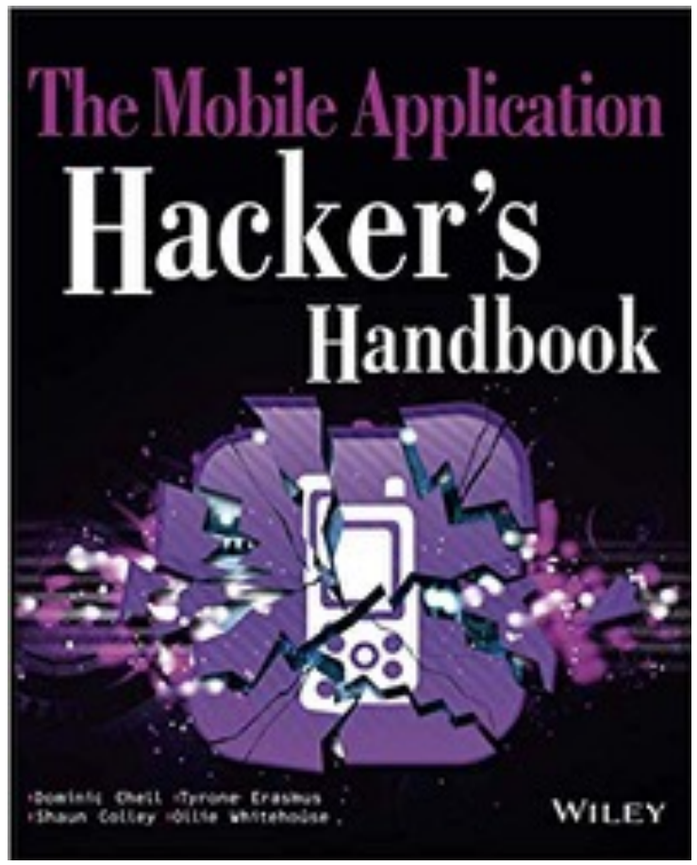
CNIT 128
Hacking Mobile Devices



8. Identifying and Exploiting Android Implementation Issues Part 1

Updated 10-24-22

Topics

- Part 1
  - Reviewing Pre-installed Applications
  - Exploiting Devices
    - Start through "Explanation of Privilege Levels" (up to p. 375)

Topics

- Part 2
  - Exploiting Devices
    - "Practical Physical Attacks" (p. 376) through
    - "Man-in-the-Middle Exploits" (up to p. 401)

Topics

- Part 3
- Exploiting Devices
  - "Injecting Exploits for JavaScript Interfaces" (p. 401) and following
- Infiltrating User Data

Reviewing Pre-Installed Applications

Root Access

- Each installed app has its own attack surface
- But when you exploit an app, you get access with the privileges of that app
  - Not root access
- But you can often exfiltrate user data without root access

Find Powerful Apps

INSTALL PACKAGES

- Exploiting an app with this permission allows an attacker to install a Trojan app
- Permission level **signature|system**
  - Defined by the **android** package

Drozer on an Emulator

```
drozer Console (v2.4.4)
dz> run app.package.list -p android.permission.INSTALL_PACKAGES
com.android.vending (Google Play Store)
com.google.android.packageinstaller (Package installer)
com.android.managedprovisioning (Work profile setup)
com.android.shell (Shell)
dz>
```

- Real devices have many more apps with this dangerous permission

Apps Running as System

- On an emulator
- Many more on a real device (66 in book)

```
dz> run app.package.list -u 1000
com.genymotion.systempatcher (com.genymotion.systempatcher.SystemPatcherApp)
android (Android System)
com.android.providers.settings (Settings Storage)
com.android.inputdevices (Input Devices)
com.android.server.telecom (Call Management)
com.android.keychain (Key Chain)
com.android.settings (Settings)
com.android.wallpaperbackup (com.android.wallpaperbackup)
com.genymotion.genyd (com.genymotion.genyd.GenydServiceApp)
com.android.location.fused (Fused Location)
com.genymotion.superuser (Superuser)
com.android.development (Dev Tools)
```

