October 21, 2016
CODE BLUE 2016

IoTとしての自動車とセキュリティ：リモートサービスのセキュリティ評価とその対策の検討＜補足資料＞

株式会社ＦＦＲＩ
Naohide Waguri

脆弱性のリスク評価とスコアリング

発見した脆弱性のCVSS V3基本値

- CVSSとは？
  - CVSS は “Common Vulnerability Scoring System（共通脆弱性評価システム）” の略。
  - 脆弱性の特性を評価し、スコアリングする汎用的でオープンな手法の１つ。
  - V3 では、脆弱性の影響範囲を加味するため、V2に比べて脆弱なコンポーネントの特性にフォーカスしている。
  - より詳細を知りたい人は「参考資料」スライドを参照。

脆弱性 #1:
ユーザ情報を含むHTTP通信

基本値
7.3 (重要)

備考：
- 攻撃者は通信データからユーザ情報を得たり改ざんを行う事が出来るが、これによってコンポーネントに直接重大な影響を与える事はない。

Metrics	Assigned
攻撃元区分 (AV)	ネットワーク
攻撃条件の複雑さ (AC)	低
必要な特権レベル (PR)	不要
ユーザ関与レベル (UI)	不要
スコープ (S)	変更なし
機密性への影響 (C)	低
完全性への影響 (I)	低
可用性への影響 (A)	低

脆弱性 #2:
脆弱なサーバ証明書検証

基本値
6.4 (警告)

備考：
- SSLサーバー証明書の検証を行っていないため、このアプリはMITM攻撃をうける可能性がある。
- その結果、攻撃者によって保護されるべき通信データを傍受される可能性がある。（例：認証クレデンシャルなど）

Metrics	Assigned
攻撃元区分 (AV)	隣接
攻撃条件の複雑さ (AC)	高
必要な特権レベル (PR)	不要
ユーザ関与レベル (UI)	要
スコープ (S)	変更なし
機密性への影響 (C)	高
完全性への影響 (I)	高
可用性への影響 (A)	低

なぜユーザ関与レベルが「要」なのか？
該当の脆弱性をもつアプリは、アプリ起動時ではなくログイン情報を入力後に「ログイン」ボタンを押さないとサーバーとのHTTPS通信を行わない。
また、ログイン後の自動ログインについてもチェックボックス上でユーザーが自動ログインを許可しない限り行われない。

