How Secure Is This Thing Anyway? A Guide Into Bug Bounties and Mobile Security

JASMINE JACKSON

About Me

- Hometown: Berkeley, CA
- Reside: Charlotte, NC
- Penetration Tester focus on AppSec
- Interested in Mobile and started learning at the beginning of this year
- Twitter: thefluffy007

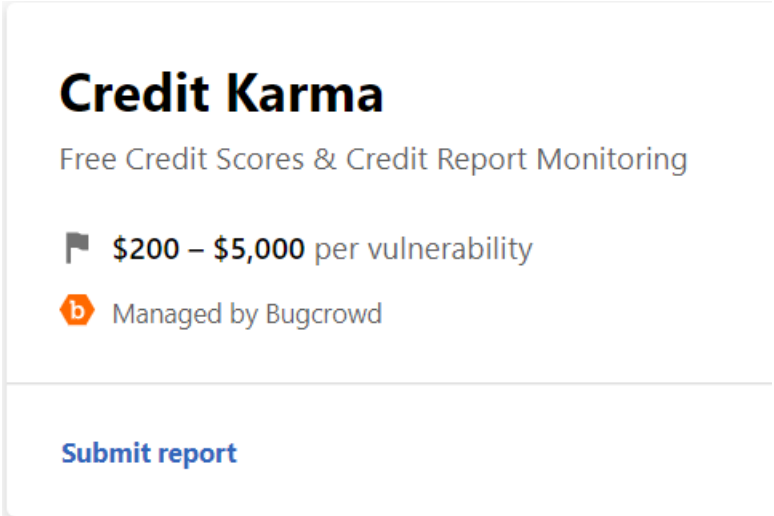
Goal of Presentation

Become empowered to start your journey in bug bounties


WHAT HAVE I GOTTEN
MYSELF INTO?
imgflip.com

DISCLAIMER: images used in presentation are not from bug bounty programs

What Is A Bug Bounty?



Credit Karma is a personal finance technology company with more than 85 million members in the United States and Canada, including almost half of all millennials. The company offers a suite of products for members to monitor and improve credit health and provides identity monitoring and auto insurance estimates. Since 2007, we have been knocking down barriers that block the path to financial health, helping our members make informed choices and feel confident about their opportunities.

40 vulnerabilities rewarded

Validation within **4 days**

75% of submissions are accepted or rejected within 4 days

$250 average payout (last 3 months)

Types of Bug Bounties

Created by icon 54
from Noun Project

Reward

Created by Graphic Tigers
from Noun Project

Swag

Reward

- Compensation based
- Based on vulnerability and platform
- Higher the severity, higher the payout

Rewards:

$	API, IOS, ANDROID	WEB
P1	$5,000	$3,000
P2	$2,250	$1,800
P3	$700	$600
P4	$250	$200

Swag

- Clothes
  - Hats
  - Shirts
  - Hoodies
- Gadgets
  - USB Drives
  - Flashlights
- Backpacks
- Stickers

Bug bounty stats

Bounty Guidance

Platform-wide guidance	Critical Severity Vulnerability	High Severity Vulnerability	Med Severity Vulnerability	Low Severity Vulnerability
LEVEL ONE	$2,000	$750	$300	$150
LEVEL TWO	$3,000	$1,000	$500	$250
LEVEL THREE	$5,000	$2,000	$750	$250 - 500
LEVEL FOUR	$7,500	$3,000	$1,000	$250 - 700
LEVEL FIVE	$10,000	$5,000	$2,500	$250 - 1000

Source: HackerOne

Things To Do On A Mobile Device

- Bank information
- Shopping
- Browse Internet
- Social media

Attack Vector for Mobile Devices

INSIDE THE MOBILE ATTACK SURFACE

THE DEVICE

BROWSER
- Phishing
- Framing
- Clickjacking
- Man-in-the-Middle
- Buffer overflow
- Data caching

