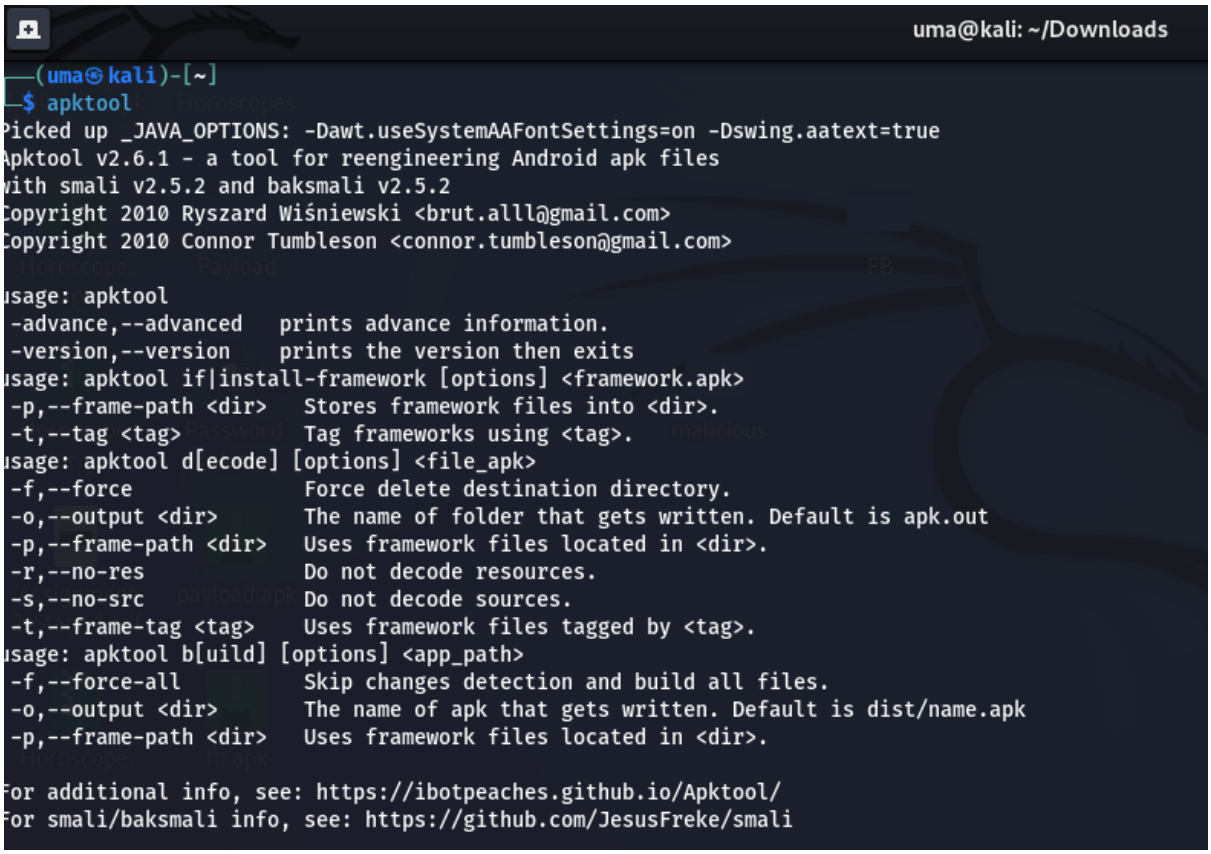
Malicious APK File Analysis
No. 1

1. To open the apktool we use the following command in command prompt- "***apktool***".



2. After that go to the path where apk file is located. Our apk file is located in the Downloads. Then decompile the apk file using the below command – "***apktool d Instagram.apk***", where d is use to decompile the apk file.

```
┌──(uma㉿kali)-[~]
└─$ cd Downloads

┌──(uma㉿kali)-[~/Downloads]
└─$ apktool d instagram.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1 on instagram.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/uma/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

┌──(uma㉿kali)-[~/Downloads]
└─$
```

3. After decompiling the given apk file, the following permissions were granted to the application as observed in "***AndroidManifest.xml***".

➔ Permissions were granted such as Internet, Location, Phone, Contacts, Settings, Storage and etc.

4. There is only one activity named "***MainActivity***" that is intended by the APK as observed in the strings.xml file. This can be a main triggering activity for the APK that can further invoke other actions.

