LAB MANUAL ON PERFORMING REVERSE ENGINEERING & ANALYSIS OF ANDROID APPS

ESTABLISHMENT OF ADVANCED LABORATORY FOR CYBER SECURITY TRAINING TO TECHNICAL TEACHERS

DEPARTMENT OF INFORMATION MANAGEMENT AND EMERGING ENGINEERING

MINISTRY OF ELECTRONICS AND INFORMATION TECHNOLOGY

GOVERNMENT OF INDIA

Principal Investigator: Prof. Maitreyee Dutta

Co Investigator: Prof. Shyam Sundar Pattnaik

PREPARED BY:

Prof. Maitreyee Dutta and Ms. Shweta Sharma (Technical Assistant)

Table of Contents

MANUAL-7: PERFORMING REVERSE ENGINEERING & ANALYSIS OF ANDROID APPLICATIONS

INTRODUCTION TO ANDROID APPLICATIONS

- Among all smartphone devices, Android is the most widely used operating system [1] in the world where malware penetrate in the form of malicious applications.
- All Android applications (shown in Figure 1) are in the form of APKs (i.e. Android Package Kits). The APK format is used by Android operating system to install applications in Android based smartphones.
- Malware stands for malicious software which includes virus, worms, Trojan horses, etc.
- A malware can penetrate into smart-phone devices, personal computers, Internet of Things devices, etc.

Figure 1: Android applications

INTRODUCTION TO REVERSE ENGINEERING

- Reverse engineering is performed to reverse the process of packaged file. For example, the zipped file is unzipped to reveal the files it contains.
- In a similar fashion, reverse engineering is applied on APKs to reveal the files it contains.
- The purpose of reverse engineering is to recover the original files written by the application developers.
- The malware analysis team can perform reverse engineering on Android APKs to recover the files for malware analysis.
- Malware analysis is usually accomplished by performing reverse engineering on Android APKs to reveal information such as permissions, operational codes used by the application.

OBJECTIVES

The objective are as follows:

- To collect Android APKs from third party app store.
- To download and install reverse engineering tools.
- To perform reverse engineering on Android APKs.
- To perform analysis of Android APKs.

APK-TOOL

- An APK-tool [2] (shown in Figure 2) is a reverse engineering tool to reverse the process of an APK file.

- This tool provides the original files after performing reverse engineering on Android APKs.

Figure 2: APK-Tool

PERFORMING REVERSE ENGINEERING ON ANDROID APKs

The reverse engineering process can be performed with the following steps:

<u>Step 1:</u> Open APKpure on personal computer/laptop by browsing the website https://apkpure.com/app as shown in Figure 3.

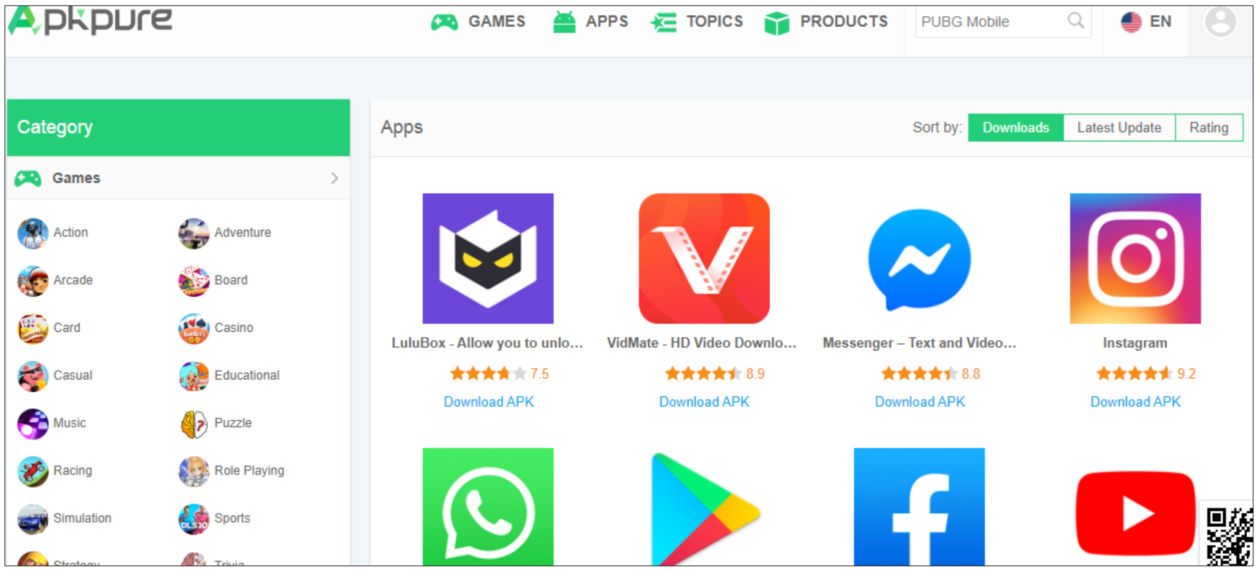


Figure 3: Opening APKpure

<u>Step 2:</u> Download APKs by typing the name of the application in search box (e.g., PUBG) as shown in Figure 4. One can directly download the APKs which are available on the home page without searching for it. Rename the downloaded application as "PUBG" on the personal computer/ laptop.

