Android Forensics

and Security Testing

Shawn Valle
shawnvalle at gmail dot com
September 2012

Introductions

whoami?

15 years in IT and security (CISSP, MCP, LCP)

Course developer / trainer at IBM's Catapult Software Training & independently

JavaScript, HTML, web app development, content management, identity management, Lotus Domino

Began working with mobile computing in 2006 (PalmOS app/ROM development)

Joined MITRE in 2008 working in network and app security

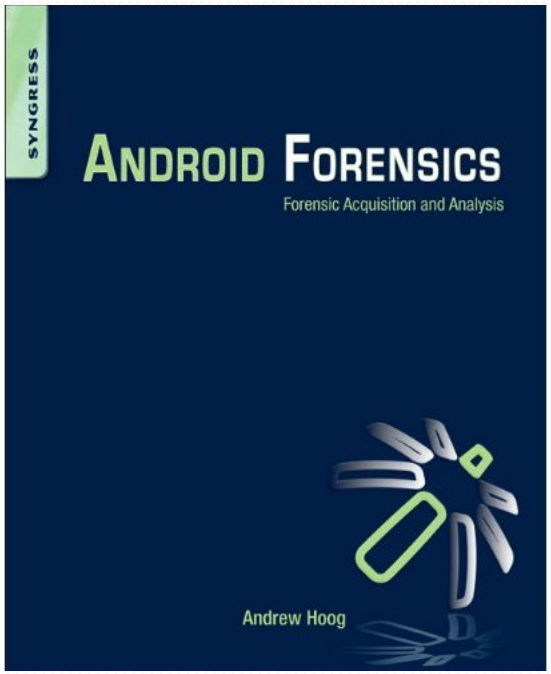
Co-established MITRE's Mobile Security Practice in 2010, leading engineering and coordination in several mobile computing projects

Learning Objectives

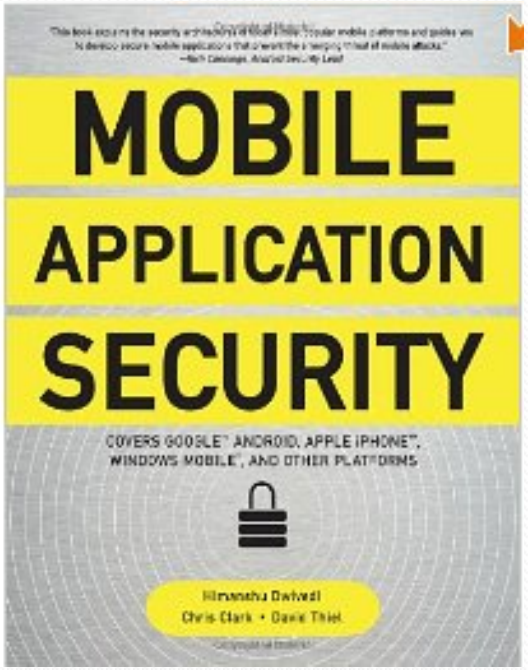
By the end of this course, you will be able to:

1. Extract and analyze data from an Android device
2. Manipulate Android file systems and directory structures
3. Understand techniques to bypass passcodes *NEW!*
4. Utilize logical and physical data extraction techniques
5. Reverse engineer Android applications
6. Analyze acquired data

Books



Hoog, Andrew (2011). Android Forensics, Syngress.



Dwivedi, Himanshu, Clark, Theil (2010). Mobile Application Security, McGraw-Hill.

Agenda

DAY 1

- Forensic Introduction
- Course Setup – Linux, OS X, and Windows
- Android Overview
- SDK and AVD
- Android Security Model
- ADB and shell Introduction

BREAK

- File System and Data Structures

LUNCH

- Device Handling
- Circumvent passcode
- Gain Root Access
- Recovery Mode
- Boot Loaders

BREAK

- Logical Forensic Techniques
- Open Source Tools
- Commercial Tools

Agenda

DAY 2

- Physical Forensic Techniques & Tools

BREAK

- Forensic Analysis

LUNCH

- Application Penetration Testing Setup
- Reverse Apps

BREAK

- ...more Reversing
- Document Findings

Prerequisites

- Introduction to Android Development

and / or

- Introduction to Linux

Legalities

- Possibility of Android devices being involved in crimes
- Easily cross geographical boundaries; multi-jurisdictional issues
- Forensic investigator should be well aware of regional laws
- Data may be altered during collection, causing legal challenges
- Fully document justification for data modification

http://www.forensicfocus.com/downloads/windows-mobile-forensic-process-model.pdf

Forensic Investigations

Terms and Definitions

- Mobile Forensics is defined as "the science of recovering digital evidence from a mobile phone under forensically sound conditions using accepted methods." (NIST)

- A **penetration test**, occasionally **pentest**, is a method of evaluating the security of a computer system or network by simulating an attack from malicious outsiders (who do not have an authorized means of accessing the organization's systems) and malicious insiders (who have some level of authorized access). (Wikipedia)

- **A vulnerability assessment** is the process of identifying, quantifying, and prioritizing (or ranking) the vulnerabilities in a system.

What is Mobile Forensics & Why Should I Care?

Android Forensics

- The acquisition and analysis of data from devices,
- Internal corporate investigations, civil litigation, criminal investigations, intelligence gathering, and matters involving national security.
- Arguably the fastest growing and evolving digital forensic discipline, offers significant opportunities as well as many challenges.

Forensic Overview

- General to forensics, not just Android.
- Potential scenarios:
  - Evidence gathering for legal proceedings
  - Corporate investigations
    - Intellectual property or data theft
    - Inappropriate use of company resources
    - Attempted or successful attack against computer systems
    - Employment-related investigations including discrimination, sexual harassment
    - Security audit
  - Family matters
    - Divorce
    - Child custody
    - Estate disputes
  - Government security and operation
    - Cyber Threats, Advanced Persistent Threat
    - Stopping cyber attacks
    - Investigating successful attacks
    - Intelligence / Counter-intelligence gathering

Source: Andrew Hoog, Android Forensics, Elselvier 2010

Forensic Considerations

- Important items to consider during investigation:
  - Chain of custody
  - Detailed notes and complete report
- Validation of investigation results, using tools or other investigators

Source: Andrew Hoog, Android Forensics, Elselvier 2010

Android Overview & History

...in five minutes

Android Overview & History

- Google Mobile SVP Andy Rubin reported that over 850,000 Android devices were being activated each day as of February 2012
- 500,000 increase per day over just one year ago

Android Overview & History

Operating System	3Q11 Market Share (%)	3Q10 Market Share (%)
Android	52.5	25.3
Symbian	16.9	26.3
iOS	15	16.6
RIM	11	15.4

- Worldwide Smartphone Sales to End Users

Source: http://www.gartner.com/it/page.jsp?id=1848514
Market Share: Mobile Communication Devices by Region and Country, 3Q11

Android Overview & History

Date	Event
July 1, 2005	Google acquires Android, Inc.
November 12, 2007	Android launched
September 23, 2008	Android 1.0 platform released
February 13, 2009	Android Market: USA takes paid apps
April 15, 2009	Android 1.5 (Cupcake) platform released
September 16, 2009	Android 1.6 (Donut) platform released
October 5, 2009	Android 2.0/2.1 (Eclair) platform released
May 20, 2010	Android 2.2 (Froyo) platform released
December 6, 2010	Android 2.3 (Gingerbread) platform released
February 2, 2011	Android 3.0 (Honeycomb) preview released
November 14, 2011	Android 4.0 (Ice Cream Sandwich), 3.0 source released
July 9, 2012	Android 4.1 (Jelly Bean) platform released

1.5 Cupcake

1.6 Donut

2.0/2.1 Eclair

2.2 Froyo

2.3 Gingerbread

3.0/3.1 Honeycomb

... IceCream Sandwich

Android Overview & History

- Android Feature Introduction
  - More details come later
  - 1st Primary feature, always connected: GSM, CDMA, LTE, WiMax, WiFi
  - 2nd Market / Play: rich source for forensic analysts
  - 3rd Data Storage: Big part of the course
    - Flash (or NAND) memory
    - External SD card
    - Internal SD card

Android Overview & History

- Cellular Networks and Hardware
  - Global System for Mobile Communications – GSM
    - Subscriber identity module (SIM) or universal subscriber identity module (USIM) to identify the user to the cellular network
    - AT&T, T-Mobile
  - Code Division Multiple Access – CDMA
    - Sprint, Verizon
  - Integrated Digital Enhanced Network – iDEN
    - Sprint
  - Worldwide Interop for Microwave Access – WiMax
    - Sprint
  - Long Term Evolution – LTE
    - AT&T, Sprint, T-Mobile, Verizon

Android Overview & History

- Apps
  - As of January 2012, over 400,000 Android apps have been developed. Doubled since January 2011.
  - Apple maintains tight control over their App Store, requiring developers to submit to a sometimes lengthy review process and providing Apple with the final approval for an app. Apps can be denied based on a number of criteria, most notably if they contain any content Apple feels is objectionable.
  - Google, on the other hand, requires very little review to publish an app in the Android Market. While Google has the ability to ban a developer, remove an app from the Android Market, and even remotely uninstall apps from Android devices, in general their approach to app management is hands off. (Hoog)

Source: http://www.theverge.com/2012/1/4/2681360/android-market-400000-app-available

Android Open Source Project

- The Android Open Source Project (AOSP) is led by Google, and is tasked with the maintenance and development of Android.
- It is good experience to download and install AOSP from source.
- Not critical for all forensics analysts to get this deep into Android. May be helpful for deep analysis.
- We won't be doing that in this course…

Source: http://en.wikipedia.org/wiki/Android_(operating_system)#Android_Open_Source_Project

Brief Linux Overview

Source:
http://www.talkandroid.com/wp-content/uploads/2011/05/LinuxAndroid.png?3995d3

Linux, Open Source Software & Forensics

Android Forensics

- Open source software has had a tremendous impact on the digital forensics discipline. Forensic tools that are released as free open source software have tremendous advantages over closed source solutions including the following:
  - The ability to review source code and understand exact steps taken
  - The ability to improve the software and share enhancements with entire community
  - The price
- Linux is not only a critical component of Android but can also be used as a powerful forensic tool.

Source: Andrew Hoog, Android Forensics, Elselvier 2010

Linux Commands

- man
- help
- cd
- mkdir
- mount
- rmdir/rm
- nano
- ls
- tree
- cat
- dd
- find
- chmod
- chown
- sudo
- apt-get
- grep
- | and >

… see Linux Commands handout for valuable commands

Android and Forensics

Source:
http://viaforensics.com/services/mobile-forensics/android-forensics/

Android & Forensics

- Relatively new, emerged in ~2009
- Best known expert in the field is Andrew Hoog
- Other leaders in the Android Security field include Jon Oberheide and Zach Lanier
- Community is rapidly growing
- In-house investigations on pilot / prototype apps
- Penetration tests
- Vulnerability assessments
- Funded research

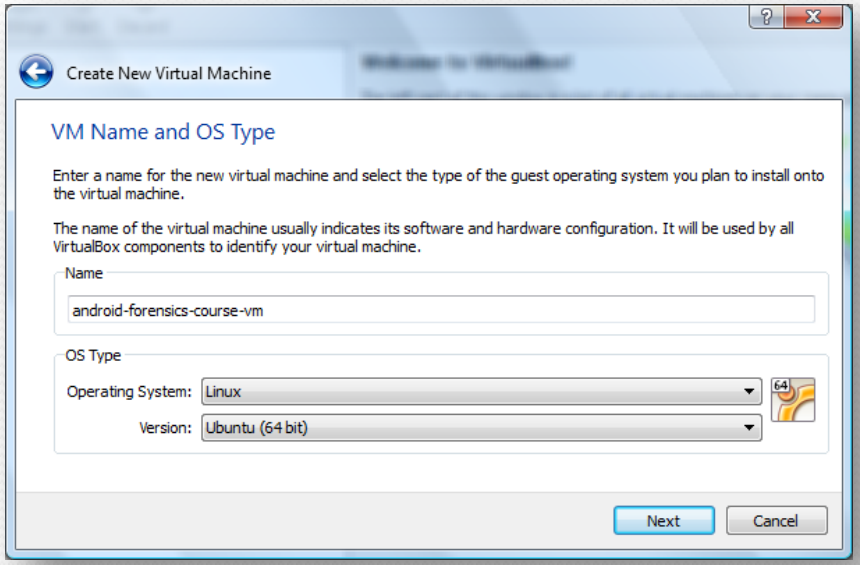
Course Setup

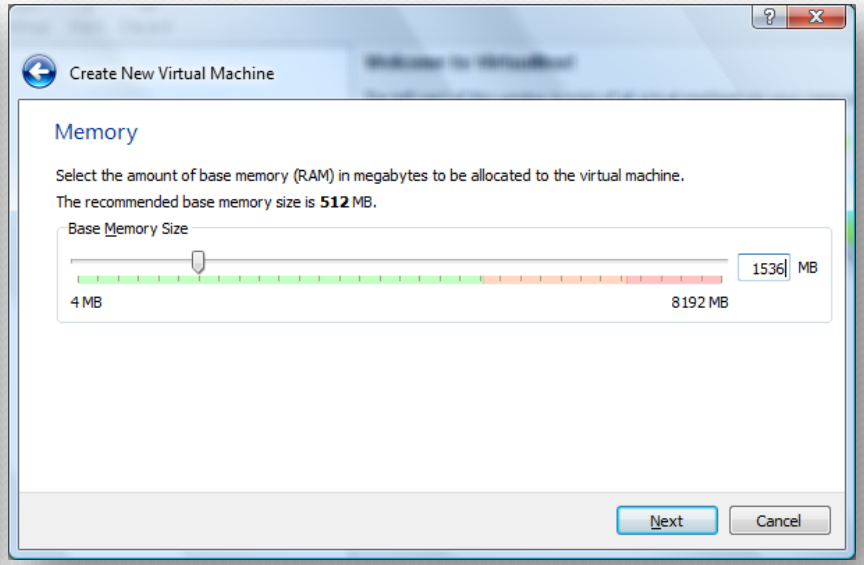
Source:
http://viaforensics.com/services/mobile-forensics/android-forensics/

Course Setup

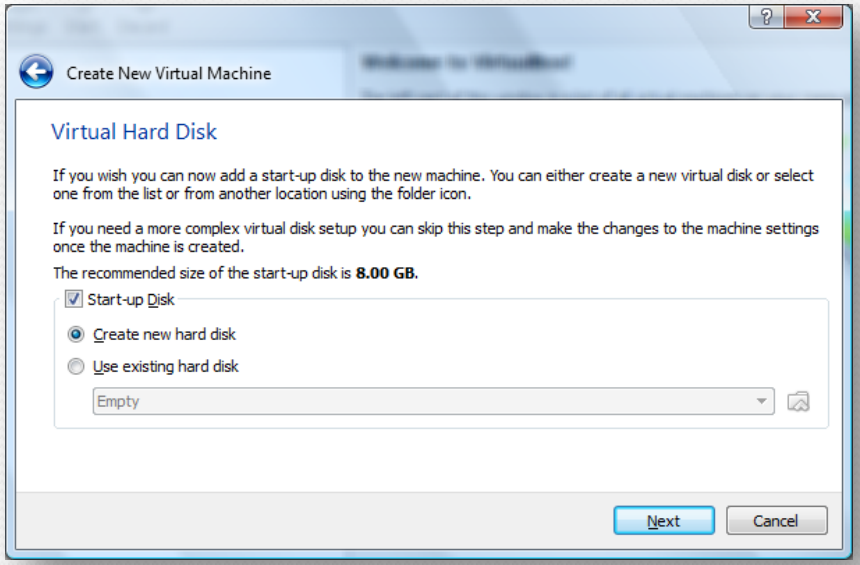
- Ubuntu Linux distribution with Android SDK
  - Ubuntu 11.10 32-bit running on VMWare. Fully functional free, open-source environment for you to keep after the course is over.
  - http://www.vmware.com/
  - Need 20GB hard drive space and 2+GB RAM for the VM
  - http://www.ubuntu.com/download/ubuntu/download
- Windows for some commercial tools

