

---

# Permission Model Analysis of Android Applications

---

*A B.Tech Dissertation report submitted in fulfilment of the requirements  
for the degree of Bachelor of Technology*

*by*

Arnima Sharma(2015UCP1009),  
Kushal Kriplani (2015UCP1017),  
Akshayaa Suresh (2015UCP1035)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

May 2019

# Certificate

We,

Arnima Sharma(2015UCP1009),

Kushal Kriplani (2015UCP1017),

Akshayaa Suresh (2015UCP1035),

Declare that this thesis titled, “Permission Model Analysis of Android Applications” and the work presented in it are our own. I confirm that:

- This project work was done wholly or mainly while in candidature for a B.Tech. degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT).
- Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this Dissertation is entirely our own work.
- We have acknowledged all main sources of help.

Signed:

---

Date:

---

Dr. Vijay Laxmi

Professor

Department of Computer Science and Engineering

Malaviya National Institute of Technology Jaipur

*May 2019*

# *Abstract*

---

Name of the students:

**Arnima Sharma(2015UCP1009),**

**Kushal Kriplani (2015UCP1017),**

**Akshayaa Suresh (2015UCP1035)**

Degree for which submitted: **B.Tech.**

Department: **Computer Science and Engineering**

Thesis title: **Permission Model Analysis of Android Applications**

Thesis supervisor: **Dr. Vijay Laxmi**

Month and year of thesis submission: **May 2019**

---

Android is one of the most popular mobile operating systems nowadays, whose popularity however, also attracts skilled developers to develop malicious software to exploit illegitimate means for profit. The entire responsibility of the decision making process, when it comes to an Android Application, is delegated onto the user. The user might often lack the required knowledge set to ascertain whether an application is legitimate. This calls for an invasive assessment into the world of Android, especially that pertaining to its Permission Model. Various Applications can be assessed against some predetermined standards and tests for their legitimacy. This dissertation, titled, 'Permission Model Analysis of Android Applications', aims to shed some light upon how some Android Applications can deceive unsuspecting users into giving more permissions, and hence information than intended.

# Plagiarism Report



## Plagiarism Checker X Originality Report

Similarity Found: 12%

Date: Sunday, May 19, 2019

Statistics: 1291 words Plagiarized / 11174 Total words

Remarks: Low Plagiarism Detected - Your Document needs Optional Improvement.

Permission Model Analysis of Android Applications A B.Tech Dissertation report submitted in fulfillment of the requirements for the degree of Bachelor of Technology by Arnima Sharma(2015UCP1009), Kushal Kriplani (2015UCP1017), Akshayaa Suresh (2015UCP1035) DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR May 2019 Certificate We, Arnima Sharma(2015UCP1009), Kushal Kriplani (2015UCP1017), Akshayaa Suresh (2015UCP1035), Declare that this thesis titled, "Permission Model Analysis of Android Applications" and the work presented in it are our own. I confirm that: \_ This project work was done wholly or mainly while in candidature for a B.Tech.

degree in the department of computer science and engineering at Malaviya National Institute of Technology Jaipur (MNIT). \_ Where any part of this thesis has previously been submitted for a degree or any other qualification at MNIT or any other institution, this has been clearly stated. Where we have consulted the published work of others, this is always clearly attributed. Where we have quoted from the work of others, the source is always given.

With the exception of such quotations, this Dissertation is entirely our own work. \_ We have acknowledged all main sources of help. Signed: Date: May 2019 Dr. Vijay Laxmi Professor Department of Computer Science and Engineering Malaviya National Institute of Technology Jaipur i Abstract Name of the students: Arnima Sharma(2015UCP1009), Kushal Kriplani (2015UCP1017), Akshayaa Suresh (2015UCP1035) Degree for which submitted: B.Tech.

Department: Computer Science and Engineering Thesis title: Permission Model Analysis of Android Applications Thesis supervisor: Dr. Vijay Laxmi Month and year of thesis

## *Acknowledgement*

It is a matter of great pleasure and privilege for us to present our B. Tech Project report on “Permission Model Analysis of Android Applications”. First of all, we would like to express our deep sense of respect and gratitude towards our supervisor, Dr Vijay Laxmi, Professor, Department of Computer Science and Engineering, Malaviya National Institute of Technology Jaipur, who has been the guiding force behind this work. We want to thank her for introducing us to the field of Android Applications and giving us the opportunity to work under her. Her undivided faith in this topic and ability to bring out the best of analytical and practical skills in people has been invaluable in tough periods. Without her invaluable guidance and cooperation throughout the period, it would not have been possible for us to complete this thesis. She provided us with continuous encouragement and support. She has been the principle guiding force behind this.

We also express our heartfelt gratitude towards all the teachers of Department of Computer Science and Engineering and technical staff for all their guidance and sympathetic throughout our stay at MNIT Jaipur.

# Contents

<b>Certificate</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Plagiarism Report</b>	<b>iii</b>
<b>Acknowledgement</b>	<b>iv</b>
<b>Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>viii</b>
<b>List of Tables</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Objectives . . . . .	2
1.3 Thesis Organisation . . . . .	3
<b>2 Android Survey</b>	<b>4</b>
2.1 Android Playstore . . . . .	4
2.2 APK File . . . . .	5
2.3 Android Manifest File . . . . .	6
2.3.1 Elements of the Manifest File . . . . .	6
2.4 Android Permission Types . . . . .	7
2.4.1 Normal Permissions . . . . .	7
2.4.2 Signature Permissions . . . . .	8
2.4.3 Dangerous Permissions . . . . .	9
2.5 Android Permission Groups . . . . .	9
2.6 SDK Level . . . . .	10
2.6.1 Android:minSdkVersion . . . . .	11
2.6.2 Android:maxSdkVersion . . . . .	11
2.6.3 Android:TargetSdkVersion . . . . .	11

2.7	Android API	12
<b>3</b>	<b>Project Implementation</b>	<b>13</b>
3.1	Task Overview	13
3.2	Manual Analysis of Applications	14
3.2.1	Steps Involved	14
3.2.2	Observations and Challenges	15
3.2.3	Example of Applications' Manual Analysis	16
3.2.3.1	WhatsApp Messenger	17
3.2.3.2	Instagram Application	18
3.3	Creation of Database	18
3.3.1	Schema of the Database	19
3.3.2	Adding Entries to the Database	19
3.3.3	Observations and Challenges	20
3.4	Using the Database to Retrieve Permissions	21
3.4.1	Using Nouns	21
3.4.2	Using Verbs	21
3.4.3	Using Nouns and Verbs	22
3.4.4	Comparison of the Approaches	23
3.4.4.1	Comparing Permissions from Manual Assessment with Noun Approach	23
3.4.4.2	Comparing Permissions from Manual Assessment with Verb Approach	25
3.4.4.3	Comparing Permissions from Manual Assessment with Noun and Verb Approach	26
3.5	Using the Manifest to Retrieve Permissions	26
3.5.1	Steps Involved	27
3.5.2	Observations and Challenges	28
3.6	API to Permission Mapping	28
3.6.1	Steps Involved	29
3.6.2	Observations and Challenges	29
<b>4</b>	<b>Results and Conclusions</b>	<b>31</b>
4.1	Results	31
4.2	Conclusions	36
4.2.1	Inadequacy of Application Descriptions	36
4.2.2	Over-declaration of Permissions in the Manifest File	36
4.2.3	Storage and Phone Permissions Predominate Others	37
4.2.4	Users lack basic knowledge of the Android Permission Model	37
4.3	Future Scope	37
<b>5</b>	<b>Appendix A: Playscraper</b>	<b>38</b>
<b>6</b>	<b>Appendix B: Androguard</b>	<b>39</b>

---

6.1	Features . . . . .	39
6.2	Retrieve Manifest file of an Android application from its APK File . . . . .	40
6.3	Retrieve classes and methods from the Dex File . . . . .	40
6.4	Retrieve the SDK version from the Manifest File . . . . .	41
6.5	Retrieve API-Permission Mapping using SDK Version . . . . .	41

<b>Bibliography</b>	<b>42</b>
---------------------	-----------



# List of Figures

2.1	Screenshot of an unzipped APK File . . . . .	6
2.2	Association between Permission Groups, Permissions and API Calls . . . . .	9
3.1	Description of Fruit Puzzle - Link Line . . . . .	16
3.2	Schema of the Database . . . . .	19
3.3	Database Entries . . . . .	19
3.4	Python script written to add the stemmed form of the nouns . . . . .	20
3.5	Permission Groups were retrieved using Nouns . . . . .	21
3.6	Permission Groups retrieved using Verbs only . . . . .	22
3.7	Permission Groups retrieved using Nouns and Verbs . . . . .	23
3.8	Screenshot of an unzipped APK File . . . . .	27