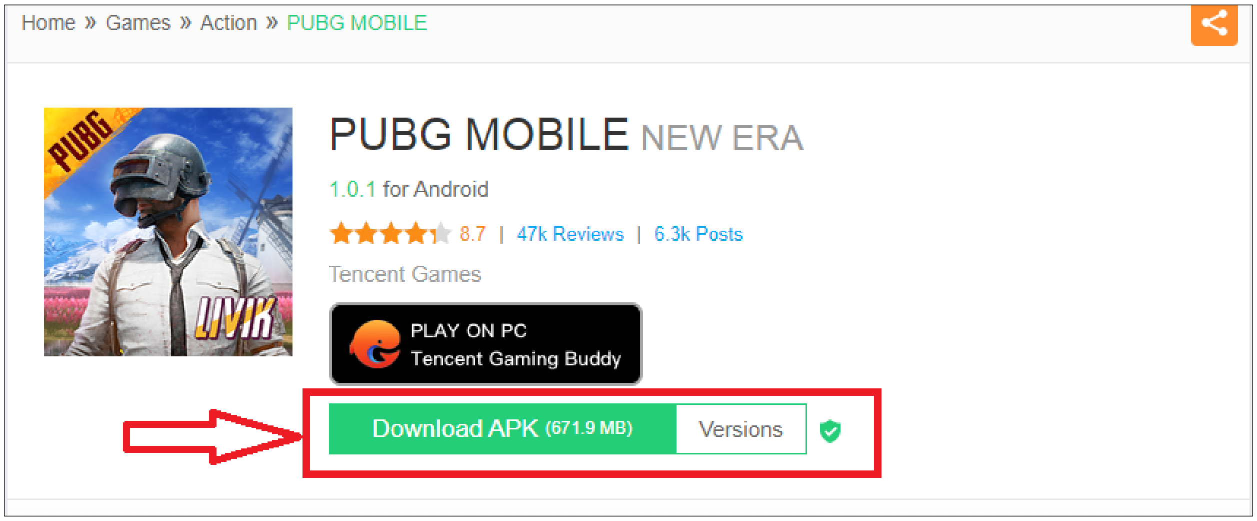
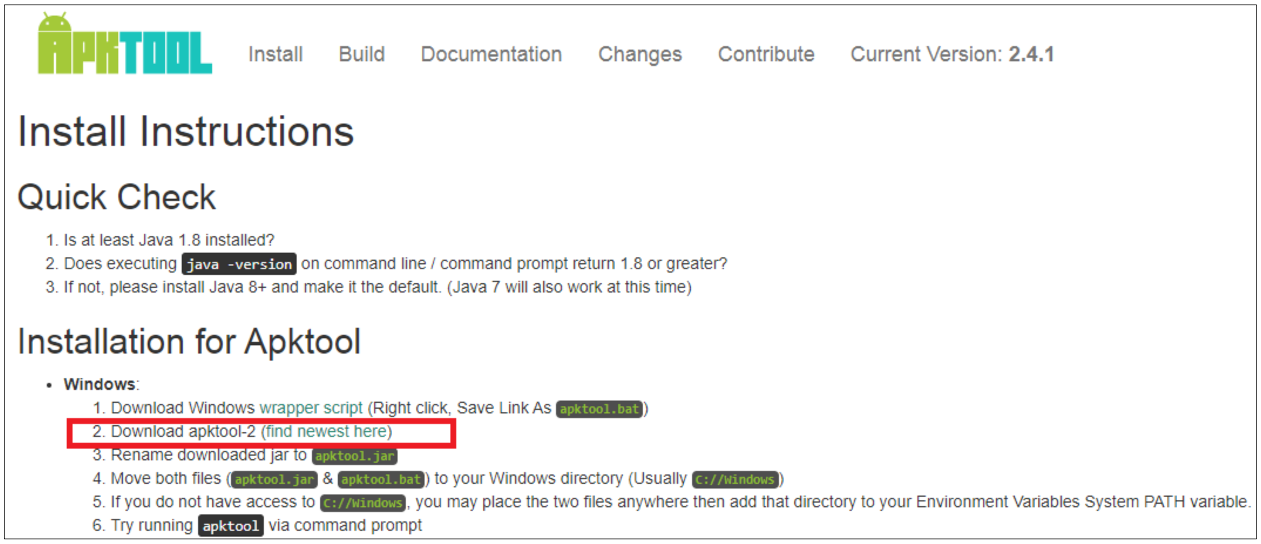


Figure 4: Download PUBG APK

<u>Step 3:</u> Download the APK-Tool on Windows operating system by opening the URL https://ibotpeaches.github.io/Apktool/install/ as shown in Figure 5. Also, download the wrapper script and save it in a text document.


Install Build Documentation Changes Contribute Current Version: **2.4.1**

Install Instructions

Quick Check

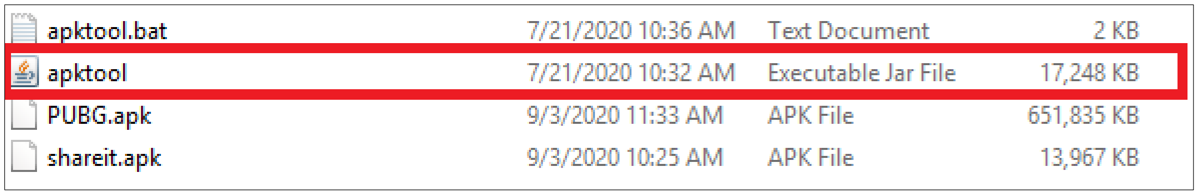
1. Is at least Java 1.8 installed?
2. Does executing `java -version` on command line / command prompt return 1.8 or greater?
3. If not, please install Java 8+ and make it the default. (Java 7 will also work at this time)

Installation for Apktool

- **Windows**:
  1. Download Windows wrapper script (Right click, Save Link As `apktool.bat`)
  2. Download apktool-2 (find newest here)
  3. Rename downloaded jar to `apktool.jar`
  4. Move both files (`apktool.jar` & `apktool.bat`) to your Windows directory (Usually `C://Windows`)
  5. If you do not have access to `C://Windows`, you may place the two files anywhere then add that directory to your Environment Variables System PATH variable.
  6. Try running `apktool` via command prompt