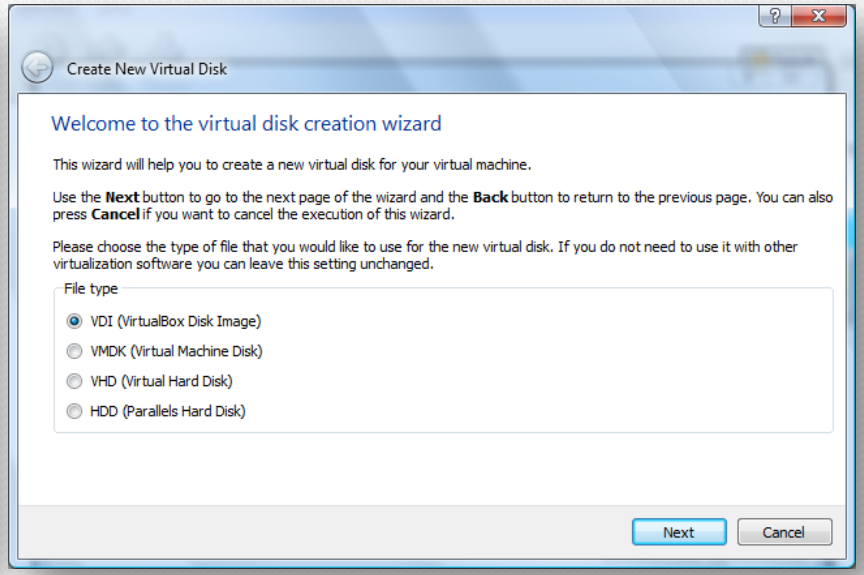
VM Setup – 1 of 7



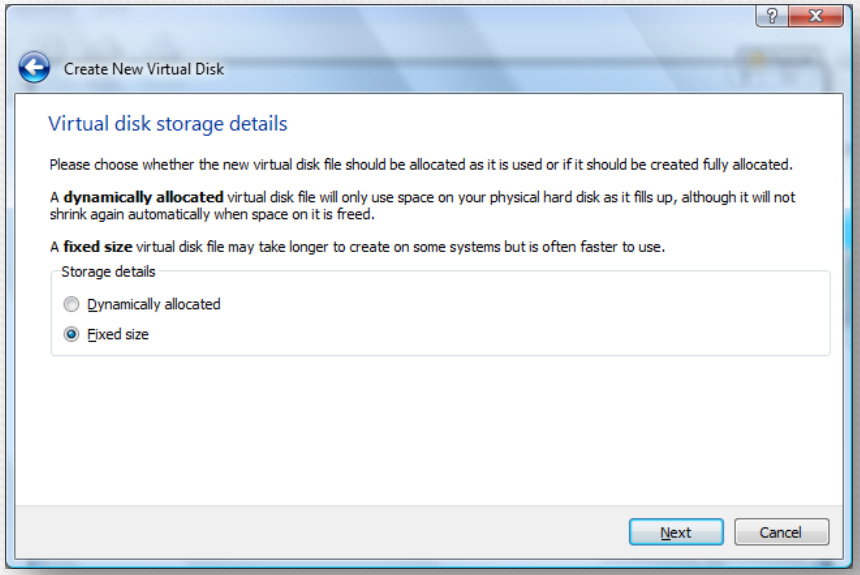


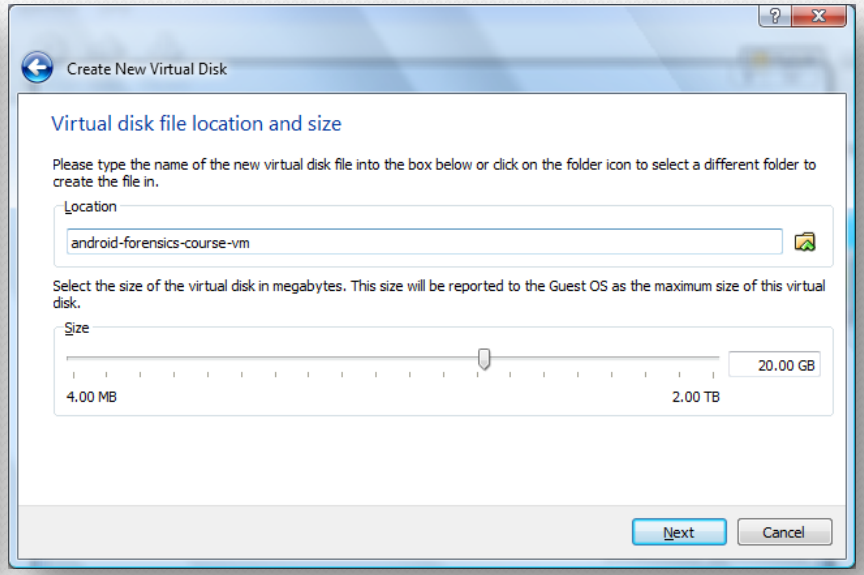
VM Setup – 2 of 7



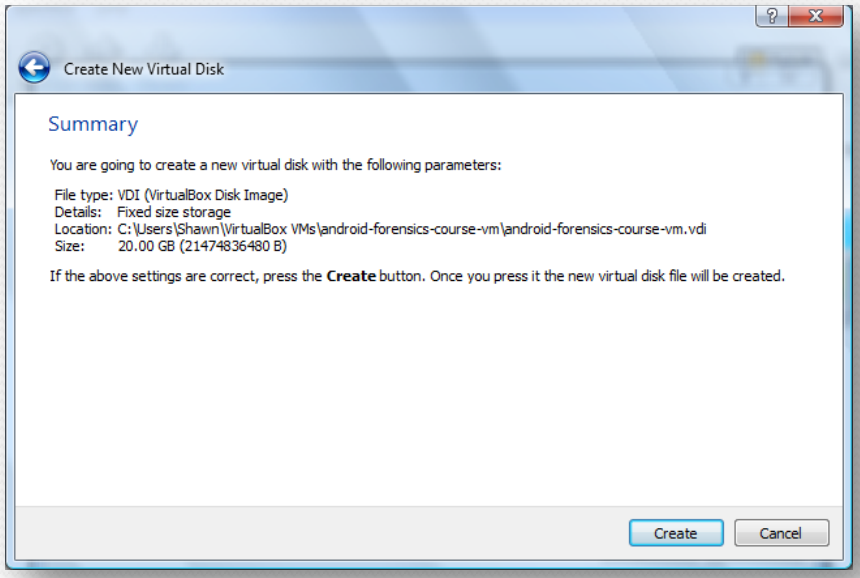


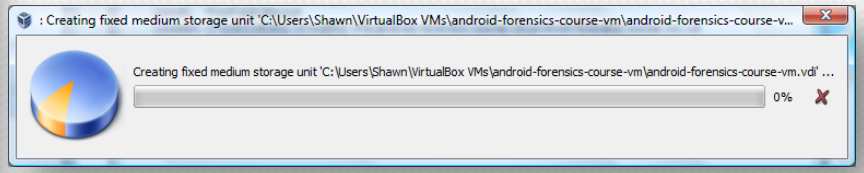
VM Setup – 3 of 7



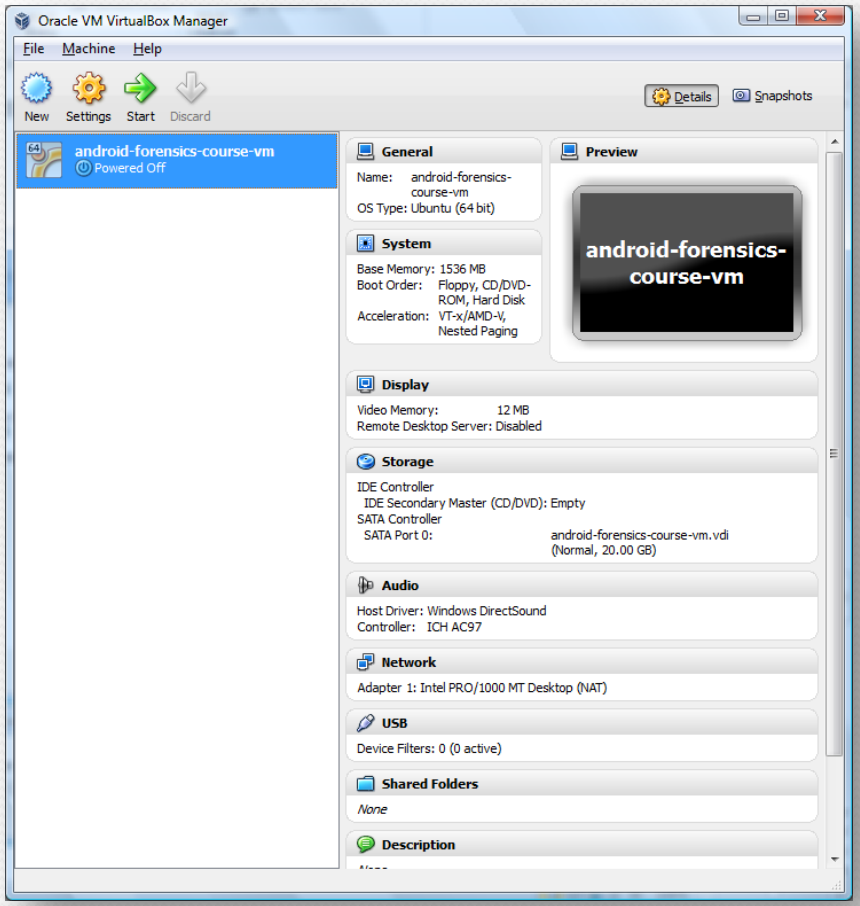


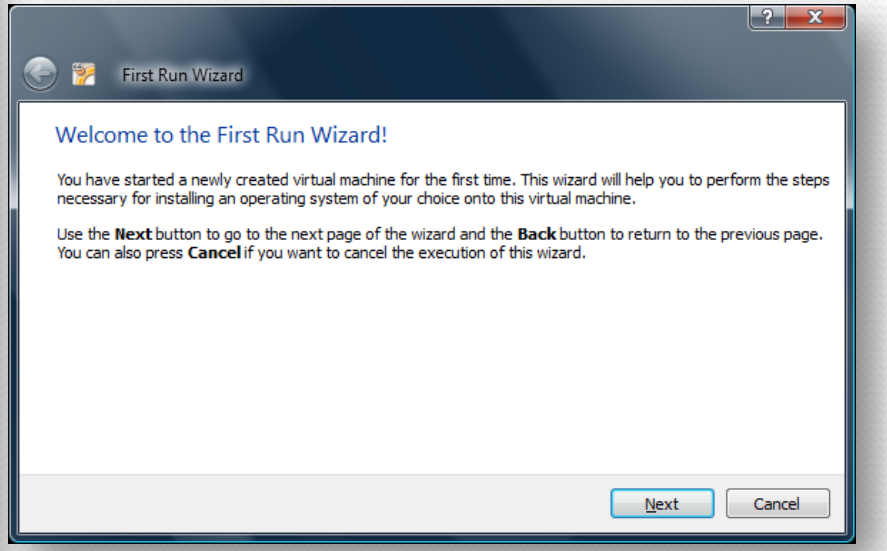
VM Setup – 4 of 7



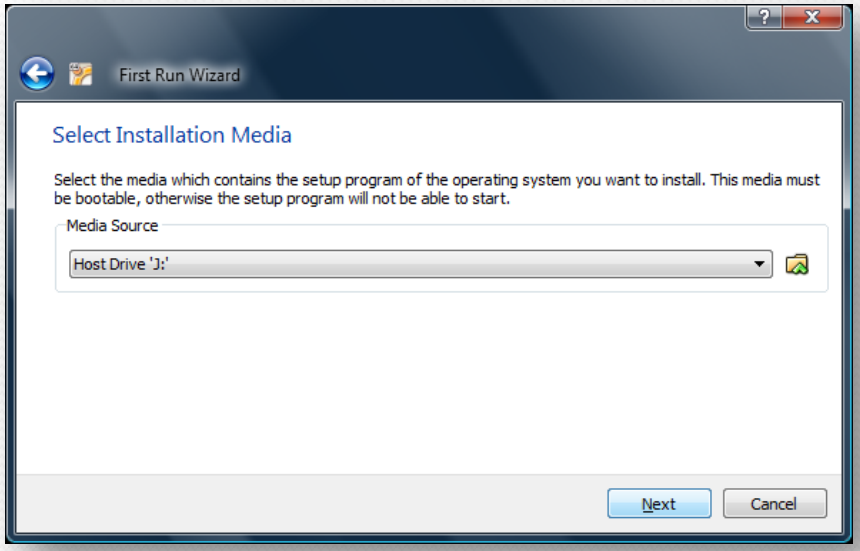


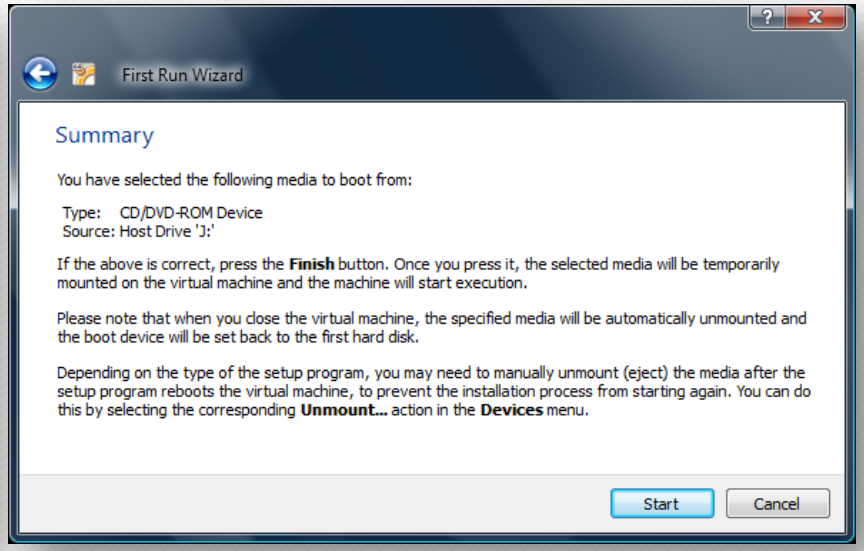
VM Setup – 5 of 7



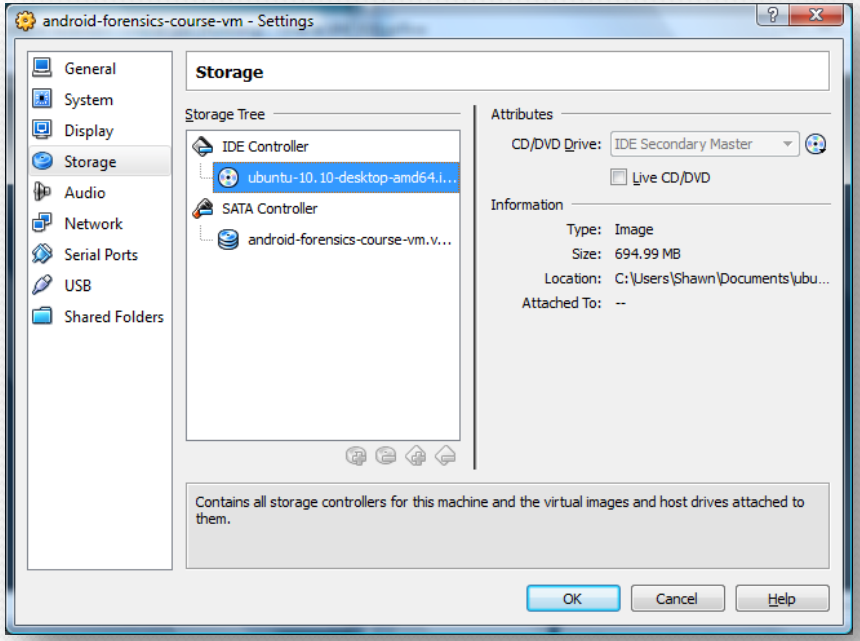


VM Setup – 6 of 7





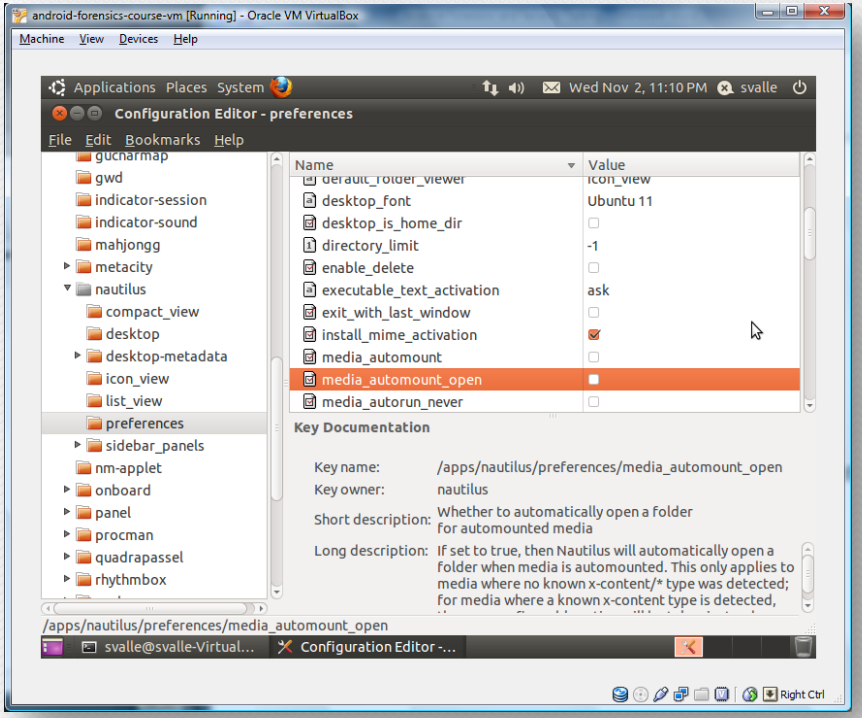
VM Setup – 7 of 9



- Ubuntu
  - User name: student
  - Password: password1

Forensic Workstation Setup

- Disable Automount
  - Command: gconf-editor
  - Navigate to apps > nautilus > preferences, uncheck "media_automount" and "media_automount_open"

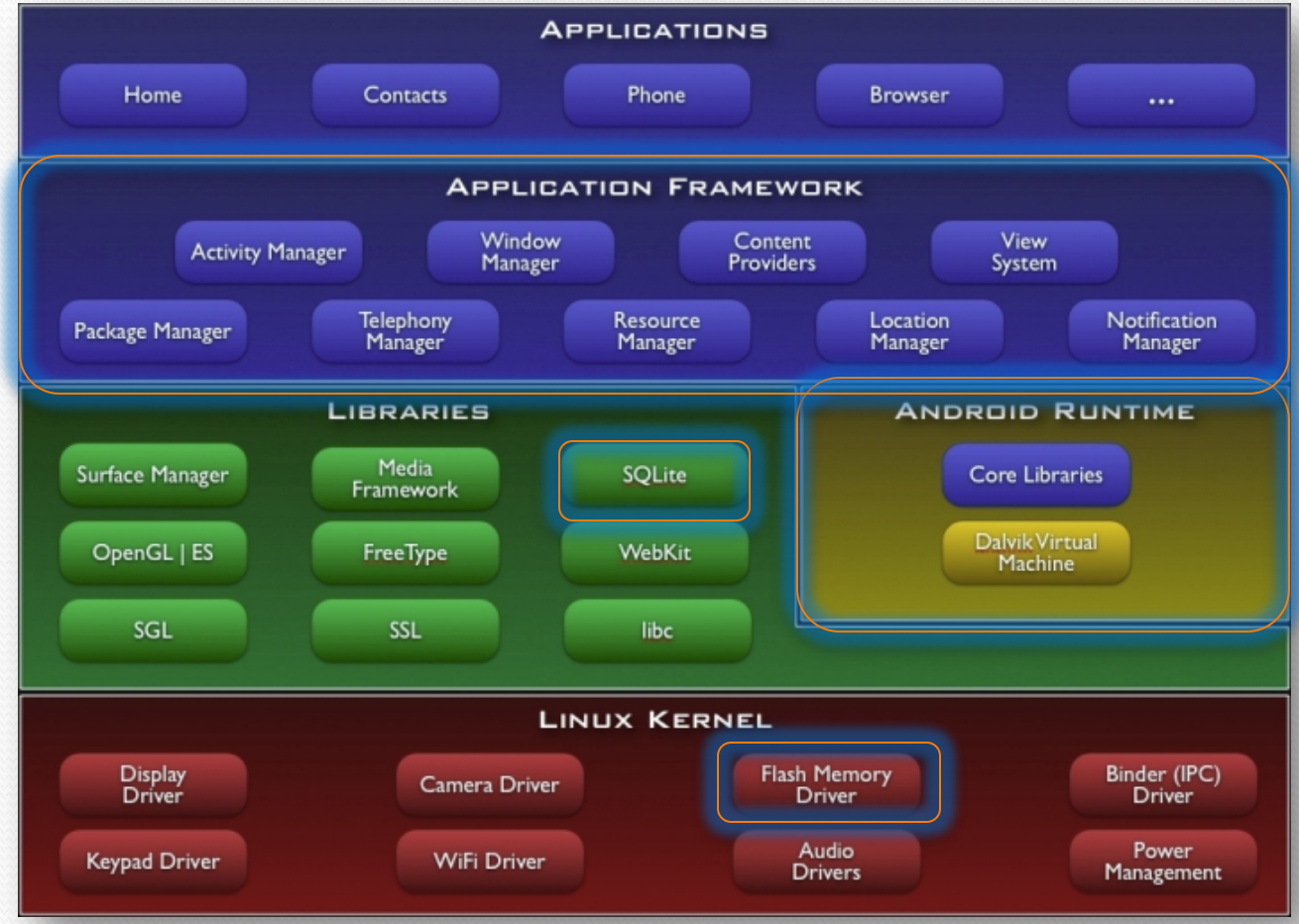



```
student@ubuntu:~$ gconf-editor
The program 'gconf-editor' is currently not installed.  You can install it by typing:
sudo apt-get install gconf-editor
```

Android Architecture

Source:
http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Got Android?



http://developer.android.com/guide/basics/what-is-android.html

Much ado about hardware

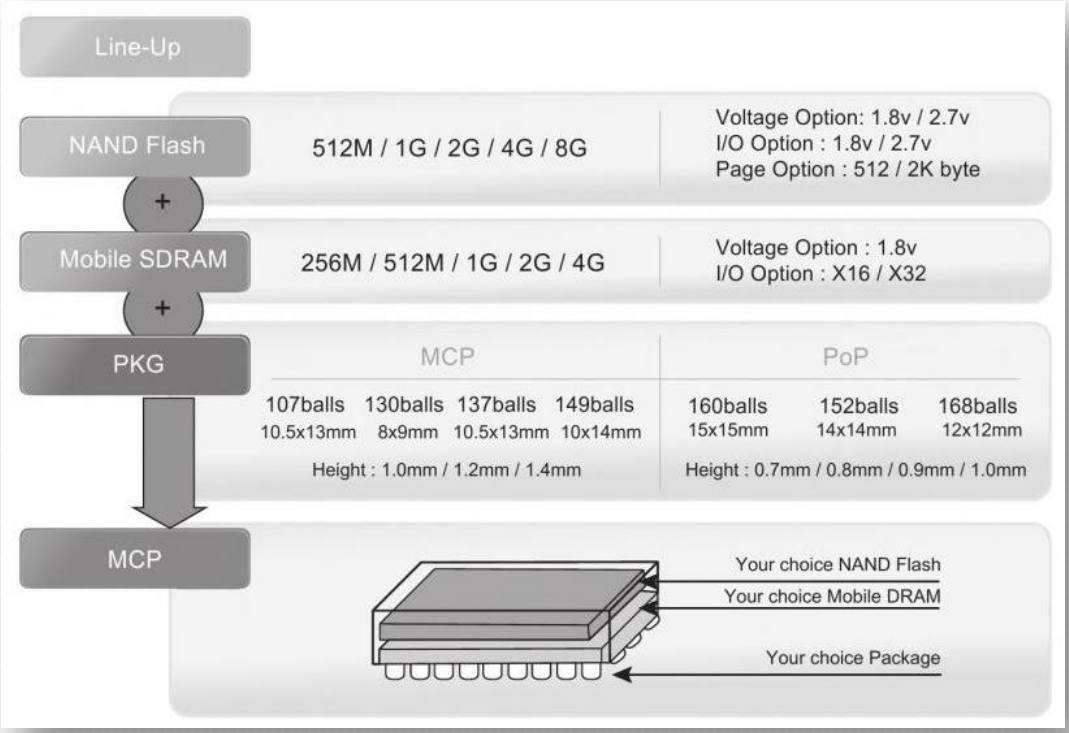
http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Hardware - core

- CPU
- Radio
- Memory (RAM & NAND Flash)
- GPS
- WiFi
- Bluetooth
- SD Card
- Screen
- Camera(s)
- Keyboard
- Battery
- USB
- Accelerometer / Gyroscope
- Speaker
- Microphone
- SIM

More Memory

- Memory (RAM & NAND Flash)
- Manufactured together into multichip package (MCP)

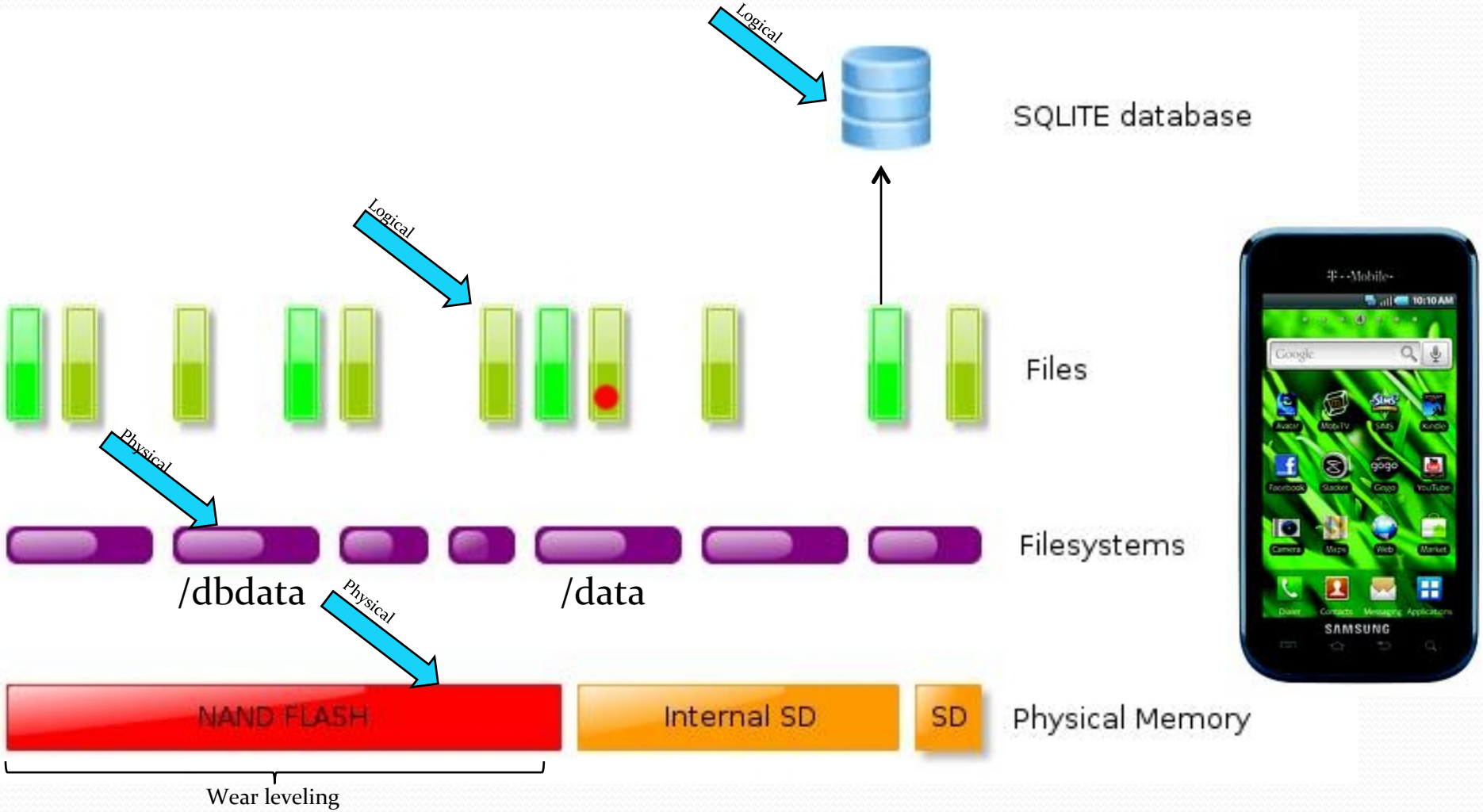


http://www.hynix.com/products/mobile/mcp.jsp?menuNo=1&m=4&s=4

Samsung Vibrant Galaxy S

Android 2.2 (Froyo)

Android Forensics



Source: Mark Guido, MITRE

Hardware - devices

- Smartphones
- Tablets
- Google TV
- Vehicle Stereos
- Standalone GPS
- Kindle Fire
- B&N Nook

- 700+ Android devices

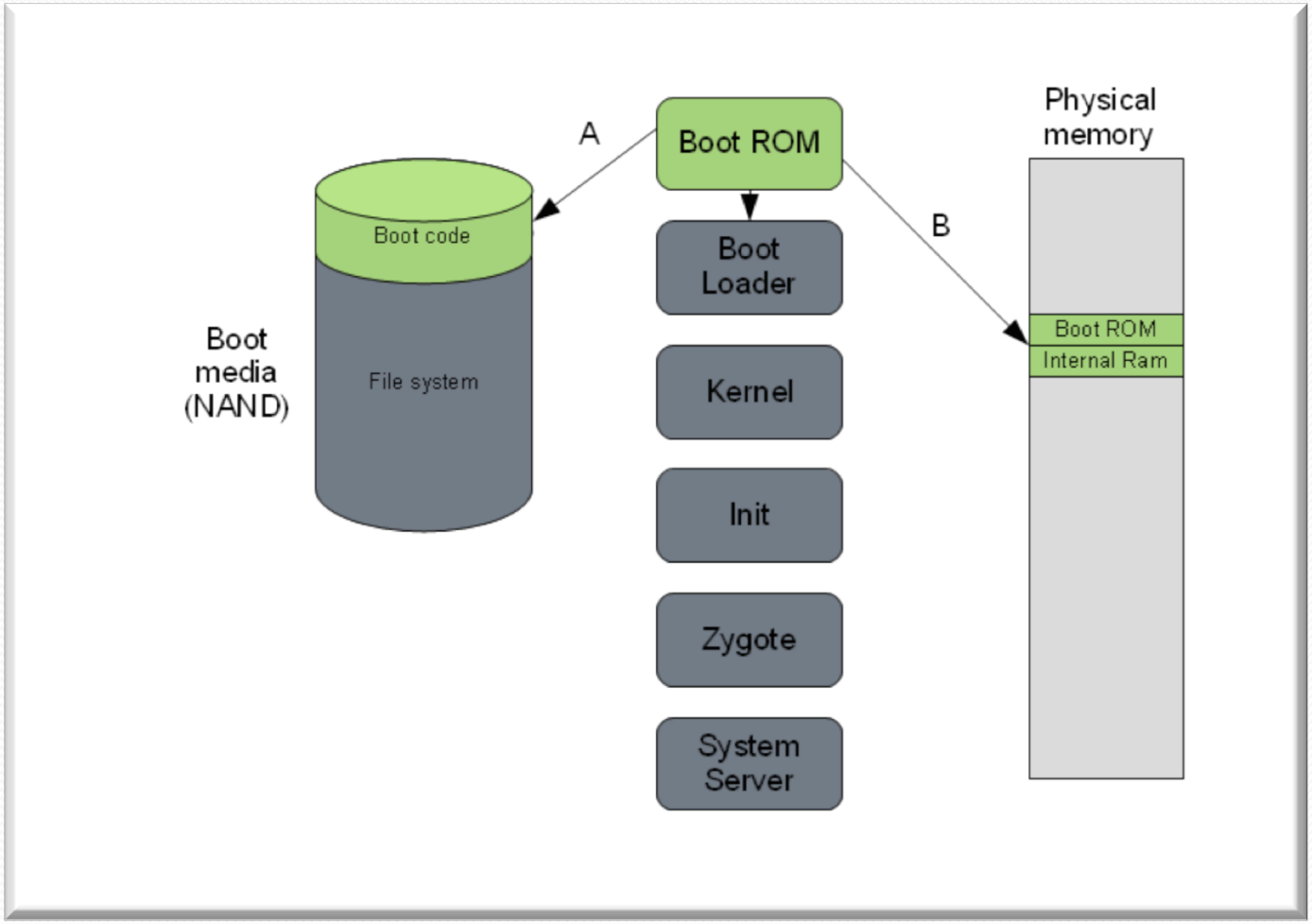
ROM & Boot Loaders

- ROM varies by manufacturer
- Contains boot process

- seven key steps to the Android boot process:
  1. Power on and on-chip boot ROM code execution
  2. The boot loader
  3. The Linux kernel
  4. The init process
  5. Zygote and Dalvik
  6. The system server
  7. Boot complete

Source: "The Android boot process from power on" by Mattias Björnheden of the Android Competence Center at Enea

ROM & Boot Loaders



Source: "The Android boot process from power on" by Mattias Björnheden of the Android Competence Center at Enea

Security Model

Security Model

- At app (.apk) installation, Android checks for developers unique signature.
  - NOTE: Not signed by a CA.
  - Key is the responsibility of the developer.
- After signature validation, Android check the permissions (AndriodManifest.xml) needed for the app, designated by the developer.
  - For example: network access, GPS access, access to storage
- Checking an app's permissions, compared to its functionality could give a clue if an app has potential malicious intent. Important area to look at for forensics analysis.

