Researching Android Device Security with the Help of a Droid Army

Joshua J. Drake

June 24th, 2014

Shakacon VI

Honolulu, HI

Agenda

- Introduction
- Building a Droid Army
- Inside the Visionary
- Doing your Bidding
- DEMO
- Conclusion / Q & A

INTRODUCTION

Who, Why and What…

About Joshua J. Drake aka jduck

- Focused on vulnerability research and exploit development for the past 15 years
- Current affiliations:
  - Lead Author of Android Hacker's Handbook
  - Director of Research Science at Accuvant LABS
  - Founder of the droidsec research group
- Some might know me from my work at:
  - Rapid7 Metasploit, VeriSign iDefense Labs

Motivations

- I want to help others overcome the biggest challenge in Android security research…

FRAGMENTATION

aka

a very heterogeneous device pool

Causes of Fragmentation

- Device models differ from each other
  - Hardware, Code changes, Compilation settings (ARM vs. Thumb), …and more!
- Android development is scattered
  - Different parties make changes when developing a particular device for release

(see my previous presentations for details)

Effects of Fragmentation I

- Many vulnerabilities only present on a single device model or a subset of device models

- Some bugs are only exploitable on a subset

Effects of Fragmentation II

- Both research and test time is multiplied
- The code behind a given attack surface could be **COMPLETELY** different
  - It's almost guaranteed to have small differences
    - Possibly more bugs introduced
    - Possibly some fixes back-ported
- Physical devices become a **REQUIREMENT**

What is a Droid Army?

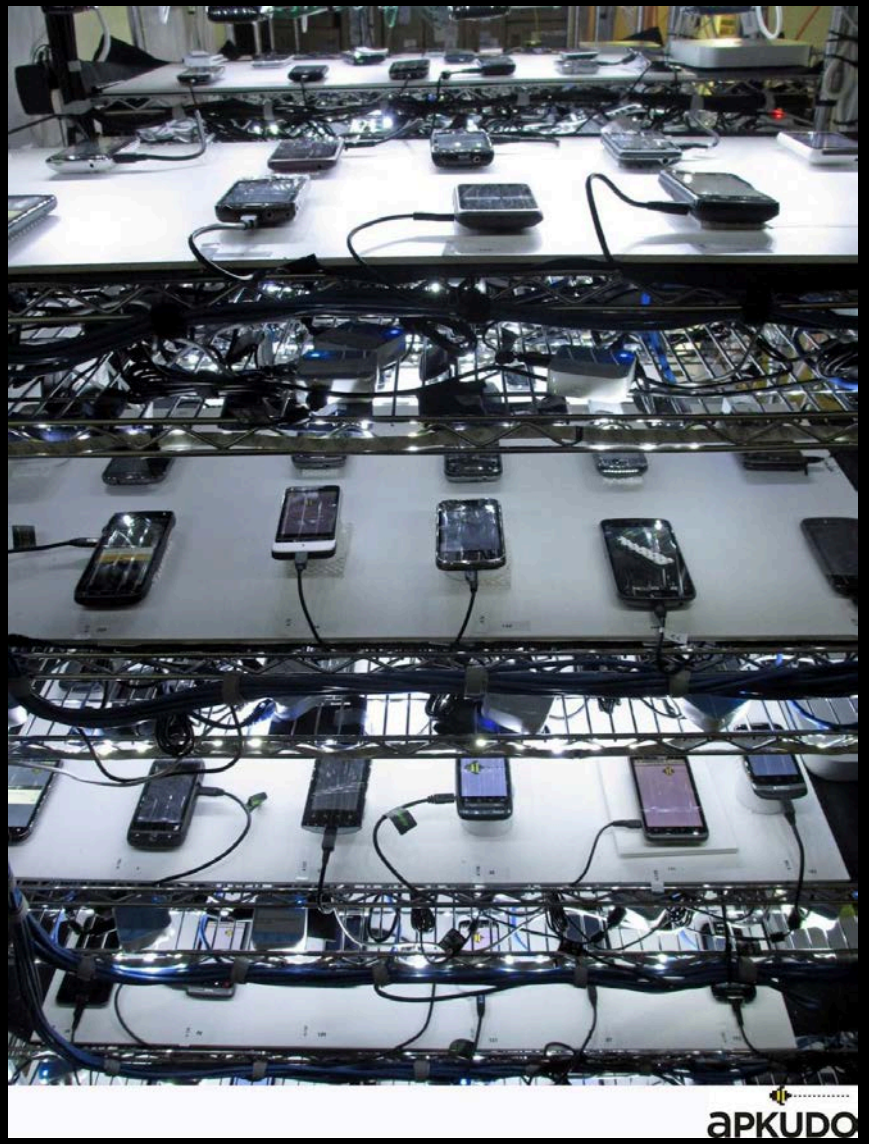
Droid Army (noun):