Figure 5: Download APK-Tool

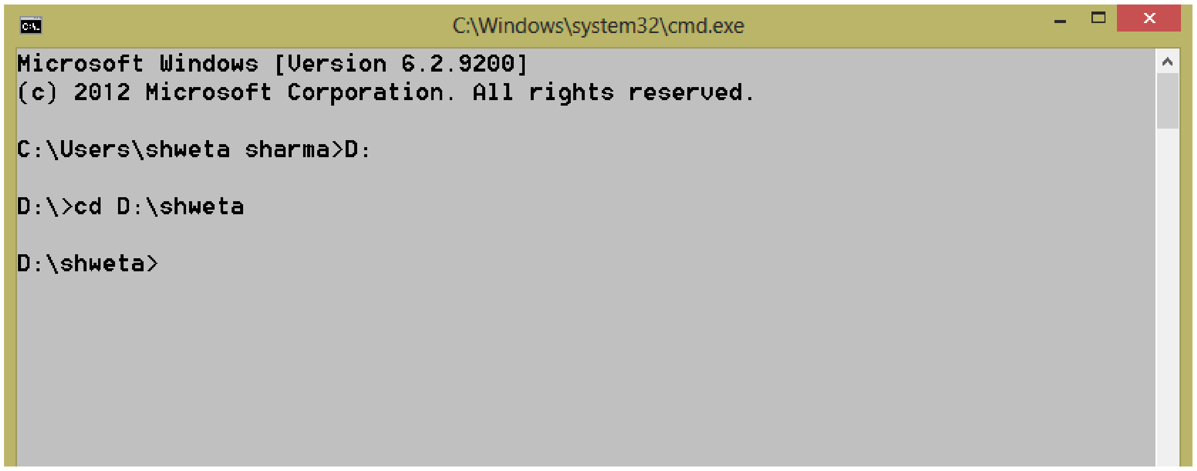
<u>Step 4:</u> Rename the tool as "apktool" which is an executable JAR file as shown in Figure 6. The wrapper script is saved as a batch file "apktool.bat" as shown in Figure 4.



apktool.bat	7/21/2020 10:36 AM	Text Document	2 KB
apktool	7/21/2020 10:32 AM	Executable Jar File	17,248 KB
PUBG.apk	9/3/2020 11:33 AM	APK File	651,835 KB
shareit.apk	9/3/2020 10:25 AM	APK File	13,967 KB

Figure 6: APK-Tool and batch file

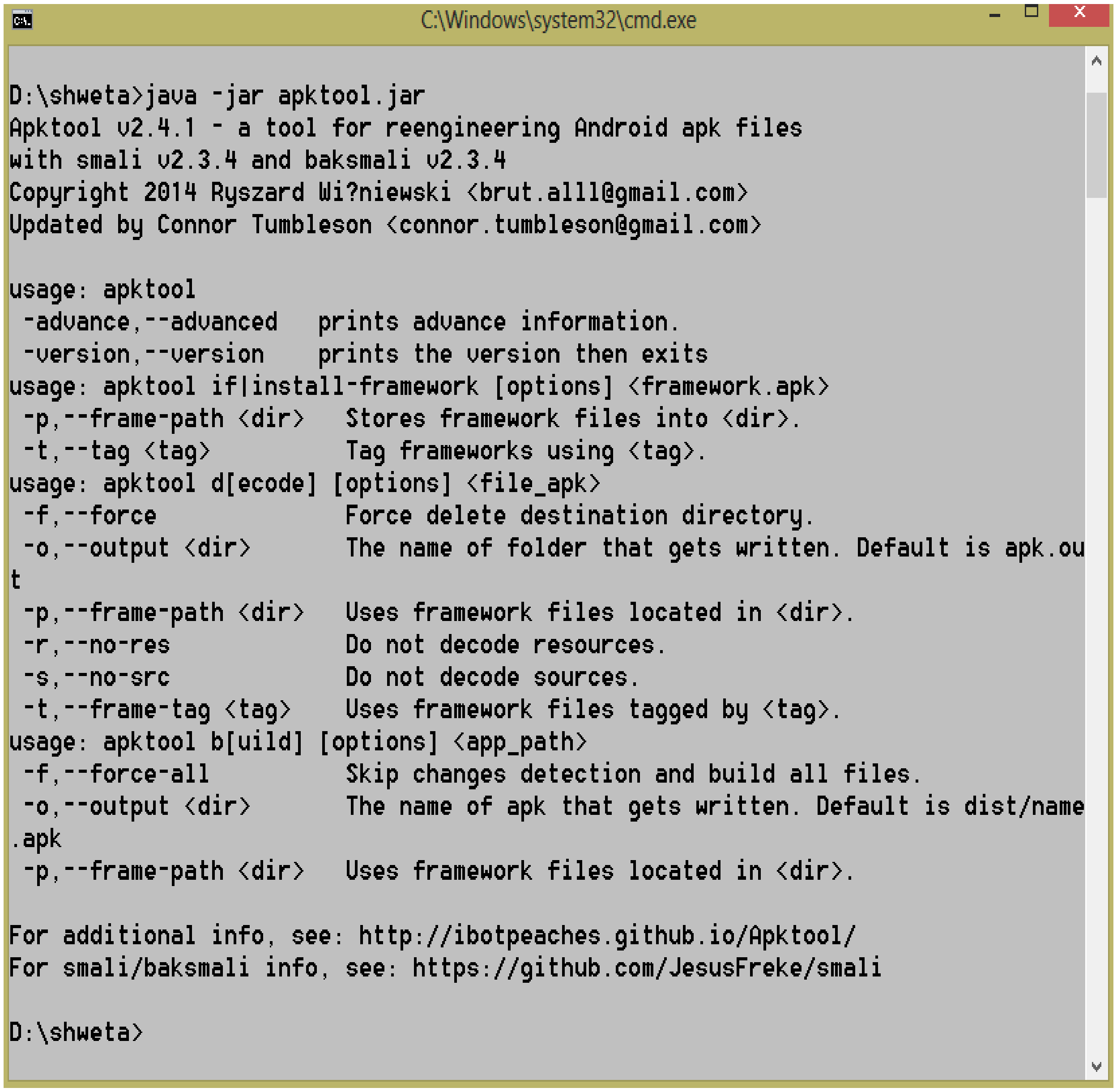
<u>Step 5:</u> Open a command prompt in personal computer or laptop. Type the path of the folder where APK-Tool and PUBG APK is stored as shown in Figure 7.



```
Microsoft Windows [Version 6.2.9200]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Users\shweta sharma>D:

D:\>cd D:\shweta

D:\shweta>
```