PHONE/SMS
- Baseband attacks
- SMS phishing

APPS
- Sensitive data storage
- No/Weak encryption
- Improper SSL validation
- Configuration manipulation
- Dynamic runtime injection
- Unintended permissions
- Escalated privileges
- UI overlay/pin stealing
- Third-party code
- Intent hijacking
- Zip directory traversal
- Clipboard data
- URL schemes
- GPS spoofing
- Weak/No Local authentication
- Integrity/tampering/repacking
- Side channel attacks
- App signing key unprotected
- App transport security
- XML serialization
- JSON-RPC
- SQLite database

SYSTEM
- No/Weak passcode
- Android rooting/iOS jailbreak
- OS data caching
- Passwords & data accessible
- Carrier-loaded software
- No/Weak encryption
- User-initiated code
- Confused deputy attack
- TEE/Secure Enclave Processor
- Side channel leak
- Multimedia/file format parsers
- Kernel driver vulnerabilities
- Resource DoS
- GPS spoofing
- Device lockout

MALWARE

THE NETWORK

- Wi-Fi (no/weak encryption)
- Rogue access point
- Packet sniffing
- Man-in-the-middle
- Session hijacking
- DNS poisoning
- SSL Strip
- Fake SSL certificate
- Baseband
- Wifi (chip/firmware attack)
- BGP hijacking
- IMSI-catcher
- LTE
- HTTP Proxies
- VPNs

CLOUD / DATA CENTER

WEB SERVER
- Platform vulnerabilities
- Server misconfiguration
- Cross-site scripting
- Cross-site request forgery
- Weak input validation
- Cross origin resource sharing
- Brute force attacks
- Side channel attacks
- Hypervisor attack
- VPN

DATABASE
- SQL injection
- Privilege escalation
- Data dumping
- OS command execution

NowSecure™

Fragmentation

- Android devices are SEVERLY fragmented
- Android is open source
- Mobile providers can add custom code on top of the Android operating system for specific devices
  - Ex: Galaxy s6 possibly not having the same code as the Note
- Leads to interoperability issues

Android Security

- Built on Linux kernel
- Provide the following security features
  - A user-based permission model
  - Process isolation
  - The ability to remove unnecessary and potentially insecure parts of the kernel

Application Sandboxing

- Isolate apps from each other
- Assigns a unique user id (UID) to each Android application and runs in its own process
- Uses this UID to set up the kernel level application sandbox
- Kernel enforces security between apps and system at the process level through user and group IDs that are assigned to apps
- By default, apps can't interact with each other, and have limited access to the operating system

Application Signing

- Allows developers to identify the author of the application
- Provide a more streamline process to update the application
- Every application that run on the Android platform must be signed by the developer
- Applications that attempt to install on a device without being signed will be rejected by Google Play or the Package Installer
- Application signing is the first step to placing an application in the application sandbox
- Signing defines the user ID that is associated with the application
- Different applications run as different user IDs

The Importance of Rooting A Device

- Android only allow a small subset of applications to run with root permissions
- Android does not stop users or applications to modify the operating system, kernel, or any other applications
- Root has full access to applications and all application data on the device
- Users that allow applications to run as root increase their threat exposure to malicious applications and application flaws

Components of an Android Application

- Activities
- Services
- Content Providers
- Broadcast Receivers

Activities

- Activities
  - Represents a single screen with a user interface
  - Similar to a window in a desktop application
  - An application can contain one or more activities

Services

- General purpose entry point for keeping an app running in the background
  - Does not need a user interface
- Perform long-running operations or perform work remote processes
  - Internet downloads
  - Data processing