# List of Tables

2.1	Permissions and their corresponding Permission Groups . . . . .	10
3.1	Manual Analysis of Whatsapp . . . . .	17
3.2	Manual Analysis of Instagram . . . . .	18
3.3	Comparison of Permissions retrieved from Manual Assessment with Noun Approach for Whatsapp and Zomato . . . . .	24
3.4	Comparison of Permissions retrieved from Manual Assessment with Verb Approach for Whatsapp and Zomato . . . . .	25
3.5	Comparison of Permissions retrieved from Manual Assessment with Noun and Verb Approach for Whatsapp and Zomato . . . . .	26
4.1	Overview of all comparisons done . . . . .	35

# Chapter 1

## Introduction

Android Applications find various uses in our daily lives. Android users can face the risk of downloading and installing bad applications on their device. In fact, several applications might either hide malware, or their expected behaviour does not completely follow the expectations of the user. Generally, the cause of this is how during install time, the user often skips the alert message when he or she is warned of a potential security threat of the application. The complexity of the permission system of Android is to blame here, which may be difficult to grasp. Through this dissertation, we propose an evaluation of Android Applications based on multiple criteria, to help the user to easily gauge the trustworthiness of applications. We validate the proposed approach by testing it on more than 120 applications. This Chapter outlines the motivation (section 1.1) and the objective of this research (section 1.2). In the final section, Thesis Organization (section 1.3), we give the outline of the remaining chapters in this thesis.

### 1.1 Motivation

The world is shrinking with the growth of mobile phone technology. As the number of users are increasing day by day, facilities of communication are as well. We began with simple handsets which were merely used for making calls, but now, mobiles have changed our lives and become an integral part of it. Now they are not used just for making calls but they also find countless number of applications. They can be used as a Camera, Music player, Navigation System, T.V., etc. With the emergence of new technologies, there is a need for a new software and operating systems as well. Android is one such powerful Operating System supporting a large number of applications. These applications make life

comfortable and provide several uses to its users. Android comes with a market which is an online software store that allows its users to select, and download and use applications developed by third party developers. It was developed by Google and launched on 23rd September 2008. Android applications at present make up for over 85% of the market share offering, and Android application developers have flexibility in design, function, and hardware compatibility, while helping enhance user experience by providing users everywhere with a platform that is engaging, easy to use, and with close to limitless functionality. But even with its many advantages, Android app development services are not free from cybersecurity threats, which served as our main motivators for the project:

1. Hackers are lurking wherever there is an upsurge in digital activity. They are constantly looking to pry into personal and sensitive information of users everywhere.
2. Malicious Applications pose an undeniable threat to the Android Environment. Developers of such applications can over-declare the permissions needed by the application for its functionality. Security breaches of this nature can prove to be fatal for users and the Android Environment alike.
3. Unsuspecting Users are largely unaware of the Android permission model and lack adequate knowledge needed to gauge an application and its permission requests. Such users often grant all permissions that an Android applications requests, unknowingly handing over all information needed by an exploiter on a silver platter.
4. Unsuspecting Users might also take the description of an Application on Google Play Store at its face value, without delving deeper into the intentions of an application.

## 1.2 Objectives

As discussed in the above section, permission model related security concerns served as our main motivators for the project. The vagueness of the permission model and the inexperience of the general user together pose an even greater threat. It is therefore evident as to what our analysis aims at doing:

1. Identifying whether application descriptions are enough to provide insight into the permission requirements of an application. In other words, understanding whether a general user would be able to gauge enough from app descriptions.

2. Determining whether the manifest file of a given application covers extra permissions when compared to those retrieved from the description.
3. Checking whether the declared permissions have corresponding API calls within the source code of the application. This ensures and checks proper utilization of permissions.
4. Identifying whether there are certain permission groups that are being over-declared frequently, and why that might be the case.

### 1.3 Thesis Organisation

The thesis is separated in chapters. Chapter 2 gives an overview of the Android Environment. Chapter 3 gives a detailed account of our work. It entails all tasks related to the assessment of application descriptions, and also those related to application permissions and their corresponding APIs. Chapter 4 serves as a conclusion of the project. Following the Conclusion is a list of references and a Bibliography. Additional works have been added in the Appendix section, which is at the end of the thesis.

## Chapter 2

# Android Survey

The survey consists of all the topics, pertaining to Android, that were looked into to get a thorough understanding of the different approaches that shall be discussed in the upcoming chapters. The observations made regarding the advantages and disadvantages of different methods and comparing them with our use case provided us the key decision making points to select the best approach.

It is well known that the Android Operating System is an open source operating system that is mainly utilized in mobile and cellular devices. The system has been majorly written in Java and is based on Linux. It was initially developed by Android Inc. and was subsequently purchased by Google in 2005.

We will now discuss some of the topics, pertaining to Android, that served as building blocks for our project.

### 2.1 Android Playstore

Google Play Store is an online shop for Android applications, games, music, e-books, movies, and more. The Play Store app gives access to all such facilities on any Android device. Standard apps appear in the Android system tray.

All Android Applications that are present on Play Store consist of all data relevant to the application, like, its name, developer, category, and most importantly – description.

All Android Applications need to follow certain Google Play Store guidelines:

1. Length of Application Title should be 50 characters long.
2. The Short Description should be 80 characters long.
3. The Full Description should be 4000 characters long.

Potential users can benefit from these descriptions to learn what the app does, its features, benefits, costs (if any), and much more. Google Play also uses these app descriptions to rank applications based on their content, their target age group, their features, and more.

## 2.2 APK File

A file with the APK file extension is an Android Package file that's used to distribute applications on Google's Android operating system. APK files are saved in a ZIP format and are generally downloaded directly onto Android devices.

Some of the content found in an APK file is:

1. `AndroidManifest.xml`: An additional Android manifest file, describing the name, version, access rights, referenced library files for the application. This file may be in Android binary XML that can be converted into human-readable plaintext XML with tools such as `AXMLPrinter2`, `apktool`, or `Androguard`.
2. `classes.dex`: The classes compiled in the dex file format understandable by the Dalvik virtual machine and by the Android Runtime.
3. `Resources.arsc`: a file containing precompiled resources, such as binary XML for example.

In order to use an APK file, simply locate the APK file you want to see and click it. APK files are stored in a .ZIP file, which compresses all the information within the APK file into a single file. This is used to save storage on your device. By clicking, or unzipping, the file, you'll be able to view the contents within the file. Note, however, that you may have to rename the file ".zip" before opening. Alternatively, you can also open the file through the Zip application's "open dialogue" box.

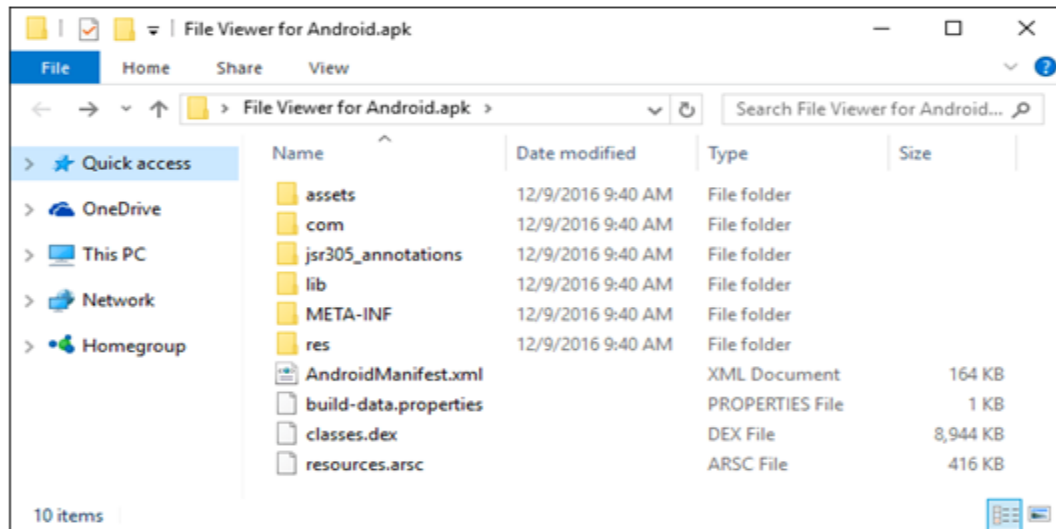


FIGURE 2.1: Screenshot of an unzipped APK File

## 2.3 Android Manifest File

Manifest file for an android application is a resource file which contains all the details needed by the android system about the application. It is an integral part of the application and works as a connection between the developer and the platform. It helps the developer in passing on the functionality and requirements of his/her application to Android. Some rules need to be followed, one of which is that an xml file must be named as `AndroidManifest.xml` and placed at the application root. Every Android application should have an `AndroidManifest.xml` file. `AndroidManifest.xml` allows us to define the packages, API, libraries needed for the application. It consists of:

1. The foundation of an application like activities, services, and more.
2. Details about permissions.
3. Set of classes needed before launch.