Application Security

- Quick intro/review of Android security model
- Every application (.apk) gets a unique Linux user ID and group ID
- Apps run with their unique user ID
- Each running app gets its own dedicated process and a dedicated Dalvik VM
- Each app has its own storage location in /data/data/<app>, only accessible by the unique user ID and group ID
- Apps can share data with other apps using Content Providers (see Intro to Android App Dev for details)

Source: Geary Sutterfield, MITRE

Android SDK


Android 4.0.3
Android 4.0.3 is an update to the Ice Cream Sandwich release that adds a handful of new features for users and developers. Check out the Platform Highlights for an overview of all features in Android 4.0.x.
For information about API changes in 4.0.3 (API level 15), read the platform notes and diff report. If you're new to Android, get started with the SDK starter package.

Android Tools Needed

- Android SDK (Software Development Kit)
  - Though we are not going to use any of the development tools for device forensics
- AVD (Android Virtual Device)
- ADB (Android Debug Bridge)

SDK Setup

- Android 4.1 (newest) , 2.3.3, and 2.2 (most used) SDK is already installed on Ubuntu workstation
- For more: http://blog.markloiseau.com/2011/06/how-to-install-the-android-sdk-and-eclipse-in-ubuntu/
- Eclipse installed, not needed for device forensics, but will be used for later application reverse engineering

SDK Install via command

```
svalle@svalle-VirtualBox:~$ sudo apt-get install ia32-libs
[sudo] password for svalle:
Reading package lists... Done
Building dependency tree
Reading state information... Done
ia32-libs is already the newest version.
```

copy the android sdk to /opt
sudo -s cp -r android-sdk_r20.0.3-linux.tgz /opt

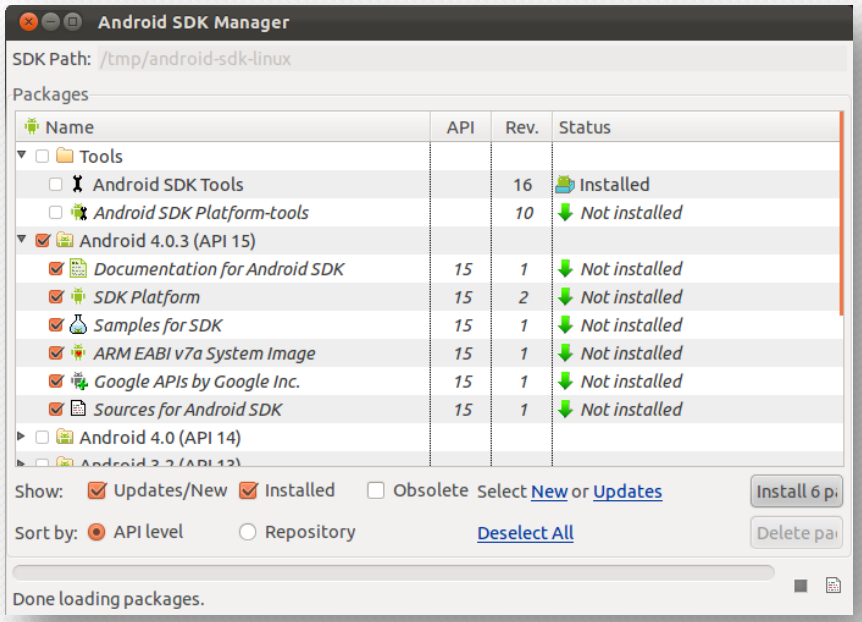
change you into the Android working directory
cd /opt

unpack your Android SDK
sudo -s tar xvzf android-sdk_r20.0.3-linux.tgz

make the /opt directory and the Android SDK writable and executable for all users
sudo -s chmod -R 755 /opt/android-sdk-linux

SDK Manager

- Starting up Android SDK and Android Virtual Device (AVD) manager from terminal
  - Icon on desktop, or
  - `$ cd /opt/android/tools`
  - `$ ./android`

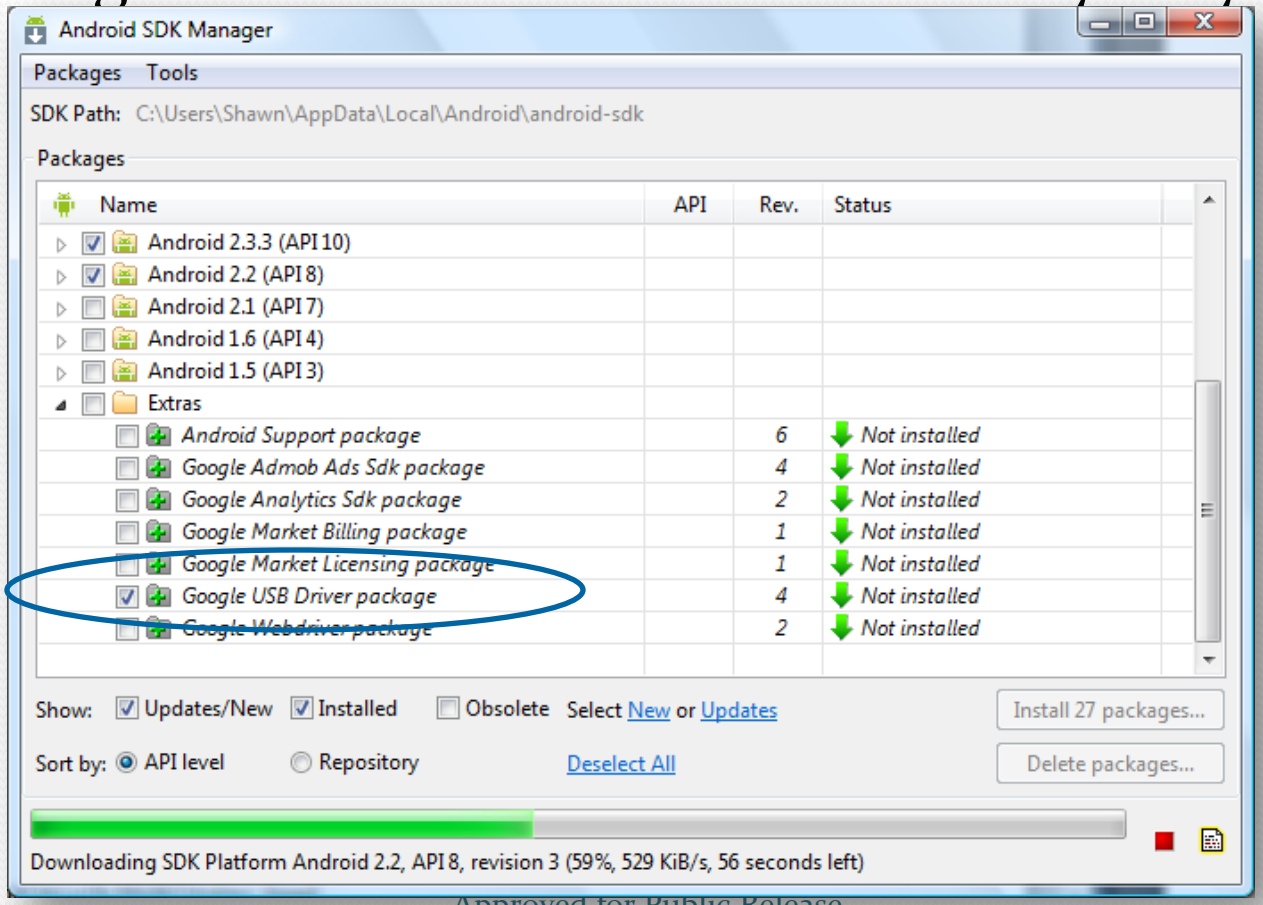


SDK Plugins

- Download the SDK plugins you want.
  - For us: 4.1 (newest), 2.3.3, and 2.2 (most used)
  - Choose whichever SDK is appropriate for the device you are analyzing.

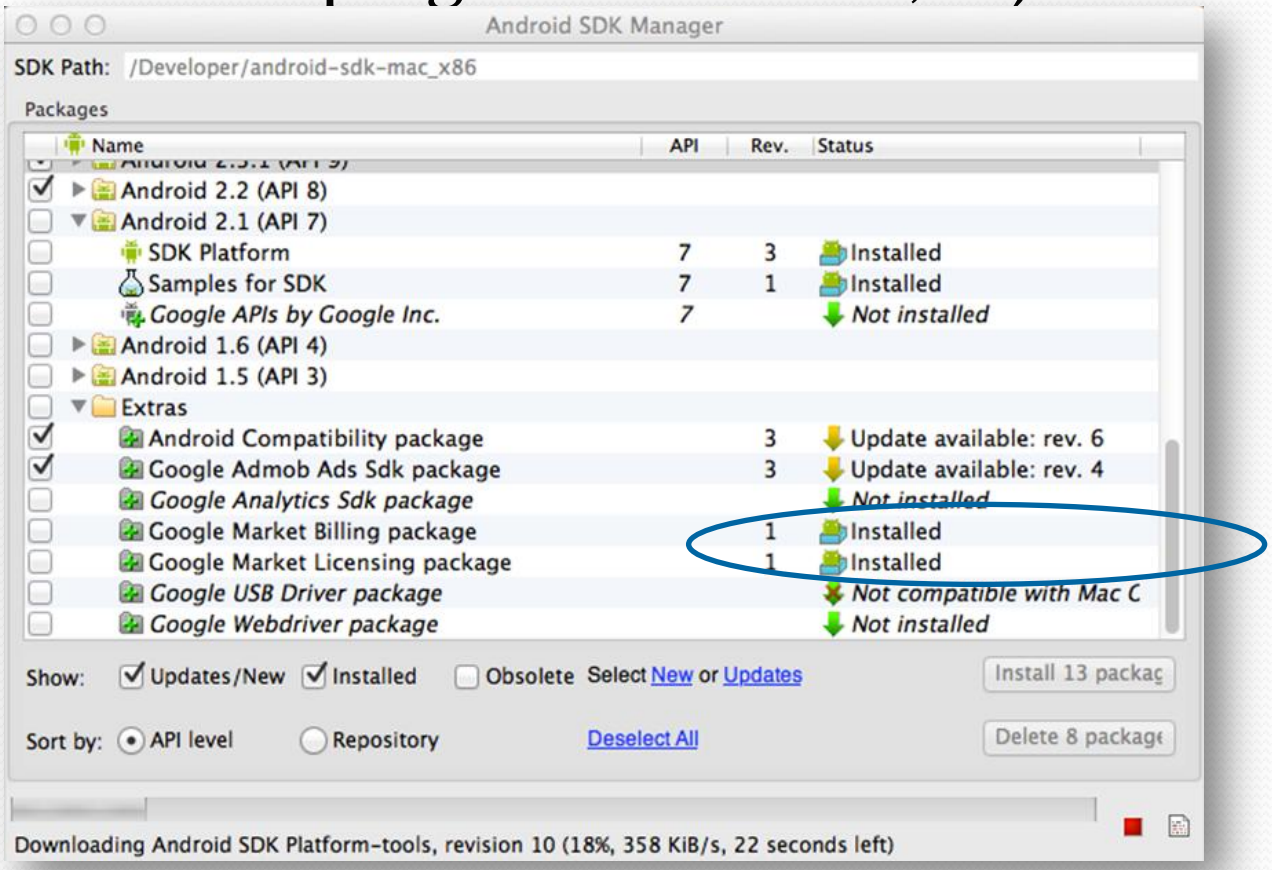
USB Drivers in Windows

- Adding USB Drivers in Windows is very easy.



USB Drivers in OS X Lion (1 of 2)

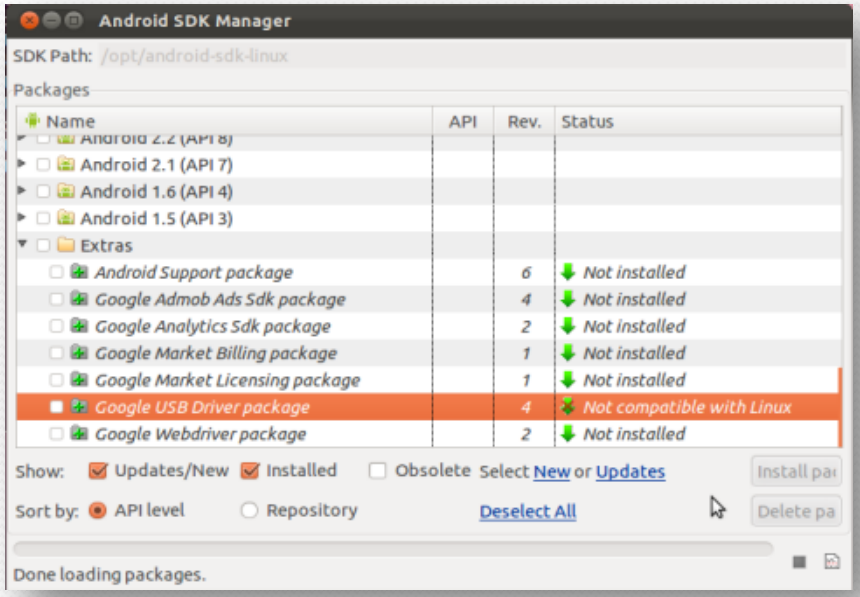
- If you're developing on Mac OS X, it just works.



USB Drivers in OS X Lion (2 of 2)

- Update PATH for Android tools
  - `nano -w ~/.bash_profile`
  - `export PATH=${PATH}:<sdk>/tools:<sdk>/platform-tools`
  - Close / reopen Terminal

USB Drivers in Linux



- Add a udev rules file
- Contains a USB configuration for each type of device

USB Vendor IDs

- This table provides a reference to the vendor IDs needed in order to add USB device support on Linux. The USB Vendor ID is the value given to the ATTR {idVendor} property in the rules file, as described above.

Company	USB Vendor ID
Acer	0502
ASUS	0B05
Dell	413C
Google	18D1
HTC	0BB4
Lenevo	17EF
LG	1004
Motorola	22B8
Nook	2080
Samsung	04E8
Toshiba	0930

UDEV Rules

- Log in as root and create this file:
  - `sudo nano -w /etc/udev/rules.d/51-android.rules`
    Use this format to add each vendor to the file:
    `SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", MODE="0666", GROUP="plugdev"`
- I used:
  - `#HTC`
  - `SUBSYSTEM=="usb", SYSFS{idVendor}=="0bb4", MODE="0666"`

Final UDEV Touches

- Make file readable to all:
  - `sudo chmod a+r /etc/udev/rules.d/51-android.rules`
- UDEV Rules Overview:
  - http://reactivated.net/writing_udev_rules.html

Android Virtual Device (AVD)

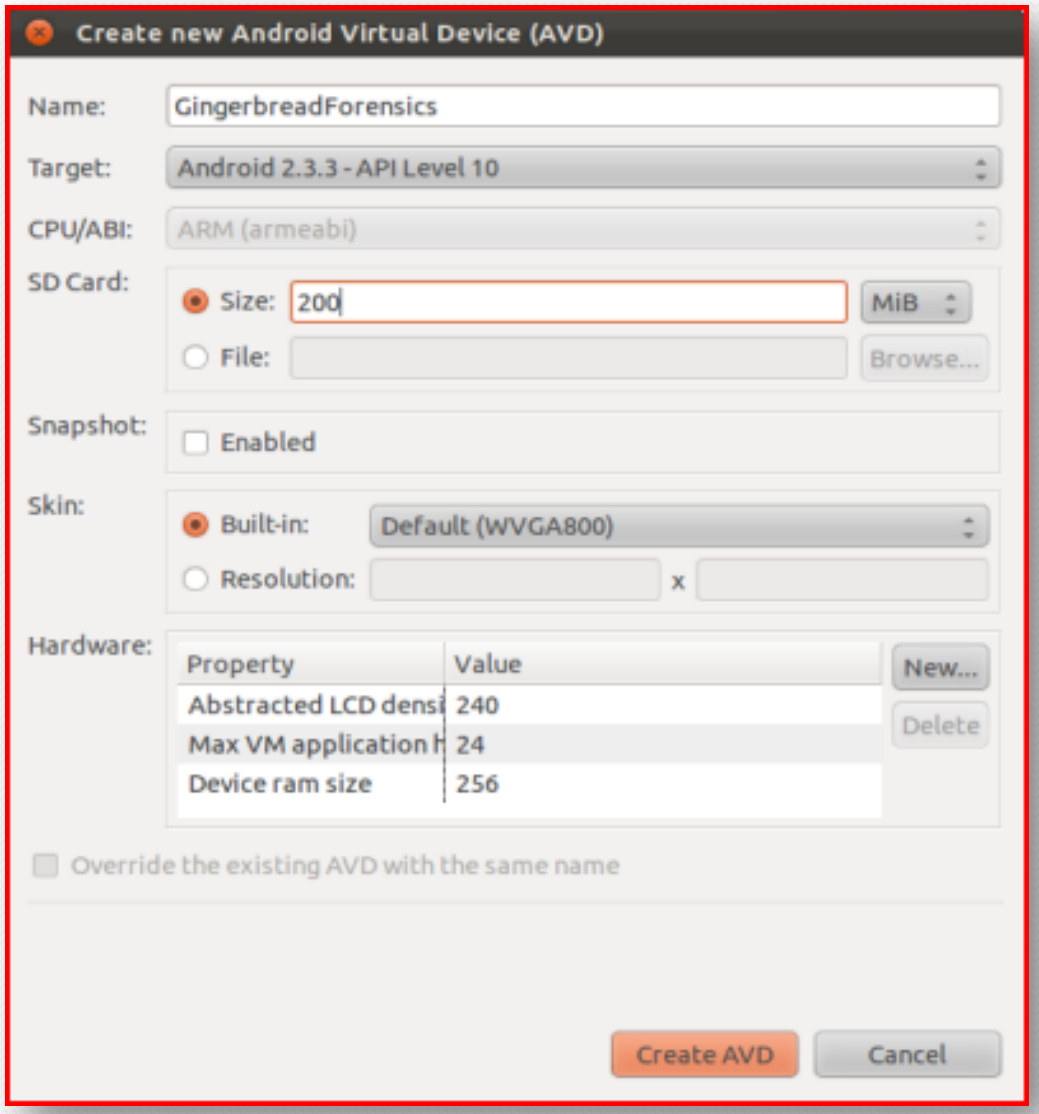
AVD (Emulator) and connecting devices

- Forensics analysts utilize AVD/emulator to learn app execution on a device
- Useful for validating investigation findings
- Useful for testing a forensics or reverse engineering tool an Android device or app
- Terminal: `android` (will start up AVD)

Create AVD

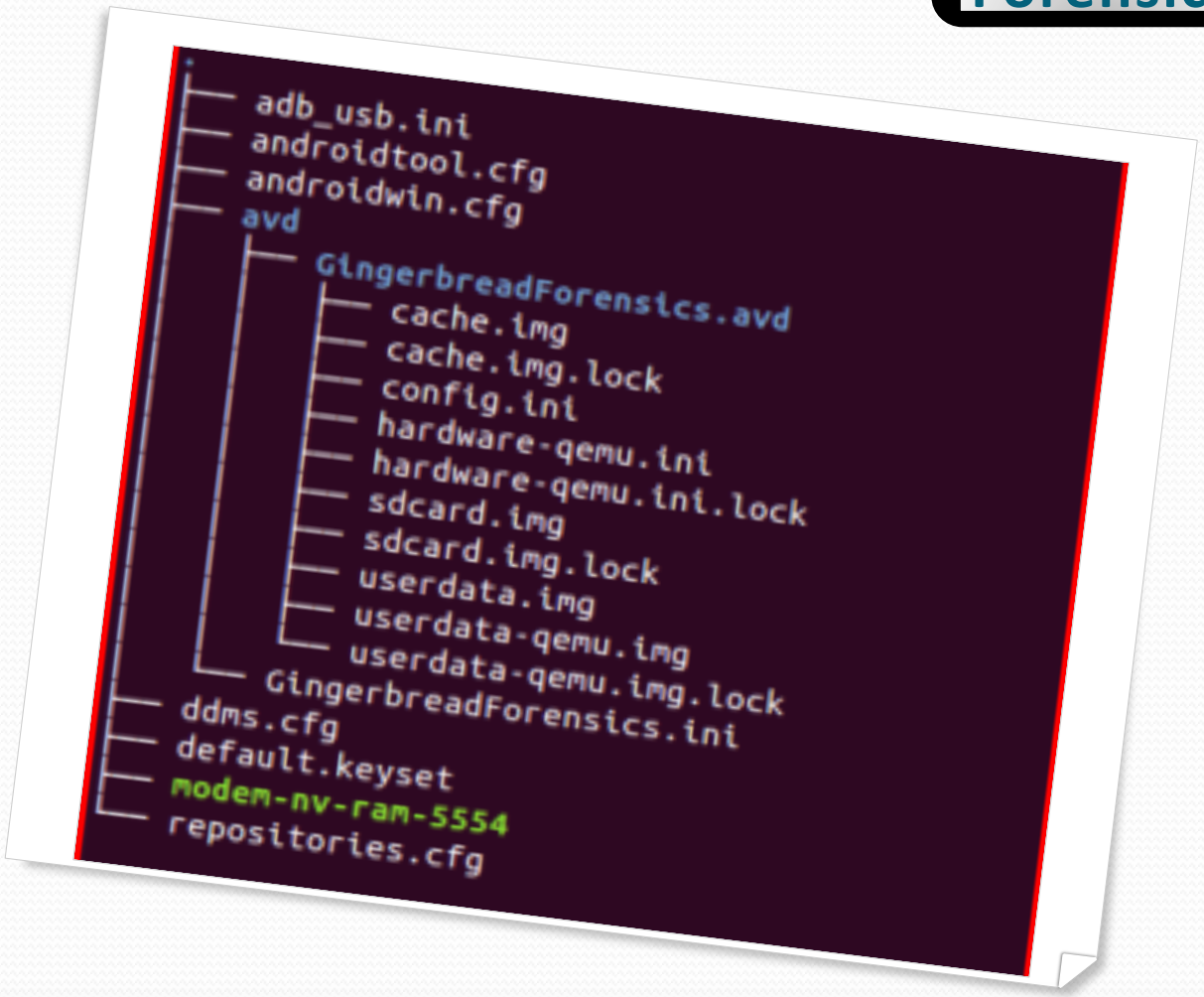
Location of AVD Files:

Desktop OS	AVD Data Location
Ubuntu	/home/<username>/.android
Max OS X	/Users/<username>/.android
Windows 7	C:\Users\<username>\.android



/.android Directory Tree

Command: `tree`




```
The program 'tree' is currently not installed.  You can install it by typing:
sudo apt-get install tree
```

Interesting Files

- **cache.img**: disk image of /cache partition
- **sdcard.img**: disk image of SD card (if created during AVD setup)
- **userdata-qemu.img**: disk image of /data partition

```
svalle@svalle-VirtualBox:~/.android/avd/GingerbreadForensics.avd$ file sdcard.img
sdcard.img: x86 boot sector, code offset 0x5a, OEM-ID "MSWIN4.1", Media descriptor 0xf8, sectors 409600 (volumes
> 32 MB) , FAT (32 bit), sectors/FAT 3175, reserved3 0x800000, serial number 0x13f23c05, label: "     SDCARD"
svalle@svalle-VirtualBox:~/.android/avd/GingerbreadForensics.avd$ file cache.img
cache.img: VMS Alpha executable
svalle@svalle-VirtualBox:~/.android/avd/GingerbreadForensics.avd$ file userdata-qemu.img
userdata-qemu.img: VMS Alpha executable
```

- More details on these directories later

REVIEW

- Learned a brief overview of Android and Linux
- Defined the basics of forensics, penetration testing, and vulnerability assessments
- Explored the hardware components of an Android device
- Familiarized with the Forensics Workstation and Android AVD

