REPORT

Tools required to decompile APK file:

- dex2jar - http://sourceforge.net/projects/dex2jar/
- jd-gui - http://jd.benow.ca/
- Apk tool - https://ibotpeaches.github.io/Apktool/install/

Malware APK:

Malware apk taken from: https://virusshare.com/
Details about the APK used:

	MD5	11bc71c430a60e1a8d06d9f52ac90312
	SHA1	824c446d2b4696ed3d73cdd509b109c063c39c7e
	SHA256	b1c8683afd5b5c1192fc1a91381b765a0622a8e695837cc39e16c9e8123be162
SSDeep	12288:FTExPhR5gewkGf0E9fB1xKtkhol7eMcYbk+bklbkebkmTCDEg6gjQe:ZSwd0ETholeXziTD5Qe	
Size	767,421 bytes	
File Type	Zip archive data, at least v2.0 to extract	

De-compilation Steps:

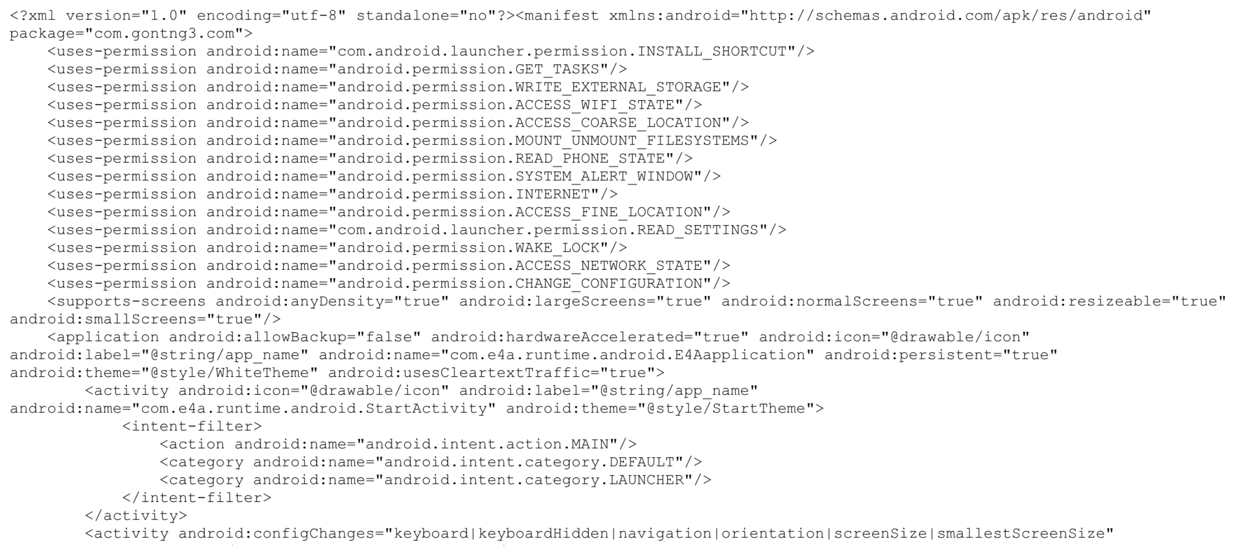
- Create a copy of the apk file and save it as .zip file.
- Extract the .zip file. A folder of the apk file gets created.
- Click on the folder and copy *classes.dex* file.
- Paste the file in the dex2jar folder after you downloaded the tool and extracted it.
- Open cmd and change the directory to where you pasted the classes.dex file and type the following command: *d2j-dex2jar classes.dex*
- After executing the command, you will get "classes-dex2jar.jar" file.

- Launch jd-gui (java decompilation) tool and open the "classes-dex2jar.jar" file in the tool.
- You can then view the .class files of the apk.
- Click on the File and select save all resources option. A classes-dex2jar.jar.src.zip file gets created.
- On extracting the file, we can then view all the java files in the folder which are used by the apk.
- Rename the classes-dex2jar.jar.src folder to src. Rename the apktool jar file to apktool.jar.
- Remove the apk folder created in step 2.
- Open cmd, and cd to where apktool is saved. Run the command:
  apktool d <name of the apk file.apk> (I named the apk "malware")

- A folder named after your apk gets created. Copy the src folder and paste it in your apk folder. Now you got the complete code for your apk application.
- We analyse the code by reading the AndroidManifest.xml file and .java files to find out the intent of the apk.