- A collection of always accessible Android devices used to enable large scale security research.

- QUICK DEMO ☺

Existing Solutions I

- App Developers know this problem well…
- Apkudo (260+)
  - Inspired me
- Testdroid (258)
- AppThwack (231)
- Xamarin test cloud (?)



Existing Solutions II

- These can be used for some tasks, but not all.
- Drawbacks
  - Focused on App testing, not security.
  - Legality concerns
    - Is it ok to root their devices?
      - “We never root … -AppThwack”
    - Is it ok to ex-filtrate data?
  - Physical proximity requirements
  - OPSEC fail
- The answer?
  - Build your own!

BUILDING A DROID ARMY

About the hardware design and acquisition…

Original Design

- Very, very simple/crude:
  1. Get a big ass hub
  2. Obtain lots of devices
  3. Connect everything together
- Initial hardware purchase:
  - Big ass hub: $75 via Amazon
- Had a few devices, sought more…

Acquiring Devices

0 or $ 1. Ask around!

$$ 2. eBay
- Fairly easy to get a good deal
- Esp. damaged but functional devices
  - bad ESN, cracked screen, etc.

$$ 3. Facebook Garage Sales

$$$ 4. Craig's List, Swappa.com, etc.
- Too pricey IMHO

$$$$$ 5. Buy NEW / Off contract
- Very pricey (sometimes unavoidable)

$$ NOTE: new prepaid phones are cheap

e.g. VZW Moto G - $100 @ BestBuy

THANK YOU!

The following persons contributed Android devices:

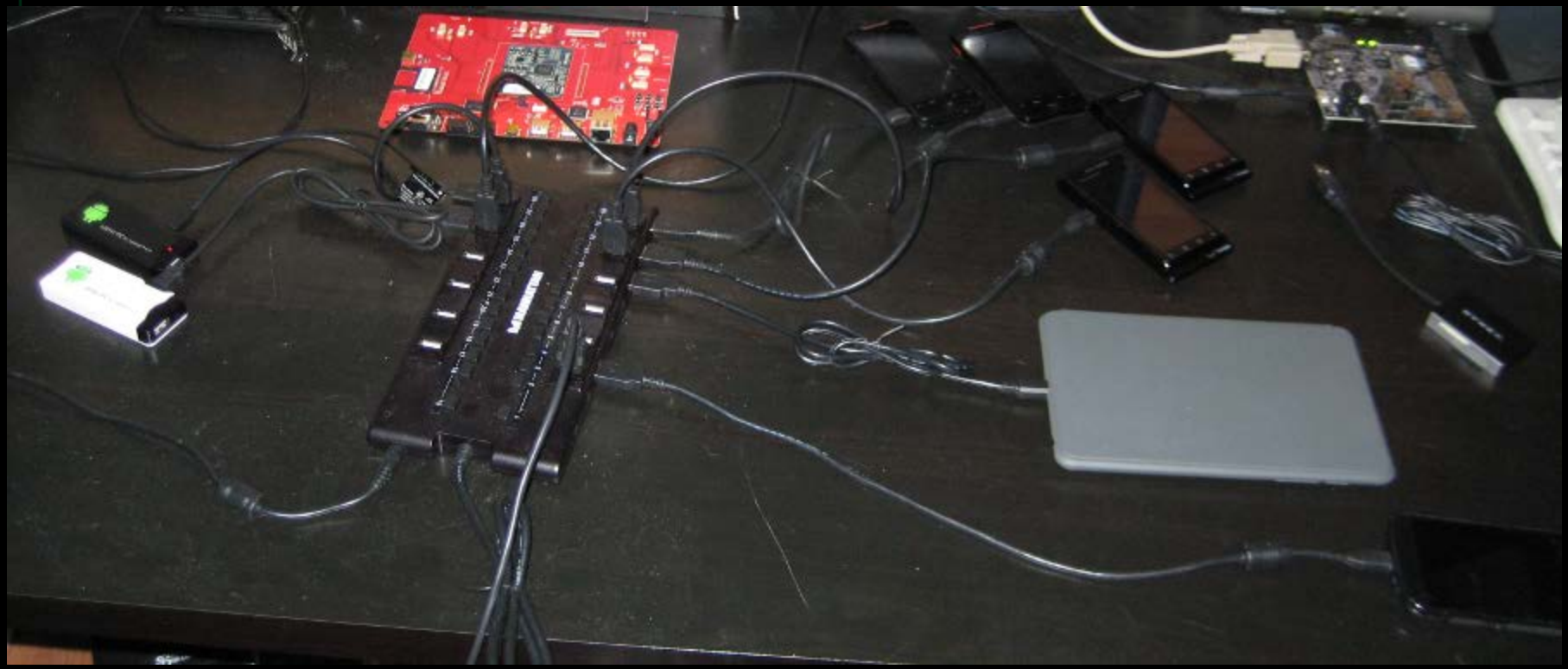
Accuvant LABS
Aarika Rosa
Brent Cook
Charlie Miller
Craig Williams
EMH
Gabriel Friedmann
Google
James Boyd
Jonathan Cran
Justin Fisher
Kevin Finisterre
Matt Molinyawe
Rick Flores
@thedude13
Tim Strazzere

Other generous AHA! Members

Friends, family, and friends of family

Version 0.7 – Sep 2012

Version 0.8 – Oct 2012



Starting to get serious, as evidenced by the organization!

Version 1.0 – Dec 2012

I really started to realize the benefits!

Version 2.0 – July 2013

My posse’s getting big and my posse’s getting bigger!!

Oh no!

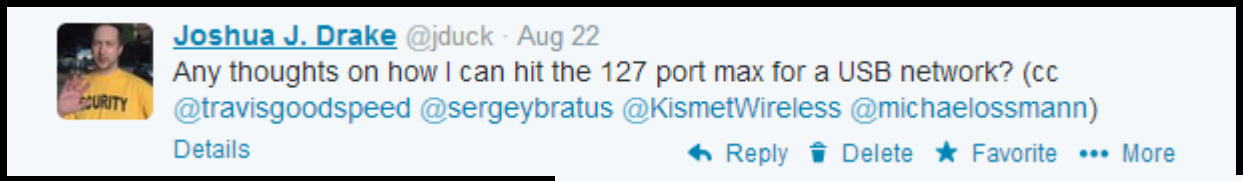
DISASTER STRIKES!!

