Towards a General Collection Methodology for Android Devices

Timothy Vidas
ECE/CyLab

Chengye Zhang
INI/CyLab

Nicolas Christin
INI/CyLab
Outline

• Motivation
• Related Work
• Android Background
• Collection Process
• Discussion & Future work
• Conclusion
Motivation

Fragmentation among mobile devices makes it more difficult to collect data

Devices have different:
• Form factors
• Operating systems
• Memory layouts
• Connectors
Android Market Share
“There are now over 500,000 Android devices activated every day, and it's growing at 4.4% w/w”

~Andy Rubin, Google
Android is a big player in the mobile smartphone arena. Every major carrier has multiple Android-based devices. Android smartphones typically get some of the “latest and greatest” technologies, such as the newer 4G networks, NFC, and screens that are easier to see in daylight.
Tablets
Android is Everywhere!
Android's Ubiquity brings unprecedented software commonality to this wide range of devices.
Related

Mobile Phone forensics is not new.

Many vendors have already or are bringing collection products to market

Techniques borrowed from the phone hacking or phone modding community
To Root or not to Root

“Phone modding” definition: 
*somehow obtaining elevated privileges on a phone in it's normal operating mode.*

Many methods of rooting exist, rooting is quite popular among tech savvy and non technical people alike.
To Root or not to Root

Rooting may not be desirable for Digital Forensics, for example

• Often device and software version specific

• Data in user data stores may be modified

• A rooted device may be left in a lesser security state
Before getting into the technique, it's worth looking at the android boot process a bit...
Many parts of the Android boot process are akin to that of a typical Linux system.

A) The Boot ROM (software on the board) locates boot media.

B) Boot code is loaded from media into memory. Execution is transferred to the boot code.
Boot Process

Like many boot loaders (think GRUB on a typical Linux machine), the loader here is staged. Stage1 and Stage2.

A) Stage1 will setup RAM

B) Stage1 will load Stage2 into RAM

C) Stage2 is executed (loads other binary images such as “modem” for use by the phone hardware, initializes hardware, etc)

D) Stage2 loads the Linux kernel from boot media, decompresses, configures options,

E) transfers execution to the kernel

Graphic: elinux.org
Recovery Mode

Recovery mode is a standard boot mode invoked by holding keys during the Stage1 bootloader altering the typical boot process.
Recovery Mode

Instead of loading the typical boot image, execution is diverted to a special recovery boot image.

<table>
<thead>
<tr>
<th>Path</th>
<th>Name</th>
<th>FS</th>
<th>Mount Pt</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/mtd/mtd0</td>
<td>pds</td>
<td>yaffs2</td>
<td>/config</td>
<td>Config data</td>
</tr>
<tr>
<td>/dev/mtd/mtd1</td>
<td>misc</td>
<td></td>
<td></td>
<td>Memory Partitioning Data</td>
</tr>
<tr>
<td>/dev/mtd/mtd2</td>
<td>boot</td>
<td>bootimg</td>
<td></td>
<td>Typical boot image</td>
</tr>
<tr>
<td>/dev/mtd/mtd3</td>
<td>recovery</td>
<td>bootimg</td>
<td></td>
<td>Recovery mode boot image</td>
</tr>
<tr>
<td>/dev/mtd/mtd4</td>
<td>system</td>
<td>yaffs2</td>
<td>/system</td>
<td>System files, System apps, etc</td>
</tr>
<tr>
<td>/dev/mtd/mtd5</td>
<td>cache</td>
<td>yaffs2</td>
<td>/cache</td>
<td>Cache files</td>
</tr>
<tr>
<td>/dev/mtd/mtd6</td>
<td>userdata</td>
<td>yaffs2</td>
<td>/data</td>
<td>User Data (apps, settings, etc)</td>
</tr>
<tr>
<td>/dev/mtd/mtd7</td>
<td>kpanic</td>
<td></td>
<td></td>
<td>Crash log</td>
</tr>
</tbody>
</table>
Recovery Collection

A typical recovery image allows for applying updates, wiping user data, etc.

No user data is stored in the recovery partition.

The contents of the recovery image has no effect on the typical boot and operation of the device.

No Radio
Recovery Collection

A custom bootimg is crafted that extends the functionality of the a typical bootimg

Binaries are added:
  adb, su, nanddump

Files are edited:
  default.prop, init.rc
  (enable adb, start the daemon, etc)

Filesystem permissions are adjusted to facilitate the above
Recovery Collection

Executing in this context gives full access to data on the phone.

Using nanddump permits the collection of OOB (eg spare) data that would likely otherwise be lost.
Recovery Collection

Technique was used for DFRWS2011 challenge! :-D
Discussion

This technique:

- Applies to a wide range of devices
- Does not modify any storage areas that hold user data
- Permits a priori setup

- Doesn't take volatile data into account
- Requires different flashing tools based on manufacturer
Moving forward

- Extended functionality/usability
- More devices
- “Unified bootimg”
- Comprehensive list of boot modes (and behaviors)
Conclusion

- Recovery boot method takes advantage of commonality across Android devices

- Comprehensively collects data with no impact to data areas used under normal operation
Thanks!

Tim Vidas

tvidas@cmu.edu