Androidアプリの
リバースエンジニアリングの基本

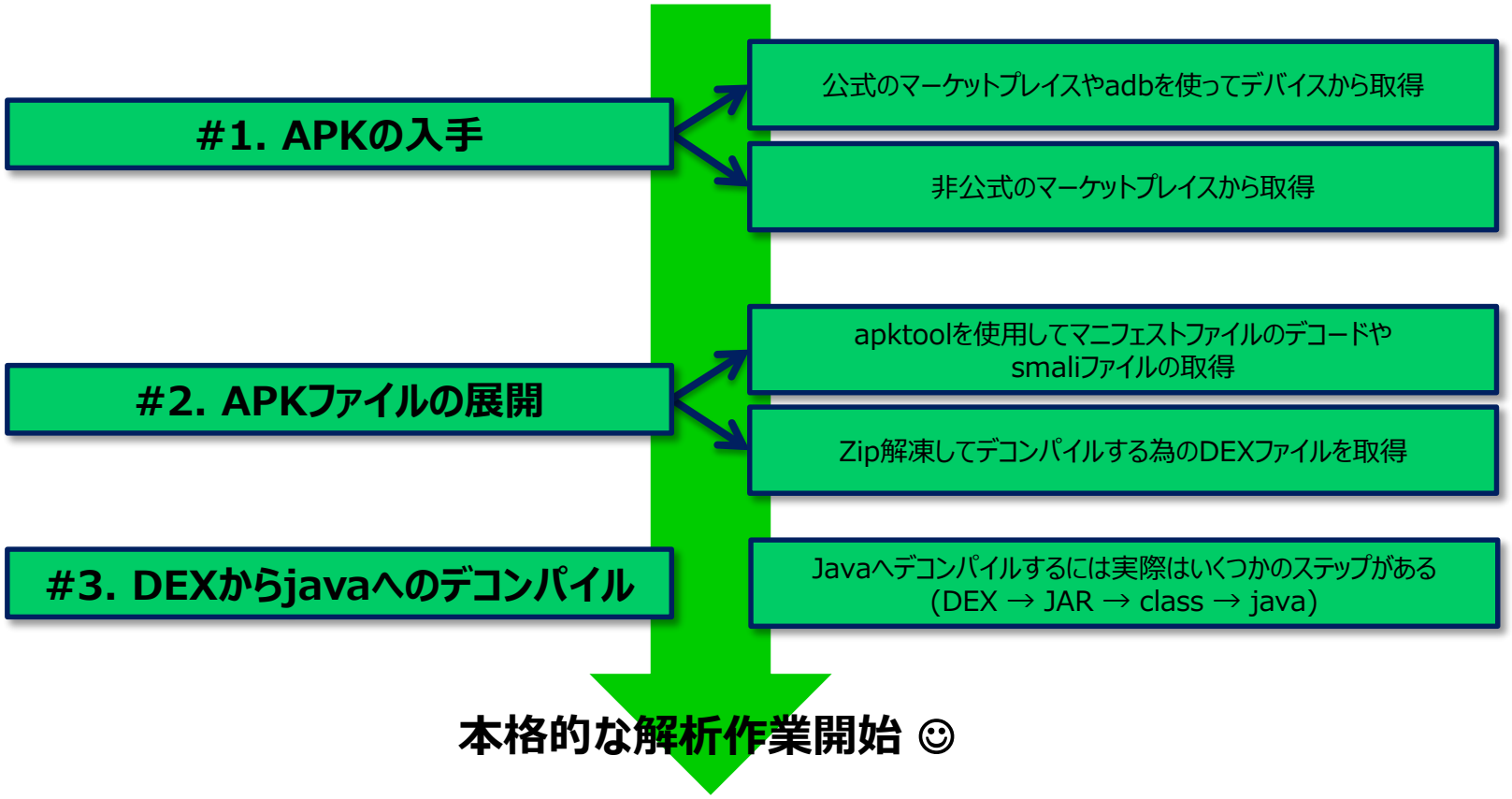
Androidアプリをリバースエンジニアリングする方法

- CODE BLUE 2016 では AndroBugs が検出した脆弱性に対する Android アプリのリバースエンジニアリング方法には触れなかった。

- 本スライドでは、Android アプリをリバースエンジニアリングする為に使用したツールについて紹介する。

Android アプリをリバースエンジニアリングする流れ

- Android アプリをリバースエンジニアリングするには3つのステップがある



#1. APKの入手

- APK ファイルの入手には２つの方法がある。
  - adb（Android Debug Bridge）を使用してデバイスから取得。
  - 非公式のマーケットプレイスなどからダウンロードする。

- adb を使用するには事前に Android SDK をインストールしておく必要がある。

- adb を Windows OS で使う場合、”grep”のようなコマンドをインストールしておくと、対象 APK を探すのに役立つ。

#1. APKの入手 (続き)

- **Step1. デバイスにインストールされているパッケージのチェック**
  - “pm list packages” は対象デバイスにあるパッケージを列挙できる。
  - “-f” オプションはパッケージに関連するファイルを出力する。

