ADDRESSING SECURITY ISSUES IN LATEST ANDROID VERSIONS AND APPLICATIONS

Bharath Ravi Prakash[1]

[1]*Department of Electronics and Communication, Jyothy Institute of Technology, Bengaluru-560082, Karnataka, India*

Abstract

In present generations, smart phones are said to be ruling the world. There are many smart phone OS platforms running in market currently. The number of Android OS users are more than any other mobile OS platform. As the number of users for a particular OS increase, the amount of threats and vulnerabilities to that particular OS will also increase. Android OS because of its popularity among the users also has an equal number of threats and other vulnerabilities that the OS is prone to.

We shall discuss and study various security issues that the Android OS is prone to, some of the methods to analyze the issues and some of the methods to tackle and overcome the issues. Also we shall discuss the issues that are addressed in the latest version of Android OS and the issues that still needs to be addressed.

Keywords: *Android, APK, Manifest, Permissions and Services*

---***---

1. INTRODUCTION

Android, which is one of the leading Operating Systems for mobile platform was designed by Google Inc [1]. It is based on Linux kernel and is mainly designed to work on touch screen phones most commonly smart phones and tablets. Android is not only designed to work on touchscreen devices, but Google has gone a step further and has developed Android TV to run on Televisions, Android Auto which runs in Cars and Android Wear designed to be used in wrist Watches. All these products have a unique interface.
Android OS was initially developed by Android Inc. founded by Andy Rubin, Rich Miner, Nick Sears, and Chris White. In July 2005, Google Inc. acquired Android Inc. [2]

1.1 Android Version History with Important Changes

There were two initial releases of Android which was not released. They were Android ALPHA and Android BETA. Each Android version is decided to be named after the name of a dessert or a sugary treat INALPHABETICAL ORDER.

2. ANDROID APPLICATION- AN OVERVIEW

Android applications are mainly built by using JAVA as the programming language. But nowadays applications are being built on other languages too. [4]

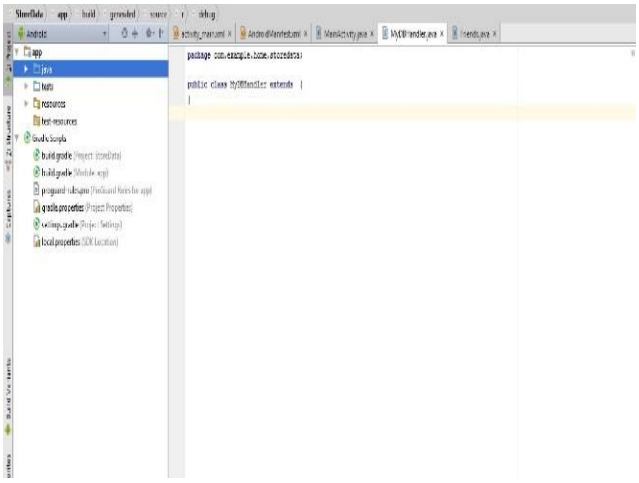
Eclipse IDE was the Application Software that was used to build android applications until Android Studio became the officially supported software application by Google Inc. to develop android applications. However there are many other online and offline tools to develop android application.

2.1 More Detailed Look

An android application mainly consists of three components. They are resource (which consists of the layouts, drawables, which are the images and other png files used and strings), the java codes, and AndroidManifest.xml [6].

But these can be viewed and will be visible only during the time of building of the application. Once the application is built into a compressed format called APK (Application Package Kit), then, all java codes are merged into a single file known as classes.dex, all resources are merged into a single file known as resource.arsc.

In order to view them, either one must get access to the whole project or the project should be reverse engineered
Fig 2.1 shows the layout of an Android application that is being developed with the help of Android Studio. It also shows the structure of application.