Finding Remote Attack Vectors

Techniques

- Trick user into installing a malicious app
- **Server-side**: exploit a listening port
- **Client-side**: open a malicious document

Browsers and Document Readers

- Frequently vulnerable
  - Complex parsers written in native code
  - Fuzzers can fund vulnerabilities
- Samsung has Polaris Viewer for PDFs by default
  - No PDF reader on my emulator

```
dz> run app.activity.forintent --action android.intent.action.VIEW
--mimetype application/pdf
```

BROWSEABLE Activities

- Allows users to open content inside an installed app rather than the browser
- App stores installed on the device use this functionality
  - To open links that point to apps

Example

- Manifest from a rogue Drozer agent
- Opening a link starting with **pwn://** will open this activity
  - But can't be used in an iframe anymore

```
<activity
    android:name="com.mwr.dz.PwnActivity">
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.DEFAULT" />
        <category android:name="android.intent.category.BROWSABLE" />
        <data android:scheme="pwn" />
    </intent-filter>
</activity>
```

Two Methods

- Via **pwn://** URI or "web intent"

```
<a href="pwn://me">Start drozer<a>

<a href="intent://me/#Intent;scheme=pwn;end">
Start Drozer</a>
```

Many apps use BROWSABLE filters on my emulator

```
dz> run scanner.activity.browsable
Package: com.android.cts.priv.ctsshim
  Invocable URIs:
  Classes:
    .InstallPriority

Package: com.fidelity.android
  Invocable URIs:
    fidelity://open
    https://@2131755937
  Classes:
    com.fidelity.android.activity.AppConfigLoaderActivity

Package: com.android.messaging
  Invocable URIs:
    sms://
    mms://
  Classes:
    .ui.conversation.LaunchConversationActivity

Package: com.blackboard.android.central.berkeleycollege
  Invocable URIs:
    com.blackboard.android.central.berkeleycollege://
    kurogo://auth/ (PATTERN_LITERAL)
  Classes:
    modolabs.kurogo.activity.ModuleActivity
    modolabs.kurogo.activity.LoginActivity
```

Custom Update Mechanisms

- Apps often write their own update mechanisms
  - Rather than using the Play Store
  - This requires the **INSTALL_PACKAGES** permission
  - Code may be vulnerable
  - May check for a new file over HTTP or broken HTTPS

Remote Loading of Code

Developers SIGN IN

DexClassLoader

```
open class DexClassLoader : BaseDexClassLoader
```

kotlin.Any
↳ java.lang.ClassLoader
↳ dalvik.system.BaseDexClassLoader
↳ dalvik.system.DexClassLoader

A class loader that loads classes from `.jar` and `.apk` files containing a `classes.dex` entry. This can be used to execute code not installed as part of an application.

Do not cache optimized classes on external storage. External storage does not provide access controls necessary to protect your application from code injection attacks.

- Link Ch 8b

Remote Loading of Code

- Apps can load new code at runtime
  - Using the Java Reflection API
  - With the **DexClassLoader** class
  - May load code over the network, or from a local location that can be overwritten by other applications
- May cause code injection vulnerabilities

WebViews

- Recipe for disaster
  - Using a WebView
  - Defining a JavaScript interface
    - Loading from a cleartext source or having SSL bypass code
    - Targeting API versions prior to 17 or using an Android version earlier than 4.2
- May allow JavaScript code injection

CVE-2012-6636 Detail

Current Description

The Android API before 17 does not properly restrict the WebView.addJavascriptInterface method, which allows remote attackers to execute arbitrary methods of Java objects by using the Java Reflection API within crafted JavaScript code that is loaded into the WebView component in an application targeted to API level 16 or earlier, a related issue to CVE-2013-4710.

Google's Fix

Caution: If you've set your `targetSdkVersion` to 17 or higher, you must add the `@JavascriptInterface` annotation to any method that you want available to your JavaScript, and the method must be public. If you do not provide the annotation, the method is not accessible by your web page when running on Android 4.2 or higher.