### 2.3.1 Elements of the Manifest File

Following are some of the elements that can appear in `AndroidManifest.xml`:

1. `uses-permission`: used to specify permissions that are requested for the purpose of security.



2. Permission: used to set permissions to provide access control for some specific component of the application.
3. permission-group: does the same as above for a set of components.
4. permission-tree: refer one specific name of the component which is the owner or parent of the set of component.
5. Instrumentation: enables to know interaction between Android system and application.
6. uses-sdk: specifies the platform compatibility of the application.

## 2.4 Android Permission Types

The purpose of permission is to protect the privacy of an Android user. Android applications are required to request permission to access sensitive user data (such as contacts and SMS), and certain system features (such as camera and internet). The system might grant the permissions automatically or might prompt the user to approve the request. This decision depends on the feature that needs to be accessed.

A major design area of the Android security architecture is that no app, by default, has permission to perform any operations that would negatively impact other applications, the operating system, or the user. This includes reading or writing the user's private data (such as contacts or emails), reading or writing another app's files, performing network access, keeping the device awake, and so on.

Permissions are categorized into multiple protection levels. The protection level affects whether runtime permission requests are required. There are three protection levels that affect third-party apps: normal, signature, and dangerous permissions

### 2.4.1 Normal Permissions

Normal permissions focus on areas where application needs to access data or resources outside the its sandbox, but where there's limited risk to the user's privacy, or the operation of other applications. For example, the permission to set time zone is considered a normal permission. If an applications declares in its manifest that it requires a normal permission, the system automatically grants the app that permission at the time of installation. The

system does not send a prompt to the user to grant normal permissions, and users cannot revoke such permissions.

As of Android 9 (API level 28), some of the permissions are classified as PROTECTION\_NORMAL:

- ACCESS\_LOCATION\_EXTRA\_COMMANDS
- ACCESS\_NETWORK\_STATE
- ACCESS\_NOTIFICATION\_POLICY
- ACCESS\_WIFI\_STATE
- BLUETOOTH
- BLUETOOTH\_ADMIN
- BROADCAST\_STICKY
- CHANGE\_NETWORK\_STATE

#### 2.4.2 Signature Permissions

The system grants these app permissions at install time, but only when the app that attempts to use permission is signed by the same certificate as the app that defines the permission.

As of Android 8.1 (API level 27), the following permissions that third-party apps can use are classified as PROTECTION\_SIGNATURE:

- BIND\_ACCESSIBILITY\_SERVICE
- BIND\_AUTOFILL\_SERVICE
- BIND\_CARRIER\_SERVICES
- BIND\_CHOOSER\_TARGET\_SERVICE
- BIND\_CONDITION\_PROVIDER\_SERVICE
- BIND\_DEVICE\_ADMIN
- BIND\_DREAM\_SERVICE

### 2.4.3 Dangerous Permissions

Dangerous permissions focus on areas where the applications requires data or resources that directly involve the user's private data, or could potentially affect the user's stored data or perhaps even the operation of other applications. For example, the ability to read the user's contact information is classified as a dangerous permission. If an applications declares that it requires a dangerous permission, the user is required to explicitly grant the permission to the application. Until the permission is approved by the user, the application cannot provide any functionality that depends on the permission in question. To utilize any dangerous permission, the application must send a prompt to the user to grant permission at runtime. Some dangerous permission is:

- WRITE\_EXTERNAL\_STORAGE
- RECORD\_AUDIO
- ACCESS\_FINE\_LOCATION

## 2.5 Android Permission Groups

Permissions are classified into groups depending on a device's capabilities or features. Under this, permission requests are handled at the group level and a single permission group corresponds to several permission declarations in the app manifest. For example, the SMS group includes both the READ\_SMS and the RECEIVE\_SMS declarations.

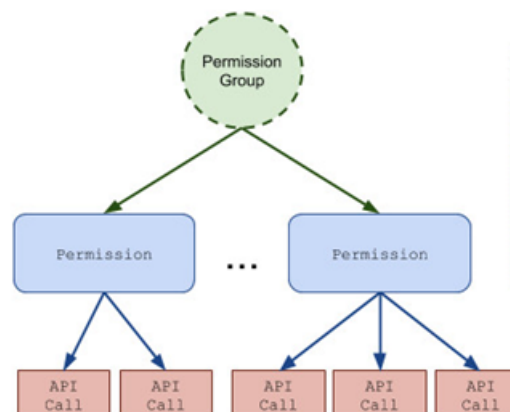


FIGURE 2.2: Association between Permission Groups, Permissions and API Calls

Permission Group	Permission
CALENDER	READ_CALENDER WRITE_CALENDER
CALL_LOG	READ_CALL_LOG WRITE_CALL_LOG PROCESS_OUTGOING_CALLS
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
PHONE	READ_PHONE_STATE READ_PHONE_NUMBER CALL_PHONE ANSWER_PHONE_CALLS ADD_VOICE_MAIL USE_SIP
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

TABLE 2.1: Permissions and their corresponding Permission Groups

Related API Calls and access to object properties are associated with a specific permission request. And related permission requests are rolled up into a Permissions Group. All dangerous Android permissions belong to permission groups. A permission's group only affects the user experience if the permission is dangerous.

## 2.6 SDK Level

Google Play uses the "uses-sdk" attribute declared in the application's manifest file to filter an application from devices that do not meet its platform version requirements.

1. It allows developers to express an application's compatibility with one or more versions of the Android platform, using an API Level integer. The API Level that is expressed by an application will be compared with the API Level of a given Android system, which may vary for Android devices.
2. Despite its name, this element is used to specify the API Level, and not the version number of the SDK (software development kit) or Android platform. The API Level is always a single integer. You cannot derive the API Level from its associated Android version number (for example, it is not the same as the major version or the sum of the major and minor versions).

Major attributes of an SDK level are discussed further.

### **2.6.1   Android:minSdkVersion**

An integer designating the minimum API Level required for the application to run. The Android system will prevent the user from installing the application if the system's API Level is lower than the value specified in this attribute. Developers are required to always declare this attribute.

### **2.6.2   Android:maxSdkVersion**

An integer designating the maximum API Level on which the application is designed to run. In Android 1.5, 1.6, 2.0, and 2.0.1, the system checks the value of this attribute when installing an application and when re-validating the application after a system update. In either case, if the application's maxSdkVersion attribute is lower than the API Level used by the system itself, then the system will not allow the application to be installed. In the case of re-validation after system update, this effectively removes your application from the device.

### **2.6.3   Android:TargetSdkVersion**

An integer designating the API Level that the application targets. If not set, the default value equals that given to minSdkVersion. This attribute informs the system that you have tested against the target version and the system should not enable any compatibility behaviours to maintain your app's forward-compatibility with the target version. The application is still able to run on older versions (down to minSdkVersion).

## 2.7 Android API

The Android Platform provides a framework API that applications can use to interact with the underlying Android system.

The framework API consists of:

1. A core set of packages and classes
2. A set of XML elements and attributes for declaring a manifest file
3. A set of XML elements and attributes for declaring and accessing resources
4. A set of intents
5. A set of permissions that applications can request

Each new version of the Android platform includes updates to the Android application framework API that it delivers. Updates to the framework API are designed in such a way that the new API remains compatible with previous versions of the API. As parts of the API are upgraded, the older parts that were replaced are deprecated but are not removed. This is done so that the existing applications can still use them.

The framework API that an Android platform delivers is specified using an integer identifier called API Level. Each Android platform supports exactly one API Level.