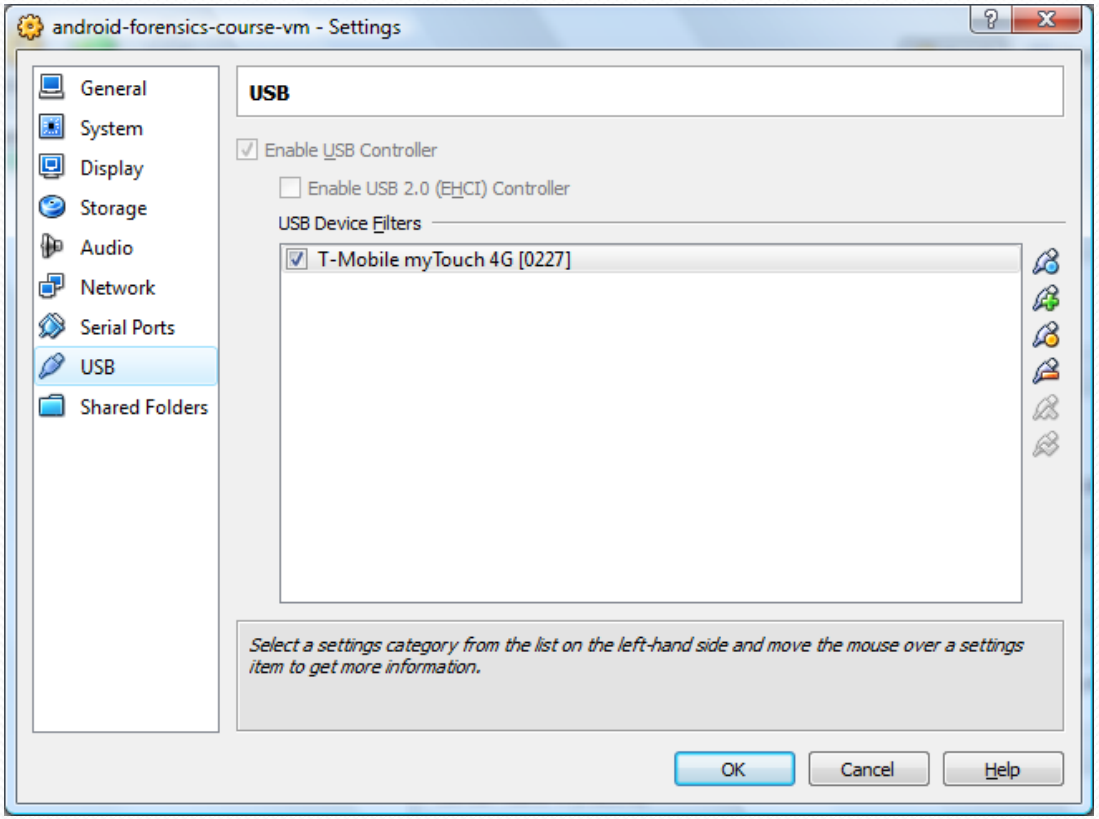
EXERCISE

- Create AVD and explore directories of interest
  - Create FroyoForensics AVD or AVD based on your own Android device
  - Explore `/.android` subdirectories
  - Locate `cache.img`

Connecting a Device for Forensics

Connecting Device to VM

- Mac OS X with VMWare Fusion
- VirtualBox



Setting up USB Interfaces

- Each device has different USB setting options when connected to a PC
- Some options are:
  - Charge only
  - Sync
  - Disk drive
  - Mobile Broadband Connect

USB Connection Test

- To ensure the device is connected and passing through the "host" OS to the Ubuntu VM
  - Open a terminal window and type **dmesg** (display message or driver message)

```
[  672.440249] scsi4 : usb-storage 1-1:1.0
[  673.469697] scsi 4:0:0:0: Direct-Access     T-Mobile myTouch 4G       0100 PQ
: 0 ANSI: 2
[  673.474216] sd 4:0:0:0: Attached scsi generic sg2 type 0
[  673.518763] sd 4:0:0:0: [sdb] Attached SCSI removable disk
```

USB Forensics Precaution

- Important to disable auto-mount to prevent automatic detection and mounting of USB mass storage
- Critical to limit and modifications to device when acquiring forensic data (more later)
- A hardware USB write blocker is an option
- To check for mounted SD cards, use `df` command.

```
svalle@svalle-VirtualBox:~$ df -h
Filesystem             Size  Used Avail Use% Mounted on
/dev/sda1               19G   13G  5.4G  70% /
udev                   743M  4.0K  743M   1% /dev
tmpfs                  301M  756K  300M   1% /run
none                   5.0M     0  5.0M   0% /run/lock
none                   752M  128K  752M   1% /run/shm
```

SD Card Info

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

SD Card

- Most developers store large data files on SD cards.
- Core application data is located in `/sdcard/data/data`

Android Debug Bridge

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

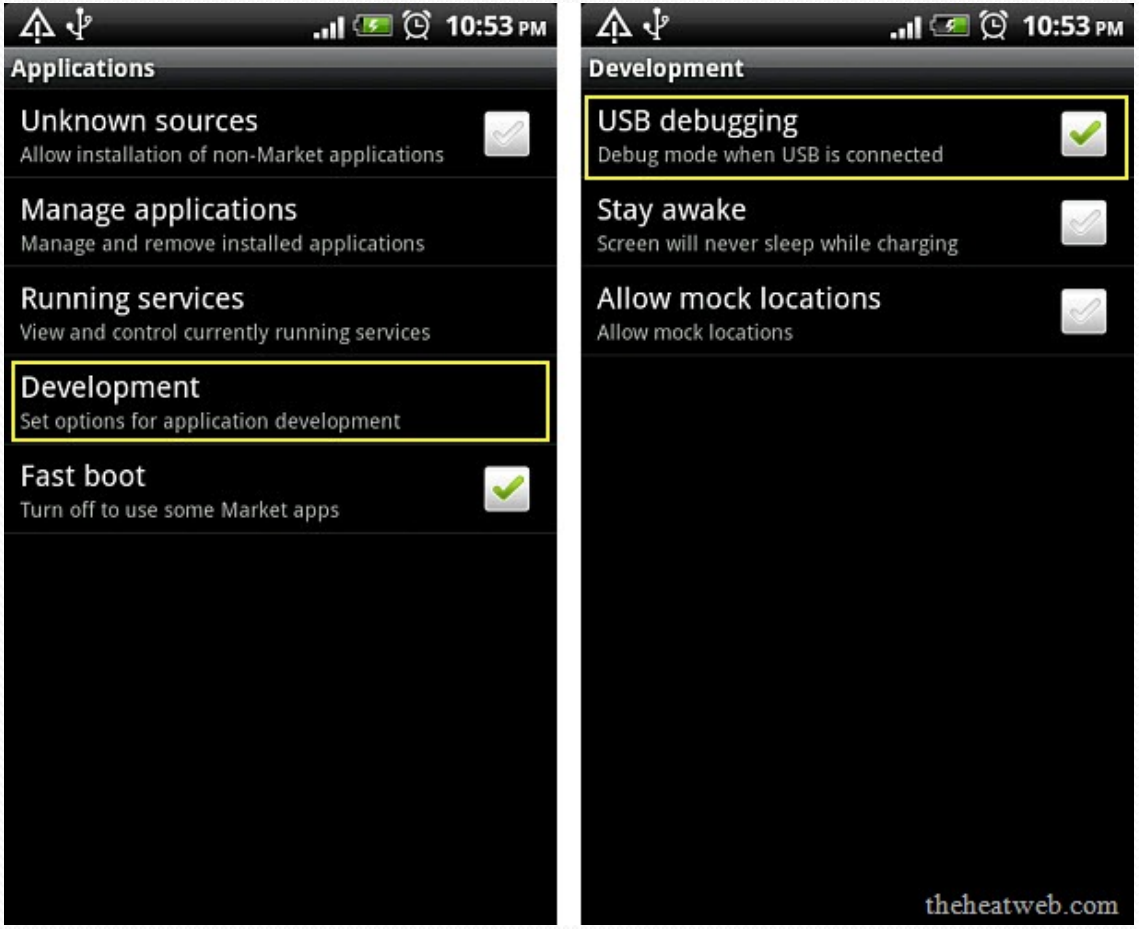
Android Debug Bridge

- One of the most important pieces of Android forensics.
- Best time to pay attention is now.
- Android Debug Bridge (ADB)
  - Developers use this, forensic analysts and security analysts rely on this.

USB Debugging

- Enable USB debugging on device
  - Applications > Development > USB Debugging
  - This will run adb daemon (adbd) on device.
  - adbd runs as a user account, not an admin account. No root access. Unless your device is rooted, then adbd will run as root.
  - If the device is locked with a pass code, enabling USB debugging is difficult.

USB Debugging



Source: http://theheatweb.com

USB Debugging

- Enable USB debugging on device
  - Applications > Development > USB Debugging
  - This will run adb daemon (adbd) on device.
  - adbd runs as a user account, not an admin account. No root access. Unless your device is rooted, then adbd will run as root.
  - If the device is locked with a pass code, enabling USB debugging is difficult.

ADB Components

- Three components
  - adbd on device
  - adbd on workstation
  - adb on workstation

- adb is free, open-source, and our primary tool for Android forensics

ADB Devices

- To identify devices connected, use command **adb devices**

```
svalle@svalle-VirtualBox:~$ adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
SH132RM00905    device
```

Bad ADB

- Sometimes adb doesn't respond properly.
- To kill adb, use command **adb kill-server**


```
svalle@svalle-VirtualBox:~$ adb kill-server
svalle@svalle-VirtualBox:~$ adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
SH132RM00905    device
```

ADB Shell

- To open an adb shell on an Android device, use command **adb shell**

```
svalle@svalle-VirtualBox:~$ adb shell
# 
```

  - Gives full shell access directly on device.
  - Once we learn more about file system and directories, adb shell will get you much of the data needed for forensic analysis

ADB Shell – example

```
svalle@svalle-VirtualBox:~$ adb shell
# cd /data/data
# ls
android.tts
com.PuppyPunch.AGWB
com.PuppyPunch.ChickenCoup
com.RefinedGames.CrossCourtTennis
com.Skillpod.GalacticStriker
com.TwistedGames.Caveman
com.amazon.mp3
com.amazon.venezia
com.android.bluetooth
com.android.browser
com.android.calculator2
com.android.calendar
com.android.camera
com.android.certinstaller
com.android.contacts
com.android.defcontainer
com.android.deskclock
com.android.development
com.android.email
```

- Full list of adb commands at http://developer.android.com/guide/developing/tools/adb.html

REVIEW

- Learned proper technique for connecting Android device to a forensic workstation
- Became familiar with USB Debugging's importance to forensics
- Explored ADB and its relevance to successful investigations

EXERCISE

- Locate data directory on an Android device
  - Connect an Android device to your VM workstation (or startup an AVD)
  - Verify USB Debugging is enabled on the device
  - Start `adb` on your forensic workstation
  - Using `adb shell`, locate directories in `/data/data`
  - Jot down the name of some interesting directories for further exploration later

File System & Data

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Forensics Data Gathered and Analyzed

- SMS History
- Deleted SMS
- Contacts (stored in phone memory and on SIM card)
- Call History
  - Received Calls
  - Dialed Numbers
  - Missed Calls
  - Call Dates & Durations
- Datebook
- Scheduler
- Calendar
- To-Do List
- File System (physical memory)
  - System Files
  - Multimedia Files
  - Java Files / Executables
  - Deleted Data
  - Notepad
  - More…
- GPS Waypoints, Tracks, Routes, etc.
- RAM/ROM
- Databases
- E-mail

File System & Data Overview

- File Systems
- Data Storage
- What Data?
- Important Directories
- Five Data Storage Methods
  - Shared Preferences
  - Internal Storage
  - External Storage
  - SQLite
  - Network
- Where else? Linux Kernel & Android Stack
  - dmesg
  - logcat
- Forensically Thinking

File Systems

- More than a dozen file systems in Android
- More than a dozen file systems in use on Android
- Forensics analysts should understand the most important
  - EXT
  - FAT32
  - YAFFS2
- Most user data live in those
- Want to find the file systems on your device?
- `adb shell cat /proc/filesystems`

Data Storage

- Explore file systems and virtual machines
- Learning the Android file systems, directory structures, and specific files will be crucial to successful Android forensics analysis

What Data?

- Apps shipped with Android (with the OS) – eg. Browser
- Apps installed by manufacturer – eg. Moto Blur
- Apps installed by wireless carrier – eg. CarrierIQ
- Additional Google/Android apps – eg. Google Play Music, Gmail
- Apps installed by the user, from Play Store or elsewhere

Important Directories

- /data/data - Apps data generally installed in a subdirectory
- Example: Android browser is named com.android.browser, data files are stored at /data/data/com.android.browser

```
# cd com.android.browser
# ls -l
drwxrwx--x    2 app_55   app_55        4096 Dec  4 20:51 app_appcache
drwxrwx--x    3 app_55   app_55        4096 Dec  4 20:51 app_databases
drwxrwx--x    2 app_55   app_55        4096 Dec  4 21:20 app_geolocation
drwxrwx--x    2 app_55   app_55        4096 Feb 11 10:50 app_icons
drwxrwx--x    2 app_55   app_55        4096 Dec  4 20:51 app_thumbnails
drwxrwx--x    3 app_55   app_55        4096 Dec  4 20:51 cache
drwxrwx--x    2 app_55   app_55        4096 Dec  4 20:51 databases
drwxr-xr-x    2 system   system        4096 Dec  4 20:45 lib
drwxrwx--x    2 app_55   app_55        4096 Dec  4 21:27 shared_prefs
```

Common Subdirectories

- /data/data/<app package name>/

shared_prefs	XML of shared preferences
lib	Custom library files required by app
files	Developer saved files
cache	Files cached by the app
databases	SQLite databases and journal files

Five Data Storage Methods

- We will be exploring these methods
  - Shared preferences
  - Internal storage
  - External storage
  - SQLite
  - Network

Source: Hoog

Shared preferences

- Key-value XML data
- use cat command to view files

```
# cd shared_prefs
# ls -l
-rw-rw----    1 app_55   app_55        119 Dec  4 21:27 BrowserBookmarksPage.xml
-rw-rw----    1 app_55   app_55        118 Dec  4 20:52 WebViewSettings.xml
-rw-rw----    1 app_55   app_55        290 Dec  4 20:51 com.android.browser_preferences.xml
```

- Can be source of data

```
# cd com.android.phone
# ls -l
drwxrwx--x    2 radio    radio         4096 Dec  4 20:45 files
drwxr-xr-x    2 system   system        4096 Dec  4 20:45 lib
drwxrwx--x    2 radio    radio         4096 Jan 11 15:18 shared_prefs
# cd shared_prefs
# ls -l
-rw-rw----    1 radio    radio          126 Dec  4 20:45 _has_set_default_values.xml
-rw-rw----    1 radio    radio         1203 Jan 11 15:18 com.android.phone_preferences.xml
# cat _has_set_default_values.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="_has_set_default_values" value="true" />
</map>
# cat com.android.phone_preferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="button_return_home" value="true" />
<boolean name="button_hide_hold_button" value="false" />
<boolean name="button_black_regex" value="false" />
<boolean name="rotate_incall_screen" value="false" />
<string name="sip_call_options_key">SIP_ASK_ME_EACH_TIME</string>
<boolean name="button_vibrate_45" value="false" />
<boolean name="button_force_touch" value="false" />
<string name="button_voicemail_provider_key"></string>
<string name="button_trackball_answer_timed">-1</string>
<boolean name="button_left_hand" value="false" />
<boolean name="button_always_proximity" value="false" />
<boolean name="button_screen_awake" value="false" />
<boolean name="button_show_organ" value="false" />
<boolean name="button_led_notify" value="true" />
<boolean name="button_vibrate_outgoing" value="true" />
<boolean name="bg_incall_screen" value="false" />
<boolean name="button_vibrate_call_waiting" value="false" />
<string name="network_selection_key"></string>
<boolean name="button_vibrate_hangup" value="true" />
<string name="network_selection_name_key"></string>
<string name="button_trackball_hangup_timed">-1</string>
</map>
```

Shared preferences – example

```
# cd org.gtmedia.seekdroid
# ls -l
drwxrwx--x    2 app_94   app_94        4096 Dec 27 18:32 files
drwxr-xr-x    2 system   system        4096 Jan 29 19:07 lib
drwxrwx--x    2 app_94   app_94        4096 Jan 31 10:30 shared_prefs
# cd files
# ls -l
# cd ..
# cd lib
# ls -l
# cd ..
# cd shared_prefs
# ls -l
-rw-rw----    1 app_94   app_94         629 Jan 31 10:30 SDPrefs_V2.xml
-rw-rw----    1 app_94   app_94         391 Jan  8 16:37 org.gtmedia.seekdroid_pref
erences.xml
-rw-rw----    1 app_94   app_94          65 Jan 31 10:30 prefs.xml
# cat SDPrefs_V2.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="DeviceID">49817</string>
<boolean name="OldPrefsRemoved" value="true" />
<string name="AndroidId">ccd923b94dab61bf</string>
<boolean name="WipePhone" value="true" />
<long name="Backoff" value="3000" />
<boolean name="REGISTERED" value="true" />
<string name="C2DMRegId">APA91bG1ENns4hCXZwEiTMf5REL1K5cH-yOCoGqsj0z3Vwx1uua_qQVJZ3
wdx0g89pYj9DfP-U8SIdLFvezBXuaud3WWojX_pkC0V3dOXG9Tixakf1XF5c4BjEy7r-6jPzoMjktdjoEhq
GGcaNxaisfW1I3bGool6g</string>
<string name="SMSCode">[redacted]</string>
<string name="AccountEmail">shawnvalle@gmail.com</string>
</map>
```

- Android device security application
- Exploring shared_prefs, and SDPrefs_V2.xml, my user name and password are stored in the clear

Shared preferences – example

```
# cd shared_prefs
# ls -l
-rw-rw----    1 app_82   app_82          259 Dec  8 15:06 FileConfig.xml
-rw-rw----    1 app_82   app_82          413 Dec  8 14:30 FileConn.xml
-rw-rw----    1 app_82   app_82          112 Dec  8 14:29 MyPrefsFile.xml
# cat MyPrefsFile.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<boolean name="dialogue" value="false" />
</map>
# cat FileCOnn.xml
FileCOnn.xml: No such file or directory
# cat FileConn.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<int name="port" value="443" />
<boolean name="remember" value="true" />
<int name="sizeNotiLabel" value="76" />
<string name="path">zdm</string>
<boolean name="welcome" value="false" />
<string name="user">svalle</string>
<string name="ip">mdmlab.org</string>
<string name="pass">[redacted]</string>
<boolean name="usePortSSL" value="true" />
</map>
```

- MDM product
- Stores entire connection string, including user name, domain, and password in clear text

Internal storage

- Common file systems used: ext3, ext4, yaffs2.
- By default, files stored in /data/data are encrypted, accessed only by the application. Commonly root access is needed to access these files.

```
# cd /data/data/com.google.android.apps.maps
# ls -l
drwxrwx--x    2 app_84   app_84        4096 Dec  9 11:44 app_sslcache
drwxrwx--x    5 app_84   app_84        4096 Dec  9 11:44 cache
drwxrwx--x    2 app_84   app_84        4096 Jan 28 10:46 databases
drwxrwx--x    2 app_84   app_84        4096 Feb 15 11:29 files
drwxr-xr-x    2 system   system        4096 Jan 28 07:34 lib
drwxrwx--x    2 app_84   app_84        4096 Jan 28 10:46 shared_prefs
```

Internal storage

```
# cd files
# ls -l
-rw-rw----    1 app_84   app_84        2000 Feb 15 11:29 DATA_Preferences
-rw-rw----    1 app_84   app_84           6 Dec  9 11:44 DATA_RECENT
-rw-rw----    1 app_84   app_84       37923 Dec  9 11:44 DATA_RemoteStringsBlock_en
-rw-rw----    1 app_84   app_84       45522 Jan 28 10:46 DATA_Restrictions
-rw-rw----    1 app_84   app_84           0 Jan 28 10:46 DATA_Restrictions_lock
-rw-rw----    1 app_84   app_84           6 Dec  9 11:44 DATA_STARRING
-rw-rw----    1 app_84   app_84          32 Dec  9 11:44 DATA_SYNC_DATA_LOCAL
-rw-rw----    1 app_84   app_84          73 Jan 28 10:46 DATA_ServerControlledParametersManager.data
-rw-rw----    1 app_84   app_84          73 Jan 28 10:39 DATA_ServerControlledParametersManager_DA.data
-rw-rw----    1 app_84   app_84           4 Feb 15 11:29 DATA_TILE_HISTORY
-rw-rw----    1 app_84   app_84         330 Jan 28 10:46 DA_DirOpt_en_US
-rw-rw----    1 app_84   app_84        5294 Dec  9 11:44 DA_LayerInfo
-rw-rw----    1 app_84   app_84         573 Dec  9 11:44 NavZoomTables.data
-rw-rw----    1 app_84   app_84          26 Dec  9 11:44 NavigationParameters.data
-rw-rw----    1 app_84   app_84        1186 Dec  9 11:44 ZoomTables.data
-rw-------    1 app_84   app_84          35 Feb 15 18:40 cp_state
-rw-rw----    1 app_84   app_84         377 Jan  7 18:07 event_store_driveabout
-rw-rw----    1 app_84   app_84         515 Jan 28 10:46 event_store_v2_driveabout
-rw-------    1 app_84   app_84          34 Feb 15 18:40 nlp_clts
-rw-------    1 app_84   app_84          61 Feb  9 22:03 nlp_params
```

- Notice user “app_84” is the owner. That user was created when Google Maps was installed
- There’s a lot of potential rich forensic maps data in these directories

External storage

- External storage (SD Card) have less permission restrictions.
- FAT32 does not have fine-grain permissions of other file systems.

```
# cd sdcard/Android/data
# ls -l
---xrwxr-x    1 system   sdcard_r       64 Feb 12 15:33 cache_fiaeaej.dat
d--xrwxr-x    3 system   sdcard_r    32768 Dec  8 06:31 com.TwistedGames.Caveman
d--xrwxr-x    3 system   sdcard_r    32768 Dec  7 20:54 com.amazon.venezia
d--xrwxr-x    3 system   sdcard_r    32768 Jan  2 11:38 com.android.providers.media
d--xrwxr-x    3 system   sdcard_r    32768 Dec  7 20:03 com.cooliris.media
d--xrwxr-x    3 system   sdcard_r    32768 Jan  8 12:43 com.daylightmap.moon.pro.android
d--xrwxr-x    3 system   sdcard_r    32768 Dec 29 21:42 com.disney.WMW
d--xrwxr-x    3 system   sdcard_r    32768 Dec 10 20:23 com.facebook.katana
d--xrwxr-x    3 system   sdcard_r    32768 Jan  2 13:58 com.gameloft.android.ANMP.GloftM3HM
d--xrwxr-x    2 system   sdcard_r    32768 Jan  7 21:59 com.google.android.apps.genie.geniewidget.news-content-cache
d--xrwxr-x    5 system   sdcard_r    32768 Dec  9 11:44 com.google.android.apps.maps
d--xrwxr-x    3 system   sdcard_r    32768 Dec  7 21:34 com.google.android.music
d--xrwxr-x    3 system   sdcard_r    32768 Jan 15 17:53 com.slacker.radio
d--xrwxr-x    3 system   sdcard_r    32768 Dec  7 20:07 com.southwindsgames.am2m
d--xrwxr-x    3 system   sdcard_r    32768 Jan  8 10:15 com.touchtype.swiftkey.phone.trial
d--xrwxr-x    4 system   sdcard_r    32768 Feb  1 23:52 com.zynga.words
```

SQLite

- Lightweight open-source relational database
- Entire database contained in a single file
- Generally stored on internal storage at /data/data/<packageName>/databases
- Browser subdirectories contain valuable data

```
# cd com.android.browser
# ls
app_appcache     app_icons        databases
app_databases    app_thumbnails   lib
app_geolocation  cache            shared_prefs
```

SQLite – commands

• sqlite3 <database name>	Runs SQLite
• .tables	Lists available tables
• .headers ON	Displays header row
• select * from <table name>;	Displays table contents
• CTRL+Z	Exits SQLite

SQLite – example

```
# sqlite3 data
SQLite version 3.7.2
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
account           category_tag      project           tran
android_metadata  currency          reminder          user_settings
budget            currency_symbol   repeat
category          license           skin
category_color    passcode          system_settings
sqlite> select * from account;
sqlite> select * from user_settings;
1|5|8:00|$|USD|USD|0|7|3:45|Yes|1|0|30|30|No|Yes
sqlite> select * from passcode;
1||No
sqlite> select * from system_settings;
1|
```

- These directories all contain one of more databases of interesting data for analysis.
- Contents include (app_geolocation) GPS positions for tracking where the device has traveled, (databases, app_databases and app_cache) stored data from visited web sites/apps.

Network

- Network storage via Java and Android network classes
- Network data is not stored locally on the device, though configuration files and related databases generally are locally stored

Where else?