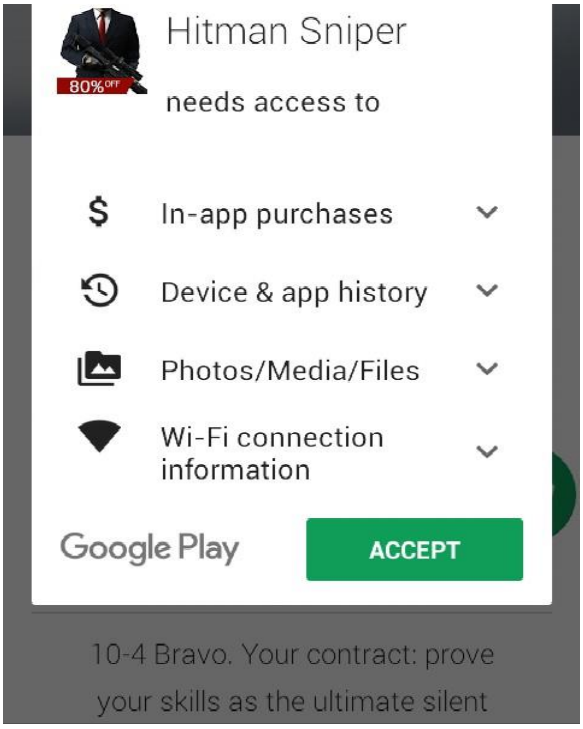
```
package com.example.home.storedata;

public class MyDBHandler extends |
|
```

Fig 2.1 Working path in Android Studio

2.2 Downloading and Installing an Application

The official place where Android apps are available are Google Play Store and Amazon Store [5]. However there are many other sources for an Android application. The general procedure for downloading and installing an application from Play Store are as follows:

- ❖ Access Google Play Store, Search for the required application
- ❖ Click on install and Grant Permission. Fig 2.2a shows screen of granting permissions.
- ❖ Application will be installed.



Hitman Sniper
needs access to

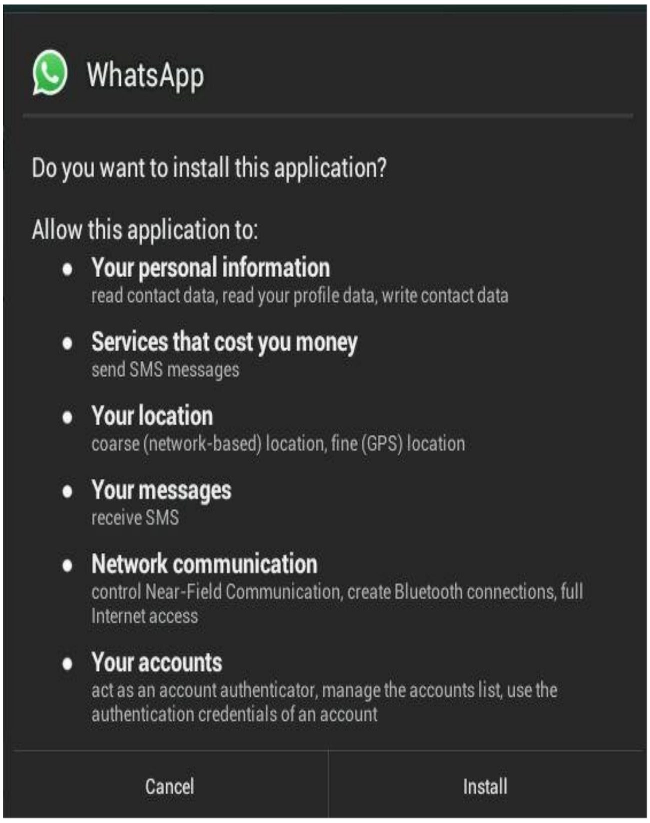
$ In-app purchases
Device & app history
Photos/Media/Files
Wi-Fi connection information

10-4 Bravo. Your contract: prove your skills as the ultimate silent

Fig. 2.2a Permission access in Google Play Store

However Android allows installation of applications from Unknown sources. The general procedure is as follows:

- ❖ Grant permission to install 3rd party applications from Security Settings
- ❖ Click on the APK file of the application.
- ❖ Accept all the permissions. Fig 2.2b shows screen of acceptation of permission and installation.
- ❖ Click on install to install the application.


WhatsApp

Do you want to install this application?