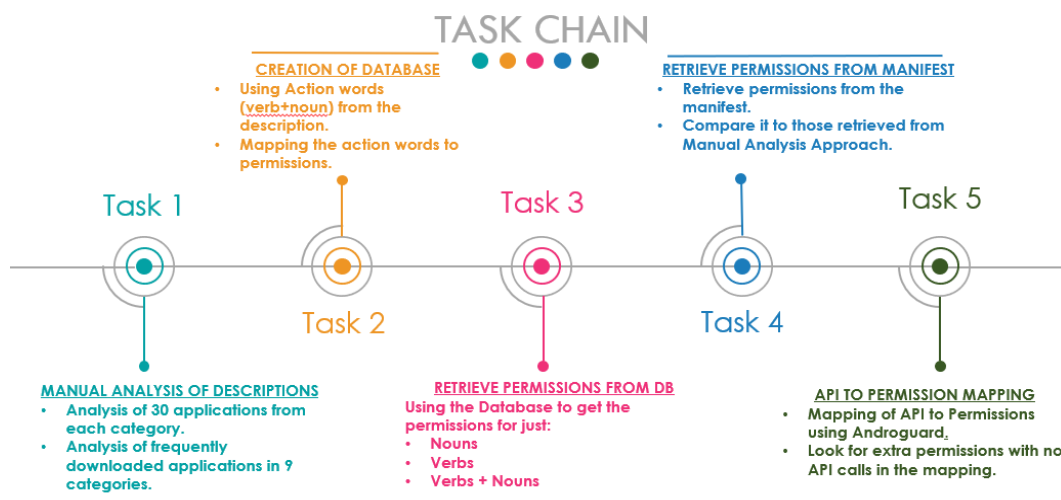
## Chapter 3

# Project Implementation

The project was implemented in five tasks which involved scouring through Application Descriptions, Manifest File and the API Calls. The following chapter consists of all the topics that were investigated to get a thorough understanding of the different approaches present. The shortcomings and observations made for each task led to the successive task.

### 3.1 Task Overview

The five tasks involved were: Manual Analysis of Applications, Creation of a Database, Retrieve Permissions from the Database, Retrieve Permissions from the Manifest, and API to Permission Mapping.



## 3.2 Manual Analysis of Applications

Around 400 Applications were manually analysed (using pen-paper approach). Their action words (verbs and nouns) and their corresponding permissions were noted. In order to improvise the set of data for a better analysis, 30 applications from each category, 90 frequently downloaded applications and 30 applications from different regions were also analysed. The different categories that were analysed were:

1. Travel
2. Tools
3. Communication
4. Photography
5. Music and Audio
6. Games
7. Education
8. Entertainment
9. Food and Drink

### 3.2.1 Steps Involved

The manual analysis began by accumulating the names of various applications. This was done in accordance to their descriptions and whether they belonged to the desired categories. The process followed from hereon can be described as follows:

1. Using a pen and paper approach, the descriptions of the applications were analysed.
2. The analysis involved identifying important action words. Action words, in this context, were words that described what the application intended to do.
3. For this scenario, the action words were noun and verb phrases.
4. These nouns and verbs, once identified, were mapped to permission groups that the application might need, if any.
5. Observations were made accordingly to address the challenges faced in this approach.



### 3.2.2 Observations and Challenges

Once the process was completed and the tables were made manually, a series of observations were made:

1. Synonyms were found for frequently occurring terms.

For example: ‘Photos’ could have ‘Pictures’, ‘Pic’, ‘Photograph’, ‘selfie’, ‘QR’, ‘snap’, ‘shot’, etc. as its synonyms. Hence, to counter this, we analysed a wide set of applications to include all the possible synonyms that convey the same meaning.

2. Some words may be nouns, as well as verbs.

For example: ‘call’: making a call (noun), and calling (verb) a person.

3. Some application descriptions don’t give anything.

For example: Google+

4. Many applications, most often gaming applications, do not have permission related descriptions. Hence, this makes it difficult to retrieve the permission groups from the description.

5. Some applications have the required permissions in the title and/or explicitly mentioned in the description.

For example: various permissions like ‘call logs’, ‘phone’, and ‘contacts’, can be extracted from the application title “Called ID, Calls, Phone book Contacts”. Flipkart has permissions explicitly mentioned in the description. Hence, to counter this, permission groups were searched directly in the title and description. If it existed, we considered it before analysing the description.

6. Certain action words may map to multiple permissions ambiguously.

For example: ‘Edit photos’, may need camera, or storage, or both.

7. Pronouns posed an issue, when encountered in the description because of their ambiguity.

For example: In Figure 3.1: The action phrase, ‘Download it’, implies that we could download the application and it would require the permission groups of internet and storage. This sort of intelligence was needed in our program as well.

8. Bogus phrases that were out of context and were not meant to be considered were hard to tackle.

For Example: The sentence, “download the ZEE5 application for a good experience”, does not mean that it needs storage permission.

Different link style, lovely fruits, interesting levels, all you want is in this new link splash game with wonderful fruits beat!

Join us now in this link and match puzzle game. Walk through the mushrooms and weeds, irrigate the flowers to make an unforgettable blossom, dig colorful diamonds, and also fill fruits into baskets, or free the caged birds. But Be Careful of the monster, it will destroy your beautiful garden. Use fruits nearby to beat them.

Even you can help the little hungry pet to get more ice-creams, but take care of the movable monsters or ghost, it will swallow the beautiful fruits as them can.

\* We can offer you \*

Plenty of Levels

Funny Goals

Smooth Control

Continuous Update

[Download it](#) and enjoy the game RIGHT NOW! No internet required, no experience required!

FIGURE 3.1: Description of Fruit Puzzle - Link Line

9. Negations had to be kept in mind.

For example: The sentence, “without needing internet or GPS”, means that it does not need internet and location permissions.

10. Certain permissions that an application would definitely need have not been mentioned, implicitly or explicitly, in the description. Therefore, there was need to analyse the manifest file, too.

For Example: Only permission groups like camera and storage were extracted from the description of Snapchat, but it is obvious that it requires permissions like access to contact list.

### 3.2.3 Example of Applications’ Manual Analysis

Using the pen and paper approach discussed earlier, the following analysis of WhatsApp Messenger and Instagram was done. The following sections discuss their results.

**3.2.3.1 WhatsApp Messenger**

Verb	Noun	Permission Groups
Messaging	App	Internet Phone
Uses	Phone's Internet Connection	
Let Message And Call	Friends And Families	Contacts
Switch	Sms	
Send And Receive	Messages, Calls, Photos, Videos, Documents, Voice Messages Storage, Microphone, Camera	
Use	Whatsapp	
Pay	Message Or Call	
Enjoy	Group Chats/Contacts	
Stay In Touch	Friends And Families	
Send And Receive	Whatsapp Messages/Computer's Browser	
Chat	Friends	
Avoid	International Sms Charges	
Say	Username And Pins	
Bother	Have	
Remember	Username And Pins	
Works	Whatsapp	
Works	Phone Number	Contacts, Phone
Integrates	Phone's Existing Phone Book	Contacts, Phone, Call Logs
Miss	Messages	Contacts, Phone, Calls Logs
Use	Address Book	
Connect	You/Your Contacts	Contacts
Need To Add	Username	
Miss	Notifications	

TABLE 3.1: Manual Analysis of Whatsapp

In table 3.1, Through this manually made table, we inferred that Whatsapp Messenger's description implies that it needs Internet, Phone, Storage, Microphone, Camera, Contacts, Call Log, and Location permission groups.

### 3.2.3.2 Instagram Application

Verb	Noun	Permission Groups
Is	Instagram	
Capture	World's Moments	Camera, Storage
Share	World's Moments	Camera, Storage
Follow	Friends And Family	Contacts
See	They	
Discover	Accounts	
Sharing	Things	
Love	You	
Join	Community	
Express	Yourself	
Sharing	Moments	Camera, Storage
Use	Instagram	
Post	Photos And Videos	Camera, Microphone, Storage
Want	You	
Keep	Profile Grid	

TABLE 3.2: Manual Analysis of Instagram

In Table 3.2, Through this manually made table, we inferred that Instagram's description implies that it needs Camera, Storage, Contacts, and Microphone permission groups.

## 3.3 Creation of Database

The next step after Manual Analysis was to create a Database using the data obtained from Manual Analysis. In order to store the data, we used a Structured Query Language (SQL) Database, as it provides well defined standards, easier portability and simplicity.

It has the following entries: Serial Number, Verb, Noun, Permissions, Stemmed Verb, and Stemmed Noun. The Serial Number was declared as Unique and Auto-Incremental.

### 3.3.1 Schema of the Database

Since the database was meant to be nothing but a rendition of the manual assessment, its schema was similar to that of the manual assessment. However, two additional columns were introduced, namely ‘Stemmed Nouns’ and ‘Stemmed Verbs’. This was done to establish uniformity of all the words in the database.

