Security in the IoT World:
Analyzing the Security of Mobile Apps for Automobiles

** Supplement **

Naohide Waguri
FFRI, Inc.
Evaluating and Scoring the Risk of Vulnerabilities
CVSS c3 base score of the found vulnerabilities

• What is CVSS?

  – CVSS is the abbreviation for the “Common Vulnerability Scoring System”.

  – It is one of the methods that is a generic and open to evaluating a risk of vulnerability.

  – v3 is focused on characteristics of vulnerable component compared to V2 for considering a scope of influence by the vulnerability.

  – To know details of CVSS, see also references page.
Vulnerability #1: HTTP communication that contains user information

Base Score 7.3 (High)

Note:
- An attacker is possible to obtain some user information and modify the communication data, but they would not be a serious impact to the component directly.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack Vector (AV)</td>
<td>Network</td>
</tr>
<tr>
<td>Attack Complexity (AC)</td>
<td>Low</td>
</tr>
<tr>
<td>Privileges Required (PR)</td>
<td>None</td>
</tr>
<tr>
<td>User Interaction (UI)</td>
<td>None</td>
</tr>
<tr>
<td>Scope (S)</td>
<td>Unchanged</td>
</tr>
<tr>
<td>Confidentiality (C)</td>
<td>Low</td>
</tr>
<tr>
<td>Integrity (I)</td>
<td>Low</td>
</tr>
<tr>
<td>Availability (A)</td>
<td>Low</td>
</tr>
</tbody>
</table>
**Vulnerability #2: Server certificate validation flaw**

**Base Score 6.4 (Medium)**

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Assigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack Vector (AV)</td>
<td>Adjacent</td>
</tr>
<tr>
<td>Attack Complexity (AC)</td>
<td>High</td>
</tr>
<tr>
<td>Privileges Required (PR)</td>
<td>None</td>
</tr>
<tr>
<td>User Interaction (UI)</td>
<td>Required</td>
</tr>
<tr>
<td>Scope (S)</td>
<td>Unchanged</td>
</tr>
<tr>
<td>Confidentiality (C)</td>
<td>High</td>
</tr>
<tr>
<td>Integrity (I)</td>
<td>High</td>
</tr>
<tr>
<td>Availability (A)</td>
<td>Low</td>
</tr>
</tbody>
</table>

Note:
- The app has a potential to be MITM attacked because it does not validate an SSL server certificate.
- There is a potential that an attacker intercepts a communication data that should be protected (e.g., authenticates credential).

**Why did we assign “Required” to User Interaction?**
- The app does not communicate to a server on HTTPS until a user taps the “login” button.
- Auto login will be performed if a user configures own.
The First Step for Reverse Engineering an Android App
How to reverse engineering an Android app

• At the CODE BLUE 2016, I did not talk detail about how to reverse engineering the apps that were detected critical vulnerabilities by AndroBugs.

• In this paper, I introduce some tools used for reverse engineering an Android app.
A Flow of reverse engineering an Android app

• There are 3 steps to reverse engineering an Android app.

#1. Getting a target APK
- Getting from official marketplace or extract APKs using adb
- Getting from unofficial marketplace

#2. Unpack an APK file
- Using the apktool to unpack and decode to manifest and smali files
- Unzip to get dex file to decompile

#3. Decompile to java from DEX
- Actually, there are some step to decompile to java (DEX → JAR → class → java)

Have fun ☺
#1. Getting a target APK file

- There are 2 ways to get an APK file.
  - Extract from a device using adb (Android Debug Bridge).
  - Download from an unofficial marketplace and so on.

- To use adb, you need to install Android SDK in advance.

- If you use adb in Windows OS, I recommend to install a grep-like command because it helps to search a target APK using adb.
#1. Getting a target APK file (cont.)

- **Step1. Check the installed packages in a device.**
  - “pm list packages” is able to enumerate packages that are in a target device.
  - “-f” option is output packages associated file.

```
jcnuts@jcnuts:$ adb shell pm list packages -f | grep google
package:/data/app/com.google.android.apps.books-1.apk=com.google.android.apps.books
package:/data/app/com.google.android.apps.docs-1.apk=com.google.android.apps.docs
```

- **Step2. Download a target package (APK) from a device.**
  - “pull” command is able to download (pull) a package to your PC.

```
jcnuts@jcnuts:~/re_apks$ adb pull /data/app/com.google.android.apps.maps-2.apk
953 KB/s (28180726 bytes in 28.863s)
jcnuts@jcnuts:~/re_apks$ ls
com.google.android.apps.maps-2.apk
jcnuts@jcnuts:~/re_apks$
```
#2. Unpack an APK file

- APK can unzip the same as ZIP.
  - However, most of the files that you obtain by unzipping are a binary format that is hard to analyze.

- The `apktool` provides some features to analyze more easily.
  - Decoding resources and manifest file.
  - Baksmaling a dex file.
#2. Unpack an APK file (cont.)

- apktool is available to download from the following link: [https://ibotpeaches.github.io/Apktool/](https://ibotpeaches.github.io/Apktool/)

```bash
jcnuts@jcnuts:~/re_apks$ apktool d sample.apk
I: Using Apktool 2.0.0-RC2 on sample.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/jcnuts/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
Cleaning up unclosed ZipFile for archive /home/jcnuts/apktool/framework/1.apk
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
jcnuts@jcnuts:~/re_apks$
```

- You can skip this process if you do not need resource and manifest files for reverse engineering.
#3. Decompile to java from DEX

• There are some steps to get to java source code.