- Linux Kernel & Android Stack
  - Android is Linux at the kernel…we know that.
  - With Linux, there is a kernel log, which may have some interesting data.
  - To access the kernel log, command `dmesg` or "display message", prints the kernel messages to the console (avd or adb shell)

dmesg

```
<6>[272367.204773] [KEY] gpio_keys_scan_keys: key 1-102, 4 (192) changed to 1
<6>[272367.358947] [KEY] gpio_keys_scan_keys: key 1-102, 4 (192) changed to 0
<6>[272368.527496] [VIB] Binder Thread #(parent:zygote): vibrates 0 msec
<6>[272372.756347] [VIB] Binder Thread #(parent:zygote): vibrates 0 msec
<6>[272379.568206] [LS][CM3602] ALS value: 0x3B, level: 5 #
<6>[272381.128173] [KEY] gpio_keys_scan_keys: key 1-102, 4 (192) changed to 1
<6>[272381.128418] [KEY] gpio_keys_scan_keys: key 1-139, 5 (193) changed to 1
<6>[272381.267791] [KEY] gpio_keys_scan_keys: key 1-139, 5 (193) changed to 0
<6>[272381.293518] [KEY] gpio_keys_scan_keys: key 1-102, 4 (192) changed to 0
<6>[272381.474823] [LS][CM3602] ALS value: 0x8, level: 3 #
<6>[272382.278381] [KEY] gpio_keys_scan_keys: key 1-102, 4 (192) changed to 1
<6>[272382.452392] [KEY] gpio_keys_scan_keys: key 1-102, 4 (192) changed to 0
<3>[272386.438018] binder: 26112: binder_alloc_buf, no vma
<6>[272386.438232] binder: 1324:1343 transaction failed 29201, size 168-0
```

- Notice [KEY] above. Possibly something logging keystrokes. May be worth further investigation
- Root access is not needed for `dmesg`, just USB debugging

…more dmesg commands

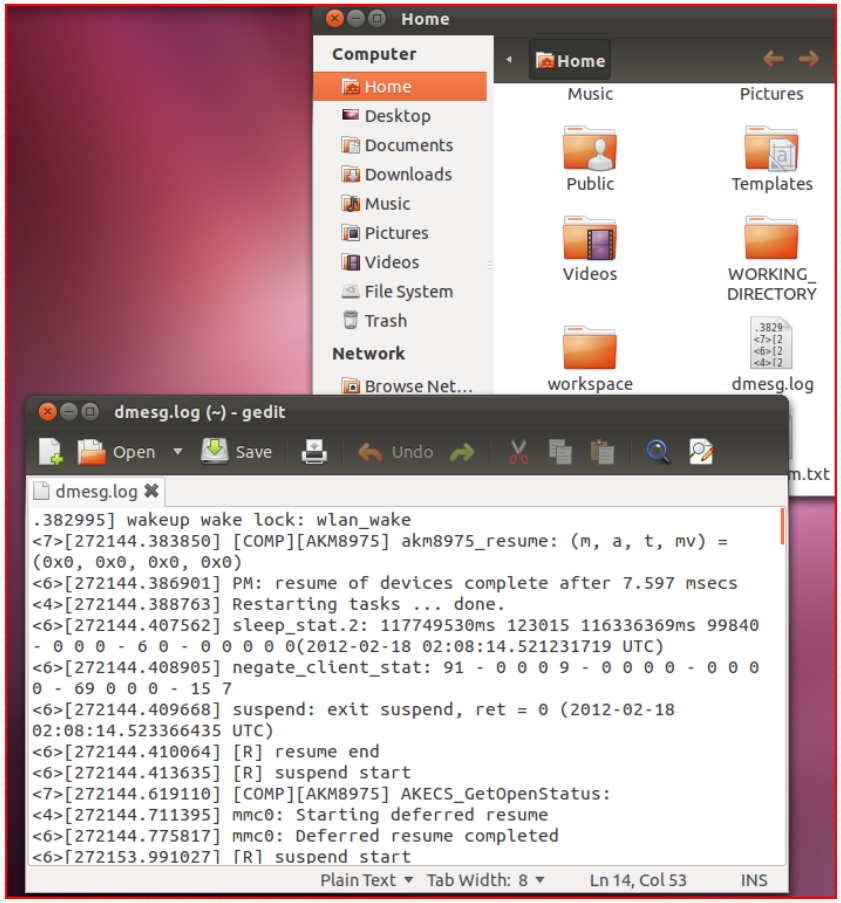
- `dmesg | wc` displays word count of log
  `-l` for line count

```
# dmesg | wc -l
2065
```

- `dmesg > dmesg.log` saves dmesg to a log file

```
svalle@svalle-VirtualBox:~$ adb shell dmesg > dmesg.log
```

dmesg.log


Home
Computer
Home
Desktop
Documents
Downloads
Music
Pictures
Videos
File System
Trash
Network
Browse Net...
Home
Music
Pictures
Public
Templates
Videos
WORKING_
DIRECTORY
workspace
dmesg.log
dmesg.log (~) - gedit
Open
Save
Undo
dmesg.log
.382995] wakeup wake lock: wlan_wake
<7>[272144.383850] [COMP][AKM8975] akm8975_resume: (m, a, t, mv) =
(0x0, 0x0, 0x0, 0x0)
<6>[272144.386901] PM: resume of devices complete after 7.597 msecs
<4>[272144.388763] Restarting tasks ... done.
<6>[272144.407562] sleep_stat.2: 117749530ms 123015 116336369ms 99840
- 0 0 0 - 6 0 - 0 0 0 0 0(2012-02-18 02:08:14.521231719 UTC)
<6>[272144.408905] negate_client_stat: 91 - 0 0 0 9 - 0 0 0 0 - 0 0 0
0 - 69 0 0 0 - 15 7
<6>[272144.409668] suspend: exit suspend, ret = 0 (2012-02-18
02:08:14.523366435 UTC)
<6>[272144.410064] [R] resume end
<6>[272144.413635] [R] suspend start
<7>[272144.619110] [COMP][AKM8975] AKECS_GetOpenStatus:
<4>[272144.711395] mmc0: Starting deferred resume
<6>[272144.775817] mmc0: Deferred resume completed
<6>[272153.991027] [R] suspend start
Plain Text
Tab Width: 8
Ln 14, Col 53
INS

logcat

- Displays a live stream of messages, system and app debug message
- Used in the CarrierIQ demonstration video on YouTube

```
I/D       (26531): Response code is 302: Found
I/D       (26531): Redirected
I/D       (26531): Beginning download of http://hw.libsyn.com/p/9/8/3/9830ff1fbc18af87/androidcentra
eid=&l_mid=2906610&expiration=1329540298&hwt=e0b8530a37a4c3cc138c029a56d3dd1f after 1 redirects and
I/D       (26531): 0 seconds have elapsed since the start of this attempt
I/D       (26531): Full download, no previous parts
W/System.err(26531): ** REQUEST **
W/System.err(26531): GET /p/9/8/3/9830ff1fbc18af87/androidcentral88.mp3?sid=ba084a88b95b00b7ab80cca8
0298&hwt=e0b8530a37a4c3cc138c029a56d3dd1f  HTTP/1.0
W/System.err(26531): Accept-Encoding:
W/System.err(26531): Accept-Language: en-us,en;q=0.5
W/System.err(26531): User-Agent: PodTrapper
W/System.err(26531): Connection: close
W/System.err(26531): Accept: */*
W/System.err(26531): Host: hw.libsyn.com
W/System.err(26531): Accept-Charset: ISO-8859-1,utf-7;q=0.7,*;q=0.7
W/System.err(26531): ** RESPONSE **
W/System.err(26531): 200: OK
W/System.err(26531): content-type: audio/mpeg
W/System.err(26531): cache-control: max-age=86400
W/System.err(26531): connection: close
W/System.err(26531): last-modified: Fri, 17 Feb 2012 07:16:11 GMT
W/System.err(26531): etag: "1329462971"
W/System.err(26531): content-length: 33965460
W/System.err(26531): accept-ranges: bytes
W/System.err(26531): date: Sat, 18 Feb 2012 03:09:56 GMT
W/System.err(26531): x-hw: 1329534596.cy016n2
I/D       (26531): Response code is 200: OK
I/D       (26531): Response content type: audio/mpeg
I/D       (26531): Server did not send range header, starting from 0 with size 33965460
D/dalvikvm(26279): GC_EXPLICIT freed 328K, 51% free 4618K/9415K, external 4133K/4552K, paused 83ms
```

logcat

- Message Indicators

Message Indicator	Description
V	Verbose
D	Debug
I	Information
W	Warning
E	Error
F	Fatal
S	Silent

Forensically Thinking

- Now that we have some idea of how to locate data
- Time to start thinking about identifying potential interesting data, forensically thinking
- What you might look for:
  - **Time stamps** – when was something modified, when did an event occur
  - **User Information** – locate user names and/or passwords in insecure prefs/logs. Locate user authentication times in log files.
  - **Image files** – identify .JPEG or other picture files, for later assessment of the picture.
  - **SD Card Files** – look for files saved to SD Card
  - **Call logs** – Who has the user been calling / receiving calls from

REVIEW

- Explored Android file system, internal and external
- Located common directories for rich forensic information
- Identified five key areas of stored persistent data
- Explored application preference files to locate important forensic data
- Explored databases in search of data for forensics analysis
- Identified sensitive data stored insecurely

EXERCISE

- Apply current Android forensics knowledge to locate data of interest
  - Using `adb shell` (or `/.android` if using an AVD), explore an applications `shared_prefs` within `/data/data`
  - Use the `cat` command to open an xml file and review the contents
  - Note anything of interest to share with the class
  - Using `sqlite3`, explore an applications databases within `/data/data`
  - Use `.tables` and `select` commands to gather data of interest, which could identify something specific about the user.
  - Note anything of interest to share with the class

Learning Objectives

By the end of this course, you will be able to:

- ✓ Extract and analyze data from an Android device
- ✓ Manipulate Android file systems and directory structures

3. Understand techniques to bypass passcodes *NEW!*
4. Utilize logical and physical data extraction techniques
5. Reverse engineer Android applications
6. Analyze acquired data

Device Handling & Modification

Source: thebransonhistory.blogspot.com

Device Handling & Modification

- **Forensics rule**: Avoid modification of the target, at all costs
- Not so easy for mobile. Drives, RAM, CPU, etc are all in non-accessible locations
- Just the act of taking the device out of sleep mode records a log (remember logcat)
- **The realization**: You cannot get a pristine mobile device, but take much precaution to minimize modification to the device

Device Acquisition

- Extend screen timeout to max, immediately (if not already locked)
- Enable Stay Awake while charging and USB debugging
- Disable network communication
- Do nothing further until in a secure location with minimal cellular / network connectivity

"What if it's already off?"

- Boot into recovery mode
- Test for connectivity and root access
- Cross your fingers that USB debugging is already enabled and/or device is already rooted

Circumventing Passcodes

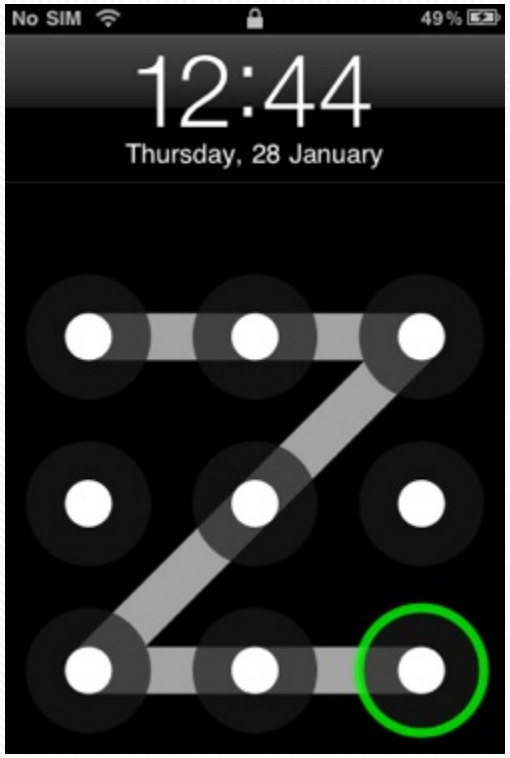
http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Circumventing Passcodes

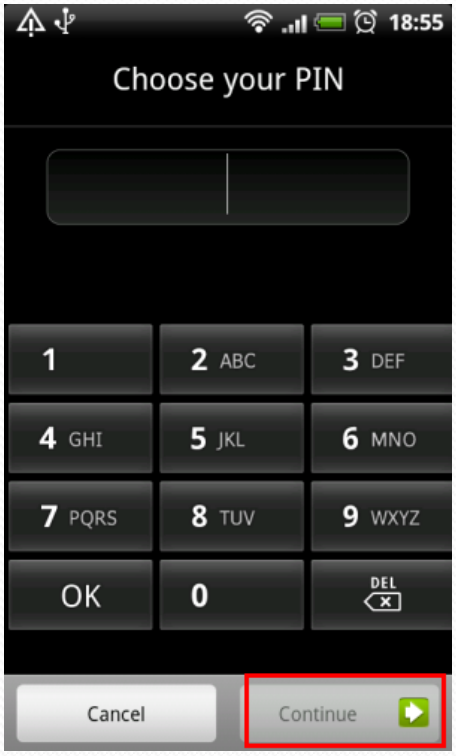
- Critical capability in forensics and security testing
- Techniques vary from platform-to-platform
- There is no panacea for circumventing passcodes on Android
  - …but we will learn a few potential techniques

Passcodes Types

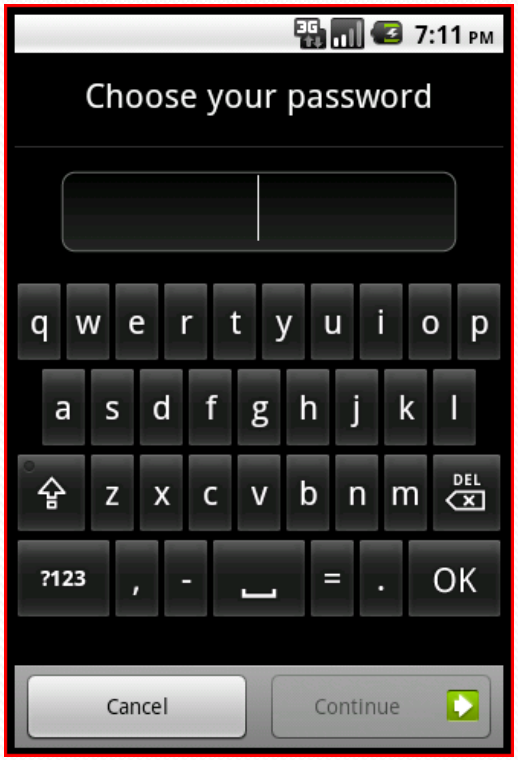
Pattern lock



PIN

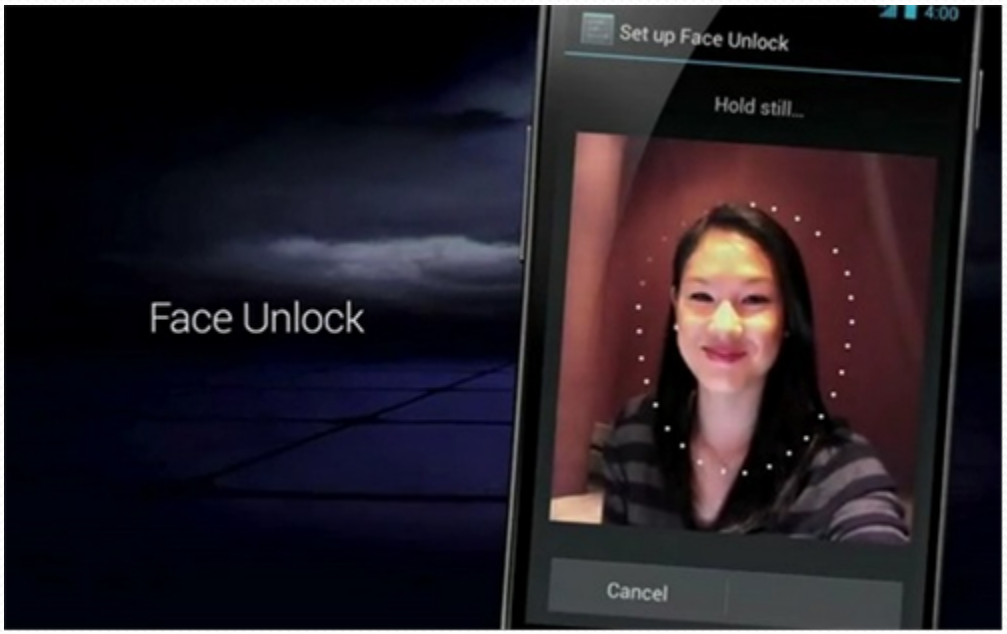


Alphanumeric



New Passcode Type

Facial recognition



"How Do We Crack Them?"

- Smudge Attack
- Pattern Lock Vulnerability
- ADB and USB Debugging, with psneuter
- Continues to evolve…

Smudge Attack

- Screens are reflective; smudge (aka pattern lock) is diffuse.
- Directional lighting and a camera capturing photos overexposed by two to three f-stops (4 to 8 times "correct" exposure)
- Creates an image displaying pattern lock
- Not 100% accurate, since other swipes of the screen may have damaged the pattern lock smudge

Smudge Attack

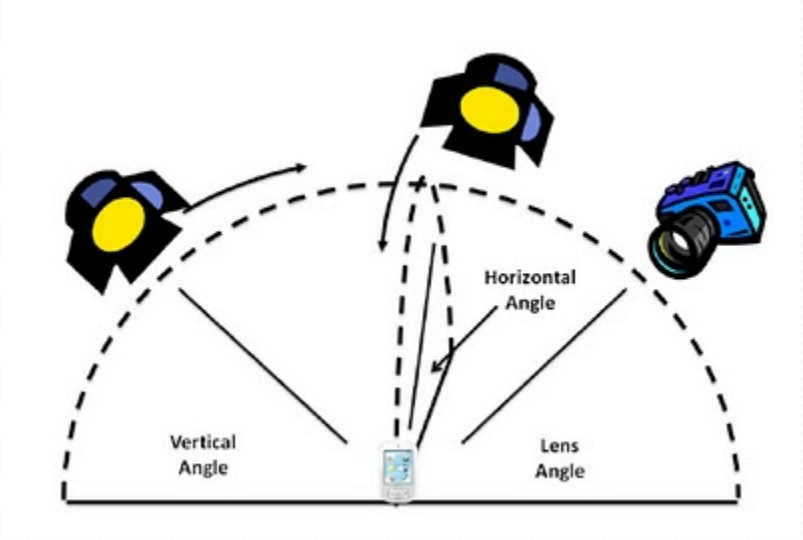


Figure A1: A phone from Experiment 2: The pattern contrasts greatly with the background noise; a grid of dots. The contrast on this image has been adjusted.

http://bcove.me/70zhp9u4

Pattern Lock Crack

- Pattern Lock creates a file `gesture.key`

```
# cd /data/system
# less gesture.key
```

- Hash of the pattern stored

```
◆◆◆TA◆["◆.◆{◆◆:E>V
```

- If custom recovery ROM is installed (i.e. ClockWork Recovery)
- Remove & recreate key to bypass pattern

```
# rm gesture.key
# touch gesture.key
```

Source: http://www.youtube.com/user/SecurityCompass

Gaining Root

http://androinica.com

Gaining Root

- Needed for many forensic techniques, including physical acquisition
- Not enabled on any device by default
- Not possible on all devices
- Gaining root isn't always the best choice in forensics
  - It will change data on the device, possibly altering evidence
  - It will be time consuming to gain root, as it's implemented differently across most devices
  - Root makes the device vulnerable to many exploits

Three Common Types of Root

- **Temp root** – roots the device only until it is rebooted, which then disables root
- **Perm root** – root persists after reboots. Commonly enabled with custom ROMs
- **Recovery mode root** – flashing (installing) a custom recovery partition, allowing root to run only in recovery mode

Temp Root

- For forensics, temp root is what we want to enable, if needed
- Suggest testing these procedures many times, ***not***, on your primary / target device

Temp Root

- Is USB debugging enabled?
- Is it already rooted?
  - `adb shell su`
    - permission denied – no root
    - `#` - root

MyTouch 4G – custom ROM

```
svalle@svalle-VirtualBox:~$ adb shell su
#
```

Droid X – stock OS

```
svalle@svalle-VirtualBox:~$ adb shell su
Permission denied
svalle@svalle-VirtualBox:~$
```

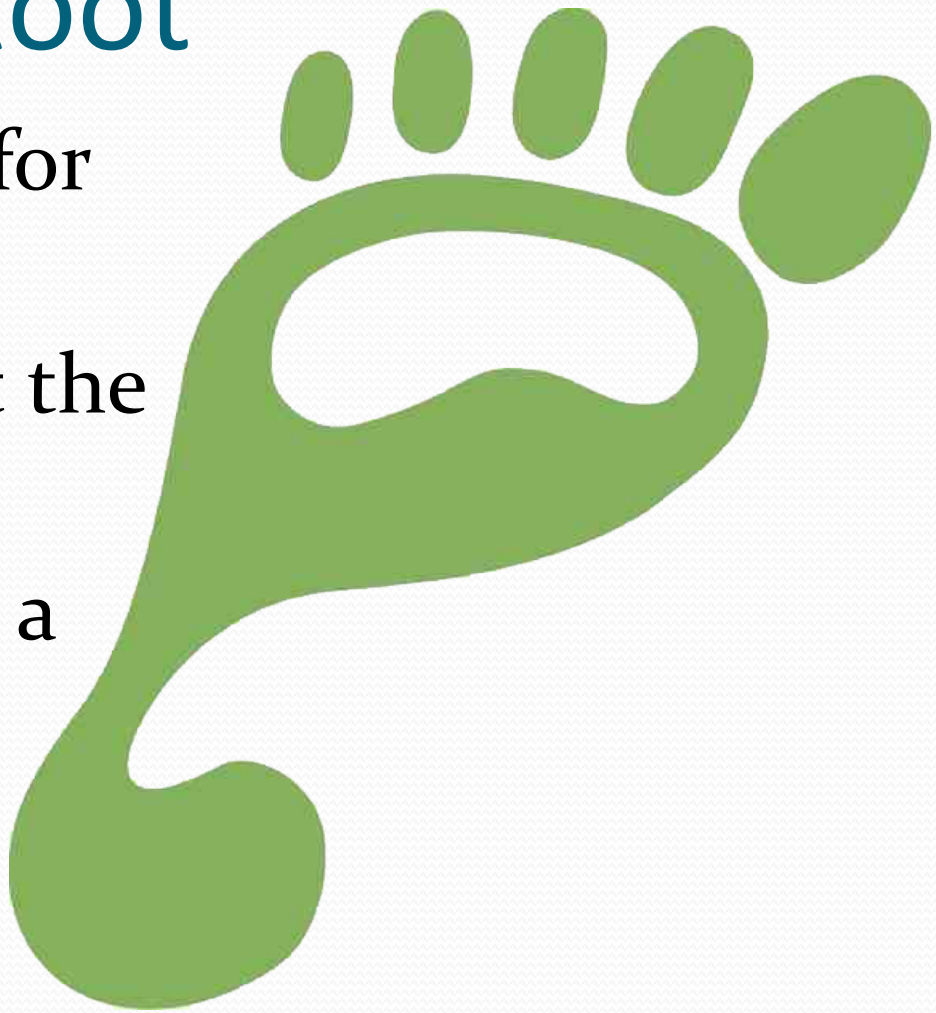
- If not rooted, start searching xda-developers.com

Property Service Neuter

- Psneuter is a form of a malicious app, but for our good
- Uses a vulnerability in Android to gain superuser access, and ultimately root
- To gain root shell (or temp root) with psneuter:
  - `adb devices`
  - `adb push psneuter /data/local/tmp`
  - `adb shell`
  - `$ cd /data/local/tmp`
  - `$ chmod 777 psneuter`
  - `$ ./psneuter`

Permanent Root

- Not as common for forensics
- We want to limit the footprint
- Perm root leaves a HUGE footprint