Figure 7: Opening command prompt

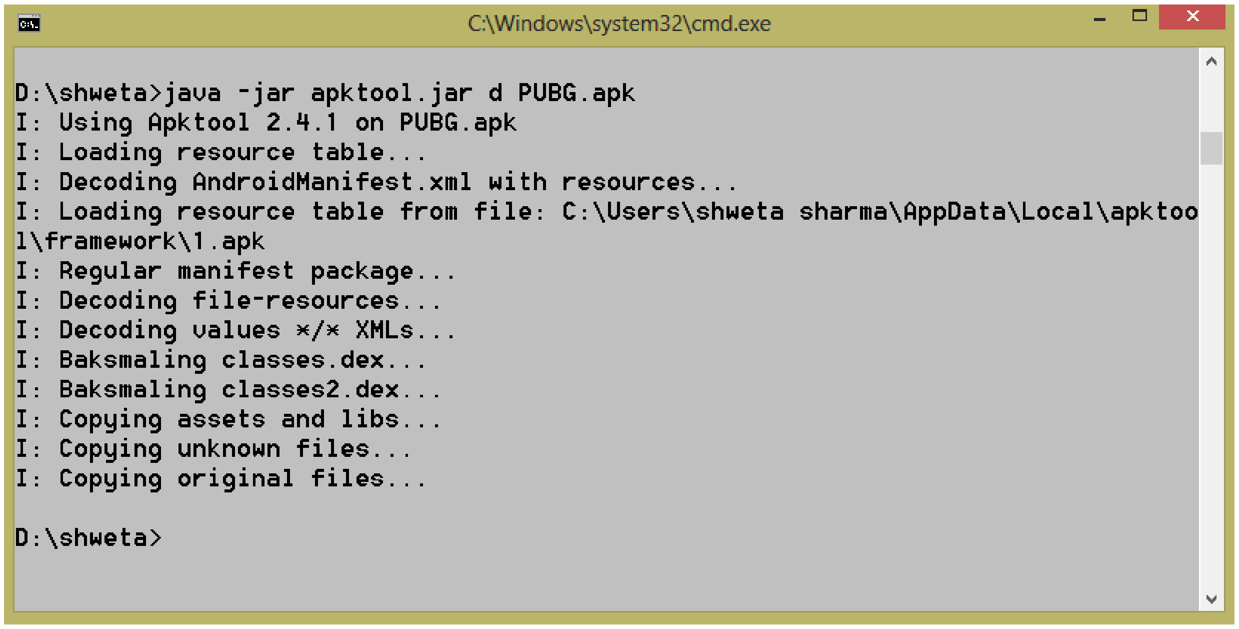
<u>**Step 6:**</u> Open APK-Tool via command prompt by typing the command "*java –jar apktool.jar*" as shown in Figure 8 where apktool.jar is the downloaded jar file of APK-tool. A manual of APK-Tool will appear on command prompt by pressing ENTER after typing the above command.



```
D:\shweta>java -jar apktool.jar
Apktool v2.4.1 - a tool for reengineering Android apk files
with smali v2.3.4 and baksmali v2.3.4
Copyright 2014 Ryszard Wi?niewski <brut.alll@gmail.com>
Updated by Connor Tumbleson <connor.tumbleson@gmail.com>

usage: apktool
 -advance,--advanced   prints advance information.
 -version,--version    prints the version then exits
usage: apktool if|install-framework [options] <framework.apk>
 -p,--frame-path <dir>   Stores framework files into <dir>.
 -t,--tag <tag>          Tag frameworks using <tag>.
usage: apktool d[ecode] [options] <file_apk>
 -f,--force              Force delete destination directory.
 -o,--output <dir>       The name of folder that gets written. Default is apk.ou
t
 -p,--frame-path <dir>   Uses framework files located in <dir>.
 -r,--no-res             Do not decode resources.
 -s,--no-src             Do not decode sources.
 -t,--frame-tag <tag>    Uses framework files tagged by <tag>.
usage: apktool b[uild] [options] <app_path>
 -f,--force-all          Skip changes detection and build all files.
 -o,--output <dir>       The name of apk that gets written. Default is dist/name
.apk
 -p,--frame-path <dir>   Uses framework files located in <dir>.

For additional info, see: http://ibotpeaches.github.io/Apktool/
For smali/baksmali info, see: https://github.com/JesusFreke/smali

D:\shweta>
```