• **Step1. Get a DEX from APK**
  – You can find a DEX file in the folder that has been created when unzipped an APK.

```
jcnuts@jcnuts:$ ls -l
total 5924
-rw-rw-r-- 1 jcnuts jcnuts 15404 Aug 18 15:55 AndroidManifest.xml
drwxrwxr-x 8 jcnuts jcnuts 4096 Oct 27 11:43 assemblies
drwxrwxr-x 4 jcnuts jcnuts 4096 Oct 27 11:43 assets
-rw-rw-r-- 1 jcnuts jcnuts 4689744 Aug 18 15:56 classes.dex
-rw-rw-r-- 1 jcnuts jcnuts 54 Aug 18 15:56 environment
drwxrwxr-x 5 jcnuts jcnuts 4096 Oct 27 11:43 lib
drwxrwxr-x 2 jcnuts jcnuts 4096 Oct 27 11:43 META-INF
-rw-rw-r-- 1 jcnuts jcnuts 157 Aug 18 15:56 NOTICE
drwxrwxr-x 3 jcnuts jcnuts 4096 Oct 27 11:43 org
drwxrwxr-x 22 jcnuts jcnuts 4096 Oct 27 11:43 res
-rw-rw-r-- 1 jcnuts jcnuts 450624 Aug 18 15:55 resources.arsc
jcnuts@jcnuts:$
```
#3. Decompile to java from DEX (cont.)

- **Step2. Convert to JAR from DEX**
  - Use a dex2jar to convert to JAR from DEX.
  - dex2jar is available to download the following link:

```
jcnuts@jcnuts:~/re_apks/sample_unzipped$ ls -l
 total 10448
 -rw-rw-r-- 1 jcnuts jcnuts 15404 Aug 18 15:55 AndroidManifest.xml
 drwxrwxr-x 8 jcnuts jcnuts  4096 Oct 27 11:43 assemblies
 drwxrwxr-x 4 jcnuts jcnuts  4096 Oct 27 11:43 assets
 -rw-rw-r-- 1 jcnuts jcnuts 4689744 Aug 18 15:56 classes.dex
 -rw-rw-r-- 1 jcnuts jcnuts 4630701 Oct 27 11:49 classes-dex2jar.jar
 -rw-rw-r-- 1 jcnuts jcnuts     54 Aug 18 15:56 environment
 drwxrwxrwx-x 5 jcnuts jcnuts  4096 Oct 27 11:43 lib
 drwxrwxr-x 2 jcnuts jcnuts  4096 Oct 27 11:43 META-INFO
 -rw-rw-r-- 1 jcnuts jcnuts  157 Aug 18 15:56 NOTICE
 drwxrwxr-x 3 jcnuts jcnuts  4096 Oct 27 11:43 org
 drwxrwxr-x 22 jcnuts jcnuts  4096 Oct 27 11:43 res
 -rw-rw-r-- 1 jcnuts jcnuts  450624 Aug 18 15:55 resources.arsc
 jcnuts@jcnuts:~/re_apks/sample_unzipped$
jcnuts@jcnuts:~/re_apks/sample_unzipped$
#3. Decompile to java from DEX (cont.)

- **Step 3. Get class files from JAR**
  - Unzip a JAR (JAR can unzip the same as ZIP) to get class files.

- **Step 4. Decompile to java from class files.**
  - Use a Java Decompiler (JD-GUI) to decompile to java from class files.
  - Java Decompiler is available to download from the following link: [http://jd.benow.ca/](http://jd.benow.ca/)
#4. What to do next?

• In this step, you ought to have already got java, smali, decoded manifest file.
  – Therefore, the next step you should be finding analysis entry point.

• How do I find an analysis entry point?
  – There are various ways to find it.
  – Deeper analysis often may be dependent on the intuition and experience (more knowledges of vulnerability and how to exploit them).

• The following examples do not depend on the intuition and experience so much 😊
  – Use vulnerability scanners like AndroBugs, they would help to find an entry point for analysis.
  – To understand common vulnerabilities, read “Android Application Secure Design/Secure Coding Guidebook”.
Introduction of other tools and distributions

- **Androguard** ([https://github.com/androguard/androguard](https://github.com/androguard/androguard))
  - Androguard is an analysis tool for Android apps that is written in full python.
  - AndroBugs, introduced at CODE BLUE, also uses Androguard.
  - Androguard might help to automate analysis of an Android app.

- **Santoku Linux** ([https://santoku-linux.com/](https://santoku-linux.com/))
  - Santoku Linux is one of the Linux distributions for mobile forensics, analysis, and security testing.
  - It was presented at RSA Conference 2014.
  - Most of the famous tools that help with forensics, analysis, and security testing, including those for a mobile device and an app, are pre-installed in Santoku Linux.
Conclusions

• The risk score of vulnerabilities that I found was high.
  – However, the results are only the base score.
  – In generally, risk score will shift to low score finally by consideration of the temporal score and the environmental score.

• Reverse engineering of an Android app is not so hard.
  – There is a lot of information on the Internet.
  – There is a lot of helpful tools for analysis.
  – But deeper analysis may need the intuition and experience.

• The analysis automation also would be possible.
  – Most of the tools provide command line execution.
  – For example, Androguard is utilized in various analysis tools.
    (e.g., Cuckoo, Viper, AndroBugs etc,...)
References

• 共通脆弱性評価システム CVSS v3概説
• Common Vulnerability Scoring System, V3 Development Update
  – https://www.first.org/cvss
• Dalvik bytecode
• Cuckoo Sandbox
  – https://cuckoosandbox.org/
• Viper
  – http://www.viper.li/
• AndroBugs
  – https://github.com/AndroBugs/AndroBugs_Framework

• If you want to know how to use Androguard, try seeing the following link:
• Part 1 – Reverse engineering using Androguard