Allow this application to:

- **Your personal information**
  read contact data, read your profile data, write contact data
- **Services that cost you money**
  send SMS messages
- **Your location**
  coarse (network-based) location, fine (GPS) location
- **Your messages**
  receive SMS
- **Network communication**
  control Near-Field Communication, create Bluetooth connections, full Internet access
- **Your accounts**
  act as an account authenticator, manage the accounts list, use the authentication credentials of an account

Fig 2.2b Installation of an Application

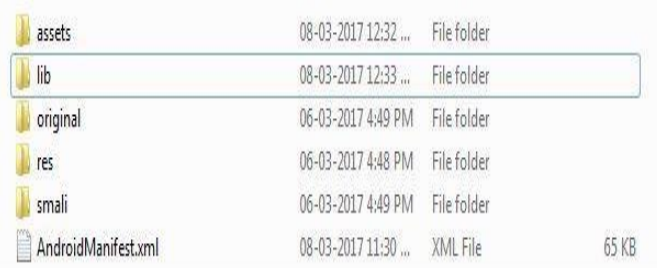
Observing the requested permission requires more attention. The reason being that there might be a lot of unwanted permissions that an application wants access. If such permissions are found then such applications are not safe to be installed.

2.3 Reverse Engineering

This is a process intended ONLY for educational purpose where the APK file is broken down to extract the contents of the file in order to access them.

By reverse engineering an APK file the codes will be available in an assembly language format called ”smali” along with all the resources.

Fig 2.3 shows the layout when an APK file is reverse engineered to extract the contents of the file.



assets	08-03-2017 12:32 ...	File folder	
lib	08-03-2017 12:33 ...	File folder	
original	06-03-2017 4:49 PM	File folder	
res	06-03-2017 4:48 PM	File folder	
smali	06-03-2017 4:49 PM	File folder	
AndroidManifest.xml	08-03-2017 11:30 ...	XML File	65 KB

Fig. 2.3 Files after reversing an APK

This method gives access to AndroidManifest.xml file as shown in Fig. 2.3.

2.4 Android Manifest and its Role in an Application

During the time of developing an application, AndroidManifest, which is a XML file plays an important role.

Each application that is build will contain this file. It can be said that AndroidManifest is the heart of any application. This file tells the android system all the information required to successfully run the application.

Few important functions of manifest are as listed below:

- It names the java package of application. It acts as unique identifier for application.
- It describes the essential components present inside the application.
- It defines the minimum and maximum API level requirements which tells the minimum android version required to run the application.
- It declares the permissions that are required by the application.

3. MANIFEST- A DETAILED LOOK

The main elements in Manifest [6] are as follows:

- Action- Used to perform a specific action
- Activity- Used to invoke a new activity
- Permission- Used to grant permissions to the application.
- Uses-sdk- Used to allocate specific SDK's API level.

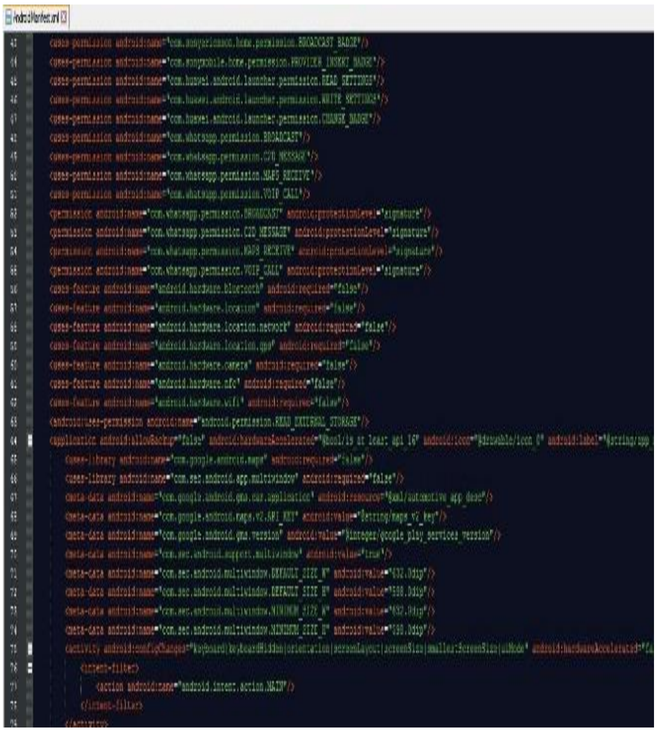
The components Activity, Services and Permissions play a very vital role in AndroidManifest.xml.