Busy Box

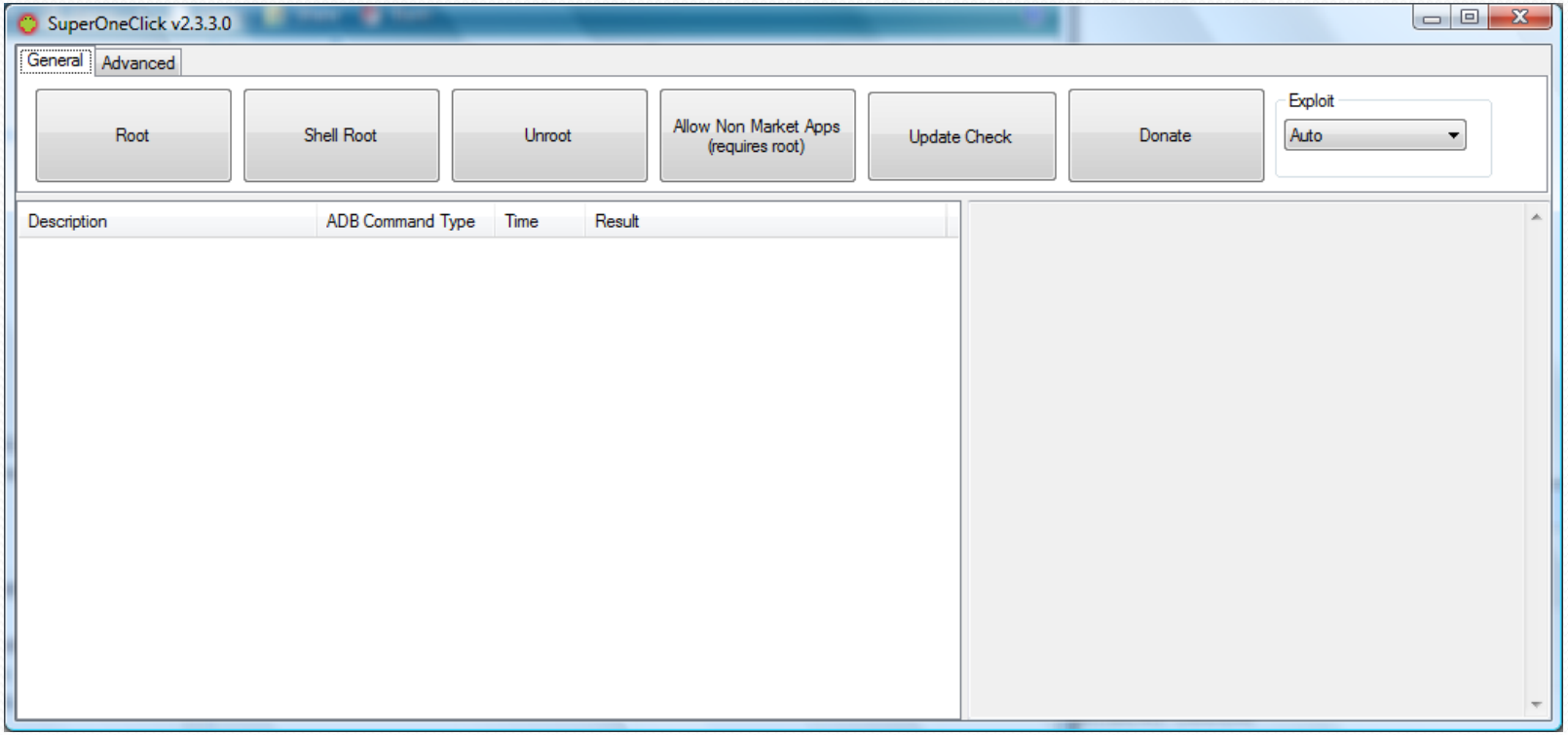
- "The Swiss Army Knife of Embedded Linux"

```
$ busybox
BusyBox v1.18.0 (2010-12-01 19:10:28 CET) multi-call binary.
Copyright (C) 1998-2009 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
   or: busybox --list[-full]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.
```

- `# mount -o remount,rw -t rfs /dev/block/st19 /system`
- `# exit`
- `adb push busybox /system/bin`
- `adb push su /system/bin`
- `adb install Superuser.apk`
- `adb shell`
- `# chmod 4755 /system/bin/busybox`
- `# chmod 4755 /system/bin/su`
- `# mount -o remount,ro -t rfs /dev/block/st19 /system`
- `# exit`
- `adb reboot`

SuperOneClick

- A simple tool for "rooting" your Android phone



SuperOneClick

- Root for perm, Shell Root for temp

Description	ADB Command Type	Time	Result
Killing ADB Server...	KillServer	1.60s	
Starting ADB Server...	StartServer	4.27s	* daemon not running. starting it now on port 5...
Waiting for device...	WaitForDevice	0.05s	
Getting manufacturer...	GetProperty	0.13s	motorola
Getting model...	GetProperty	0.13s	DROIDX
Getting version...	GetProperty	0.13s	2.3.15
Checking if rooted...	CheckIfRooted	0.14s	True

```
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
$ export PS1=""

getprop ro.build.version.release > /data/local/tmp/output 2>&1
export TEMPRANDOM=15342
export PS1=END:$TEMPRANDOM;cat /data/local/tmp/output
2.2
END:15342export PS1=""

getprop ro.product.manufacturer > /data/local/tmp/output 2>&1
export TEMPRANDOM=90536
export PS1=END:$TEMPRANDOM;cat /data/local/tmp/output
motorola
END:90536export PS1=""

getprop ro.product.model > /data/local/tmp/output 2>&1
export TEMPRANDOM=99058
export PS1=END:$TEMPRANDOM;cat /data/local/tmp/output
DROIDX
END:99058export PS1=""

getprop ro.build.version.incremental > /data/local/tmp/output 2>&1
export TEMPRANDOM=25654
export PS1=END:$TEMPRANDOM;cat /data/local/tmp/output
2.3.15
END:25654export PS1=""

ls -l /system/xbin/su > /data/local/tmp/output 2>&1
export TEMPRANDOM=58396
export PS1=END:$TEMPRANDOM;cat /data/local/tmp/output
-rwsr-xr-x root     shell       26248 2010-09-22 11:25 su
END:58396
```

A couple roots

- Acer A500
  http://www.tabletroms.com/forums/showwiki.php?title=AcerIconiaFaq:How-to-root-the-Acer-Iconia-Tab-A500
- Lenovo
  http://rootzwiki.com/topic/8722-lenovo-ideapad-k1-rooting-guide-messy/page__st__120

Agenda

DAY 1

- ✓ Forensic Introduction
- ✓ Course Setup – Linux, OS X, and Windows
- ✓ Android Overview
- ✓ SDK and AVD
- ✓ Android Security Model
- ✓ ADB and shell Introduction
- ✓ File System and Data Structures
- ➢ Device Handling
- ➢ Circumvent passcode
- ➢ Gain Root Access

Agenda

DAY 2

- Recovery Mode
- Boot Loaders
- Logical Forensic Techniques
- Open Source Tools
- Commercial Tools
- Physical Forensic Techniques & Tools
- Forensic Analysis
- Application Penetration Testing Setup
- Reverse Apps
- …more Reversing
- Document Findings

Got sqlite3?

- $ adb push sqlite3 /sdcard/

- $ adb shell
- $ su
- # mount -o remount,rw -t yaffs2 /dev/block/mtdblock3 /system
- # dd if=/sdcard/sqlite3 of=/system/bin/sqlite3
- # chmod 4755 /system/bin/sqlite3
- # mount -o remount,ro -t yaffs2 /dev/block/mtdblock3 /system
- sqlite3 binary is in SuperOneClick directory.

Recovery Mode

http://www.mymac.com

Recovery Mode

- Designed as an avenue for manufacturers to deliver and apply system updates
- Recovery partitions offer shell access and root permissions
- When booting into recovery mode, pass codes are circumvented

Recovery Not User Accessible

Recovery User Accessible

- Check adb devices on forensic workstation

```
svalle@svalle-VirtualBox:~/.android$ adb devices
List of devices attached
SH132RM00905    recovery
```

- If no adb access, search for root while in recovery mode

```
svalle@svalle-VirtualBox:~$ adb shell
~ #
```

Recovery Mode Techniques

Device	Key Combination
Motorola Droid X	Power off. Hold Home and press power button. Release power. When (!) displays release Home. Press Search button. (needs more research)
HTC Incredible	Hold volume down and press power button. Use volume down to select recovery and press power button.

Passcode Circumvention Recap

- If device is on and passcode protected, connect to USB and attempt ADB access.
- If pattern lock is present (and you have access to lighting and camera), attempt smudge attack.
- If those fail, attempt to reboot into recovery mode.
- If device is off, attempt boot into recovery mode.

REVIEW

- Identified the important of proper device handling
- Explored techniques for circumventing passcodes
- Applied rooting techniques and tools
- Located recovery partitions and benefit of recovery mode

EXERCISE

- Attempt to circumvent passcode and obtain root access
  - Document your findings to share with the class

Learning Objectives

By the end of this course, you will be able to:

- ✓ Extract and analyze data from an Android device
- ✓ Manipulate Android file systems and directory structures
- ✓ Understand techniques to bypass passcodes *NEW!*

3. Utilize logical and physical data extraction techniques
4. Reverse engineer Android applications
5. Analyze acquired data

Android Forensics Techniques

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Android Forensics Techniques

- Forensic data acquisition
- Acquiring SD card data
- Open-source and commercial forensic tools
  - qtADB
  - viaExtract
  - CelleBrite
  - Paraben

Logical vs. Physical Acquisition

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Logical vs. Physical Acquisition

- Logical vs. Physical
- Logical
  - ADB Pull
  - Other tools
- Physical
  - Hardware vs. software
  - Software technique in detail

Logical vs. Physical Acquisition

Logical

- Accesses the file system.
- Data that is readily available to a user.

Physical

- Targets the physical memory, not relying on the file systems.
- Gains much more data than logical, potentially circumvents passcodes.

Logical Acquisition

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Logical SD Card Acquisition

- User app data lives in `/data/data` directories which each sub-directory is RW protected to the app user
- SD cards are used for large storage (audio, video, maps)
- SD uses cross-platform FAT file systems
- .apk files residing on SD cards are increasingly encrypted
- Removing SD card challenges
- Unencrypted .apk's are mounted in `/mnt/asec`
  - This is an important directory to pull and analyze, if 3rd party apps are part of the investigation

ADB Pull – logical

- Command used for copying data from an emulator or device
- Primary logical acquisition tool
- adb pull on non-rooted Droid X:

```
svalle@svalle-VirtualBox:~/src/dc3dd-7.1.614$ adb pull /data adbpull
pull: building file list...
0 files pulled. 0 files skipped.
```

ADB Pull – rooted & locked

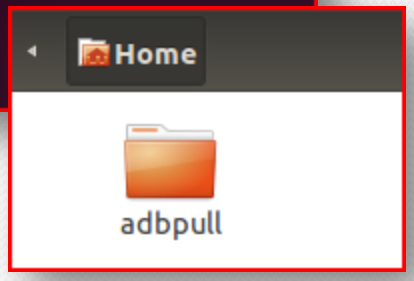
- adb pull on rooted and password locked HTC Glacier (aka T-Mobile MyTouch 4G):

```
svalle@svalle-VirtualBox:~$ adb pull /data adbpull
pull: building file list...
pull: /data/data/com.touchtype.swiftkey.phone.trial/shared_prefs/com.touchtype.swif
tkey.phone.trial_preferences.xml -> adbpull/data/com.touchtype.swiftkey.phone.trial
/shared_prefs/com.touchtype.swiftkey.phone.trial_preferences.xml
pull: /data/data/com.google.android.syncadapters.calendar/app_sslcache/android.clie
nts.google.com.443 -> adbpull/data/com.google.android.syncadapters.calendar/app_ssl
cache/android.clients.google.com.443
pull: /data/data/com.cooliris.media/databases/picasa.db -> adbpull/data/com.cooliri
s.media/databases/picasa.db
pull: /data/data/com.cooliris.media/databases/picasa.db-shm -> adbpull/data/com.coo
liris.media/databases/picasa.db-shm
pull: /data/data/com.cooliris.media/databases/picasa.db-wal -> adbpull/data/com.coo
liris.media/databases/picasa.db-wal
pull: /data/data/com.iauns.idemolished/lib/libidemandroid.so -> adbpull/data/com.ia
uns.idemolished/lib/libidemandroid.so
```

ADB Pull – rooted & locked

- ~700 MB
- ~27 minutes

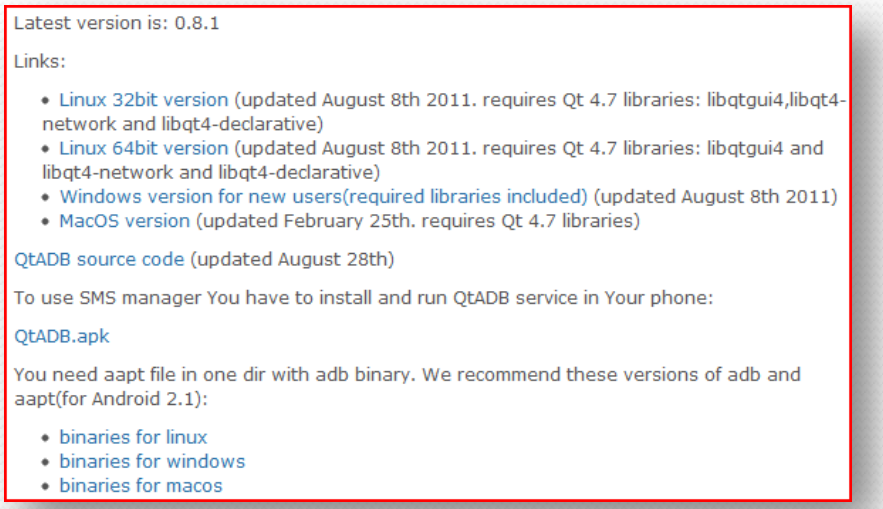

```
pull: /data/backup/processed -> adbpull/backup/processed
pull: /data/fix_permissions.log -> adbpull/fix_permissions.log
pull: /data/13_v.yuv -> adbpull/13_v.yuv
pull: /data/12_v.yuv -> adbpull/12_v.yuv
pull: /data/11_v.yuv -> adbpull/11_v.yuv
5279 files pulled. 0 files skipped.
421 KB/s (708794693 bytes in 1642.843s)
```



Tools and Time Savers

QtADB

- http://qtadb.wordpress.com/
- Graphical app based on adb
- Open-source, currently well-supported



Latest version is: 0.8.1

Links:

- Linux 32bit version (updated August 8th 2011. requires Qt 4.7 libraries: libqtgui4,libqt4-network and libqt4-declarative)
- Linux 64bit version (updated August 8th 2011. requires Qt 4.7 libraries: libqtgui4 and libqt4-network and libqt4-declarative)
- Windows version for new users(required libraries included) (updated August 8th 2011)
- MacOS version (updated February 25th. requires Qt 4.7 libraries)

QtADB source code (updated August 28th)

To use SMS manager You have to install and run QtADB service in Your phone:

QtADB.apk

You need aapt file in one dir with adb binary. We recommend these versions of adb and aapt(for Android 2.1):

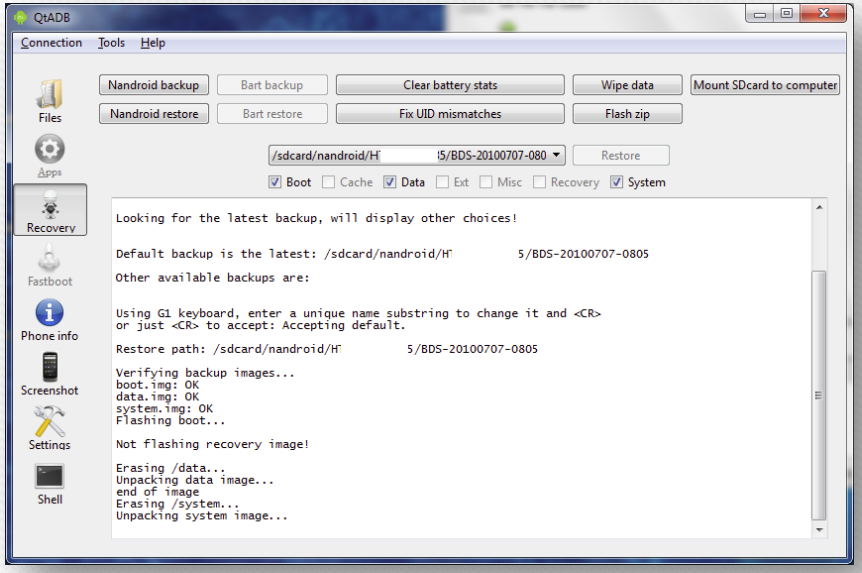
- binaries for linux
- binaries for windows
- binaries for macos

QtADB – features

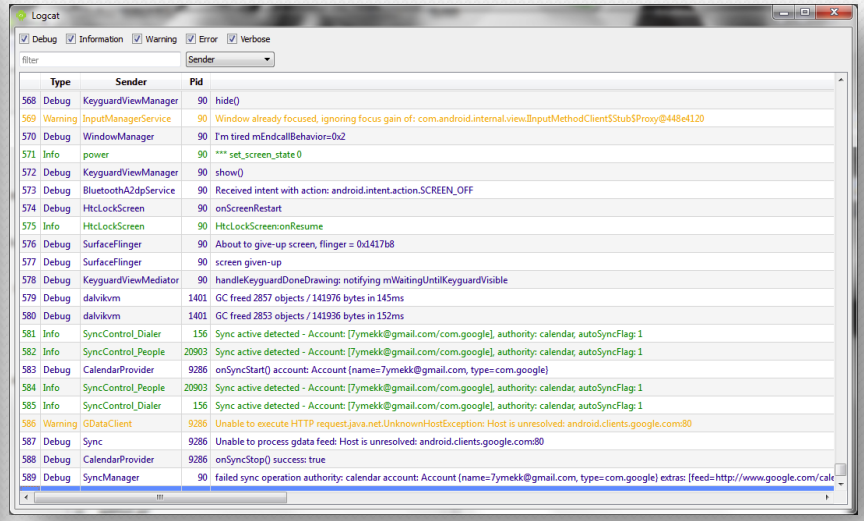
- File manager
  - copying files and dirs between phone and computer
  - removing files and dirs
  - creating new dir
  - and other
- App manager
  - installing apps
  - removing apps
  - creating backup of apps with data
  - restoring backups of apps with data
- Sms manager
  - receiving sms (baloon in tray)
  - reading sms
  - sending sms
- Shell
  - opens android shell
- Screenshot
  - take screenshot of your device
  - save screenshot to png file
- Fastboot
  - flash bootloader, radio and recovery
  - boot recovery
- Recovery
  - nandroid backup/restore
  - wipe data
  - flash rom
  - wipe battery stats
  - fix uid mismatches
- Reboot
  - to bootloader
  - to recovery
  - normal reboot
- Settings
  - set font used by app
  - set starting paths (or remember paths on exit)
  - and other
- Logcat
- Automatically detects phone (device, fastboot and recovery mode)

QtADB – in action

Recovery partition

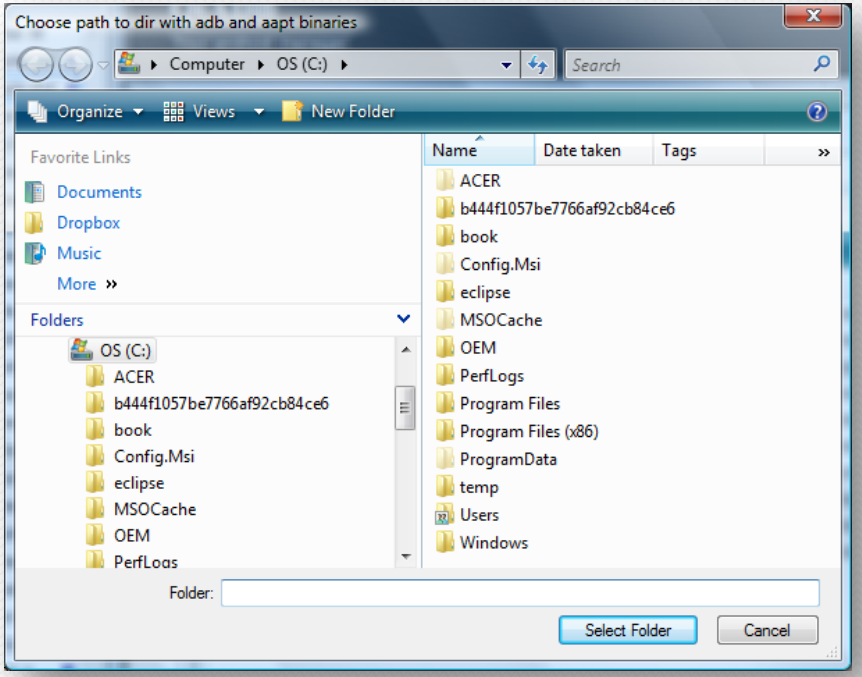


Logcat



QtADB – setup

- Windows:
  - Must have Android SDK installed
  - ZIP contain all libraries
  - Extract to a permanent directory
  - Open QtADB application
  - Choose path to directory with adb and aapt binaries (example: `C:\Users\<USERNAME>\AppData\Local\Android\android-sdk\platform-tools`)



REVIEW

- Identified the difference between logical and physical forensics
- Explored open and free tools and techniques for logically acquiring data
- Located directories and file details for SD card logical acquisition

EXERCISE

- Using either ADB or QtADB pull a logical acquisition from your device or AVD.

- Verify pull successfully completed, and document size of data acquired.