Figure 8: Opening APK-Tool manual on command prompt

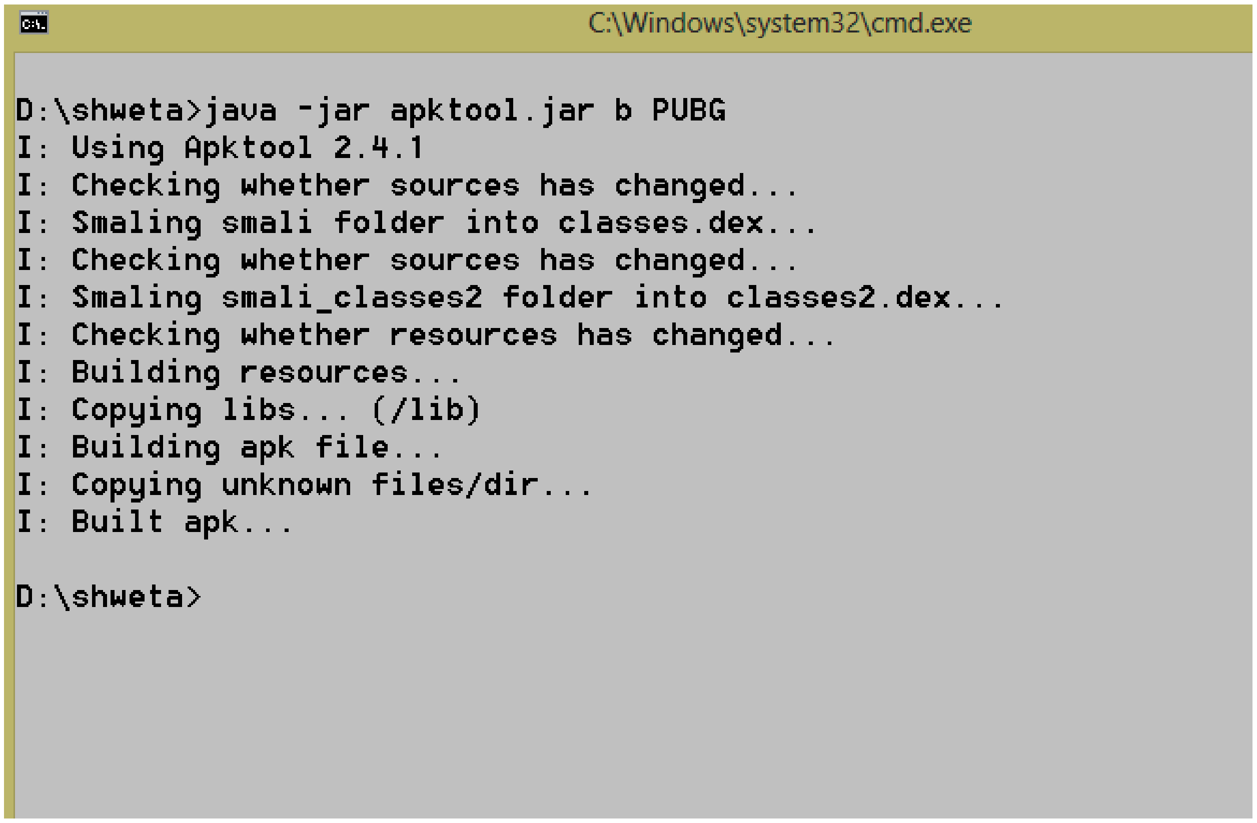
Step 7: In this step, reverse engineering will be performed on the downloaded APK (i.e. PUBG). The APK will be decompiled via reverse engineering tool (i.e. APK-Tool) by typing the command *"java –jar apktool.jar d PUBG.apk"* on the command prompt as shown in Figure 9. In the above command, d stands for decompile and PUBG.apk is the APK file downloaded in Step 2.


C:\Windows\system32\cmd.exe

```
D:\shweta>java -jar apktool.jar d PUBG.apk
I: Using Apktool 2.4.1 on PUBG.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\shweta sharma\AppData\Local\apktoo
l\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Baksmaling classes2.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

D:\shweta>
```

Figure 9: Reverse Engineering to de-compile PUBG APK

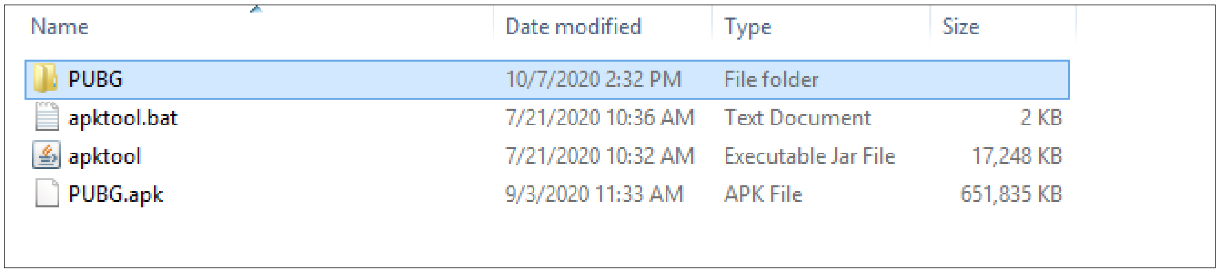
<u>Step 8:</u> In this step, reverse engineering will be performed on the downloaded APK (i.e. PUBG). The APK will be decompiled further to build other folders via reverse engineering tool (i.e. APK-Tool) by typing the command "*java –jar apktool.jar b PUBG*" on the command prompt as shown in Figure 10. In the above command, b stands for build and PUBG is the APK file downloaded in Step 2.



```
D:\shweta>java -jar apktool.jar b PUBG
I: Using Apktool 2.4.1
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether sources has changed...
I: Smaling smali_classes2 folder into classes2.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...

D:\shweta>
```