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
serial_no	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
verb	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
noun	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
permissions	VARCHAR(200)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
stemmed_noun	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
stemmed_verb	VARCHAR(200)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL

FIGURE 3.2: Schema of the Database

Fig 3.2 shows the schema of the database made. All attributes, except serial\_no, were declared to be of varchar data type with a maximum character limit of 200 characters, to accommodate bigger terms as well. Verb, noun, stemmed\_verb, and stemmed\_noun could also have NULL values, since sometimes merely the noun or the verb can convey which permission group is needed. However, keeping the same logic in mind, permissions were declared to not have NULL values.

### 3.3.2 Adding Entries to the Database

Once the schema of the Database was declared, the entire data obtained from the Manual Analysis Task was added into the database one by one. Only those entries that had a permission group were added. A python script was used to update the Stemmed Verb and Stemmed Noun for its corresponding Verbs and Nouns.

serial_no	verb	noun	permissions	stemmed_noun	stemmed_verb
24	customize	photo	camera,storage	photo	custom
25	sync	contacts	contacts	contact	synt
26	customize	location	location	loc	custom
27	enjoy	messaging	sms,contacts	mess	enjoy

FIGURE 3.3: Database Entries

With over 1300+ entries in the SQL Database, it could now be used to retrieve the Permission Groups for its corresponding Verbs and Nouns.

### 3.3.3 Observations and Challenges

Once the entries were made into the database, observations and challenges faced in this task were noted. They were as follows:

1. The form of the action words in the database and of those retrieved from the description were sometimes found to differ. For example, “Photo” and “Photos” convey the same meaning, but do not have the same form. However, if both the words are stemmed before comparing, it would result in a better comparison. Therefore, a python script was written to automatically update the Stemmed form of the word for each Noun and Verb, using the Lancaster Stemmer for precise comparisons.

```
#To fetch all the nouns from the Database
my_cursor.execute("select noun from testdb.semantic_analysis;")

noun_list=list(my_cursor.fetchall())

#Stemming the noun and updating it.
for token in noun_list:
    stemmed_noun=lanaster_stemmer.stem(str(token[0]))
    noun=str(token[0])
    my_cursor.execute("update testdb.semantic_analysis SET stemmed_noun='%s' WHERE noun='%s';" %(stemmed_noun,noun))
mydb.commit()
```

FIGURE 3.4: Python script written to add the stemmed form of the nouns

Figure 3.4 is a snapshot of the Python script used to add the stemmed form of the nouns for all the database entries. Similarly, a Python script was written to add the stemmed form of the verbs as well.

2. Certain action words were mapped to multiple permissions. Hence, the permission column was declared as a list of permissions and had to be split to access each permission.

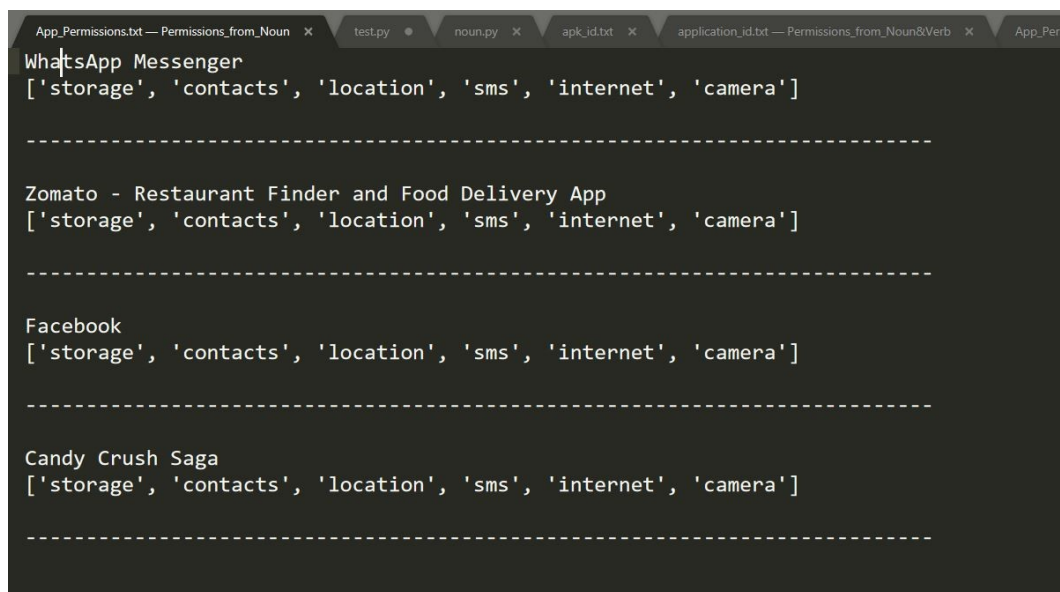
For example, for the action words “enjoy messaging”, the list of permissions required were, “SMS, Contacts”. Therefore, to access each permission, i.e. “SMS” and “Contacts”, it had to be separated.

## 3.4 Using the Database to Retrieve Permissions

Once the database was created, it was used to retrieve permissions for corresponding Verbs and Nouns. Three different approaches were used to retrieve the permission groups from the Database.

### 3.4.1 Using Nouns

In this approach, the Nouns were retrieved from the description and were stemmed using Lancaster Stemmer. The Stemmed Nouns thus retrieved were checked with the Stemmed Nouns in the database, and the corresponding permission groups were thus noted. However, this approach had a low accuracy as it gave many additional results. Thus, a different approach of using only verbs was considered.



```
App_Permissions.txt — Permissions_from_Noun x test.py x noun.py x apk_id.txt x application_id.txt — Permissions_from_Noun&Verb x App_Perm
WhatsApp Messenger
['storage', 'contacts', 'location', 'sms', 'internet', 'camera']
-----
Zomato - Restaurant Finder and Food Delivery App
['storage', 'contacts', 'location', 'sms', 'internet', 'camera']
-----
Facebook
['storage', 'contacts', 'location', 'sms', 'internet', 'camera']
-----
Candy Crush Saga
['storage', 'contacts', 'location', 'sms', 'internet', 'camera']
-----
```

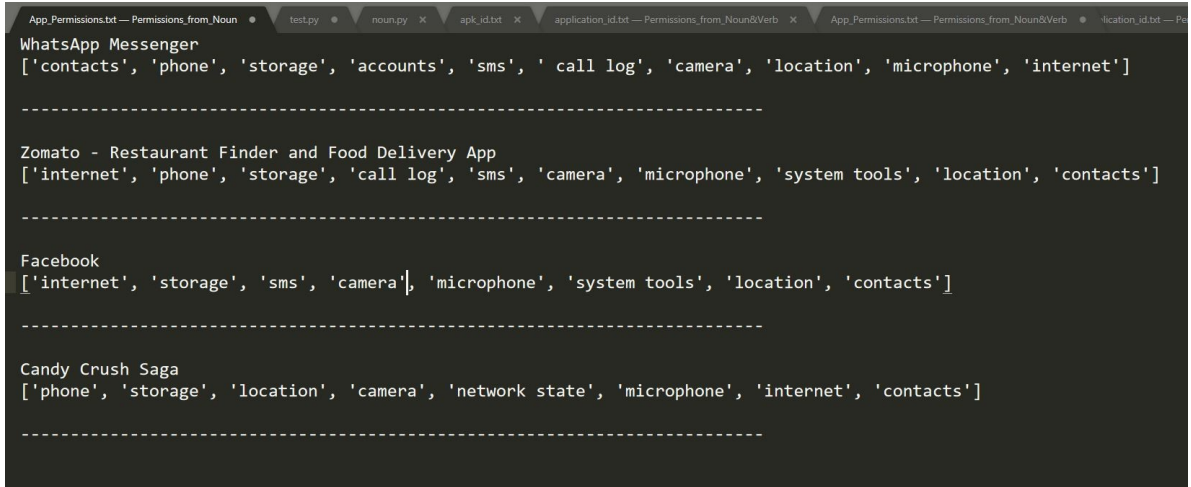
FIGURE 3.5: Permission Groups were retrieved using Nouns

Figure 3.5 illustrates the set of Permission Groups retrieved by following only Nouns approach.