- https://developer.android.com/develop/ui/views/layout/webapps/webview

Listening Services

- Android is unlikely to have listening ports
- My Genymotion has a few

```
root@kali:~# adb shell netstat -pant | grep LISTEN
tcp        0      0 127.0.0.1:5037          0.0.0.0:*               LISTEN      143/adbd
tcp        0      0 0.0.0.0:24800           0.0.0.0:*               LISTEN      293/local_camera
tcp        0      0 127.0.0.1:24801         0.0.0.0:*               LISTEN      293/local_camera
tcp        0      0 0.0.0.0:22468           0.0.0.0:*               LISTEN      292/local_opengl
tcp        0      0 0.0.0.0:24810           0.0.0.0:*               LISTEN      294/local_camera
tcp        0      0 127.0.0.1:24811         0.0.0.0:*               LISTEN      294/local_camera
tcp        0      0 0.0.0.0:6379            0.0.0.0:*               LISTEN      142/redis
tcp6       0      0 :::5555                 :::*                    LISTEN      143/adbd
tcp6       0      0 :::24296                :::*                    LISTEN      341/audioserver
tcp6       0      0 :::6379                 :::*                    LISTEN      142/redis
```

Messaging Applications

- Examples, may be vulnerable
  - Short Message Service (SMS)
  - Multimedia Messaging Service (MMS)
  - Commercial Mobile Alert System (CMAS)
  - Email clients
  - Chat clients

Finding Local Vulnerabilities

- Manual process
  - Download all installed apps
  - Convert them to readable source code
  - Use **grep** to search for vulnerabilities
- Or use Drozer's scanner modules

Drozer's SQLi Scanner

- Doesn't find the Sieve SQL injection

```
dz> run scanner.provider.injection
Scanning com.android.cts.priv.ctsshim...
Scanning com.savved.uptick...
Scanning com.android.internal.display.cutout.emulation.corner...
Scanning com.google.android.ext.services...
Scanning com.example.android.livecubes...
Scanning com.android.internal.display.cutout.emulation.double...
Scanning com.android.providers.telephony...
Scanning com.android.providers.calendar...
Scanning com.fidelity.android...
```

Exploiting Devices

Remote and Local Exploits

- Remote exploit
  - Gives attacker a foothold on the device
    - Such as software exploits, MITM attacks, or malware
- Local exploit
  - Requires a foothold on the device already
  - Local privilege escalation

Using Attack Tools

Ettercap

- Performs ARP poisoning, DNS spoofing, etc.
- We're using local proxy settings
- You need ettercap to perform real MITM attacks on a LAN

Burp

- Can inspect and modify traffic
- Sends fake TLS certificates
- Burp can be added as a "trusted CA"

PORTSWIGGER
WEB SECURITY

Installing Burp's CA Certificate in an Android Device

Before you start:

- Ensure you have configured your Android device to work with Burp.
- Ensure your Android device is able to receive email, and that your email filter does not block .cer files.

Note: Android Nougat no longer trusts user or admin supplied CA certificates. We recommend that you use an older version of Android for your testing. If you must use Android Nougat then you will need to install a trusted CA at the Android OS level on a rooted device or emulator.

Drozer

- Infrastructure Mode
  - Runs a Drozer server, as a C&C server
  - Make "rogue agents" which are like malware
    - Custom-built to phone home to the Drozer server
- Much like Metasploit

Privilege Levels

Non-System App without Context

- Ex: a shell from a Web browser
- Attacker has privileges of the compromised app
  - Can navigate filesystem under the app's user account
  - Cannot use Java libraries
  - Cannot install packages, or read SMS, etc.

Non-System App with Context

- Attacker takes over app's execution flow and can load arbitrary classes
- Attacker can retrieve app **Context**
- Can do anything the app can do

Installed Package

- Can request arbitrary permissions
  - Can be granted them, depending on protection level

ADB Shell Access

- Can install apps
- Can interact with apps as a developer

System User Access

- Running as **system** user, can
  - Install apps
  - Change device configuration
  - Access data from any app's private directory

Root User Access

- Ultimate power, can
- Install apps
- Read and write RAM
- Manipulate any aspect of the device


Kahoot!