Content Providers

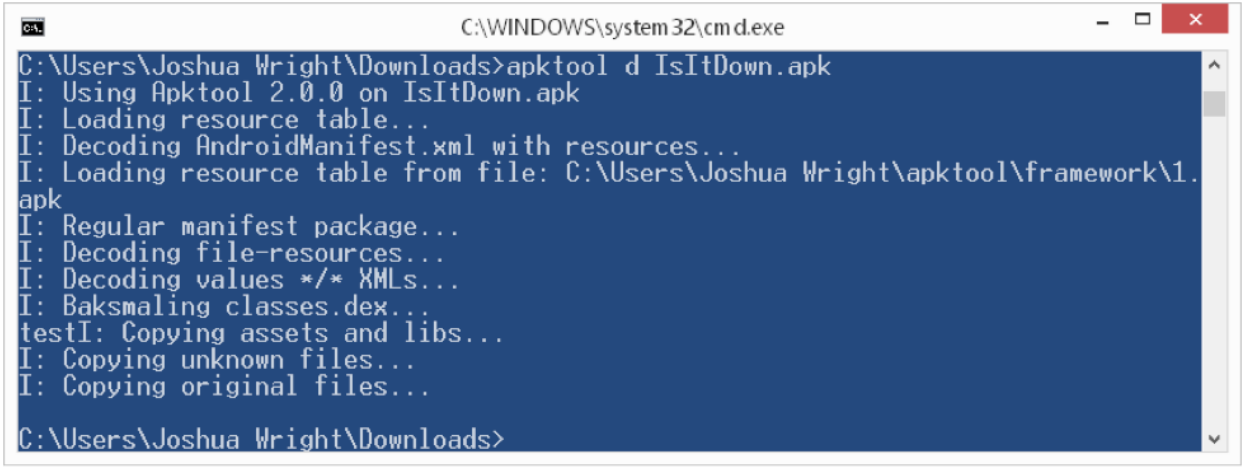
- Supplies data from one application to others on request
- Centralized content in one place and allow different applications to access it

Broadcast Receivers

- Respond to system wide broadcast announcements
- Messages are called events or intents
- Examples:
  - Screen has turned off
  - Low battery