### 3.4.2 Using Verbs

In this approach, the Verbs were retrieved from the description and were stemmed using Lancaster Stemmer. The Stemmed Verbs thus retrieved were checked with the Stemmed

Verbs in the database, and the corresponding permission groups were thus noted. However, even this approach proved to have lower accuracy.



```

App_Permissions.txt — Permissions_from_Noun • test.py • noun.py • apk_id.txt • application_id.txt — Permissions_from_Noun&Verb • App_Permissions.txt — Permissions_from_Noun&Verb • application_id.txt — Pe
WhatsApp Messenger
['contacts', 'phone', 'storage', 'accounts', 'sms', ' call log', 'camera', 'location', 'microphone', 'internet']
-----

Zomato - Restaurant Finder and Food Delivery App
['internet', 'phone', 'storage', 'call log', 'sms', 'camera', 'microphone', 'system tools', 'location', 'contacts']
-----

Facebook
['internet', 'storage', 'sms', 'camera', 'microphone', 'system tools', 'location', 'contacts']
-----

Candy Crush Saga
['phone', 'storage', 'location', 'camera', 'network state', 'microphone', 'internet', 'contacts']
-----

```

FIGURE 3.6: Permission Groups retrieved using Verbs only

Figure 3.6 illustrates the set of Permission Groups retrieved by following only Verbs approach

### 3.4.3 Using Nouns and Verbs

The entire process of retrieving permissions using both Verbs and Nouns can be subdivided into several steps:

1. The application description is retrieved using a library named Play-Scraper.
2. The description was sentence tokenized using NLTK's sentence tokenizer
3. Each word in the sentence would be assigned a POS (Part of Speech) Tag to identify its word form. The POS tag was identified using a library named Spacy. Spacy is a free, open-source library for advanced Natural Language Processing (NLP) in Python, with an accuracy of 92.6.
4. Only the Verbs and Nouns in a sentence were filtered out and stemmed. These Stemmed Verbs and Stemmed Nouns were compared with the Stemmed Verbs and Stemmed Nouns from the database, and its corresponding permission groups were noted for further assessment.



```

1 WhatsApp Messenger
2 ['CALL LOG', 'INTERNET', 'CAMERA', 'LOCATION', 'SMS', 'MICROPHONE', 'PHONE', 'CONTACTS', 'STORAGE']
3
4 -----
5
6 Zomato - Restaurant Finder and Food Delivery App
7 ['CALL LOG', 'INTERNET', 'CAMERA', 'LOCATION', 'SMS', 'MICROPHONE', 'PHONE', 'CONTACTS', 'STORAGE']
8
9 -----
10
11 Facebook
12 ['CALL LOG', 'INTERNET', 'CAMERA', 'LOCATION', 'SMS', 'MICROPHONE', 'PHONE', 'CONTACTS', 'STORAGE']
13
14 -----
15
16 Candy Crush Saga
17 ['CALL LOG', 'INTERNET', 'CAMERA', 'LOCATION', 'NETWORK STATE', 'SMS', 'MICROPHONE', 'PHONE', 'CONTACTS', 'STORAGE']
18
19 -----
20
21

```

FIGURE 3.7: Permission Groups retrieved using Nouns and Verbs

Thus, the above-mentioned approach yields better results than the only Nouns and only Verbs approach, as it retrieves only relevant results. It was noted that the accuracy of the result was only much better than the previous two approaches. Hence, this approach was taken into consideration and the other two approaches were discarded.

### 3.4.4 Comparison of the Approaches

The following sections compare the permissions retrieved from the Manual Assessment with those retrieved from the three approaches discussed above. Through the results generated, we also analysed the best approach among the three.

#### 3.4.4.1 Comparing Permissions from Manual Assessment with Noun Approach

Two applications, WhatsApp Messenger and Zomato, were used to make the comparison. The following is an overview of the comparison.

Application Name	Manual Assessment Results	Noun Approach Results
WhatsApp	Internet Phone Storage Microphone Camera Contacts Call Logs Location	Internet - Storage - Camera Contacts Call Logs Location SMS
Zomato	Location Internet Camera Contacts Phone SMS Storage Phone State	Location Internet Camera Contacts - SMS Storage -

TABLE 3.3: Comparison of Permissions retrieved from Manual Assessment with Noun Approach for Whatsapp and Zomato

From the Table 3.3 given above, we infer that the Noun Approach also declares SMS as a permission group needed for Whatsapp, which is not true as seen in the results for the manual assessment.

For Zomato, the Noun Approach failed to detect a few permission groups.

Therefore, we moved onto the Verb Approach.

**3.4.4.2 Comparing Permissions from Manual Assessment with Verb Approach**

Application Name	Manual Assessment Results	Verb Approach Results
WhatsApp	Internet Phone Storage Microphone Camera Contacts Call Logs Location	Internet Phone Storage Microphone - Contacts Call Logs Location SMS Accounts
Zomato	Location Internet Camera Contacts Phone SMS Storage Phone State	Location Internet Camera Contacts Phone SMS Storage - Call Log System Tools Microphone

TABLE 3.4: Comparison of Permissions retrieved from Manual Assessment with Verb Approach for Whatsapp and Zomato

From the Table 3.4 given above, we infer that the Verb approach is resulting in extra permission groups that were not seen in the Manual Assessment results. This called for an even better approach, which was that of combining Nouns and Verbs together.

### 3.4.4.3 Comparing Permissions from Manual Assessment with Noun and Verb Approach

Application Name	Manual Assessment Results	Verb Approach Results
WhatsApp	Internet	Internet
	Phone	Phone
	Storage	Storage
	Microphone	Microphone
	Camera	Camera
	Contacts	Contacts
	Call Logs	Call Logs
	Location	Location
Zomato	Location	Location
	Internet	Internet
	Camera	Camera
	Contacts	Contacts
	Phone	Phone
	SMS	SMS
	Storage	Storage
	Phone State	Call Log

TABLE 3.5: Comparison of Permissions retrieved from Manual Assessment with Noun and Verb Approach for Whatsapp and Zomato

From Table 3.5 given above, we infer that the Noun and Verb approach rendered better results than the previous approaches. For Example, in WhatsApp, the previous approaches had failed to detect Camera as a permission group. Noun and Verb approach successfully detected it. This reiterates our decision of selecting Noun and Verb approach for analysing the descriptions for permission groups.

## 3.5 Using the Manifest to Retrieve Permissions

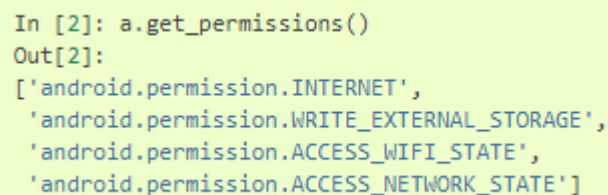
Every application must have an AndroidManifest.xml file. Manifest file describes essential information about the application to the Android Build Tools, the Android Operating System, and Google Play. The Manifest file is required to declare the permissions that the

application needs in order to access protected paths of the system or other applications. As discussed in the previous tasks, the description is often not enough to assess the permission groups required by an application. Hence, there was a need to delve deeper and look into the backend of the applications. This was done through the manifest files, which contains the permissions required by an application.

### 3.5.1 Steps Involved

This section discusses how the permissions were retrieved from the Manifest file and compared with the database result (which is the Noun and Verb approach to fetch permissions from the database). The following were the steps taken:

1. The permissions of an application are fetched using a tool called Androguard. It is a python-based tool, which can run on Linux/Windows/OSX, provided python is installed in the system. It is powerful tool which provides reverse engineering and malware analysis for Android, to disassemble and to decompile android apps. It can be used for the Static Analysis of an application using code obfuscators. The permissions are retrieved using the instruction “a.get\_permissions()” once the APK has been loaded into the function AnalyzeAPK.



```
In [2]: a.get_permissions()
Out[2]:
['android.permission.INTERNET',
 'android.permission.WRITE_EXTERNAL_STORAGE',
 'android.permission.ACCESS_WIFI_STATE',
 'android.permission.ACCESS_NETWORK_STATE']
```

FIGURE 3.8: Screenshot of an unzipped APK File

2. The permissions retrieved were classified into their corresponding permission groups. For example: “android.permission.READ\_EXTERNAL\_STORAGE” and “android.permission.WRITE\_EXTERNAL\_STORAGE” was classified under the permission group Storage
3. These permission groups were then compared with the ones retrieved from the database using the description of the application. The extra permission groups were noted.

4. However, there are a few permissions of Dangerous type that do not come under a permission group. These permissions were checked with the results of the database, and they were noted, if not found.