3.1 Permissions-A Detailed View

Permissions [7] play a very important role in Manifest file. It declares a security permission that can be used to limit access to specific components or features of an application.
The attributes inside permissions are as follows:

- Android: description- Provides a user readable description of the permission.
- Android: icon- provides reference to a drawable that can be used as a default icon for the application.
- Android: label- Provides the name for the permission that is defined. Normally the name is referenced from a string source.
- Android: protection level- Most important component of permission. There are four main levels-normal (default value. Involves lower risk), Dangerous (higher risk involving permission access to private user data), signature (level where access is granted if the application is signed with the same signature with that of requesting permission), signature or system (used for certain special situations where multiple vendors have access to same system image).

Fig 3.1 shows the permissions used by one of the most commonly used Android application. This application has four permissions whose protection levels are signature.



```
<uses-permission android:name="com.sonyericsson.home.permission.BROADCAST_BADGE"/>
<uses-permission android:name="com.sonymobile.home.permission.PROVIDER_INSERT_BADGE"/>
<uses-permission android:name="com.huawei.android.launcher.permission.READ_SETTINGS"/>
<uses-permission android:name="com.huawei.android.launcher.permission.WRITE_SETTINGS"/>
<uses-permission android:name="com.huawei.android.launcher.permission.CHANGE_BADGE"/>
<uses-permission android:name="com.whatsapp.permission.BROADCAST"/>
<uses-permission android:name="com.whatsapp.permission.C2D_MESSAGE"/>
<uses-permission android:name="com.whatsapp.permission.MAPS_RECEIVE"/>
<uses-permission android:name="com.whatsapp.permission.VOIP_CALL"/>
<permission android:name="com.whatsapp.permission.BROADCAST" android:protectionLevel="signature"/>
<permission android:name="com.whatsapp.permission.C2D_MESSAGE" android:protectionLevel="signature"/>
<permission android:name="com.whatsapp.permission.MAPS_RECEIVE" android:protectionLevel="signature"/>
<permission android:name="com.whatsapp.permission.VOIP_CALL" android:protectionLevel="signature"/>
<uses-feature android:name="android.hardware.bluetooth" android:required="false"/>
<uses-feature android:name="android.hardware.location" android:required="false"/>
<uses-feature android:name="android.hardware.location.network" android:required="false"/>
<uses-feature android:name="android.hardware.location.gps" android:required="false"/>
<uses-feature android:name="android.hardware.camera" android:required="false"/>
<uses-feature android:name="android.hardware.nfc" android:required="false"/>
<uses-feature android:name="android.hardware.wifi" android:required="false"/>
<android:uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
<application android:allowBackup="false" android:hardwareAccelerated="@bool/is_at_least_api_16" android:icon="@drawable/icon_0" android:label="@string/app_
    <uses-library android:name="com.google.android.maps" android:required="false"/>
    <uses-library android:name="com.sec.android.app.multiwindow" android:required="false"/>
    <meta-data android:name="com.google.android.gms.car.application" android:resource="@xml/automotive_app_desc"/>
    <meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="@string/maps_v2_key"/>
    <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version"/>
    <meta-data android:name="com.sec.android.support.multiwindow" android:value="true"/>
    <meta-data android:name="com.sec.android.multiwindow.DEFAULT_SIZE_W" android:value="632.0dip"/>
    <meta-data android:name="com.sec.android.multiwindow.DEFAULT_SIZE_H" android:value="598.0dip"/>
    <meta-data android:name="com.sec.android.multiwindow.MINIMUM_SIZE_W" android:value="632.0dip"/>
    <meta-data android:name="com.sec.android.multiwindow.MINIMUM_SIZE_H" android:value="598.0dip"/>
    <activity android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize|smallestScreenSize|uiMode" android:hardwareAccelerated="fa
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
        </intent-filter>
    </activity>
```

Fig 3.1 Permissions in Manifest

Since this application has feature to make voice calls, the permission: VOIP_CALL is used. Similarly to access Google Maps to receive Location content, the permission: MAPS_RECEIVE is used.