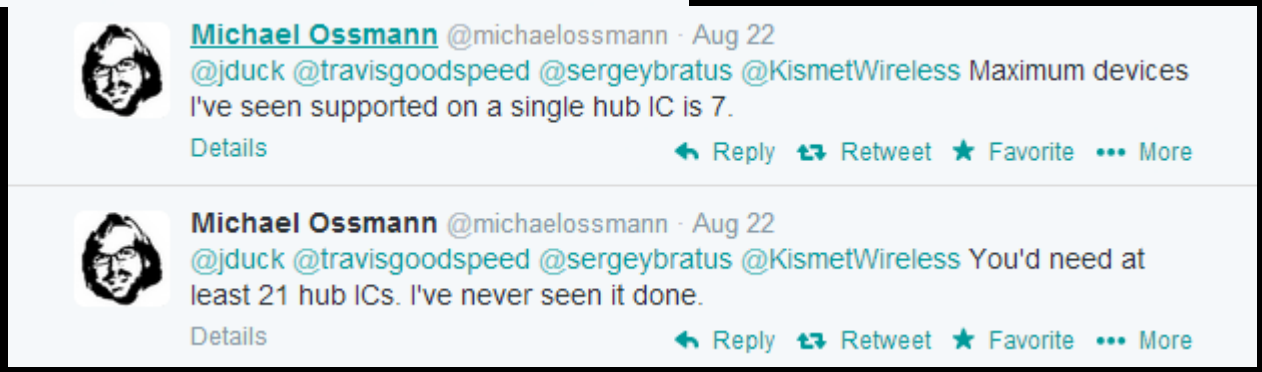
Version 2.7 – Nov 2013

The army is crippled!

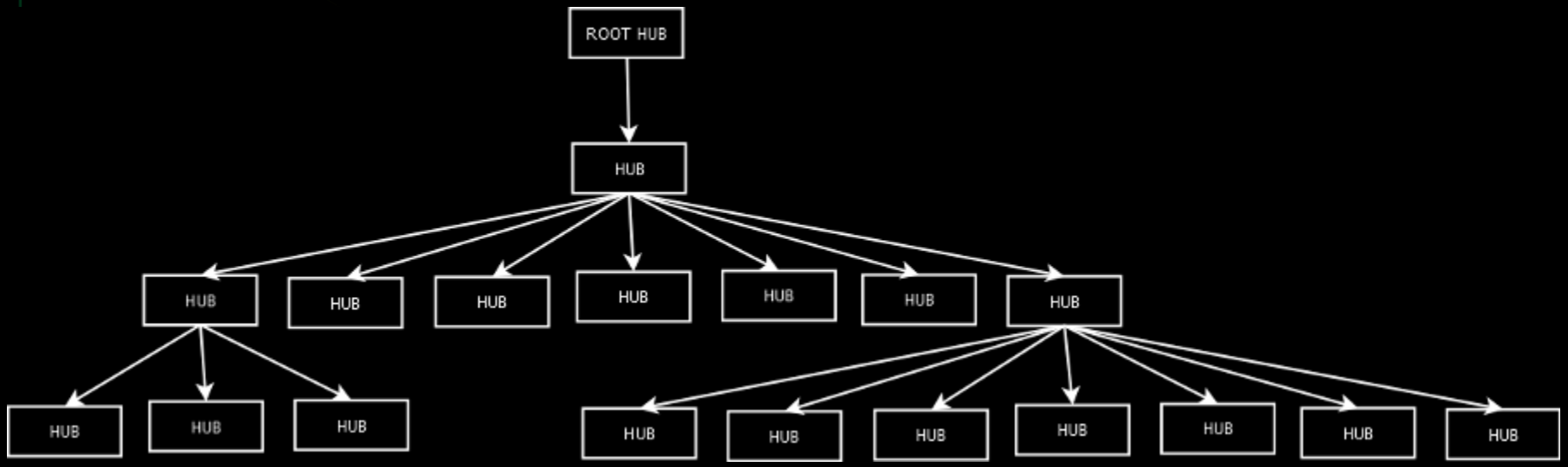
Version 3.0 – Issue I

- How many devices can we ***REALLY*** have?
- Turns out USB has some limitations!
  - Max. hub nesting depth – 7 (root hub counts!)
  - Max. devices (incl. hubs) – 127