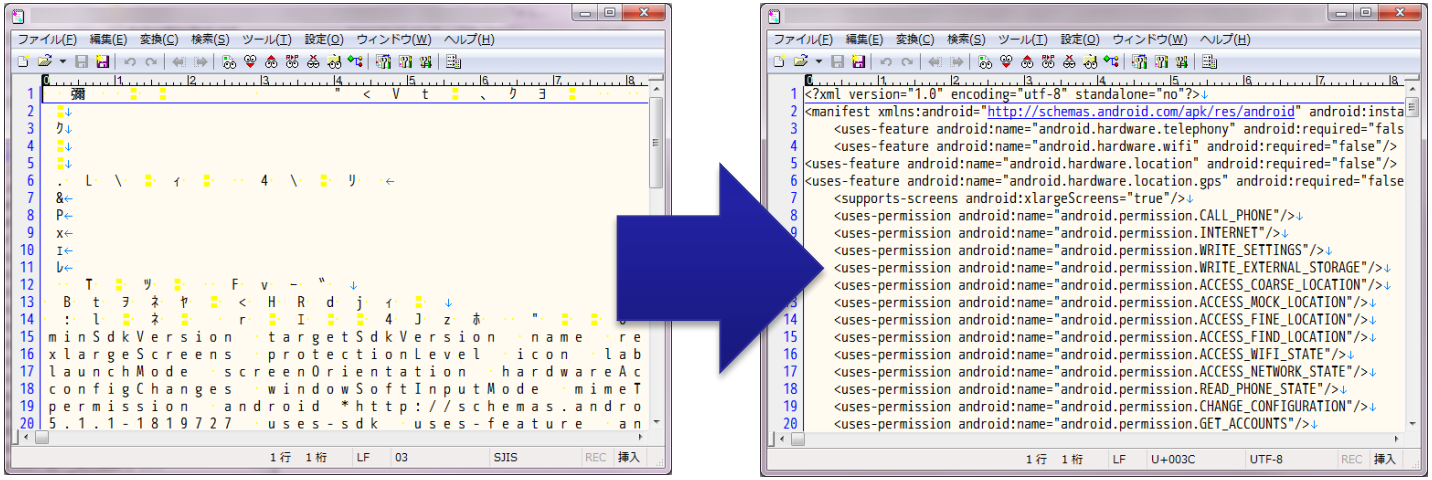
```
jcnuts@jcnuts:~$ adb shell pm list packages -f | grep google
package:/data/app/com.google.android.apps.books-1.apk=com.google.android.apps.books
package:/data/app/com.google.android.apps.docs-1.apk=com.google.android.apps.docs
```

- Step2. デバイスから対象パッケージ（APK）のダウンロード
  - “pull” コマンドはパッケージを PC にダウンロード（pull）できる。

```
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. APKファイルの展開

- APKはZIP同様に解凍できる。
  - しかし、大抵のファイルはバイナリフォーマットなので解析は難しい☹
- apktool は 解析を容易にするための機能をいくつか備えている。
  - リソースファイルやマニフェストファイルのデコード。
  - DEXファイルをsmali変換（baksmali）。



#2. APKファイルの展開（続き）

- apktool は 下記のリンクからダウンロードできる：
  https://ibotpeaches.github.io/Apktool/

```
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$
```

- リバースエンジニアリングする際にデコードされたマニフェストやリソースファイルが不要な場合はこのフェーズはスキップしても良い。

#3. DEXからjavaへのデコンパイル

- javaのソースコードを取得するにはいくつかのステップがある

- **Step1. APKファイルからDEXの取得**
  - APKファイルをZIP展開すると classes.dex が取得できる。

```
jcnuts@jcnuts:~/re_apks/sample_unzipped$ 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:~/re_apks/sample_unzipped$
```

#3. DEXからjavaへのデコンパイル（続き）

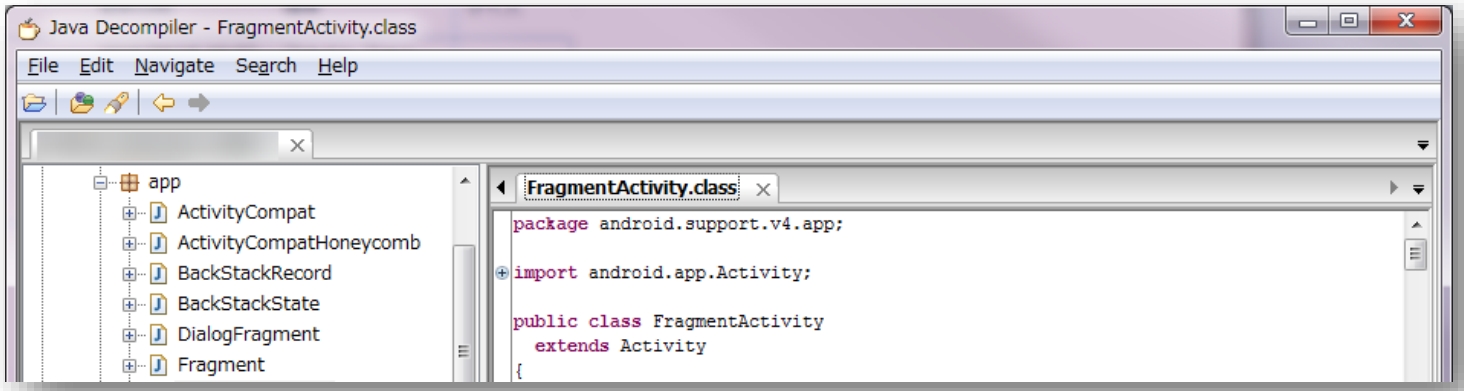
- Step2. DEXからJARへの変換
  - DEXからJARを変換するには dex2jar を使用する。
  - dex2jar は以下のリンクから入手可能：
    https://github.com/pxb1988/dex2jar