### 3.5.2 Observations and Challenges

The following observations were made for this task:

1. There is no mechanism of obtaining the APK file of the application through a code. Therefore, third party websites were used to download a set of APK files
2. Since, we retrieved a set of Permission Groups from the description, the permissions retrieved from the Manifest need to be also classified into a Permission group before comparison.

For example: “android.permission.READ\_EXTERNAL\_STORAGE” and “android.permission.WRITE\_EXTERNAL\_STORAGE” come under the permission group “Storage”.

3. Some dangerous permissions cannot be classified into a Permission Group. Hence, they were compared with the results of the database analysis, and flagged if not found.

For example: Results from Database: Internet, Phone, Storage, Microphone, Camera, Contacts, Call log, Location. Manifest Analysis: Bluetooth, android.permission.Read\_Profile, android.permission.NFC, android.permission.Manage\_Accounts, android.permission.Authenticate\_AuthenticationToken, android.permission.Change\_WIFI\_State, android.permission.USE\_CREDENTIALS. As we can see, the Permission Group ‘Bluetooth’ was not found in the results of the database for WhatsApp. Several other dangerous permissions were not found in the results.

## 3.6 API to Permission Mapping

The `androguard.core.api_specific_resources` package in `androguard` can be used to extract the mapping information between API calls and Android Permission corresponding to a specific SDK version. Androguard provides mapping information between API calls and Android Permissions for API Levels 16-25. This mapping can be used to check for Over Declared Permissions in the Manifest.

### 3.6.1 Steps Involved

In order to retrieve the over declared permissions in the manifest, the following steps were taken:

1. Only the mapping for the minimum SDK version declared in the manifest was retrieved. The minimum SDK version can be retrieved as follows

```
a.get_min_sdk_version()
```

2. Retrieve the mapping information between API calls and Android Permissions for the minimum SDK level. It can be done as follows

```
mapping = load_permission_mappings(a.get_min_sdk_version())
```

3. Retrieve the methods (API Calls) that an application makes. Both internal and external API calls are noted. This is done as follows

```
for c in dx.get_classes():
```

```
    methods=(list(map(lambda x: x.name, c.get_methods()))))
```

4. Remove the API calls and the permissions from the Mapping, that the application does not require, by comparing the API call in the Mapping to the methods retrieved above. This is done as to cut computation time and provide results quickly.
5. The Permissions in the Manifest are thus compared with the updated Mapping. Those permissions that are not found in the Updated Mapping can thus be called as Over Declared Permissions.
6. The Over Declared Permissions are flagged.

### 3.6.2 Observations and Challenges

The following observations were made during this task:

1. Several tools like Pscout, Axplorer etc. provided an outdated API to Permission Mapping. Hence, a tool named as Androguard was used as it provided API to Permission Mapping for SDK levels 16-25.

2. Every application is developed for a particular minimum SDK version. Hence, the API to Permission Mapping must be considered in accordance with the minimum SDK version specified in the Manifest.

For example: Whatsapp specifies the minimum sdk version as 16 in its manifest. Therefore, the Permission to API Mapping must be considered for the SDK version 16 only, and not some other SDK Version.

3. Mappings of many SDK Versions (or API Levels) lacked some critical permissions. For example, Androguard did not have API mappings for some of the dangerous permissions such as:

- (a) `android.permission.READ_CALL_LOG`
- (b) `android.permission.RECORD_AUDIO`
- (c) `android.permission.ACCESS_COARSE_LOCATION`
- (d) `android.permission.WRITE_EXTERNAL_STORAGE`

This therefore made it difficult to check for over declared permissions in the manifest due to the incorrect mapping. In order to counter this, the Androguard mapping was improvised for each API Level, and on an average of 500 extra API calls were added for each API level. This was done in accordance with the Android Developer Forum.

4. Several API calls were deprecated in newer API Levels. Hence, they needed to be removed from the newer API levels as a result.



## Chapter 4

# Results and Conclusions

In the final chapter, we would like to shed light upon the results derived at various stages. These results lead to important conclusions that will be discussed also be discussed in this chapter.

### 4.1 Results

The following table depicts the comparisons made between:

1. The permissions retrieved from the manifest file and those from the database.
2. The permissions retrieved from the manifest file and the API calls, in order to find the over-declared permissions in the manifest file.

<b>Application Name</b>	<b>Comparison of Manifest and Description</b>	<b>Over declared Permissions in the Manifest</b>
Fruit Ninja	Write External Storage	RECEIVE_BOOT_COMPLETED
Oman Air	Location (Fine and Coarse) Camera	SYSTEM_ALERT_WINDOW

	Read External Storage Write External Storage	
Fruit Puzzle	Read Logs Location (coarse and fine) Read External Storage Write External Storage Read Phone State	ACCESS_WIFI_STATE READ_LOGS GET_TASKS INSTALL_SHORTCUT  ACCESS_COARSE_LOCATION VIBRATE ACCESS_NETWORK_STATE READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE INTERNET READ_SETTINGS READ_PHONE_STATE ACCESS_FINE_LOCATION
Temple Run	Camera Internet Read Storage Write External Storage Read Phone State Flashlight Vibrate	RECEIVE_BOOT_COMPLETED BIND_ACCESSIBILITY_SERVICE, BIND_GET_INSTALL_REFERRER_SERVICES READ_SETTINGS  INSTALL_SHORTCUT REQUEST_INSTALL_PACKAGES GET_PACKAGE_SIZE MOUNT_UNMOUNT_FILESYSTEMS FLASHLIGHT
Hill Climb Racing	Read Phone State	ACTIVITY_RECOGNITION
Stickman Hook	Location  Vibrate	CHANGE_BADGE  READ READ_SETTINGS

		WRITE UPDATE_BADGE READ_GSERVICES WRITE_SETTINGS BADGE_COUNT_WRITE PROVIDER_INSERT_BADGE UPDATE_SHORTCUT UPDATE_COUNT BADGE_COUNT_READ READ_APP_BADGE BROADCAST_BADGE
Subway Surfers	Vibrate  Write External Storage	C2D_MESSAGE
Air India	System Alert Window Location (fine and coarse) Write External Storage Call Phone Read External Storage Read Phone State	NA
Dr Driving	NA	ACCESS_NETWORK_STATE BILLING INTERNET WAKELOCK READ_GSERVICES
CamScanner	Phone	CAMERA_ADDON ACCOUNT SYSTEM_UI_VISIBILITY_EXTENSION C2D_MESSAGE MOUNT_UNMOUNT_FILESYSTEMS SET_ALARM

FaceBook	Calendar Authenticate Accounts Manage Accounts Read Profile Phone Change WIFI State System Alert Window	ACCESS INSTALL_SHORTCUT  RECEIVE_BOOT_COMPLETED RECEIVE_ADM_MESSAGE READ_SYNC_SETTINGS WRITE WRITE_CALENDAR  WRITE_SYNC_SETTINGS GET_TASKS MANAGE_ACCOUNTS READ_EXTERNAL_STORAGE RECEIVE ACCESS READ C2D_MESSAGE SYSTEM_ALERT_WINDOW UNPROTECTED_API_ACCESS READ_GSERVICES CROSS_PROCESS_BROADCAST_MANAGER MODIFY_AUDIO_SETTINGS AUTHENTICATE_ACCOUNTS REQUEST_INSTALL_PACKAGES READ_CONTACTS READ_SETTINGS BROADCAST_BADGE CHANGE_NETWORK_STATE WRITE_CONTACTS DOWNLOAD_WITHOUT_NOTIFICATION ACTIVITY_RECOGNITION READ_CALENDAR UPDATE_SHORTCUT FB_APP_COMMUNICATION READ_PROFILE
----------	---	--

		BATTERY_STATS
Netflix	Change Wifi Multi-castState Phone Bluetooth	SET_FLAG_NOSOFTKEYS  CHANNEL_ID
Zomato	Phone Change WIFI State Use Credentials	C2D_MESSAGE READ_GSERVICES
Google Pay	Internet Storage Location Phone Camera Contacts NFC	C2D_MESSAGE MFC_ACCESS READ_GSERVICES USE_FINGERPRINT GET_PACKAGE_SIZE
Uber	Use Credentials SMS Read Profile Manage Accounts System Alert Window Bluetooth	SYSTEM_ALERT_WINDOW RECEIVE_BOOT_COMPLETED READ_SMS GET_ACCOUNTS READ_CONTACTS  C2D_MESSAGE READ_GSERVICES FOREGROUND_SERVICE

TABLE 4.1: Overview of all comparisons done

In Table 4.1, the column, ‘Comparison of Manifest and Description’ depicts the permissions that were present in the Manifest file, but not evident in the description. These are permissions that the user might possibly be unaware of while reading the description. The next column, ‘Over declared Permissions in the Manifest’ depicts the permissions present in the manifest file that do not have API calls in the code. These are permissions that are being redundantly declared in the manifest file. These comparisons hence show that developers are not transparent enough about the permissions that their applications are using. It also goes to show that developers often over-declare permissions in the manifest file that they do not require for the functionality of the application. This may pose a severe threat to the security and privacy for the user.