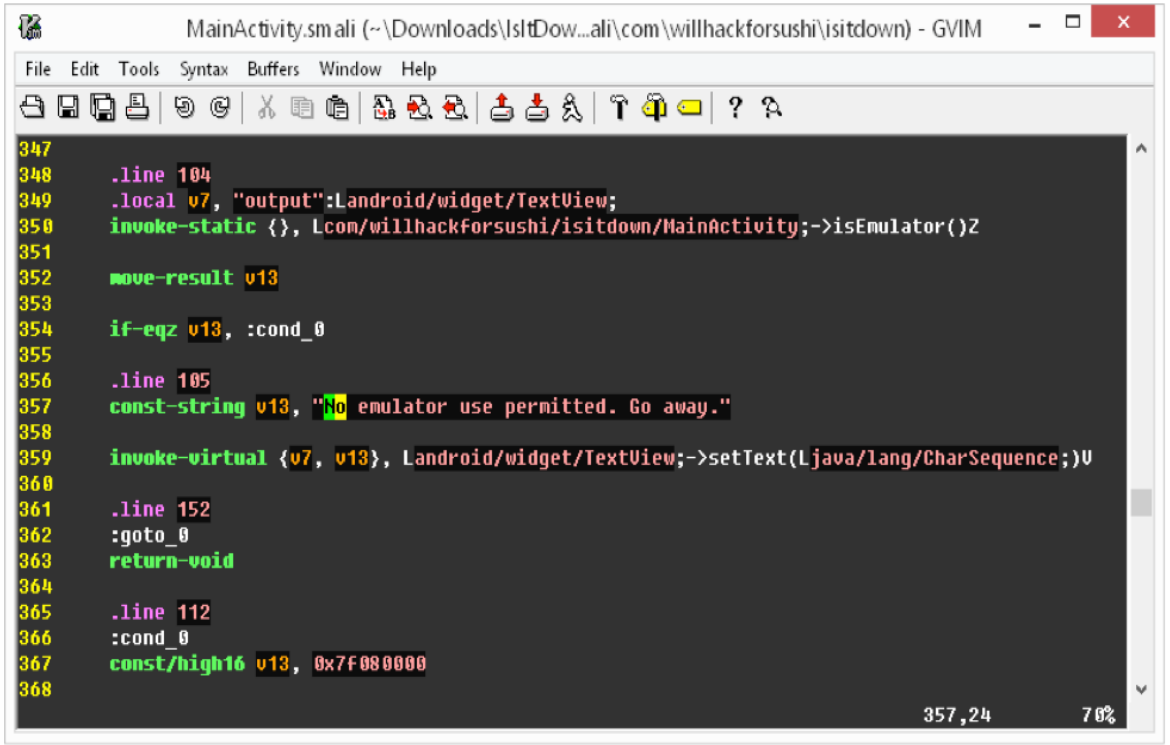
APKTool

- Used for reverse engineering 3rd party apps
- Can decode resources close to original form
- Can rebuild application after modification
- Files/folders after decompiling APK
  - smali
  - res
  - META-INF
  - AndroidManifest.xml



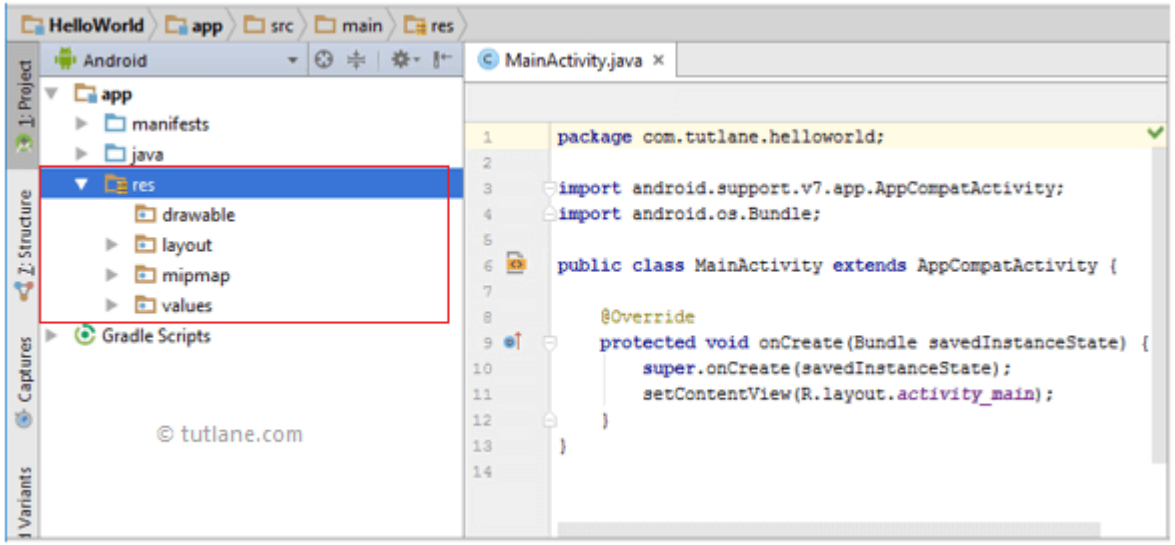
Smali

- An assembly language used by the Android Dalvik Virtual Machine
- Decompiled through a .DEX file
- Used for low-level inspection of Android app content



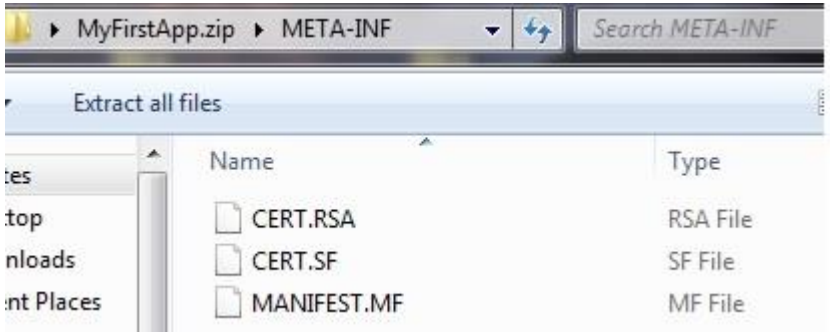
Res

- Folder for additional files and static content
  - Images (res/drawable)
  - Layout definitions (res/layout)
  - User interface strings
  - Animation strings (XML files) (res/values)