```
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
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:~/re_apks/sample_unzipped$
```

#3. DEXからjavaへのデコンパイル（続き）

- **Step3. JARからclassファイルの取得**
  - Classファイルを取得する為にJARを解凍（JARはZIP同様に解凍可能）。

- Step4. classファイルからjavaへのデコンパイル
  - ClassファイルからjavaへのデコンパイルにはJava Decompiler (JD-GUI)を使用する。
  - Java Decompiler は下記リンクから入手可能：
    http://jd.benow.ca/



#4. 次は何をするか？

- この段階で既にjavaやsmali、デコードされたマニフェストファイルを持っているはず。
  - そのため、次は解析のエントリポイントを探す作業をする。

- **解析のエントリポイントを探す方法**
  - いろいろな方法がある。
  - より深い解析には勘や経験が依存する場合がある。（より多くの脆弱性とそれらをエクスプロイトする為の知識が必要）

- **下記の例はあまり勘や経験に依存せずにエントリポイントを探せる☺**
  - AndroBugs のような脆弱性スキャナは解析のエントリポイントを探すのに役立つ。
  - 一般的な脆弱性を知るために、”Android セキュア設計・セキュアコーディングガイド”を読む。

その他のツールとディストリビューションの紹介

- **Androguard (https://github.com/androguard/androguard)**
  - Androguard は Android アプリを解析するためのツールで、全て Python で記述されている。
  - CODE BLUE 2016 で紹介した AndroBugs も Androguard を使用している。
  - Androguard はアプリの解析の自動化に役立つ可能性がある。

- **Santoku Linux (https://santoku-linux.com/)**
  - Santoku Linux はモバイルフォレンジックや解析、セキュリティテストを行う為の Linux ディストリビューションの１つ。
  - RSA Conference 2014 で発表された。
  - モバイルデバイスやアプリのフォレンジックや解析、セキュリティテストを行う為の代表的なツールがプリインストールされている。

まとめ

- **発見した脆弱性のリスク値は高い**
  - しかし、この結果は基本値のみである。
  - 一般的に、リスク値は現状値や環境値を考慮する事によって低下していく傾向にある。

- **Android アプリのリバースエンジニアリングはそんなに難しくない**
  - 多くの情報がインターネット上にある。
  - 多くの解析に役立つツールがある。
  - ただし、より深い解析には勘や経験が必要になる事がある。

- **解析の自動化も可能**
  - 多くのアプリはコマンドライン実行をサポートしている。
  - 例えば、Androguard は様々な解析ツールで利用されている。
    （例：Cuckoo、Viper、AndroBugs 等･･･）

参考資料

- 共通脆弱性評価システム CVSS v3概説
  - https://www.ipa.go.jp/security/vuln/CVSSv3.html
- Common Vulnerability Scoring System, V3 Development Update
  - https://www.first.org/cvss
- Dalvik bytecode
  - https://source.android.com/devices/tech/dalvik/dalvik-bytecode.html
- Cuckoo Sandbox
  - https://cuckoosandbox.org/
- Viper
  - http://www.viper.li/
- AndroBugs
  - https://github.com/AndroBugs/AndroBugs_Framework

- Androguard の使い方について知りたい場合は、下記を参照：
- Part 1 – Reverse engineering using Androguard
  - http://www.technotalkative.com/part-1-reverse-engineering-using-androguard/