## 4.2 Conclusions

Using the results discussed above, a set of conclusions were derived that give an insight into the inadequacy of application descriptions and the lack of responsibility from the developer's end, especially while creating the manifest files.

### 4.2.1 Inadequacy of Application Descriptions

Our first step was to analyse descriptions of various applications using pen and paper approach, in order to estimate the permission groups it might require. As discussed earlier, this data was used to build a database and automate the process. When the permissions retrieved from the database for an application was compared with those retrieved from the manifest file of the application, it was seen that there were a lot of permissions that were in the manifest but not in the manual assessment. About 120 applications were tested. Their descriptions and manifest files were compared for permissions. On average, 45.51% of permissions were identified by the manual assessment approach, when compared to the permissions in the manifest file. The formula used was to test the accuracy of any application was:

$$\frac{x * 100}{y} \quad (4.1)$$

Where x is the number of permissions identified by the manual assessment method and y is the number of permissions in the manifest file, and x is a subset of y. This goes to show that an application's description is not enough for a user to determine what permissions the application might use. Developers are, either intentionally or unintentionally, omitting vital information regarding permissions in the descriptions. This issue can largely be noticed with gaming applications, where the description gives a detailed account of the game itself and not of the application's functionality.

### 4.2.2 Over-declaration of Permissions in the Manifest File

After observing the shortcomings of application descriptions, it was deemed imperative to look into the technicalities of applications to verify whether the developers are doing enough from their end. The obvious move was use the permissions declared by developers in the manifest and check whether they are being utilized in the source code. As discussed earlier, this was done by checking for API calls for these permissions. On analysing 120

applications, it was noted that about 32.2% of permissions did not have an API call in the source. The formula used was:

$$\frac{z * 100}{y} \quad (4.2)$$

Where  $z$  is the number of permissions in the manifest file that do not have an API Call in the source code (i.e. an over-declared permission), and  $y$  is the number of permissions in the manifest file. We also need to keep in mind that  $z$  is supposed to be a subset of  $y$ . This led to the conclusion that developers are over-declaring permissions in the manifest. These permissions find no use in the source code.

### 4.2.3 Storage and Phone Permissions Predominate Others

After analysing the Manifest files of over 120 applications, it was found that Storage and Phone permission groups were the frequently occurring permission groups. This was an indication of the fact that these permissions are the most basic and vital permissions needed by most applications for proper functioning.

### 4.2.4 Users lack basic knowledge of the Android Permission Model

The common end user does not possess sufficient technical knowledge required to understand and interpret the permissions an Android application demands. Users should invest time and effort into reading and understanding permissions before granting them, so that their security is not compromised in the long run.

## 4.3 Future Scope

This research can be used to generate a rating system that rates applications based on these metrics for the purpose of safety and privacy. Applications on the lower spectrum of the scale can be flagged for further evaluation either by the platform (like Google Play Store) or by the users. These findings can also be used to tighten the regulations and guidelines that developers follow when building applications.

## Chapter 5

# Appendix A: Playscraper

It is a Python library that can be integrated into Python codes using Import Playscraper instruction. It scrapes and parses application data of Android applications from the Google Play Store. Information fetched from the Android applications:

- Details: Fetches an application's details
- Collection: Fetches a list of applications and their details, optionally filtered by category.
- Developer: Fetches a developer's offered applications.
- Suggestions: Fetches a list of auto completed query string suggestions.
- Search: Fetches a list of applications matching a search query.(max 20 apps at a time)
- Similar: Fetches an application's similar apps.
- Categories: Fetches a list of available categories.



## Chapter 6

# Appendix B: Androguard

It is a python based tool, which can run on Windows/OSX/Linux, provided that python is installed on the system. It is powerful tool which provides malware analysis and reverse engineering analysis for Android, to decompile and to disassemble android apps. It can be used for the Static Analysis of an application using code obfuscators. Using any reverse engineering tool, the obfuscation level can be measured to convert Android xml files to readable format.

### 6.1 Features

- Map and manipulate DEX/APK format into full Python objects
- Diassemble/Decompilation/Modification of DEX/ODEX/APK format
- Allows access to the static analysis of the code (basic blocks, instructions, permissions)
- To transform Android's binary XML (like AndroidManifest.xml) into classic XML
- Allows integration with external decompilers

## 6.2 Retrieve Manifest file of an Android application from its APK File

Viewing `AndroidManifest.xml` is the most important part of reverse engineering. Using `Androaxml` tool of `Androguard`, we can easily fetch the `AndroidManifest.xml` file. It converts android's binary XML (i.e. `AndroidManifest.xml` file) into the classic XML file, that is human readable.

sectionRetrieve Permissions from the Manifest For analyzing and loading APK files, the function `AnalyzeAPK(filename)` is used. `a, d, dx = AnalyzeAPK("examples/android/abcore/app-prod-debug.apk")`

The three objects needed are:

- `a`: an APK object; where all information about the APK, like permissions, package name, the `AndroidManifest.xml` or its resources are found
- `d`: an array of `DalvikVMFormat` object that corresponds to the DEX file found inside the APK file to get classes, methods etc.
- `dx`: an `Analysis` object

## 6.3 Retrieve classes and methods from the Dex File

The `androguard.core.analysis.analysis.Analysis` object has all information about the classes, methods, fields and strings inside one or multiple DEX files. Additionally it enables you to get call graphs and crossreferences (XREFs) for each method, class, field and string. This means you can investigate the application for certain API calls or create graphs to see the dependencies of different classes.

For retrieving the classes and methods from the dex files, the functions `get_classes()` and `get_methods()` are used.

- `get_classes()` returns a list of `ClassAnalysis` objects or classes
- `get_methods()` returns a list of API calls or methods corresponding to the classes retrieved using `get_classes()`

## 6.4 Retrieve the SDK version from the Manifest File

The APK object returns all information about the APK, like permissions, package name, the SDK version and the AndroidManifest.xml. The "uses-sdk" attribute in the manifest expresses an application's compatibility with one or more versions of the Android platform, by means of an API Level integer.

To retrieve the minimum, maximum and target sdk versions declared in the manifest, `get_min_sdk_version()`, `get_max_sdk_version()` and `get_target_sdk_version()` functions are used.

- `get_min_sdk_version()` returns the minimum SDK version required for the application to run
- `get_max_sdk_version()` returns the maximum SDK version required for the application to run
- `get_target_sdk_version()` returns the SDK version that the application targets

## 6.5 Retrieve API-Permission Mapping using SDK Version

The `androguard.core.api_specific_resources` package in `androguard` can be used to extract the mapping information between API calls and Android Permission corresponding to a specific SDK version. The `load_permission_mappings(api_level)` function returns an API-Permission mapping for a requested API level(SDK version). The parameter `api_level` is an integer value of the API level. The output is a dictionary of the form `MethodSignature: [List of Permissions]`

# Bibliography

- [1] S. S. Shinde and S. S. Sambare, “Analysis of android app permissions for user’s privacy preservation,” Report 5, in International Journal of Computer Sciences and Engineering, 2015.
- [2] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, *SCanDroid: Automated Security Certification of Android Applications*,. PhD thesis, University of Maryland, College Park, 2015.
- [3] K. Benton, L. J. Camp, and V. Garg, *Studying the Effectiveness of Android Application Permissions Requests*,. PhD thesis, School of Informatics and Computing, Indiana University, Bloomington, IN, 2013.
- [4] R. H. Shaozhang Niu, W. Chen, and Y. Xue2, *An Improved Permission Management Scheme of Android Application Based on Machine Learning*,. PhD thesis, Beijing University of Posts and Telecommunications, 2018.
- [5] T. Mohini, S. A. Kumar, and G. Nitesh, “Review on android and smartphone security, research journal of computer and information technology science,” Report 6, 2013.
- [6] “Android developer forum: Manifest permissions.” <https://developer.android.com/reference/android/Manifest.permission.html>.
- [7] “Androguard documentation.” <https://androguard.readthedocs.io/en/latest/>.