AFLogical

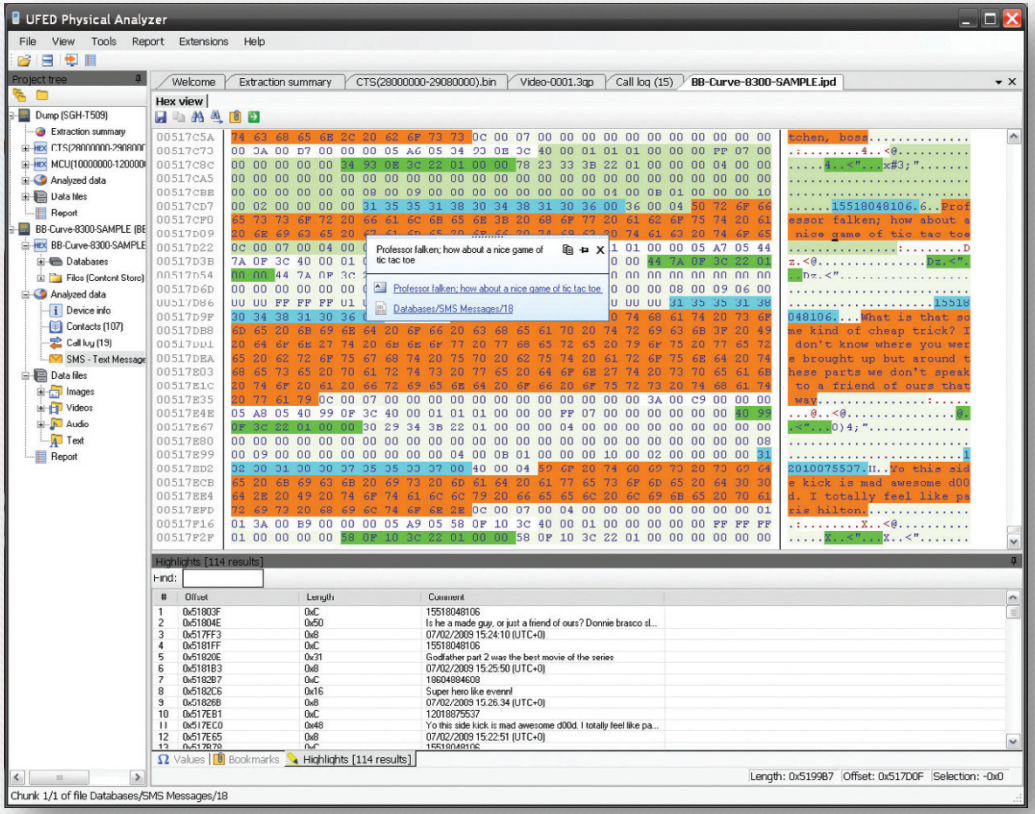
- Android forensics logical extraction tool
- Free for law enforcement and government agencies
- Leverages Content Providers
- CallLog Calls

	A	B	C	D	E	F	G	H	I
1	_id	number	date	duration	type	new	name	numberty	numberlabel
2	1	508-577-	1.32E+12	24	2	1		0	
3	2	508-577-	1.32E+12	4	2	1		0	
4	3	978-339-	1.32E+12	17	2	1		0	
5	4	508-577-	1.32E+12	23	2	1		0	
6	5	781-271-	1.32E+12	9	2	1		0	
7	6	508-577-	1.32E+12	8	2	1		0	
8	7	508-577-	1.32E+12	35	2	1		0	

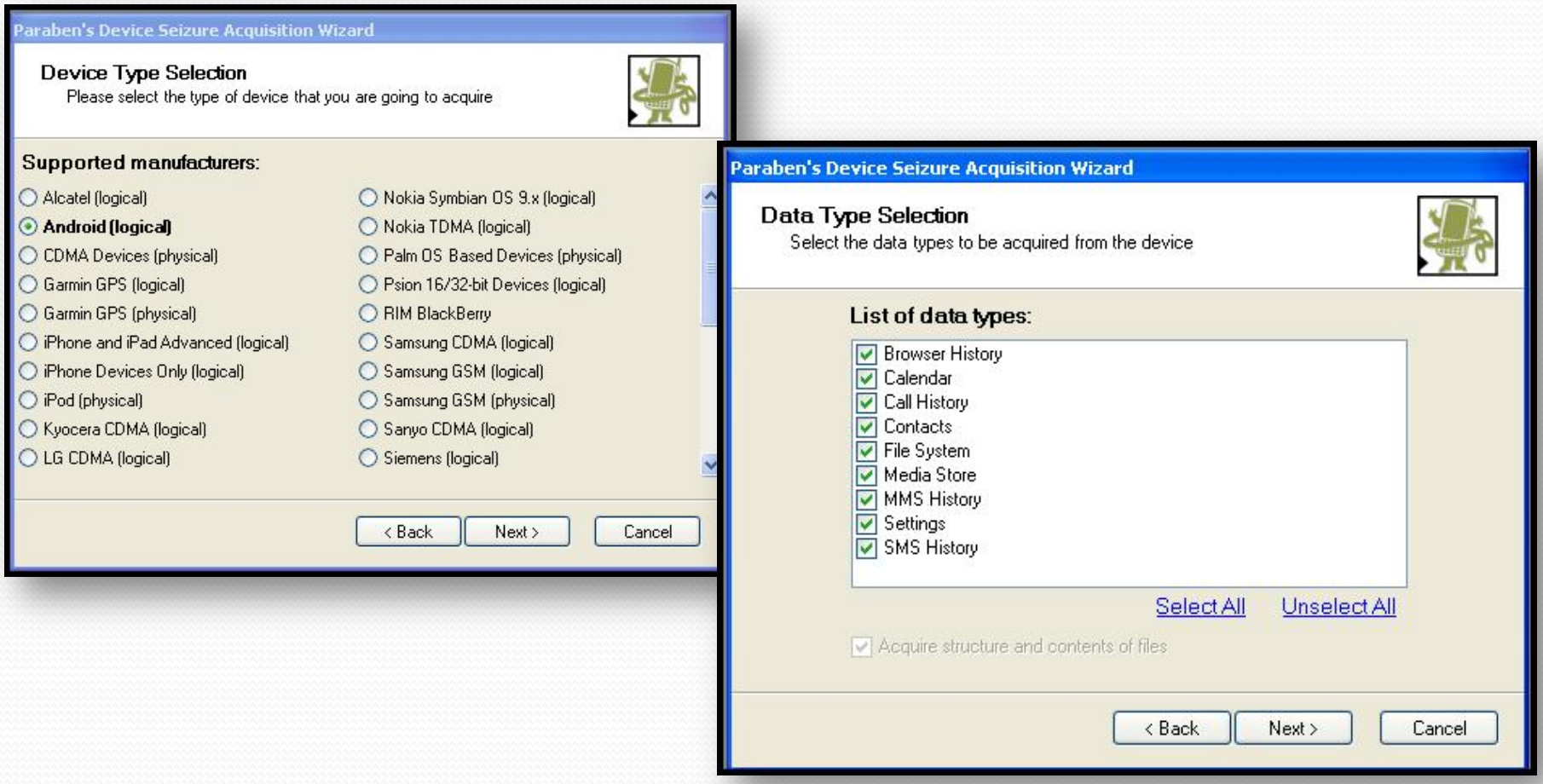
Cellebrite UFED

Vendor	Model	Phonebook Read	Call Logs Read	Calendar Read	SMS Read	ESN/IMEI Read	Pictures Read	Videos Read	Ringtones Read	Audio/Music Read	Memory Card Read	Internal SIM	Platform	Using Client	Connectivity			Date Added	Est. Time Support
															Cable	InfraRed	BlueTooth		
Motorola CDMA	Moto. MB810 Droid X (Android)	Y	Y		Y	Y	Y	Y	Y	Y	Y		Android	Y	'100'			6/1/2010	Y
Motorola CDMA	Motorola V810	Y			Y	Y									'89'			9/23/2009	
Motorola CDMA	Motorola E815	Y	Y	Y	Y	Y	Y	Y							'89'			9/23/2009	
Motorola CDMA	Motorola E815 (IL)	Y	Y		Y	Y	Y	Y	Y	Y					'89'			9/23/2009	Y
Motorola CDMA	Motorola E816	Y			Y	Y	Y	Y							'89'			9/23/2009	
Motorola CDMA	Motorola A840	Y	Y			Y	Y	Y							'89'			9/23/2009	
Motorola CDMA	Moto. W840	Y	Y	Y	Y	Y	Y		Y	Y	Y				'100'			11/19/2009	Y
Motorola CDMA	Motorola W845 (USC,CellSouth)	Y	Y		Y	Y	Y	Y	Y	Y					'100'			11/17/2009	Y
Motorola CDMA	Motorola W845 (MetroPCS)	Y	Y		Y	Y	Y		Y	Y					'100'			11/19/2009	Y
Motorola CDMA	Moto. A854 Milestone (Android)	Y	Y		Y	Y	Y	Y	Y	Y	Y		Android	Y	'100'			8/8/2010	Y
Motorola CDMA	Moto. A855 Droid (Android)	Y	Y		Y	Y	Y	Y	Y	Y	Y		Android	Y	'100'			10/6/2009	Y
Motorola CDMA	Motorola V860 Barage PTT	Y	Y	Y		Y	Y	Y	Y	Y	Y				'100'			9/23/2009	Y
Motorola CDMA	Moto. A955 Droid 2 (Android)	Y	Y		Y	Y	Y	Y	Y	Y	Y		Android	Y	'100'			8/8/2010	Y
Motorola CDMA	Moto. A956 Droid 2 Global (Android	Y	Y		Y	Y	Y	Y	Y	Y	Y		Android	Y	'100'			9/21/2010	Y
Motorola CDMA	Motorola A957 R2D2 (Android)	Y	Y		Y	Y	Y	Y	Y	Y	Y		Android	Y	'100'			10/3/2010	Y
Motorola CDMA	Motorola V950 Renegade	Y	Y			Y	Y	Y	Y	Y	Y				'100'			9/23/2009	Y
Motorola CDMA	Motorola Q	Y	Y		Y	Y	Y	Y	Y	Y	Y		WinMo	Y	'80'		Y	9/23/2009	Y
Motorola CDMA	Spirit CAR (C18)	Y													'88'			9/23/2009	

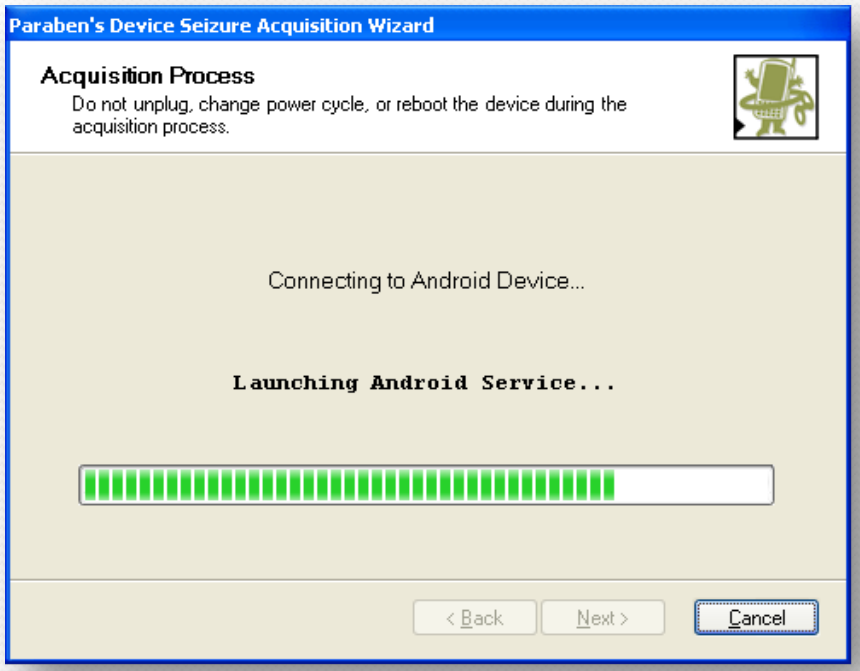
Cellebrite Physical Analyzer



Paraben Device Seizure



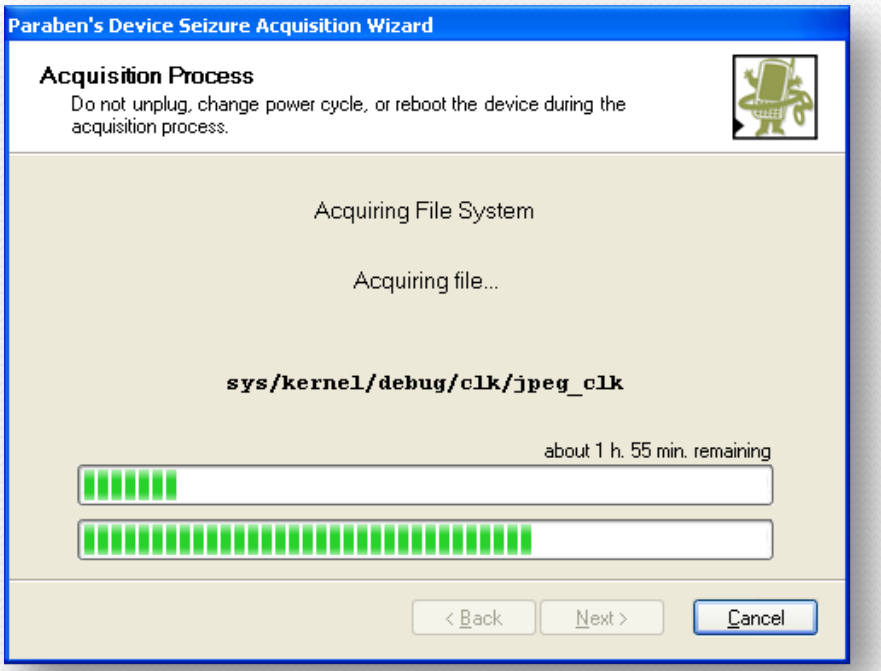
Device Seizure – Acquisition



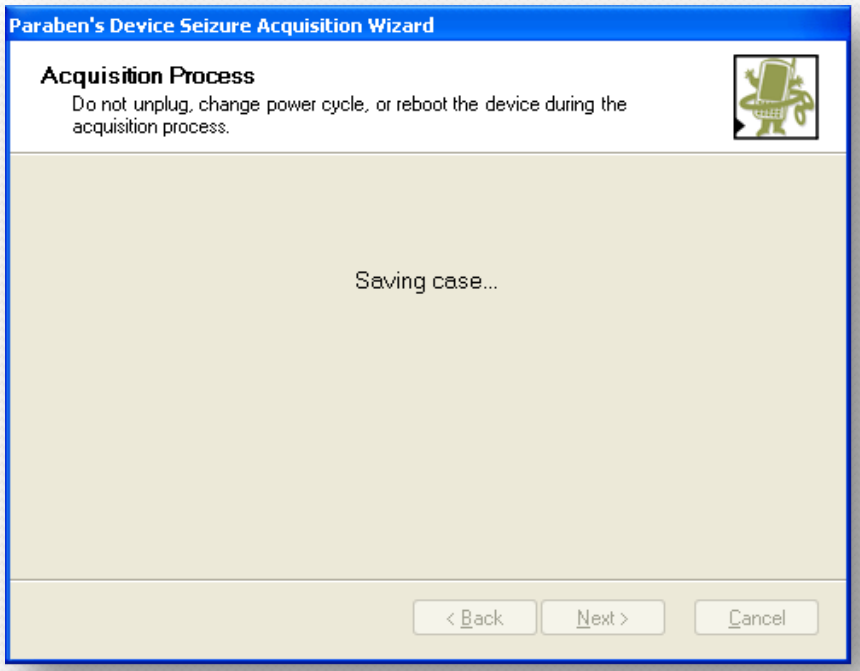
- DS acquisition temp installs Seizure Service on device. Removes automatically during completion of acquisition

Device Seizure – Acquisition

- Device Seizure hung while acquiring data after more than 11 hours
- Keep in mind, I'm acquiring from a rooted CyanogenMod ROM, and checked options to acquire all data, including the entire contents of 32GB Class 10 microSD card

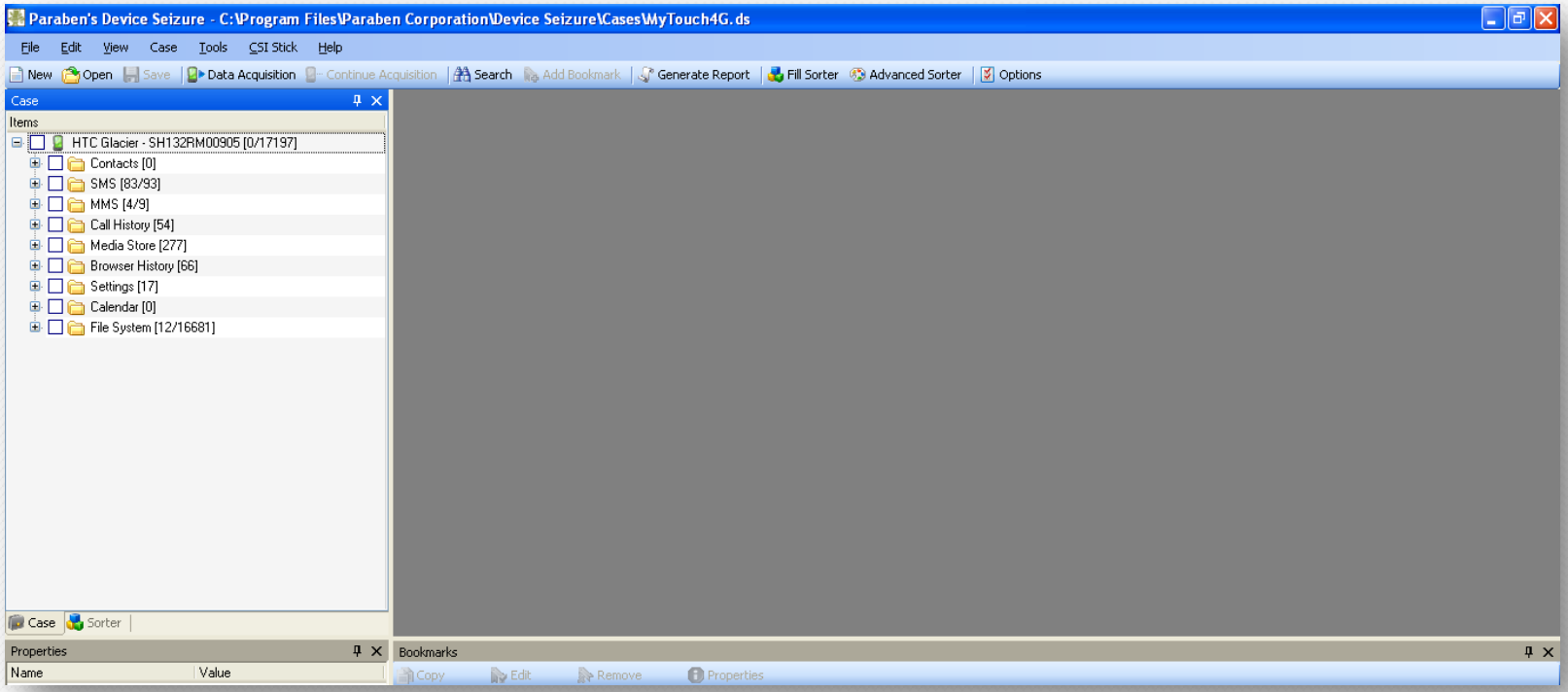


Device Seizure – Acquisition



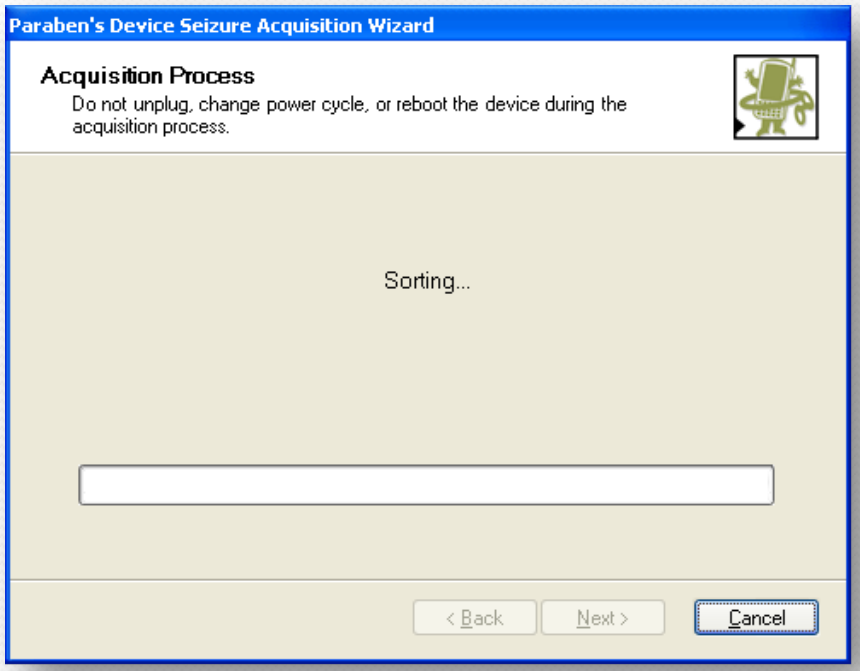
- This screen displays for considerable amount of time when completing / canceling an acquisition

Device Seizure – Results



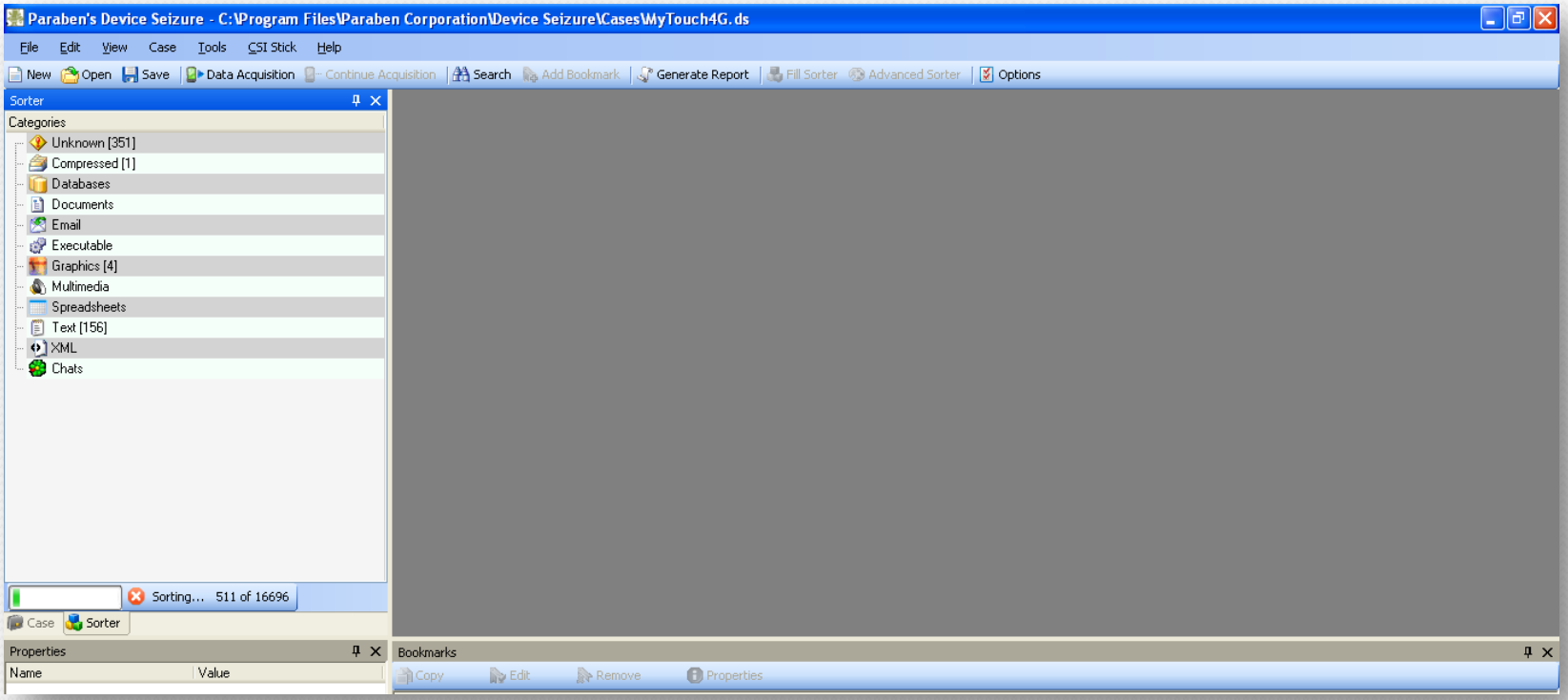
- Contacts and Calendar were empty

Device Seizure – Sorting



- After acquisition, "Do you want to fill the sorter?“
- This will take about an hour