META-INF

- CERT.RSA - Contains the signed contents of the CERT.SF file along with the certificate chain of the public key used to sign the contents.
- CERT.SF - Contains a list of all files along with SHA-1 hash
- MANIFEST.MF - Contains information for the application such as package version, build number, creator of the package, etc.



AndroidManifest.xml

- Located at the root of the project
- Manifest declares the app's component such as:
  - Package of application
  - Describes components of the application (activities, services, broadcast receivers, and content providers
  - Declares which permissions the application must have in order to interact with protected API (Application Programming Interfaces), and other applications
  - Declares the minimum level of Android API that the application requires

AndroidManifest.xml Example

```
1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.androidapp.basicElements"
4      android:versionCode="1"
5      android:versionName="1.0">
6    <application android:icon="@drawable/icon" android:label="@string/app_name">
7        <activity android:name=".BasicElements"
8                  android:label="@string/app_name">
9            <intent-filter>
10               <action android:name="android.intent.action.MAIN" />
11               <category android:name="android.intent.category.LAUNCHER" />
12           </intent-filter>
13       </activity>
14
15   </application>
16   <uses-sdk android:minSdkVersion="2" />
17
18</manifest>
```

Finding Targets

- Why hunt android applications?
  - Android is an open source platform
  - Low entry to barrier
- What's the big deal of hunting?
  - There are multiple programs (and more incoming)
  - Subset of programs - Department of Defense (DoD), Apple, Facebook, etc.
- Found bounty programs that are small
  - Most people are going after bigger programs
  - (Possibly) more vulnerabilities to find
  - Use as practice before moving to bigger targets

Reporting

- Make sure reports are clear and concise
  - Report should allow a non-bug bounty person to get the same results
  - Report is just as important as finding the bug
  - If report is not clear and concise your finding could be rejected
- Don't be discouraged with duplicates!
  - Shows you're moving in the right direction, but someone got there first
- Make sure to save notes
  - Notebook
  - Blog

How To Get Involved

- Read write-ups (if available)
- Take the plunge into bug bounties!
  - Bugcrowd
  - HackerOne
- Tutorials:
  - Hacker101 CTF: https://www.hacker101.com/
  - Bugcrowd University: https://www.bugcrowd.com/hackers/bugcrowd-university/

