Android Forensics

Session C4

Tuesday, April 3, 2012

Ming Chow
Lecturer, Department of Computer Science
Tufts University

INFOSEC WORLD
CONFERENCE & EXPO 2012

MIS TRAINING INSTITUTE

Introduction

- Over 700,000 Android phones activated per day
  - ~250 million devices activated so far
- Android is a loosely defined platform:
  - Hardware: varies (e.g., phones, tablets, appliances); manufacturers and carriers can customize it
  - Operating system: Based on Linux; over 3 major releases in the last 1.5 years (more later)
- What this presentation is: how to *acquire* and *analyze* data from an Android device

What We Will Not Cover

- Jailbreaking or rooting an Android device
- Developing apps or scripts for Android
- Fundamentals of computer forensics and investigations
- Anything specific to law enforcement or the court system
- Using commerical tools such as FTK or EnCase
- Linux internals

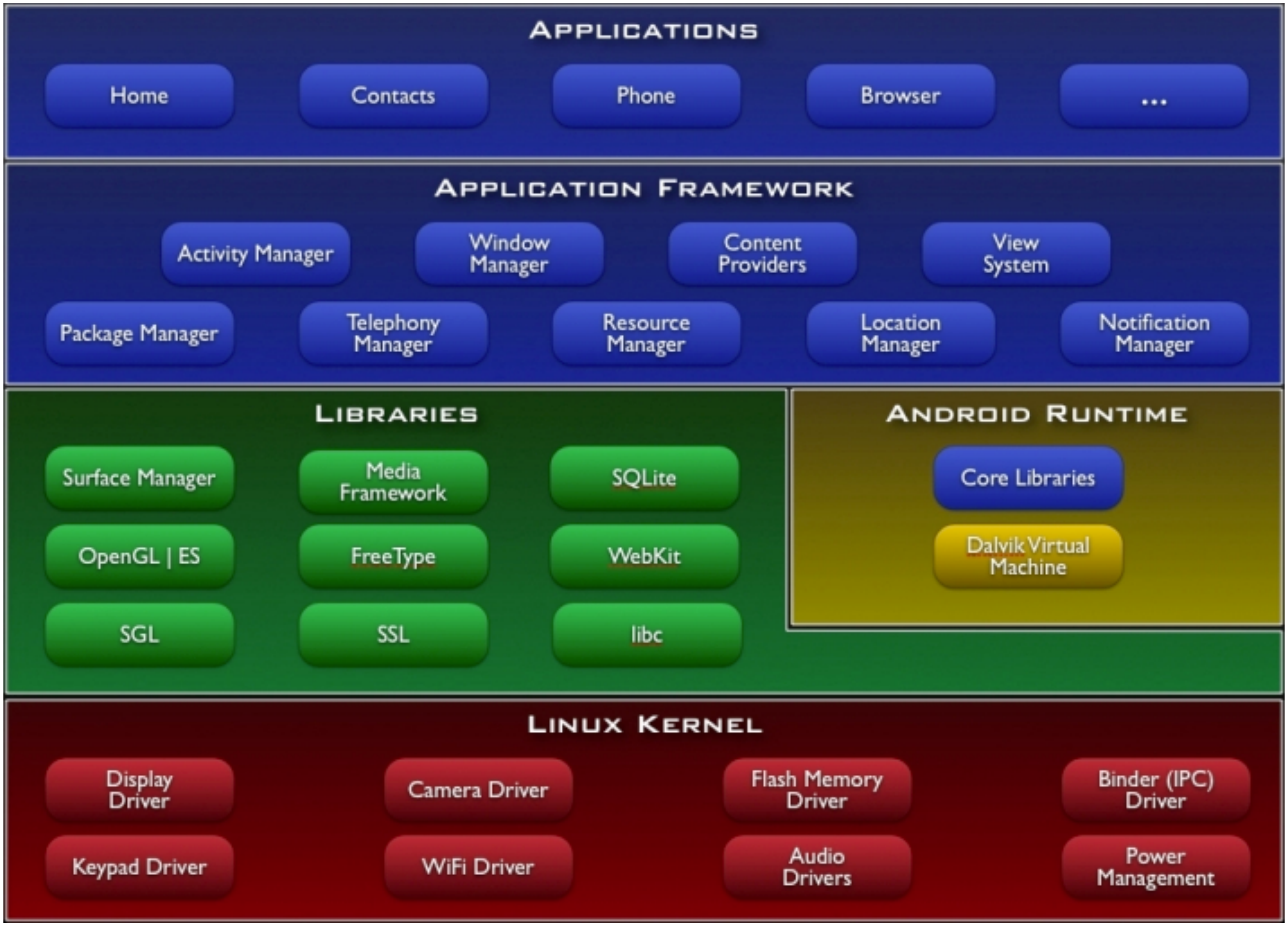
What You Will Need

- Android Standard Development Kit (SDK)
  - http://developer.android.com/sdk/index.html
- Basic *nix (Unix or Linux) command line skills

Android Architecture (continued)

- Based on Linux 2.6 for core system services (e.g., memory and process management, network stack)
- How the apps are run: by the Android Runtime System utilizes the Dalvik virtual machine (VM)
    - Allows multiple apps to run concurrently
    - Each app has its own separate VM (e.g., unique user ID and process)
    - Sandboxed apps: files created by an app cannot be viewed by another app (i.e., based on privilege separation)

Android Architecture



Applications

- Home
- Contacts
- Phone
- Browser
- ...