Open

strings.xml
~/Downloads/instagram1/res/values

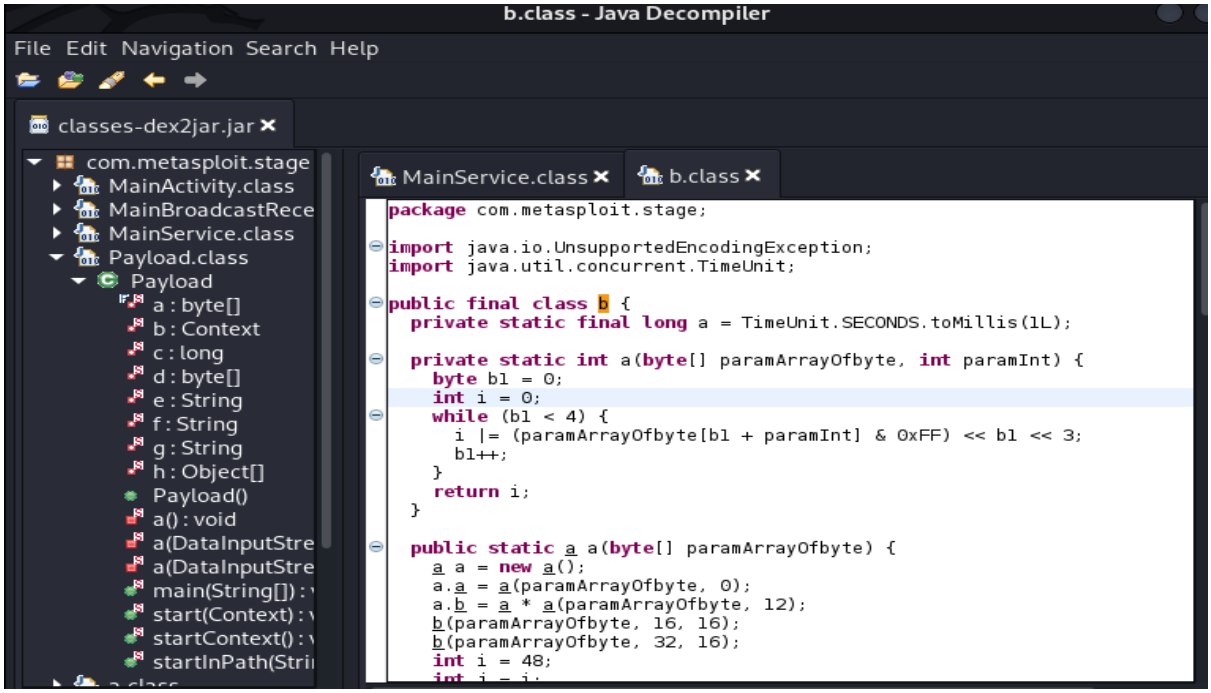
```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">MainActivity</string>
    <string name="secret">HELLO WOrlD!</string>
</resources>
```

5. When we analyze the AndroidManifest.xml further we observe the startup class for the android APK is "***MainActivity"*** as its being bound to the **intent.action.main.**

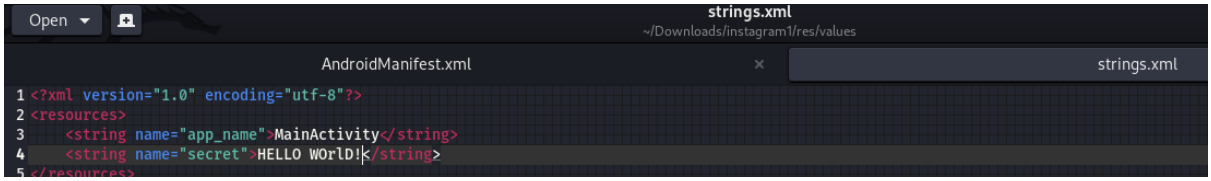
```xml
        <activity android:label="@string/app_name" android:name=".MainActivity" android:theme="@android:style/Theme.NoDisplay">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
            <intent-filter>
                <data android:host="my_host" android:scheme="metasploit"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <category android:name="android.intent.category.BROWSABLE"/>
                <action android:name="android.intent.action.VIEW"/>
            </intent-filter>
        </activity>
```

➔ When analyzing the XML, we see the onCreate service implementation would be available in the MainActivity class as mentioned in the ***andoid:name*** tag.