Version 3.0 – USB Design I



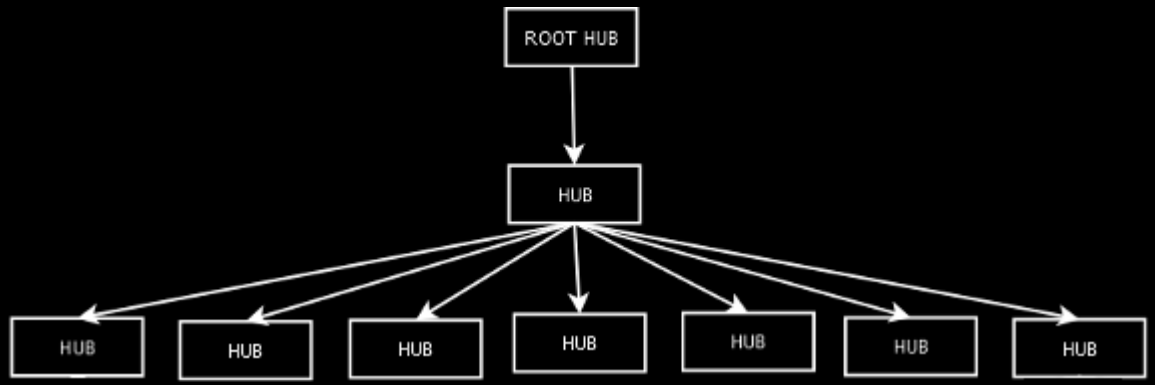
- Realistic max droidz = **108**
  - Hit 127 pretty quickly, with only 19 hubs
  - Several unusable ports :-/

Version 3.0 – USB Design II

- Built off recommendations, reports of previous success, and my own experiences
  - Thanks Charlie Miller, Sergey Bratus, others!
- Parts list:
  - 10x D-Link DUB-H7 hubs (Amazon - $26 ea)
    - 7 ports, remarkably stable
    - Software power control!
  - 70x Micro-USB cables (Monoprice - $1-2 ea)
    - Some 1.5 ft, some 3 ft
    - Some w/ferrite core, some w/o
    - NOTE: a 6ft cable helps if touching a device is needed

Version 3.0 – USB Design III

- Currently topology:



  - root -> 7 port hub -> 7 hubs -> droidz
  - Supports ~ 49 USB devices

- Another issue becomes apparent…

Version 2.7 – Issue II

Wall Warts + Power Strip = FAIL

Version 3.0 – Power Design I

- Modeled after some Bitcoin miner's projects
  - https://bitcointalk.org/index.php?topic=74397.0
- Parts list:
  1. An ATX power supply (surplus ☺)
  2. 10x Male Molex connectors
     - From FrozenCPU or 3D print 'em!
  3. 40x Molex Pins (FrozenCPU)
  4. 10x wired barrels (two options)
     1. Butcher power supplies that came with the hubs
     2. Order some (DigiKey CP-2191-ND)
- I ordered new and assembled my own. The result…

Version 3.0 – Power Design II

The fancy Molex to Barrel cable

Version 3.0 – Power Design III

The power cables all wired up.

More Scale Issues

- More than 108 devices
  - More USB host adapters – PCI-X slot limits
  - Use a small ARM box (ODROID?)
    - Connect via Ethernet
    - **Achieves ~Limitless scale !!**
- Running out of physical space!
  - Pondering a vertical solution
- Maybe power phones without batteries?

Version 3.0 – Dec 2013

The result of the version 3.0 overhaul

Version 3.3 – Current

TODAY!

INSIDE THE VISIONARY

About the Android Cluster Toolkit…

Android Cluster Toolkit I

- No tools like this existed…
  - …or at least none were available
  - …guess it's time to build them!
- Features:
  - Provision new devices quickly/easily
  - Manage devices by human-friendly names
  - Handle transient devices (not always connected)
  - Perform tasks against one or more device
- https://github.com/jduck/android-cluster-toolkit

Android Cluster Toolkit II

- Requirements: ADB binary and Ruby
- Scripts wrap Android Debug Bridge (ADB)
  - README.md covers details and usage
- Simple but elegant and powerful
  - 1 device, multiple devices, all devices
- Recommended I:
  - Minor patch to ADB: https://gist.github.com/jduck/8849310