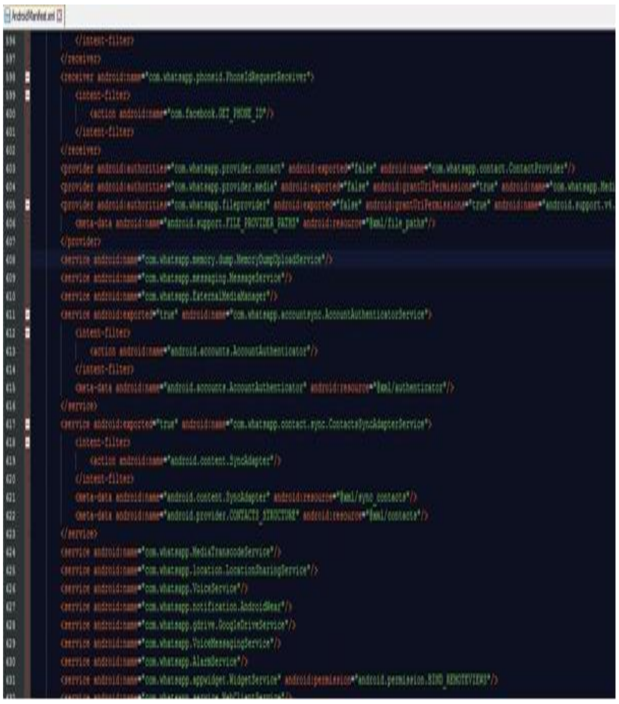
3.2 Services-A Detailed View

Services [7] declares a service as one of the application's components. They are used to implement long running background operations or a rich communication API. It must be a class that is present inside the application.

Attributes inside services are as follows:

- Android: description- string that describes the service to users.
- Android: enabled- a Boolean value that determines if the service is enabled or not.
- Android: label- A name for the service that can be displayed to users. Normally the name is referenced from a string source.
- Android: permission- Provides details about permission that an entity must have in order to launch the service or bind to it.

Fig 3.2 shows some of the services used by one of the most commonly used Android application.



```
        </intent-filter>
    </receiver>
    <receiver android:name="com.whatsapp.phoneid.PhoneIdRequestReceiver">
        <intent-filter>
            <action android:name="com.facebook.GET_PHONE_ID"/>
        </intent-filter>
    </receiver>
    <provider android:authorities="com.whatsapp.provider.contact" android:exported="false" android:name="com.whatsapp.contact.ContactProvider"/>
    <provider android:authorities="com.whatsapp.provider.media" android:exported="false" android:grantUriPermissions="true" android:name="com.whatsapp.Medi
    <provider android:authorities="com.whatsapp.fileprovider" android:exported="false" android:grantUriPermissions="true" android:name="android.support.v4.
        <meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/file_paths"/>
    </provider>
    <service android:name="com.whatsapp.memory.dump.MemoryDumpUploadService"/>
    <service android:name="com.whatsapp.messaging.MessageService"/>
    <service android:name="com.whatsapp.ExternalMediaManager"/>
    <service android:exported="true" android:name="com.whatsapp.accountsync.AccountAuthenticatorService">
        <intent-filter>
            <action android:name="android.accounts.AccountAuthenticator"/>
        </intent-filter>
        <meta-data android:name="android.accounts.AccountAuthenticator" android:resource="@xml/authenticator"/>
    </service>
    <service android:exported="true" android:name="com.whatsapp.contact.sync.ContactsSyncAdapterService">
        <intent-filter>
            <action android:name="android.content.SyncAdapter"/>
        </intent-filter>
        <meta-data android:name="android.content.SyncAdapter" android:resource="@xml/sync_contacts"/>
        <meta-data android:name="android.provider.CONTACTS_STRUCTURE" android:resource="@xml/contacts"/>
    </service>
    <service android:name="com.whatsapp.MediaTranscodeService"/>
    <service android:name="com.whatsapp.location.LocationSharingService"/>
    <service android:name="com.whatsapp.VoiceService"/>
    <service android:name="com.whatsapp.notification.AndroidWear"/>
    <service android:name="com.whatsapp.gdrive.GoogleDriveService"/>
    <service android:name="com.whatsapp.VoiceMessagingService"/>
    <service android:name="com.whatsapp.AlarmService"/>
    <service android:name="com.whatsapp.appwidget.WidgetService" android:permission="android.permission.BIND_REMOTEVIEWS"/>
```

Fig 3.2 Services in Manifest

Since this application has feature to transfer media files, access voice service, access Google drive service, the services Media Transcode Service, Voice Service, and Google Drive Service are used.