Figure 10: Reverse Engineering to build resource folder of PUBG APK

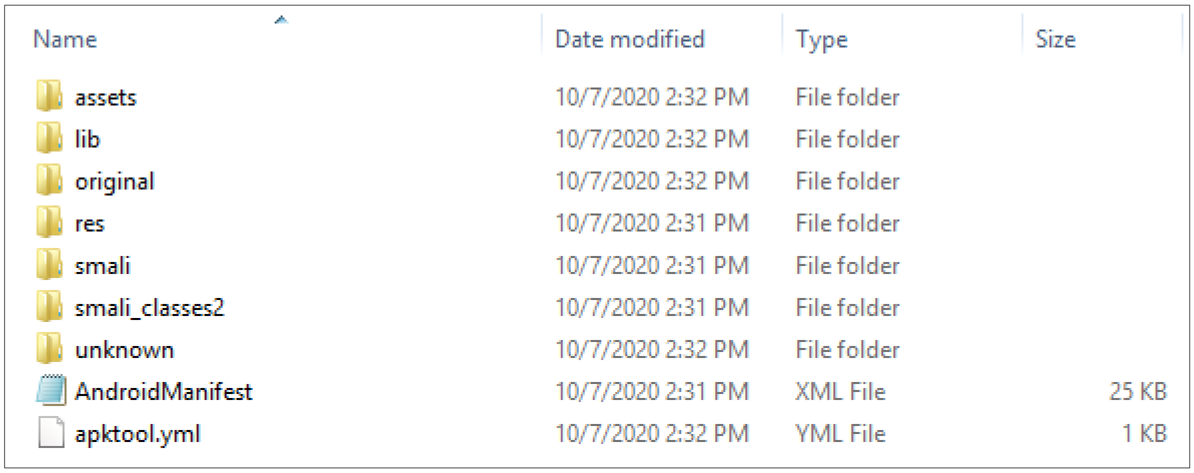
<u>Step 9:</u> A folder with name “PUBG” will appear on the same location where the APK-Tool and PUBG APK has been downloaded shown in Figure 11.



Name	Date modified	Type	Size
PUBG	10/7/2020 2:32 PM	File folder	
apktool.bat	7/21/2020 10:36 AM	Text Document	2 KB
apktool	7/21/2020 10:32 AM	Executable Jar File	17,248 KB
PUBG.apk	9/3/2020 11:33 AM	APK File	651,835 KB

Figure 11: PUBG folder

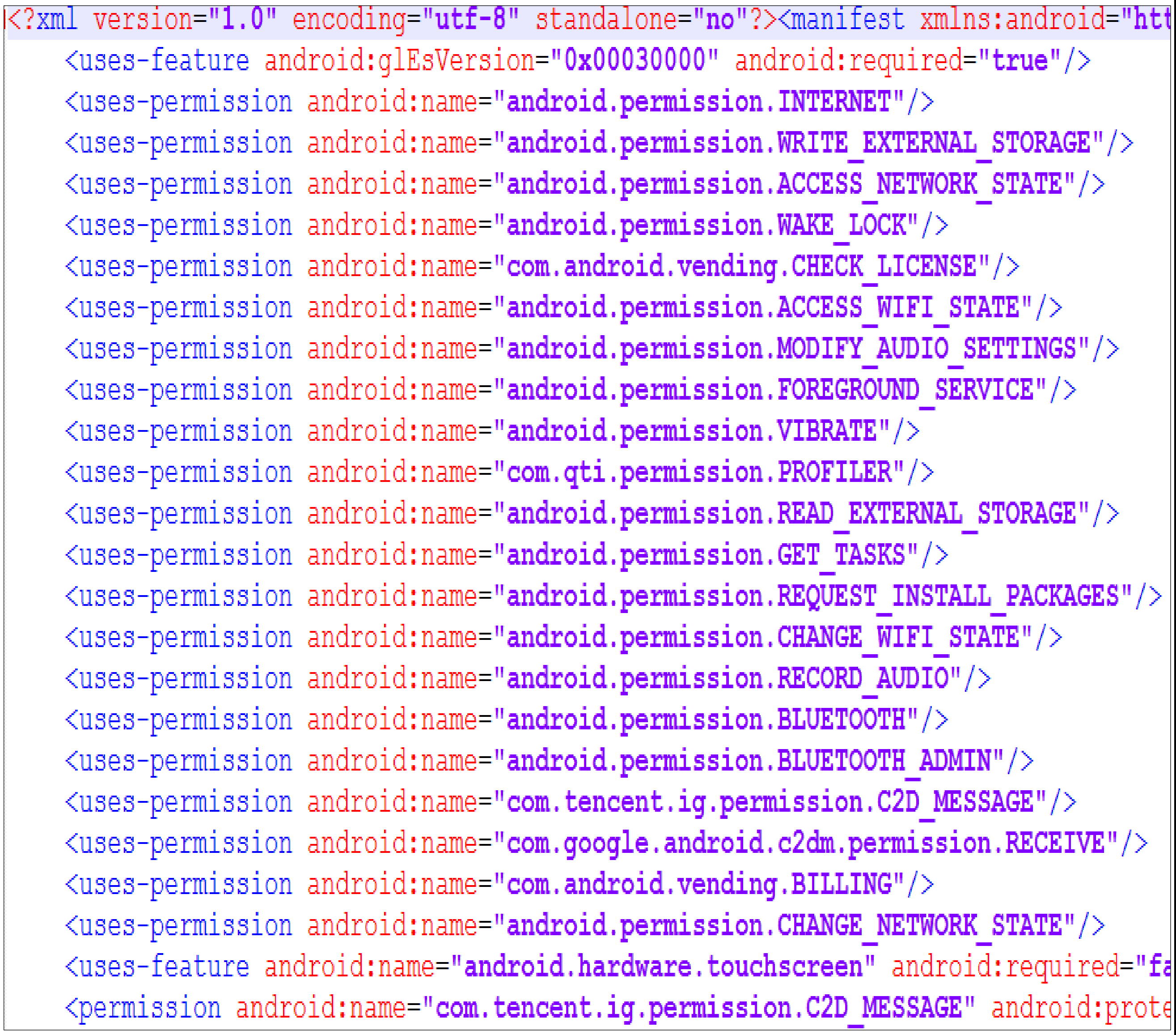
Step 10: This folder (PUBG) contains several files and folders as shown in Figure 12.