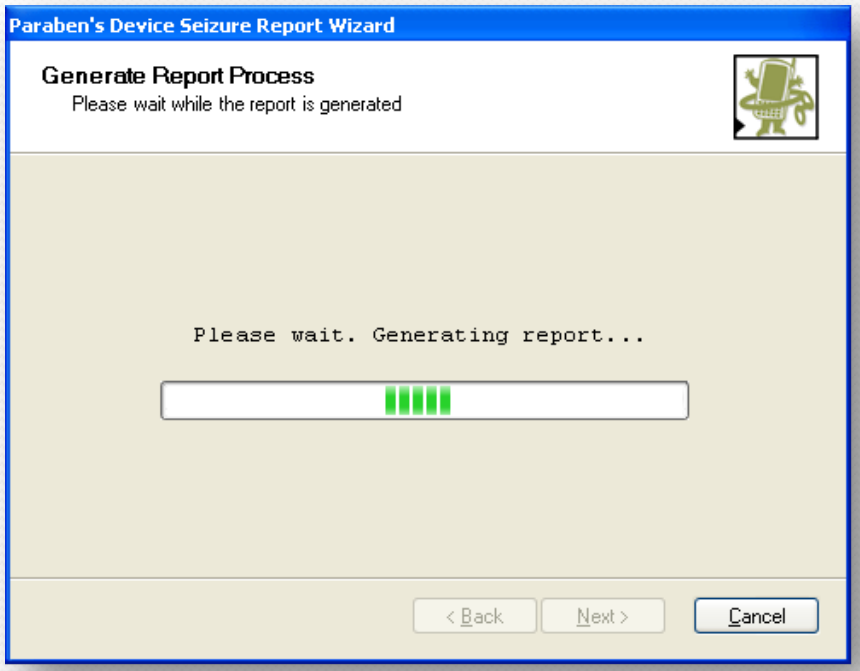
Device Seizure – Sort Results



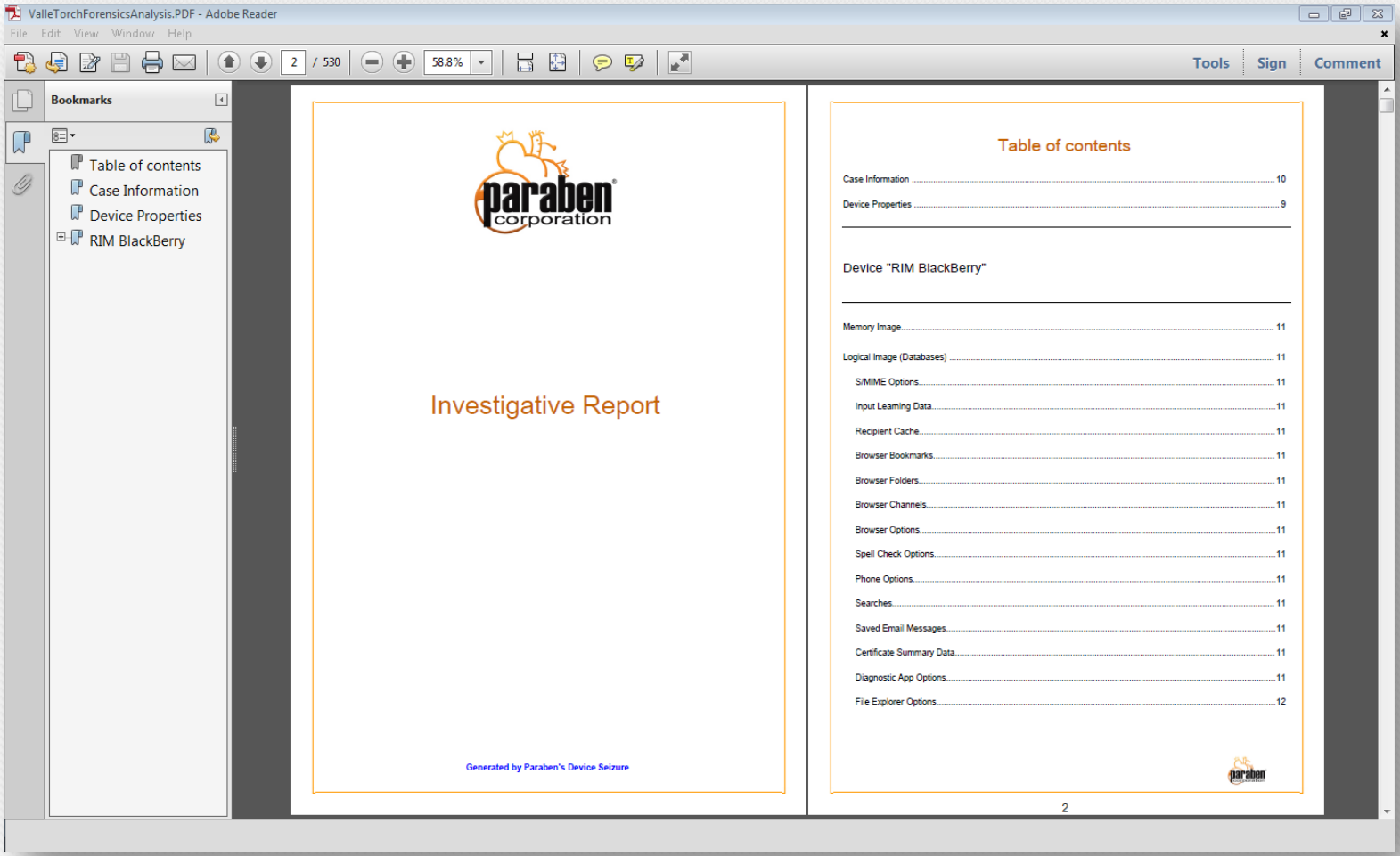
- Sorting all the findings

Device Seizure – Reports

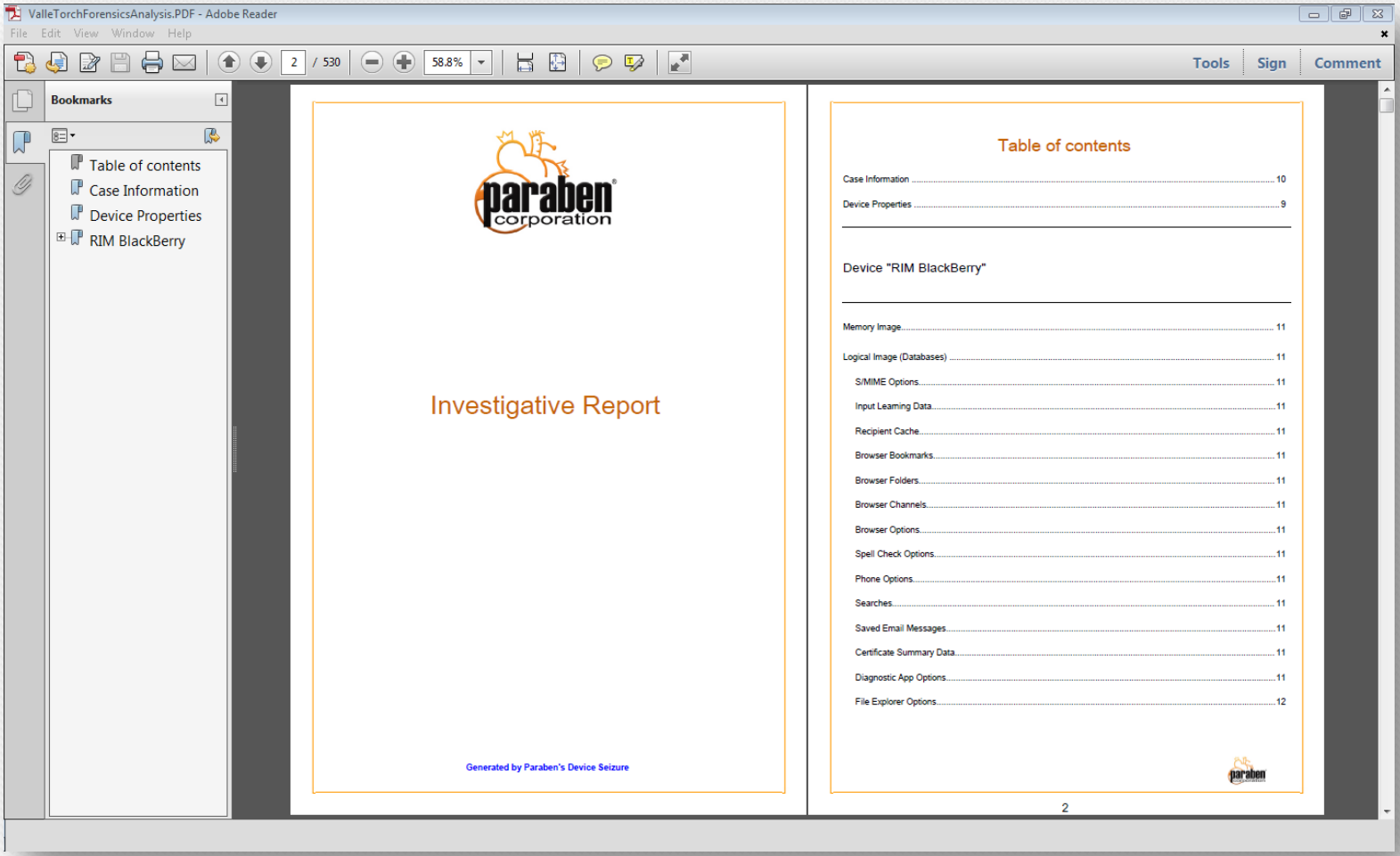
- Creating a PDF report of the entire case



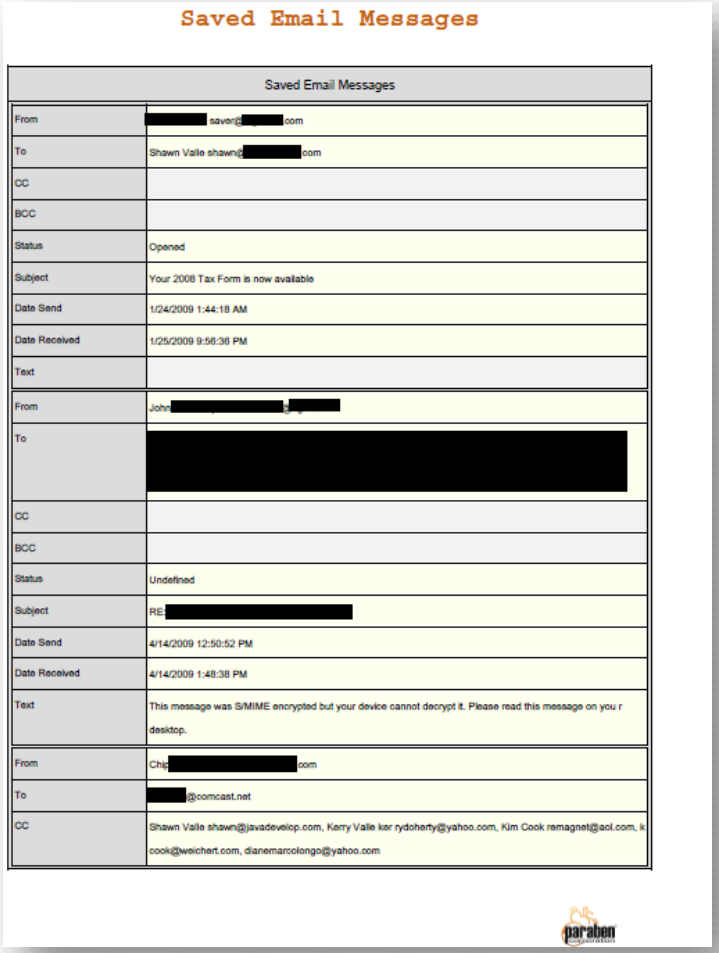
Device Seizure – Report



Device Seizure – Report



Device Seizure – Report



Saved Email Messages

Saved Email Messages	
From	[redacted] saven@[redacted].com
To	Shawn Valle shawn@[redacted].com
CC	
BCC	
Status	Opened
Subject	Your 2008 Tax Form is now available
Date Send	1/24/2009 1:44:18 AM
Date Received	1/25/2009 9:56:36 PM
Text	
From	John[redacted]
To	[redacted]
CC	
BCC	
Status	Undefined
Subject	RE:[redacted]
Date Send	4/14/2009 12:50:52 PM
Date Received	4/14/2009 1:48:38 PM
Text	This message was S/MIME encrypted but your device cannot decrypt it. Please read this message on you r desktop.
From	Chip[redacted].com
To	[redacted]@comcast.net
CC	Shawn Valle shawn@javadevelop.com, Kerry Valle ker rydoherty@yahoo.com, Kim Cook remagnet@aol.com, k cook@weichert.com, dianemarcolongo@yahoo.com

Physical Acquisition

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Software Physical Acquisition

- Let's get a full NAND acquisition of the user accessible data partition

- For time's sake, and now that we know of open-source and commercial tools, let's take advantage of them for the physical acquisition

Forensic Analysis

http://www.geeky-gadgets.com/wp-content/uploads/2010/08/android3.jpg

Forensic Analysis

- Analyzing acquired data
  - File System Analysis
  - SQLite Analysis
  - Directory Structure
  - FAT Analysis
  - SD Card Analysis
  - YAFFS2 Analysis

Forensic Analysis - photos

- Common location for storage of photos in JPG format

```
# ls /mnt/sdcard/DCIM/Camera
IMG_20111224_191659.jpg  IMG_20111224_192558.jpg  IMG_20120203_193731.jpg
IMG_20111224_192056.jpg  IMG_20111224_192602.jpg  IMG_20120203_193755.jpg
IMG_20111224_192103.jpg  IMG_20111224_192614.jpg  IMG_20120203_193808.jpg
IMG_20111224_192109.jpg  IMG_20111224_192628.jpg  IMG_20120203_193917.jpg
IMG_20111224_192115.jpg  IMG_20111224_192632.jpg  IMG_20120217_111427.jpg
IMG_20111224_192142.jpg  IMG_20111224_192724.jpg  IMG_20120217_111432.jpg
IMG_20111224_192215.jpg  IMG_20111224_192806.jpg  IMG_20120217_111600.jpg
IMG_20111224_192225.jpg  IMG_20111224_192824.jpg  IMG_20120220_163843.jpg
IMG_20111224_192230.jpg  IMG_20111224_192829.jpg  IMG_20120221_081528.jpg
IMG_20111224_192237.jpg  IMG_20111224_192856.jpg  IMG_20120221_081605.jpg
IMG_20111224_192307.jpg  IMG_20120103_073240.jpg  IMG_20120221_081632.jpg
IMG_20111224_192311.jpg  IMG_20120103_073252.jpg  IMG_20120221_081643.jpg
IMG_20111224_192400.jpg  IMG_20120103_073316.jpg  IMG_20120221_104028.jpg
IMG_20111224_192422.jpg  IMG_20120107_154502.jpg  IMG_20120221_104034.jpg
IMG_20111224_192426.jpg  IMG_20120107_195400.jpg  IMG_20120221_104843.jpg
IMG_20111224_192449.jpg  IMG_20120107_195405.jpg  IMG_20120221_105239.jpg
IMG_20111224_192511.jpg  IMG_20120107_195413.jpg  IMG_20120310_120309.jpg
IMG_20111224_192515.jpg  IMG_20120107_195419.jpg  IMG_20120310_120313.jpg
IMG_20111224_192523.jpg  IMG_20120107_214905.jpg  VID_20120217_200653.m4v
IMG_20111224_192527.jpg  IMG_20120107_214910.jpg  VID_20120311_141101.m4v
IMG_20111224_192554.jpg  IMG_20120107_214931.jpg
# adb pull /mnt/sdcard/DCIM/Camera/*.jpg files-2012-03-11
```

```
svalle@svalle-VirtualBox:~$ adb pull /mnt/sdcard/DCIM/Camera*.jpg-2012-03-11
```

Important Directories Recap

- `/cache/`
  - Previewed Gmail attachments
  - Downloads (Market and messages)
- `/data/`
  - dalvik-cache: applications (.dex) that have been run
  - app: .apk files
  - data: subdirectories per app with SQLite databases and XML shared preferences
  - misc: protocol info
  - system:
    - installed applications (or packages.xml)
    - accounts database
    - device and app login details, .key files
- `/proc` & `/sys` – list of device filesystems, web history, device info
- `/mnt/sdcard/DCIM/Camera` - images
- `/sdcard/android` or `sdcard/data/data` – FAT32, limited permission

REVIEW

- Explored several commercial Android forensics products
- Identified the benefits and acquisition steps of physical forensics
- Located the most important directories for analysis

EXERCISE

- Determine what the user does for work and fun
  - (in groups) Now that you have acquired data many different ways, analyze the data using one of the forensics tools (adb, adb shell, Device Seizure, QtADB, etc) to get a fresh data acquisition from your device
  - Look at earlier exercises for commands, as a refresher
  - Explore data in directories like `/data/` and `/cache/`
  - As a forensic analyst, document findings that would help you determine the users profession and hobbies
  - Be prepared to share your findings with the class

Learning Objectives

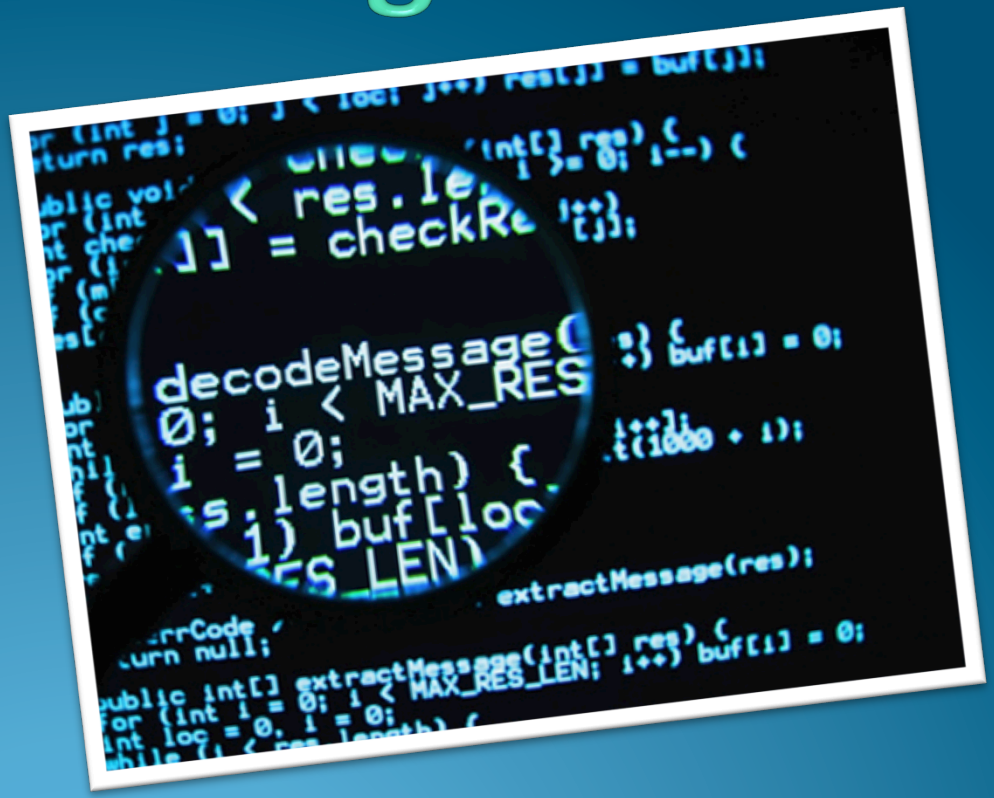
By the end of this course, you will be able to:

- ✓ Extract and analyze data from an Android device
- ✓ Manipulate Android file systems and directory structures
- ✓ Understand techniques to bypass passcodes
- ✓ Utilize logical and physical data extraction techniques

4. Reverse engineer Android applications
5. Analyze acquired data

Application Testing

Reverse Engineering Apps



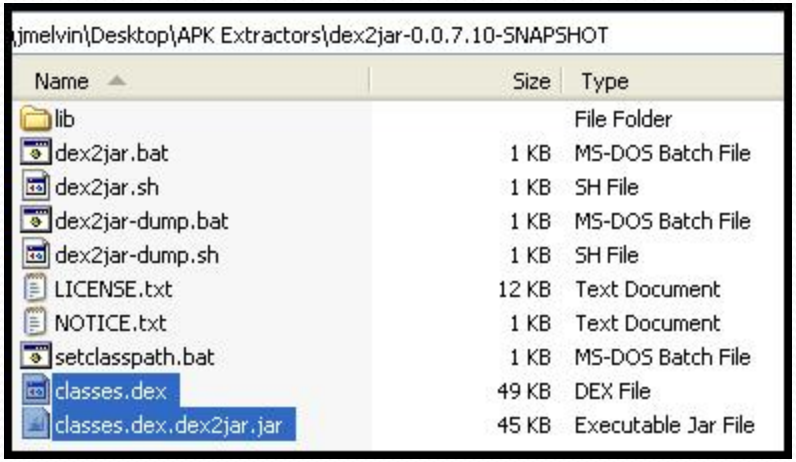
http://www.areamobile.it/wp-content/uploads/2011/12/defend-reverse-engineering.jpg

Analyzing APKs

- Byte code is reverted to source
- First extracting each of the classes.dex files
- Using dex2jar.bat, a jar file is created

- Batch file used to convert dex files to jar files

```
C:\Documents and Settings\jmelvin\Desktop\APK Extractors\dex2jar-0.0.7.10-SNAPSH
OT>dex2jar classes.dex
0 [main] INFO com.googlecode.dex2jar.v3.Main - version:0.0.7.10-SNAPSHOT
15 [main] INFO com.googlecode.dex2jar.v3.Main - dex2jar classes.dex -> classes.d
ex.dex2jar.jar
578 [main] INFO com.googlecode.dex2jar.v3.Main - Done.
```



More Analyzing APKs

- Java Decompiler used to create a zip file containing all of the Java source code
  - Used to view class files and convert them to java
- The remaining content of each of the APK files is extracted
- Yes, it's a painful process!
- How can we make it easier?

APK Reversing

- Rename Android app (.apk) to .zip.
- Extract .zip.
- Run Dex2Jar desktop script (.bat or .sh) on extracted .dex file
- Dex2Jar decompiles .dex to .jar (Java Archive)
- Open .jar in Java Decompiler desktop app to review source

http://en.wikipedia.org/wiki/Step_by_Step_(TV_series)

APKTool

- Powerful tool for forensic analysts
- Tool for reverse engineering Android binaries
- Available at code.google.com

```
student@ubuntu:~/Downloads$ apktool d CryptSQL.apk -f
I: Baksmaling...
I: Loading resource table...
I: Loaded.
I: Loading resource table from file: /home/student/apktool/framework/1.apk
I: Load
I: Deco
I: Deco
I: Done
```

```
student@ubuntu:~/Downloads/CryptSQL$ ls -l
total 24
-rw-rw-r-- 1 student student  605 2012-04-15 20:48 AndroidManifest.xml
-rw-rw-r-- 1 student student   91 2012-04-15 20:48 apktool.yml
drwxrwxr-x 3 student student 4096 2012-04-15 21:02 build
drwxrwxr-x 2 student student 4096 2012-04-15 21:02 dist
drwxrwxr-x 7 student student 4096 2012-04-15 20:48 res
drwxrwxr-x 3 student student 4096 2012-04-15 20:48 smali
```

androguard

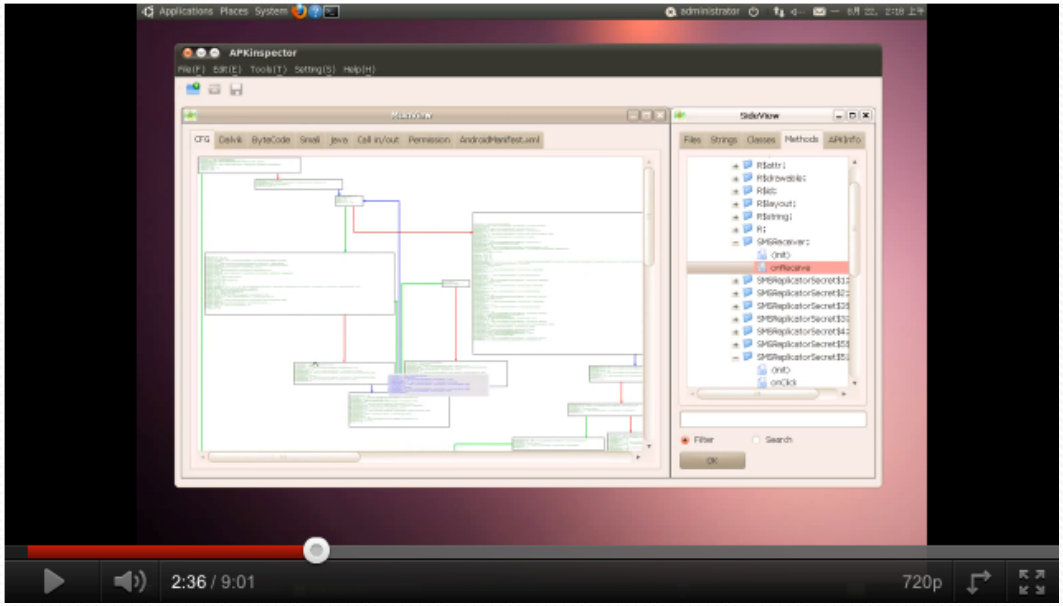
- Reverse engineering, Malware and goodware analysis of Android applications … and more !
- Check for permissions and usage
- Available at code.google.com

```
student@ubuntu:/opt/androguard$ ./androlyze.py -x -i /home/student/ApiDemos.apk
```

```
PERM :  CAMERA
        Lcom/example/android/apis/graphics/CameraPreview; onResume ()V (@onResume-BB@0x0-0x6)  ---> Landroid/hardware/Camera; open ()Landroid/hardware/Camera;
PERM :  NFC
        Lcom/example/android/apis/nfc/ForegroundDispatch; onCreate (Landroid/os/Bundle;)V (@onCreate-BB@0x0-0x3a)  ---> Landroid/nfc/NfcAdapter; getDefaultAdap
ter (Landroid/content/Context;)Landroid/nfc/NfcAdapter;
        Lcom/example/android/apis/nfc/ForegroundDispatch; onPause ()V (@onPause-BB@0xe-0x12)  ---> Landroid/nfc/NfcAdapter; disableForegroundDispatch (Landroid
/app/Activity;)V
        Lcom/example/android/apis/nfc/ForegroundNdefPush; onCreate (Landroid/os/Bundle;)V (@onCreate-BB@0x0-0x8)  ---> Landroid/nfc/NfcAdapter; getDefaultAdapt
er (Landroid/content/Context;)Landroid/nfc/NfcAdapter;
        Lcom/example/android/apis/nfc/ForegroundNdefPush; onPause ()V (@onPause-BB@0xe-0x12)  ---> Landroid/nfc/NfcAdapter; disableForegroundNdefPush (Landroid
/app/Activity;)V
        Lcom/example/android/apis/nfc/ForegroundNdefPush; onResume ()V (@onResume-BB@0xe-0x16)  ---> Landroid/nfc/NfcAdapter; enableForegroundNdefPush (Landroi
d/app/Activity; Landroid/nfc/NdefMessage;)V
```

APKinspector

- Powerful tool for forensic analysts
- Graphically reverse engineer and analyze apps
- Available at code.google.com



REVIEW

- Explored reversing tools for Android
- Reverse engineered app back to source code
- Explored code and data for an APK

EXERCISE

- Reverse engineer an app and locate critical data
  - Use APKInspector
  - Reverse engineer Facebook or F-Droid, mobile app market, application
    - Both apps located in Documents directory on workstation
  - Locate the database where user ID's are stored

Learning Objectives

By the end of this course, you will be able to:

- ✓ Extract and analyze data from an Android device
- ✓ Manipulate Android file systems and directory structures
- ✓ Understand techniques to bypass passcodes *NEW!*
- ✓ Utilize logical and physical data extraction techniques
- ✓ Reverse engineer Android applications
- ✓ Analyze acquired data