4. DETECTION OF SECURITY THREATS

There are many ways of Threat detection that can be implemented. But AndroidManifest is the best way to detect security issues that are existing in an android application. Security issues are mainly related to unnecessary and unwanted permissions and services that are declared by the developers while developing the application. [3][8]

4.1 Manifest's Role in Security Detection

There are a lot of critical permissions and services that can causes security issues while using an application. Hence it is very much essential to consider these critical issues. Since manifest plays an important role in successful run of an application it is very much necessary to be very careful while handling manifest.

4.2 Method of Detection

The most important procedure while detecting suspicious permissions and services is to analyze the manifest file. The components inside a manifest file will be different for each application depending on the nature and type of the application.

It is very important to take necessary care while attempting to safely remove unwanted and unnecessary components present in the manifest.

It is not easy to access AndroidManifest.xml file. Reason being the developers will not easily share their project to get access to the file. This is where concept of reverse engineering helps to get access to the file.

Permissions that are totally not related to the application can be safely removed. For example, an application which does not require an internet connection may contain permissions requesting access to internet. These kinds of permissions are not related to the application. Removing such permissions and services will in no way affect the stability or performance of the application.

The brief procedure to be followed once access to AndroidManifest.xml is obtained can be discussed as follows:

- ❖ Carefully observe all the permission requests, services, activities, and other components that are used by the application.
- ❖ If a particular activity, service, or a permission is found to be irrelevant to the nature of application, then it can be safely removed.
- ❖ Care must be taken such that the complete component should be removed in order to avoid errors while compiling back to the APK package.[10]
- ❖ Once the task is completed, it must be tested so that the application functions in the same way as how it was functioning before editing.

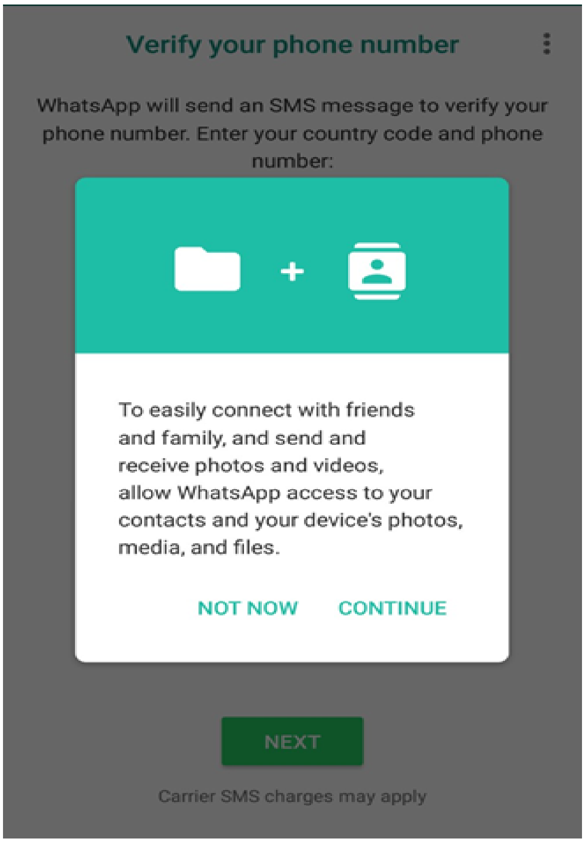
Considering Fig 3.1, if permission VOIP_CALL has to be removed then, the line "<permission android: name="com.whatsapp.permission.VOIP_CALL" android: protectionLevel="signature" />" has to be removed completely.

4.3 Android's Initiative to Improve Security with respect to Permission Management

There was a major improvement to Android's security with the release of Android's Lollipop version which was further improved with the release of Marshmallow and Nougat version. [9]

Some of the critical permissions like accessing messages, making phone call and similar permissions were given a two-step verification. The user's permission is again asked while the application requires use of critical permissions.

Fig 4.3a shows a screen where a particular application is asking for permission to access GALLERY and MEDIA.