Name	Date modified	Type	Size
assets	10/7/2020 2:32 PM	File folder	
lib	10/7/2020 2:32 PM	File folder	
original	10/7/2020 2:32 PM	File folder	
res	10/7/2020 2:31 PM	File folder	
smali	10/7/2020 2:31 PM	File folder	
smali_classes2	10/7/2020 2:31 PM	File folder	
unknown	10/7/2020 2:32 PM	File folder	
AndroidManifest	10/7/2020 2:31 PM	XML File	25 KB
apktool.yml	10/7/2020 2:32 PM	YML File	1 KB

Figure 12: Files and folders in PUBG

Step 11: Open the AndroidManifest file in notepad/notepad++ as shown in Figure 13. The AndroidManifest file contains permissions obtained by the particular app.



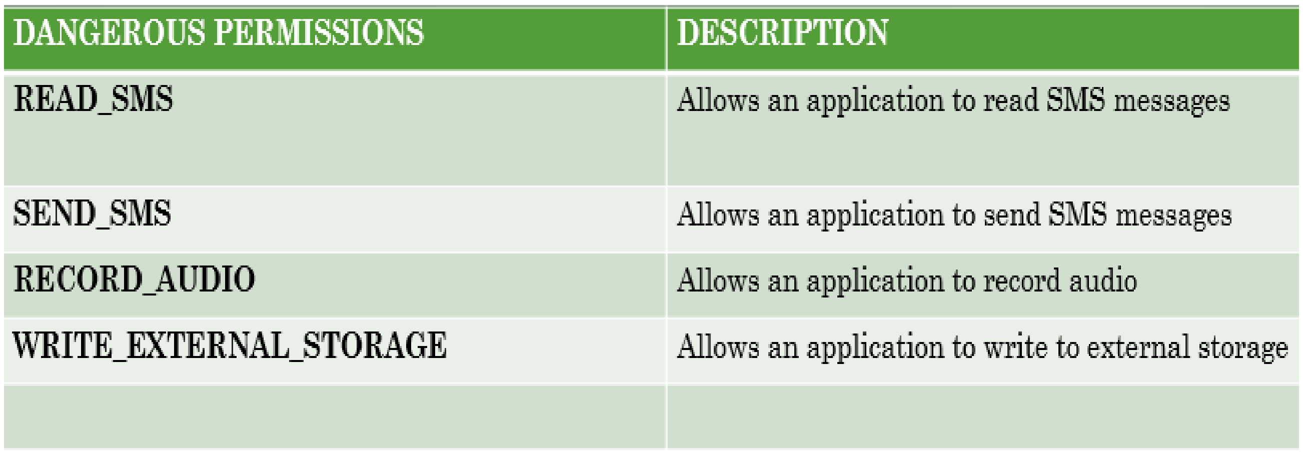
```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="htt
    <uses-feature android:glEsVersion="0x00030000" android:required="true"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="com.android.vending.CHECK_LICENSE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE"/>
    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="com.qti.permission.PROFILER"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.GET_TASKS"/>
    <uses-permission android:name="android.permission.REQUEST_INSTALL_PACKAGES"/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.RECORD_AUDIO"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission android:name="com.tencent.ig.permission.C2D_MESSAGE"/>
    <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"/>
    <uses-permission android:name="com.android.vending.BILLING"/>
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-feature android:name="android.hardware.touchscreen" android:required="fa
    <permission android:name="com.tencent.ig.permission.C2D_MESSAGE" android:prote
```

Figure 13: AndroidManifest file of PUBG app

ANALYSIS OF ANDROID-MANIFEST FILE

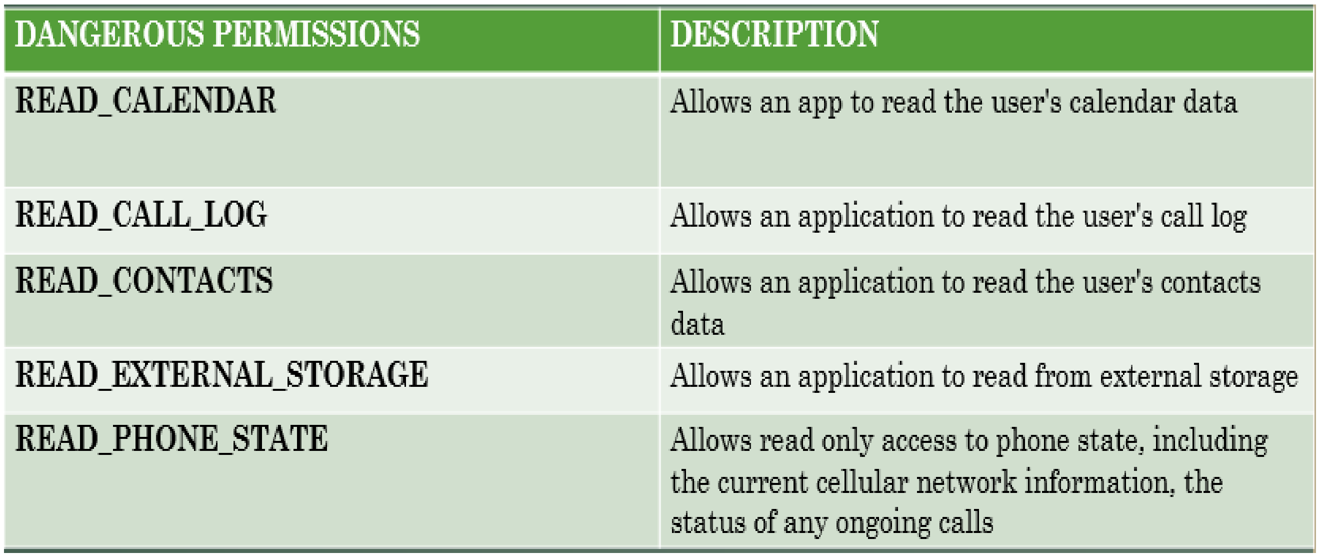
The following analysis has been performed on AndroidManifest file:

- **Dangerous permissions:** Dangerous permissions are the permissions which after granting access from the user can steal personal data of users or affect the phone storage [3]. There are several dangerous permissions in Android operating system as shown in Figure 14, Figure 15, and Figure 16.
- **Malicious application:** An application can unnecessarily ask for any of these dangerous permissions, even if not required. The users should not grant these dangerous permissions, unless an app actually needs it. For example, if a battery saver application is asking for READ_CONTACTS permission, then the user should not grant it. This is because the battery saver application doesn’t require to read the contact of users.



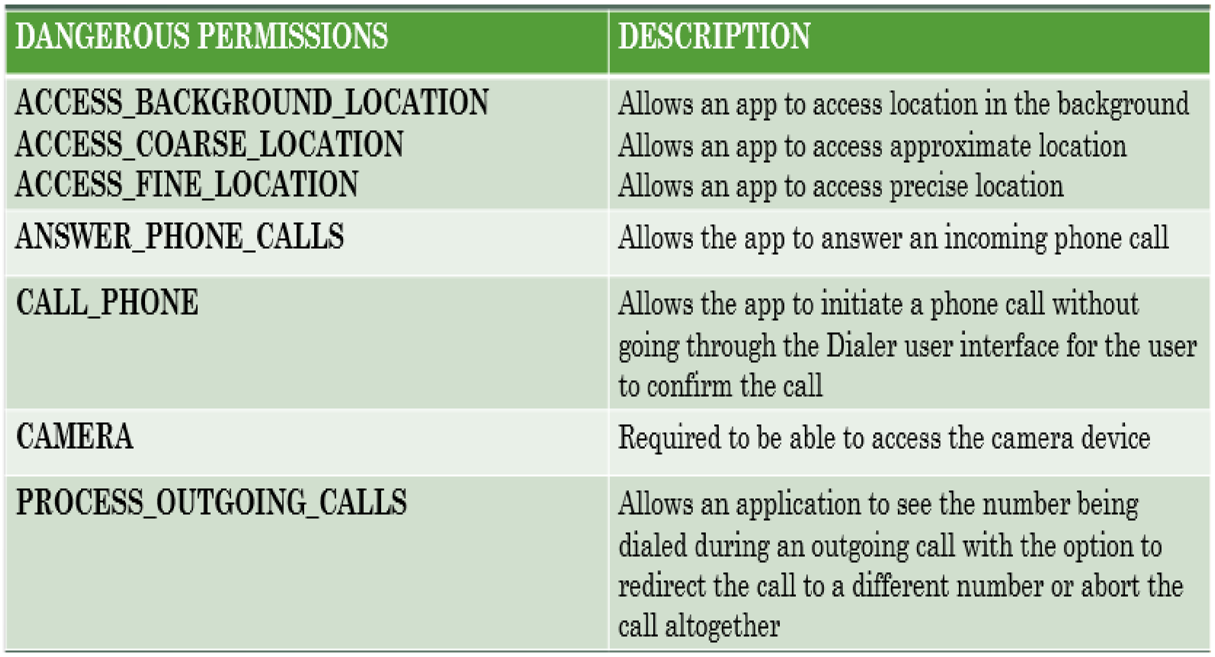
DANGEROUS PERMISSIONS	DESCRIPTION
READ_SMS	Allows an application to read SMS messages
SEND_SMS	Allows an application to send SMS messages
RECORD_AUDIO	Allows an application to record audio
WRITE_EXTERNAL_STORAGE	Allows an application to write to external storage

Figure 14: Dangerous permissions-I



DANGEROUS PERMISSIONS	DESCRIPTION
READ_CALENDAR	Allows an app to read the user's calendar data
READ_CALL_LOG	Allows an application to read the user's call log
READ_CONTACTS	Allows an application to read the user's contacts data
READ_EXTERNAL_STORAGE	Allows an application to read from external storage
READ_PHONE_STATE	Allows read only access to phone state, including the current cellular network information, the status of any ongoing calls

Figure 15: Dangerous permissions-II



DANGEROUS PERMISSIONS	DESCRIPTION
ACCESS_BACKGROUND_LOCATION ACCESS_COARSE_LOCATION ACCESS_FINE_LOCATION	Allows an app to access location in the background Allows an app to access approximate location Allows an app to access precise location
ANSWER_PHONE_CALLS	Allows the app to answer an incoming phone call
CALL_PHONE	Allows the app to initiate a phone call without going through the Dialer user interface for the user to confirm the call
CAMERA	Required to be able to access the camera device
PROCESS_OUTGOING_CALLS	Allows an application to see the number being dialed during an outgoing call with the option to redirect the call to a different number or abort the call altogether

Figure 16: Dangerous permissions-III

REFERENCES

[1] Statista, “Number of smartphone users worldwide from 2016 to 2020,” 2020. https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/ (accessed Jun. 14, 2020).

[2] iBotPeaches, “APKTOOL,” *Github*, 2019. .

[3] Android, “Manifest.permission,” *Android*, 2018. https://developer.android.com/reference/android/Manifest.permission%5C#GET%5C_ACCOUNTS.