Capita selecta: Android security

André Jacobs — r0370664

5 december 2016

1 Application analysis

For this first step I have chosen the application tiny flashlight [1][2]
Using apktool to decompile the apk following permissions were found in the `AndroidManifest.xml` file:

```
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<uses-permission android:name="android.permission.WAKE_LOCK"/>
<uses-permission android:name="android.permission.VIBRATE"/>
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="com.devuni.flashlight.CONTROL_LIGHT"/>
```

I used the included `pyparser.py` script I have written to first parse the `ALL_API_CALLS.txt` file and then look through all the smali files using a series of `grep` operations. The result of this script can be found in the file called `used_apicalls`.
This shows that following permissions were used:

- android.permission.READ_SMS: this seems to be used for reading notitications regarding download requests. This permission is not stated in the manifest.
- android.permission.CHANGE_WIFI_STATE: This is used to process some sort of transaction. This permission is also not stated in the manifest.
- android.permission.NFC: Not stated in the manifest
- android.permission.VIBRATE: used by the AudioManager and NotificationManager of the application. This permission is requested in the manifest.
- com.android.browser.permission.READ_HISTORY_BOOKMARKS: not requested
- android.permission.CAMERA: used to access flashlight, requested in manifest
- android.permission.INTERNET: Used by http client component, requested in manifest
- android.permission.WRITE_EXTERNAL_STORAGE: Used by UI component and to save settings not requested in manifest
- android.permission.ACCESS_FINE_LOCATION: used by the LocationManager, probably for ads. Not requested in manifest.
- android.permission.KILL_BACKGROUND_PROCESSES: used for placing ads in the main UI thread. Not requested in manifest.
- android.permission.READ_PHONE_STATE: Not requested
- android.permission.ACCESS_NETWORK_STATE: requested in manifest
- android.permission.SYSTEM_ALERT_WINDOW: not requested in manifest
- android.permission.WRITE_SETTINGS: not requested in manifest
- android.permission.WAKE_LOCK: requested in manifest

I did the same for malware application `4f4ee687c683e889f204b1a0c86878f198380513`.
Following permissions are defined in the manifest:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
```

Following permissions are used by the application (see: `used_apicalls_mallware`:

- android.permission.VIBRATE: used by the notitication component
- android.permission.INTERNET: used by a notifier
- android.permission.USE_CREDENTIALS: feels like this is the malware part of the app
- android.permission.MANAGE_ACCOUNTS: also part of the malware, trying to steal all kinds of account information.
- android.permission.READ_PHONE_STATE: stealing the phone information (deviceId for example)
- android.permission.GET_ACCOUNTS: more information to steal.
- android.permission.WAKE_LOCK: anysnchronous component of the app.

The malware application seems to be underpriviliged, as for example ethe MANAGE_ACCOUNTS and USE_CREDENTIALS permissions are not listed in the maniffest.

2 Reverse engineering

To remove the calls to the advertisement library I have removed the `AdActivity` call from the `AndroidManifest`. Included apk file `flashlight_no_ads.apk` demonstrates the result.

3 Static analysis

```
Found a flow to sink specialinvoke $r5.<java.net.URL: void <init>(java.lang.String)>($r1)
from the following sources:
- $r4 = virtualinvoke $r3.<android.os.Bundle: java.lang.String getString(java.lang.String)>("referrer")
        (in <com.typ3studios.airhorn.MyReferrerReceiver:
        void onReceive(android.content.Context,android.content.Intent)>)
- $r9 = virtualinvoke $r8.<android.accounts.AccountManager:
        android.accounts.Account[] getAccounts()>()
        (in <com.typ3studios.airhorn.MyReferrerReceiver: void getUserInfo(android.content.Context)>)
- $r3 = virtualinvoke $r6.<android.telephony.TelephonyManager:
        java.lang.String getLine1Number()>()
        (in <com.typ3studios.airhorn.MyReferrerReceiver: void getUserInfo(android.content.Context)>)
- $r1 := @parameter0: android.content.Context
        (in <com.typ3studios.airhorn.MyReferrerReceiver:
        void onReceive(android.content.Context,android.content.Intent)>)
- $r3 = virtualinvoke $r6.<android.telephony.TelephonyManager: java.lang.String getDeviceId()>()
        (in <com.typ3studios.airhorn.MyReferrerReceiver: void getUserInfo(android.content.Context)>)

Sink virtualinvoke $r1.<android.content.Context: android.content.ComponentName
startService(android.content.Intent)>($r2)
from the following sources:
- $r1 := @parameter0: android.content.Context (in <com.and.snd.StartAtBootServiceReceiver:
        void onReceive(android.content.Context,android.content.Intent)>)
```

4 Instrumentation for dynamic analysis

5 Testin and triggering

I used monkey to do automated testing, running the `adb monkey` command with my application installed on the emulator.

Referenties

[1] https://play.google.com/store/apps/details?id=com.devuni.flashlight&utm_source=www.apk4fun.com

[2] https://www.apk4fun.com/apk/1823/