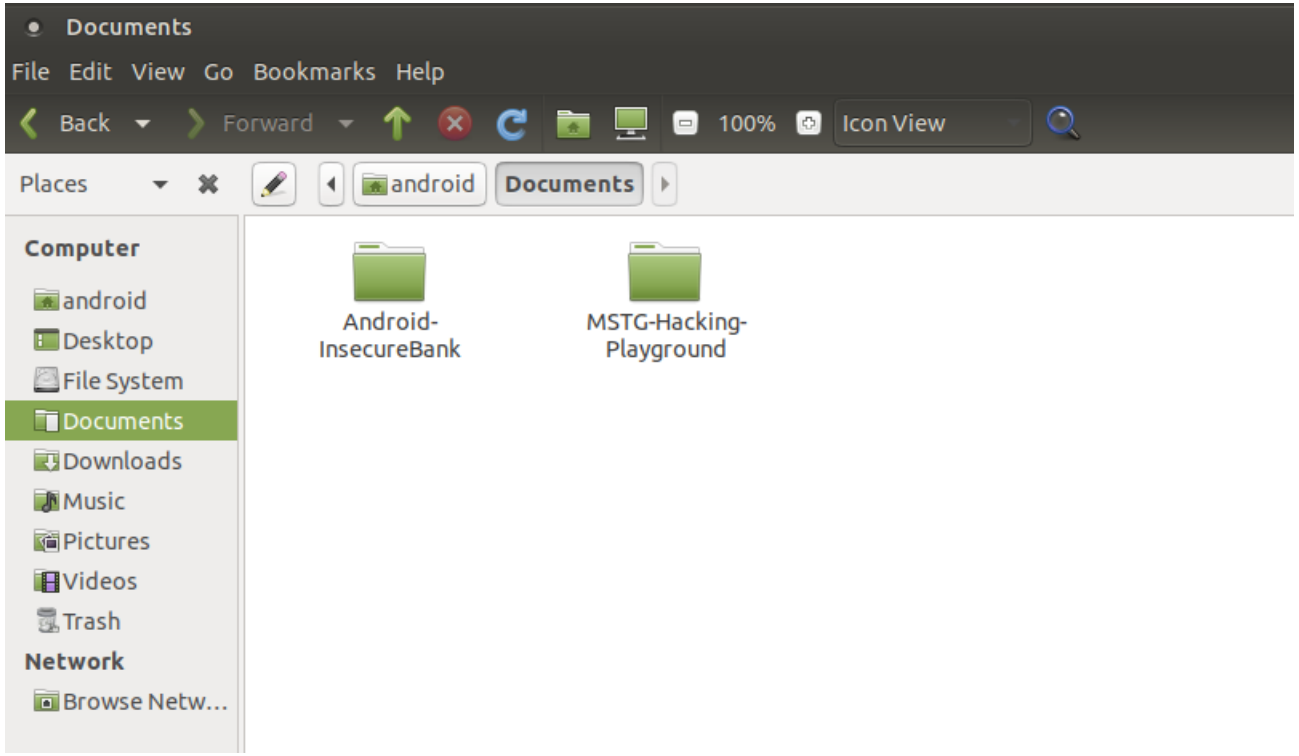
DEMO

Summary

- Attack vector is wider in mobile apps
- Barrier to entry is low with Android
- Able to decompile Android apps and look “under the hood”
- Bug bounties are not a one-size fits all
- Frida is a good tool to use for dynamic analysis as you can change logic on the fly
- There is no silver bullet application with mobile security

Continued Learning

- IntroToAndroidSecurity VM has two additional apps
  - Android-InsecureBank
  - MSTG-Hacking-Playground
  - Will add DIVA in next release
- Push the apps to Android emulator using Android Debug Bridge (adb)



Continued Learning Pt 2

- Working with TryHackMe (http://tryhackme.com) to add VMs on their site
- Download IntroToAndroidHacking VM from SourceForge:
  - https://sourceforge.net/projects/introandroidsecurity

Resources

- Mobile security challenges:
  - MOBISEC: https://mobisec.reyammer.io/challs
  - MSTG (Mobile Security Testing Guide) Hacking Playground: https://github.com/OWASP/MSTG-Hacking-Playground
  - OWASP Uncrackable Mobile Apps: https://github.com/OWASP/owasp-mstg/tree/master/Crackmes
  - DIVA (Damn Insecure and Vulnerable App for Android) - https://github.com/payatu/diva-android

Suggested Reading

- System and Kernel Security: https://source.android.com/security/overview/kernel-security#rooting-devices
- Application Signing: https://source.android.com/security/apksigning
- AVD Manager: https://developer.android.com/studio/command-line/avdmanager
- SDK Manager: https://developer.android.com/studio/command-line/sdkmanager
- Android Debug Bridge (ADB): https://developer.android.com/studio/command-line/adb

Suggested Reading

- Android Application Fundamentals: https://stuff.mit.edu/afs/sipb/project/android/docs/guide/components/fundamentals.html
- Intro to JavaScript – https://www.w3schools.com/js/js_intro.asp
- Intro to Java - https://www.w3schools.com/java/java_intro.asp
- Frida - https://www.frida.re/docs/home/