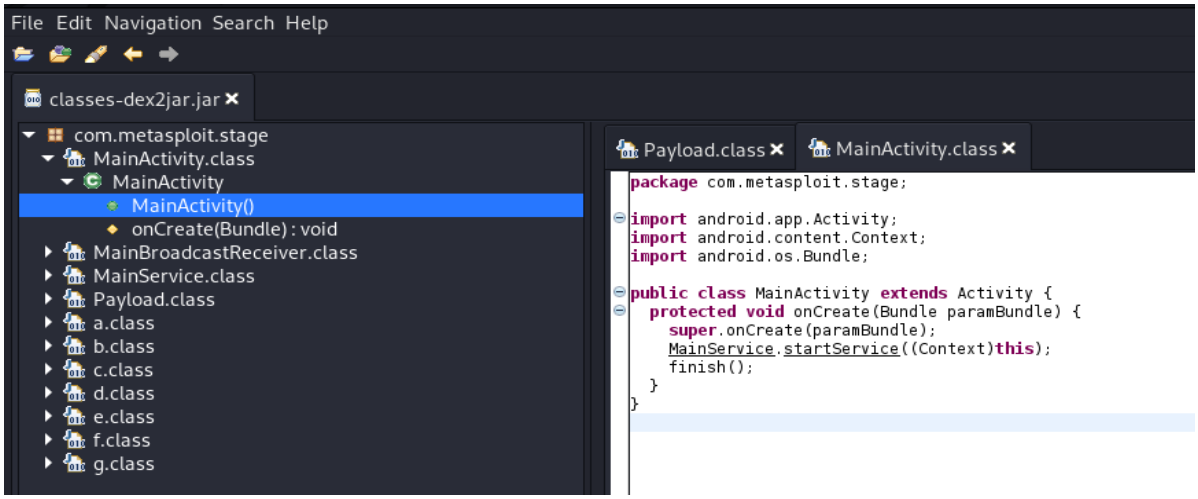
6. This code might be suspicious and it seems to be obfuscated.



7. The secret code we have found for this application as "***HELLO WOrlD!***" as observed in the "***Strings.xml***".

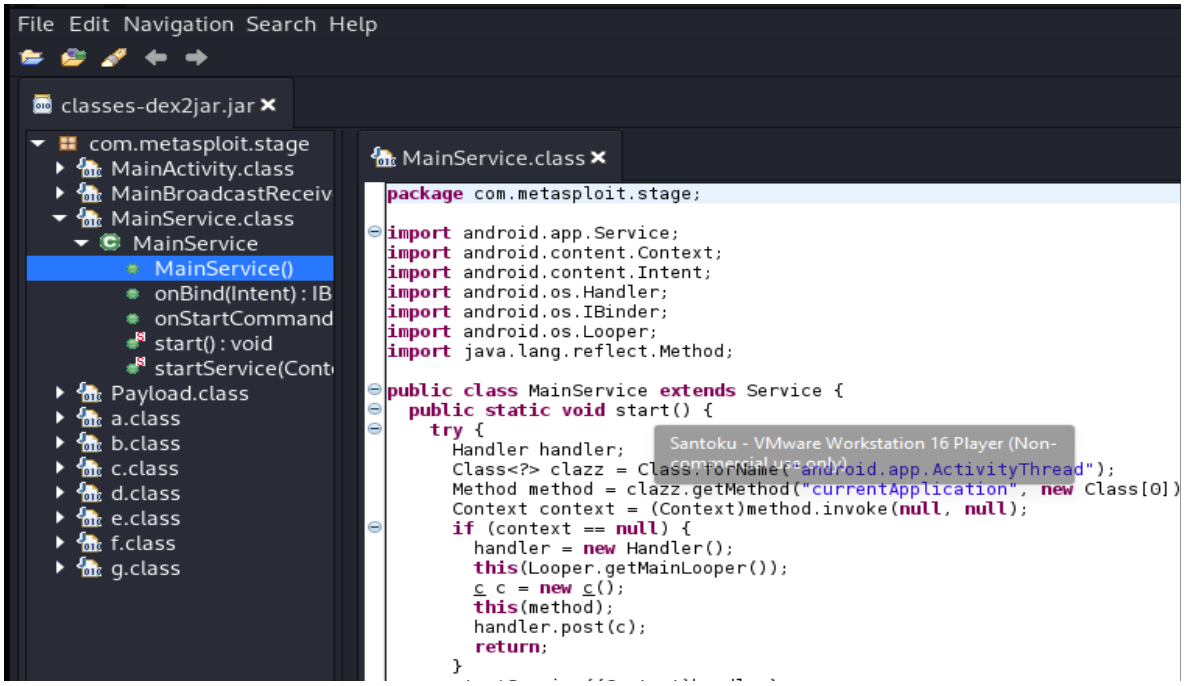


8. The MainActivity class when decompiled with dex2jar on the classes.dex obtained by unzip command on the APK we observe it as follows:



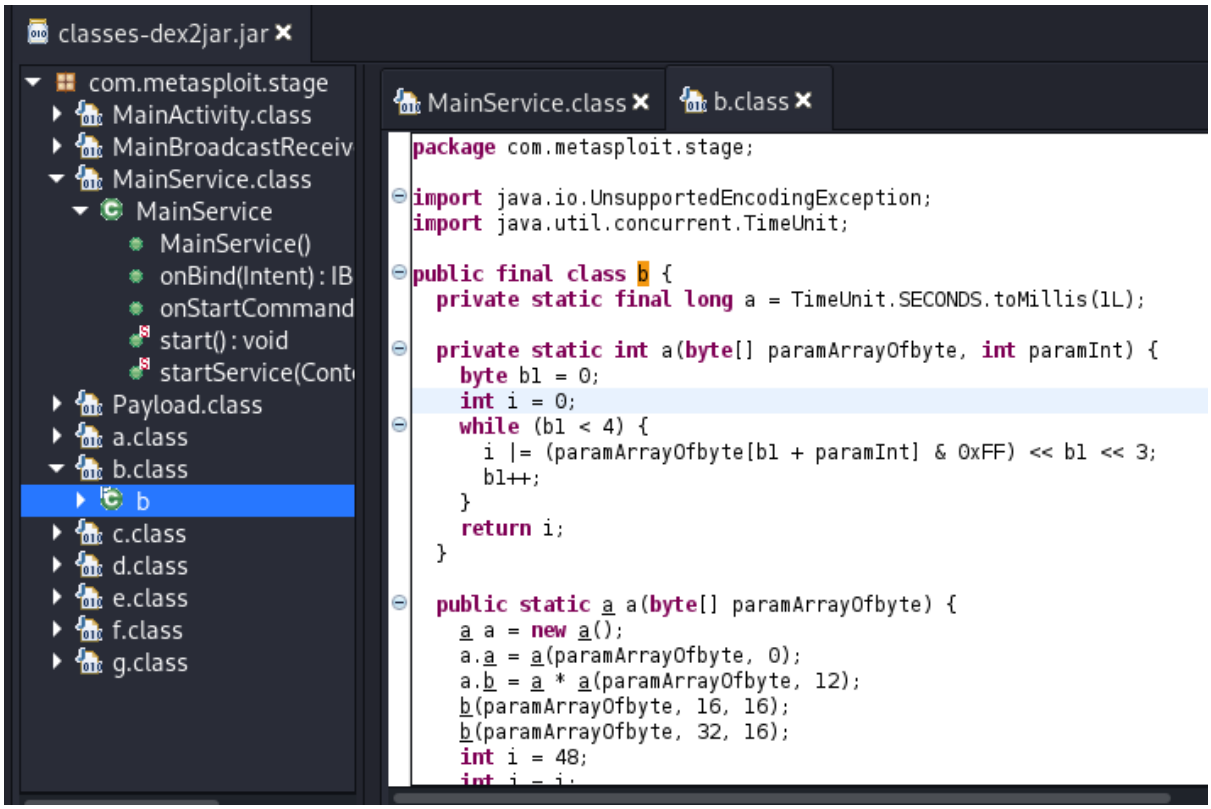
➔ Here the ***MainActivity*** is calling the ***MainService*** class to start service with the activity context when the Booting of the android is completed within the ***onCreate*** method.

9. Now let's see the ***MainService*** class implementation:



➔ Here we can see the start method initiates a thread for the current application and as soon as a context is created, we attach the context to the given handler for posting.

10. Inside b.class generated we se as follows:



➔ There are some offsets being used for the http URL under consideration and some operations like ***Arrays.copyOf*** being applied upon these offsets showing a possible scenario of obfuscation.