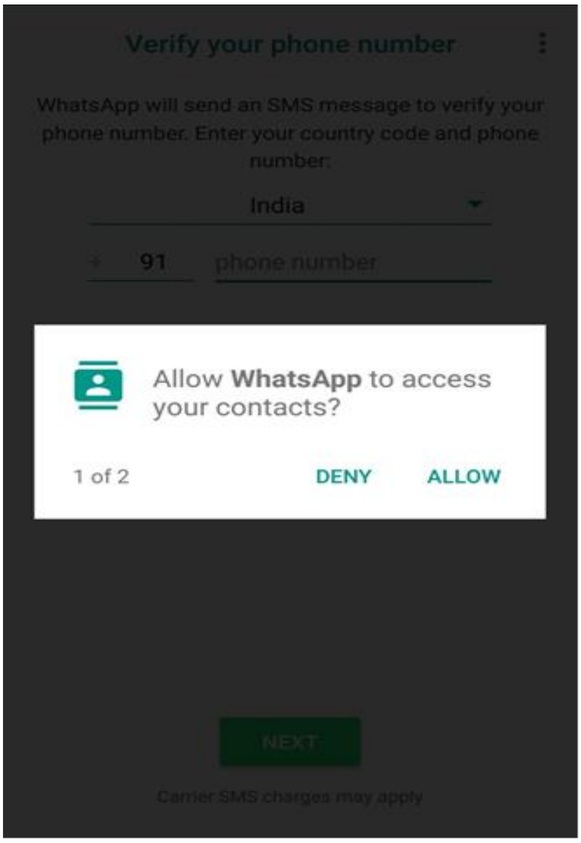
Verify your phone number

WhatsApp will send an SMS message to verify your phone number. Enter your country code and phone number:

To easily connect with friends and family, and send and receive photos and videos, allow WhatsApp access to your contacts and your device's photos, media, and files.

Fig 4.3a Media Access

Fig 4.3b shows a screen where a particular application is asking for permission to access CONTACTS.



Verify your phone number

WhatsApp will send an SMS message to verify your phone number. Enter your country code and phone number:

India

+ 91 phone number

Allow **WhatsApp** to access your contacts?

Fig 4.3b Contact access

5. FUTURE DEVELOPMENTS

The proposed future work is:

- ❖ It might be tedious to reverse engineer every application. A simpler method is to make use of a separate application designed exclusively to view, analyze, and remove the components.
- ❖ There are certain applications available to perform this task.[11][12]
- ❖ But, currently available applications provide facility only to remove components for an application. There are no applications available to rebuild and reinstall the newly modified application.

6. CONCLUSION

Android is the most widely used mobile operating system in today's works. The amount of threats and vulnerabilities that it is prone to is also high. Hence it is very much necessary to give emphasis on the security issues. Applications that are not safe to be used should be avoided.

Care must be taken to eliminate the use of applications involving critical permission request which are not used by the application and are not required.

REFERENCES

[1]. Martínez-Pérez, Borja, Isabel De La Torre-Díez, and Miguel López-Coronado. "Privacy and security in mobile health apps: a review and recommendations."Journal of medical systems 39.1 (2015): 181.

[2]. Elgin, Ben. "Google buys Android for its mobile arsenal." Bloomberg Businessweek 16 (2005).

[3]. Lawton, George. "Is it finally time to worry about mobile malware?." Computer41.5 (2008).

[4]. Zheng, Min, Mingshen Sun, and John CS Lui. "Droid analytics: a signature based analytic system to collect, extract, analyze and associate android malware." Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on. IEEE, 2013.

[5]. Barrera, David, and Paul Van Oorschot. "Secure software installation on smartphones." IEEE Security & Privacy 9.3 (2011): 42-48.

[6]. Android official. http://developer.android.com/

[7]. Android Manifest. https://developer.android.com/guide/topics/manifest/manifest-intro.html

[8]. Miller, Charlie. "Mobile attacks and defense." IEEE Security & Privacy 9.4 (2011): 68-70.

[9]. Batyuk, Leonid, et al. "Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications."Malicious and Unwanted Software (MALWARE), 2011 6th International Conference on. IEEE, 2011..

[10]. Zhou, Wu, et al. "Detecting repackaged smartphone applications in third-party android marketplaces." Proceedings of the second ACM conference on Data and Application Security and Privacy. ACM, 2012.

[11]. Zhou, Yajin, et al. "Hey, you, get off of my market: detecting malicious apps in official and alternative android markets." NDSS. Vol. 25. No. 4. 2012.
[12]. Chin, Erika, et al. "Analyzing inter-application communication in Android."Proceedings of the 9th international conference on Mobile systems, applications, and services. ACM, 2011.