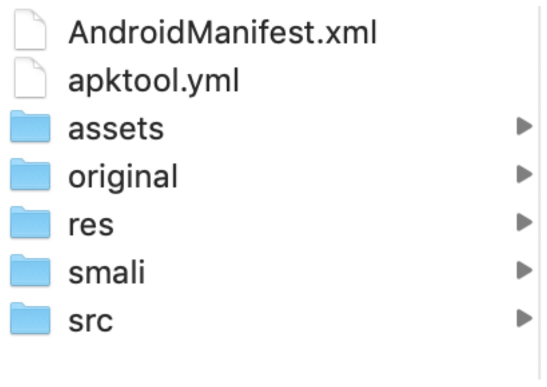
ANALYSIS:

From manifest.xml file, we can see the name of the application file which is "`com.e4a.runtime.android.E4Aapplication`" under the tag <android: label>.Moreover, we can also see what permissions does this app ask for once installed in the victim phone.
Some of the suspicious ones I can see is the one where it asks to access coarse location, get_tasks,read_phone_state,write_external_storage, system_alert_window to name a few.
Below is the screenshot the for xml file:



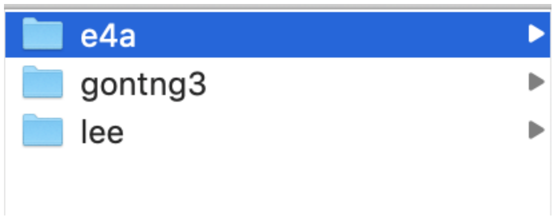
```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.gontng3.com">
    <uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
    <uses-permission android:name="android.permission.GET_TASKS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="com.android.launcher.permission.READ_SETTINGS"/>
    <uses-permission android:name="android.permission.WAKE_LOCK"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.CHANGE_CONFIGURATION"/>
    <supports-screens android:anyDensity="true" android:largeScreens="true" android:normalScreens="true" android:resizeable="true"
android:smallScreens="true"/>
    <application android:allowBackup="false" android:hardwareAccelerated="true" android:icon="@drawable/icon"
android:label="@string/app_name" android:name="com.e4a.runtime.android.E4Aapplication" android:persistent="true"
android:theme="@style/WhiteTheme" android:usesCleartextTraffic="true">
        <activity android:icon="@drawable/icon" android:label="@string/app_name"
android:name="com.e4a.runtime.android.StartActivity" android:theme="@style/StartTheme">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:configChanges="keyboard|keyboardHidden|navigation|orientation|screenSize|smallestScreenSize"
```

Folders inside the malware apk folder



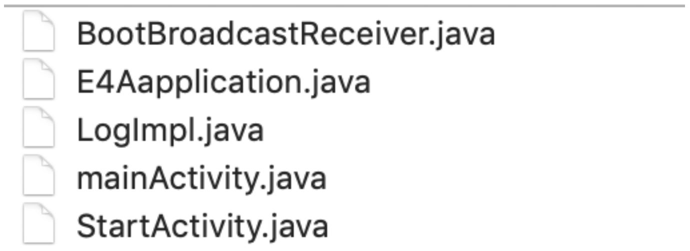
```
AndroidManifest.xml
apktool.yml
assets
original
res
smali
src
```

If you go inside src/com/
You will see 3 folders named:

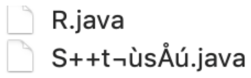

e4a
gontng3
lee

We can all the java files in these folders by following this path:

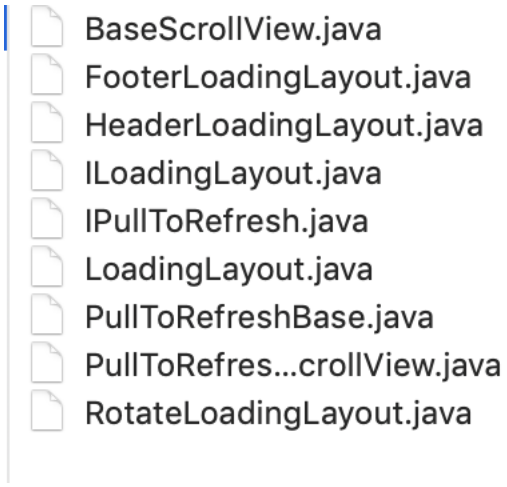
1. e4a/runtime/android/


BootBroadcastReceiver.java
E4Aapplication.java
LogImpl.java
mainActivity.java
StartActivity.java

2. gotng3/com/


R.java
S++t¬ùsÅú.java

3. lee/pullrefresh/ui/


BaseScrollView.java
FooterLoadingLayout.java
HeaderLoadingLayout.java
ILoadingLayout.java
IPullToRefresh.java
LoadingLayout.java
PullToRefreshBase.java
PullToRefres...crollView.java
RotateLoadingLayout.java