Application Framework

- Activity Manager
- Window Manager
- Content Providers
- View System
- Package Manager
- Telephony Manager
- Resource Manager
- Location Manager
- Notification Manager

Libraries

- Surface Manager
- Media Framework
- SQLite
- OpenGL | ES
- FreeType
- WebKit
- SGL
- SSL
- libc

Android Runtime

- Core Libraries
- Dalvik Virtual Machine

Linux Kernel

- Display Driver
- Camera Driver
- Flash Memory Driver
- Binder (IPC) Driver
- Keypad Driver
- WiFi Driver
- Audio Drivers
- Power Management

Android File System

- File system is Yet Another Flash File System 2 (YAFF2)
- Base file system is "/"; generally recreated everytime using ramdisk content
- **`/cache`** => Used as scratch pad by OS place dex optimized dalvik bitecode
- **`/data`** => Contains USER Data Stored as a separate partition in mtdblocks mounted at bootup
- **`/default.prop`** => Default property settings, values restored from this file on every restart
- **`/proc`**
- **`/sbin`**
- **`/sys`**
- **`/system`**
- **`/sdcard`** => The removable sdcard directory
- Interactive: http://anantshri.info/andro/file_system.html

Memory and Storage

- SIM card
- Removable Flash
- RAM (on the device itself)

Forensics Caveats

- Can’t “pull the plug”
- Devices are always online (either using Wi-Fi or provider’s network)
- Data stored on the device and in the cloud
- Android devices are strongly coupled with Google services (e.g., Gmail, Calendar, Voice)

Anatomy of an Android App

- Android apps are developed using Java and the Android SDK
- An app use least-privilege permissions to access various components on device (e.g., camera, networking, GPS, flashlight)
- The binary: a signed **`.apk`** file; you can unzip it!
  - **`AndroidManifest.xml`**: details about the app including permissions, version number, and main class
  - **`res/`**: resources such as images
- Each app runs its own Dalvik VM

Anatomy of an Android App (continued)

- Data stored in **`/data/data/`** of where the app is installed
  - Each app has a package name (such as **`com.google.dev`** or **`edu.tufts.cs.mchow`**; URL reversed)
  - Subdirectories
    - **`lib/`** - Custom library files or dependencies
    - **`files/`** - Files used by the app
    - **`cache/`** - Cached files, often from the browser
    - **`databases/`** - Namely SQLite databases

Secure the Device

1. Unlock device

u Enter or break pass code

u Increase screen timeout

2. Isolate device from network

u Put device in Airplane Mode

3. Enable USB debugging

u On the device, go to Settings > Applications > Development > check off "USB debugging"

4. Remove SIM card

5. Remove SD card

6. Find the right USB and power cables

Logical Acquisition

- Download latest version of Andrew Hoog’s AFLogical open source at http://code.google.com/p/android-forensics
  - Unzip .apk file and send to device
  - Instructions: http://code.google.com/p/android-forensics/wiki/WikiPageUse
- Information acquired include browser history, call logs, metadata of various media files, MMSes, SMSs, apps installed (with version), contacts; results to CSV files
- Information about the device saved to `info.xml` file

Physical Acquisition

- Bit-by-bit copy of an entire physical store or SD card (FAT32)
  - Gold mine of deleted and active personal data including photos, music, downloads, app data
- Use **`dd`**

Online Analysis with Android Debug Bridge (`adb`)

- http://developer.android.com/guide/developing/tools/adb.html
- Command line tool; found in **`<sdk>/platform-tools/`**
- Client-server based; communication between your computer and the device
- Make sure “USB debugging” is enabled on device
- Commands:
  - **`adb devices`** => see list of connected devices
  - **`adb shell`** => interact with with device
  - You can push and pull files to and from the device via **`adb push`** and **`adb pull`**
  - **`adb logcat`** => print system log (includes app stuff)

Online Analysis with the Dalvik Debug Monitor Server (DDMS)

- http://developer.android.com/guide/developing/debugging/ddms.html
- Command line tool; found in `<sdk>/tools/`
- Again, make sure “USB debugging” is enabled on device
- Graphical
- Can take screenshots of device
- Overlaps with `adb` (e.g., `logcat`)
- Can emulate phone operations, location
- Can spoof calls and text messages
- Can dump application state

Conclusion

- Challenges
  - Fragmentation
    - Many different Android OSes
    - Many different carriers and devices
  - Varies file systems used by Android(YAFFS2, FAT32, etc.)
  - Rooted vs. un-rooted devices
- Still a very young field (mobile forensics)
- Both logical and physical techniques are necessary
- Android continues to grow --fast

References and Resources

- “Android Forensics: Simplifying Cell Phone Examinations,” Lessard & Kessler, Small Scale Digital Device Forensics Journal, Vol. 4, No. 1, September 2010, http://www.ssddfj.org/papers/SSDDFJ_V4_1_Lessard_Kessler.pdf
- “Introduction to Computer Forensics and Android Forensics,” Simson Garfinkel http://simson.net/ref/2011/2011-07-12%20Android%20Forensics.pdf
- “Android Forensics: Investigation, Analysis, and Mobile Security for Google Android,” Andrew Hoog, Syngress Press, June 2011. http://my.safaribooksonline.com/book/-/9781597496513
- “Android: Forensics and Reverse Engineering,” Raphael Rigo, https://deepsec.net/docs/Slides/DeepSec_2010_Reverse_Forensics.pdf
- http://computer-forensics.sans.org/blog/2010/03/01/open-source-android-digital-forensics-application/
- http://code.google.com/p/android-forensics/
- http://www.dfinews.com/article/introduction-android-forensics
- http://viaforensics.com/services/mobile-forensics/android-forensics
- https://viaforensics.com/android-forensics/htcia-android-forensics-training-presentation-february-14-2012.html
- http://techcrunch.com/2011/12/22/android-700000/
- http://developer.android.com/sdk/index.html