Recommendation II - BusyBox

- The tools on an Android devices are limited
  - e.g., some don’t have “grep”
- BusyBox solves this problem
- Best BusyBox binary out there (AFAIK):
  - Provided by saurik (Jay Freeman)
  - Only works on devices >= Android 2.3.x
  - Features:
    - More busybox tools (SELinux!!)
    - Built against bionic (shows users/groups correctly)

http://cache.saurik.com/android/armeabi/busybox

Supporting Data

- Firmware images for devices (“stock roms”)
  - Restore your devices to factory settings
  - Extracting offsets, addresses offline
- Source code
  - AOSP checkout
    - Compiler toolchain, etc
    - Base source for Android devices
    - Exact code for Nexus devices
  - GPL releases
    - Linux kernel for device kernels
- More info in AHH and slides from previous talks

DOING YOUR BIDDING

Deploying your army for security research…

…NOW WITH **DEMOS**!

Tasks I

- All device interaction!!
- Query for:
  - ~~“fingerprint”~~
  - ~~Linux kernel version~~
  - System-on-Chip
  - ADB user privileges
  - Root status

Tasks II

- Auditing tasks:
  - Check for driver (exynos-mem, pvrsrvkm)
  - Comparing devices
    - Processes
    - File system
    - init scripts
    - Key files
      - Manifests
      - /system/etc/permissions/platform.xml
    - Plenty more!

Tasks III

- Other tasks:
  - Install an app
  - Push files to all devices
  - Pull files from all devices
    - Offline interaction
  - Test exploits (CVE-2013-6282)
- Subset interaction!!

Tasks IV

- Final demo
  - Running scripts
    - e.g., kernel config – heap selection
- Other tasks (w/o demo):
  - Send Intents
  - Fuzzing
  - Checking compatibility
    - Tested “PatchDroid” by Dr. Collin Mulliner
  - Testing addJavascriptInterface

CONCLUSION

These are the facts you are looking for.

Lessons Learned

- Various problems appeared over time
- Occasionally disappearing devices
  - Require intervention, sometimes manual :-/
- Random sounds emanating from cluster
  - Distracting!
- Li-Ion batteries do not like overcharging!
  - Swollen, scary, need replacing
  - Seem to live ~ 2 years

Future Directions I

- MOAR DEVICES!!@#$%!
  - Please donate!
  - http://www.droidsec.org/donate/
- Further automation
  - privmap, canhazaxs, device diffing, etc
  - Automated firmware switching, setup
- I’m open to suggestions!
  - Email me ;-)

