avoid-malware-by-understanding-androids-app-permissions/

[88]Scott Alexander-Bown, "Android Security: Adding Tampering Detection to Your App"available at https://www.airpair.com/android/posts/adding-tampering-detection-to-your-android-app

[89]https://nakedsecurity.sophos.com/2016/01/15/malvertising-why-fighting-adblockers-gets-users-backs-up/

[90]Google PlayHelp, "Review app permissions thru Android 5.9" available at https://support.google.com/googleplay/answer/6014972?hl=en

[91]Feb, 2014 "Is Adblock (Plus) a security risk?" available at http://security.stackexchange.com/questions/52361/is-adblock-plus-a-security-risk

[92] Symantec Corporation, April 2014, "Internet Security Threat Report 2014 available at http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v19_21291018.en-us.pdf

[93] 2014, AV-Comparatives, "Mobile Security Review" available at http://www.av-comparatives.org/wp-content/uploads/2014/09/avc_mob_201407_en.pdf

[94] Michael I. Gordon et al. , 2015, "Information-Flow Analysis of Android Applications in DroidSafe" available at http://www.internetsociety.org/sites/default/files/02_1_2.pdf

[95] Android Developers "Security Tips"available at https://developer.android.com/training/articles/security-tips.html

[96] Margaret Jones, "Android app security FAQ: Keeping devices safe from Android threats" available at http://searchmobilecomputing.techtarget.com/tip/Android-app-security-FAQ-Keeping-devices-safe-from-Android-threats

[97] Decompling App, "TOP 10 MOBILE SECURITY RISKS"available at http://www.decompilingandroid.com/mobile-app-security/top-10-mobile-security-risks/

[98] Marco Gruteser and Dirk Grunwald, "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking" available at http://www.winlab.rutgers.edu/~gruteser/papers/gruteser_anonymous_lbs.pdf

[99]"The amount of personal data you may hold (Principle 3)" avaialable at https://ico.org.uk/for-organisations/guide-to-data-protection/principle-3-adequacy/

[100] D. Hutter, W. Stephan, and M. Ullmann. Security and privacy in pervasive computing: State of the art and future directions. In D. Hutter, G. M¨uller, and W. Stephan, editors, Security in Pervasive Computing, volume 2802 of Lecture Notes in Computer Science, pages 284–289. Springer, 2004.

APPENDIX A

QUESTIONNAIRE

In-Advertisement Applications

Embedded Adware	1	2	3	4	5
Do you think Embedded ads in Android applications are helpful for shopping and they make your life easier?					
Third party advertising agencies should be able to compile my usage behaviour across different web sites for direct marketing purposes					
Free or low cost apps that are supported by in-app purchases (such as games that prompt you to make real-money purchases to score more points) are more appealing to be downloaded?					
More expensive apps that have no ads or in-app purchases are my choice of download?					
Some of the Applications have Advertisements in them. These are irritating.					
Preventive Measures					
If AppChoices is introduced as a downloadable tool to manage the relevant advertising preferences in mobile applications, you will prefer to use it?					
Ad-blocker should demand less permissions and stop displaying the annoying ads?					
There should be a mechanism to view the hidden permissions of the particular application and those who can separate the permissions of Android app and embedded					

adware?					
Advertisement networks or other third parties should know about our “Exact” location?					
Advertisement attacks					
I often click on those advertisements intentionally or unintentionally?					
In general, **CONVENIENCE** is more important than **PRIVACY?**					
Do you think metrics to measure "how secure" a specific mobile application rated be of any help or value to you?					
Have you installed an Application from 3rd party source other than "Google Play Store”?					
We understand all the permissions demanded at the time of App installation?					

APPENDIX B

LIST OF ACRONYMS

OS- Operating System

App- Application

Ad- Advertisement

DEX- Dalvik Executable Bytecode

OEM- Orignal Equipment manufacturer

ADB- Android Debug Bridge

CPC- Cost per Click

CPM- Cost per Mile

CPA- Cost per Action

MiMT- Man-in-the Middle Attack

SSL- Secure Socket Layer

VPN- Virtual Private Network

APK- Android Pacakge Kit

HTTP- Hypertext Transfer Protocol Secure

HTTPS- Hypertext Transfer Protocol Secure

DNS-Domain Name System

CA- Certificate Authority

EA- Embedded Adware

AT- Advertisement Attacks

PM- Preventive Measures