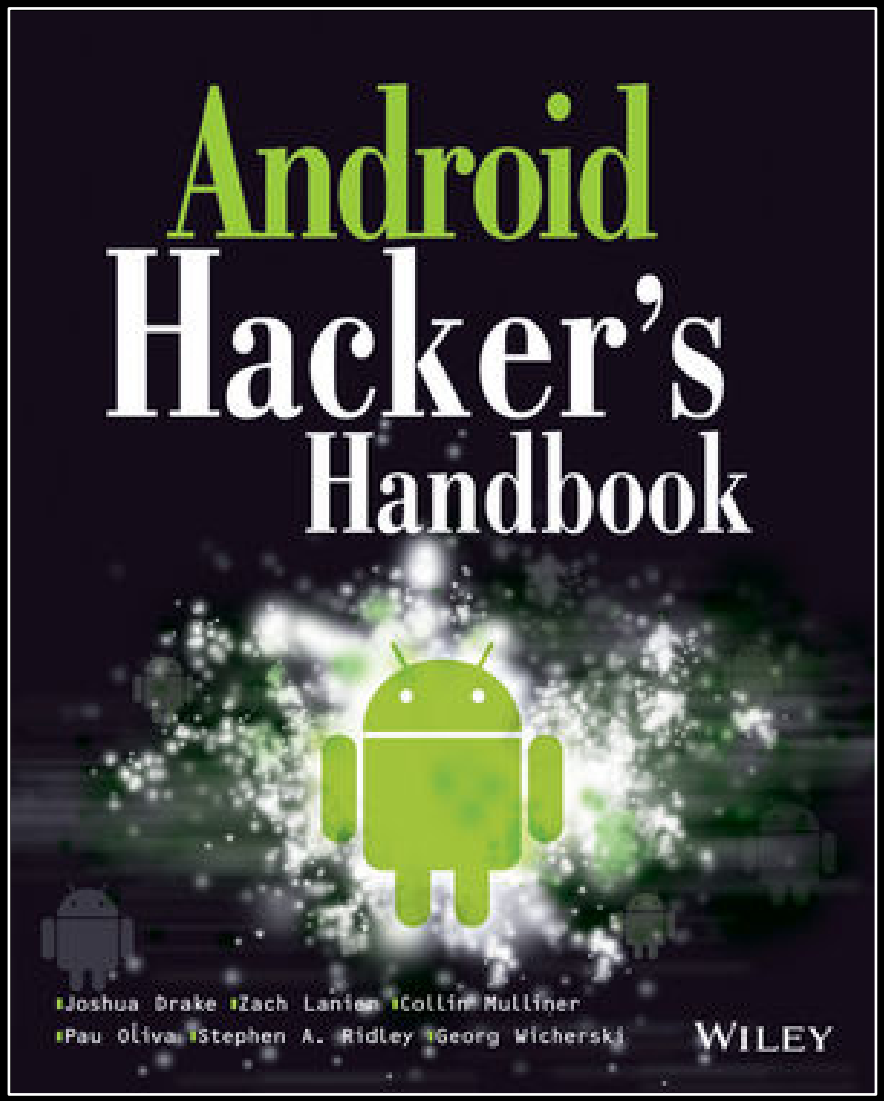
Conclusions

- Device differences complicate security research.
- Building and using a Droid Army helps you scale your research!
  - Provide quick and easy access to any particular device, version of Android, etc.
- It’s worth the investment!

Recommendations

- Use the recommended hardware design!
- Ask around for old/unused devices
- Follow device buying guidelines
- Use / contribute to the tools!
- Join and contribute to droidsec ;-)

Book Giveaway!



ASK ME ANYTHING!

Joshua J. Drake
jdrake [at] accuvant.com
jduck on Twitter, IRC, etc.

Accuvant Headquarters
1125 17th Street, Suite 1700, Denver, CO 80202
800.574.0896
www.accuvant.com

BONUS SLIDES

These didn’t make the cut…

Causes of Fragmentation (detailed)

- Device models differ from each other
  - Hardware
    - SoC, peripherals, CPU features, RAM size, etc.
  - Code changes
    - Made by various ecosystem players
      - GOOG, SoCs, OEMs, carriers, third parties, etc.
    - Android OS / Framework, Linux kernel, etc.
  - Compilation settings (ARM vs. Thumb)
  - …and more!

Provisioning New Devices

- Device databases
  - devices-orig.rb
    - maps device serial numbers to names
  - devices.rb
    - generated from devices-orig.rb by reconfig.rb
  - scan.rb
    - shows you devices that are in ‘adb devices’ but not in your database

Provisioning a New Node

1. Plug the device in
2. If not running ADB as root:
    1. Get USB Vendor:Product
    2. Add to udev scripts
    3. Replug :-/
3. Run ./scan.rb
4. Add to devices-orig.rb
5. Run ./reconfig.rb
6. Upload busybox
7. Root the device
8. Do some research!

Where do you get firmware/src?

This stuff is spread allllll over the place :-/

Various places, step-by-step directions
Google/OEM download sites
Snagging OTA updates
community ROM collection sites
random searching - "stock roms" etc.

See AHH Appendices or my 2013 slide decks

Maintenance Tasks

- Fixing problems as they appear (seldom)
- Acquiring more devices is time consuming
- Provisioning new devices
  - Quick and easy with the toolkit!
- Updating firmware / source code
  - Also time consuming (slow downloads!)
  - Sometimes requires re-rooting :-/
  - Infrequent updates reduce the workload ☺