DFRWS 2013

A Study of User Data Integrity During Acquisition of Android Devices

Namheun Son, Yunho Lee, Dohyun Kim, Joshua I. James, Sangjin Lee, and Kyungho Lee

2013-08-05

Dohyun Kim
exdus84@gmail.com

Digital Forensic Research Center
Center for Information Security Technologies, Korea University
Introduction

- Android
- Booting Mode
Android Forensics

- **Android OS Market Share (Q3, 2013)**

<table>
<thead>
<tr>
<th>Operating System</th>
<th>3Q12 Shipments (in Millions)</th>
<th>3Q12 Volumes (in Millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>16</td>
<td>21</td>
</tr>
<tr>
<td>iOS</td>
<td>51</td>
<td>34</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>Symbian</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>Windows Phone 7/</td>
<td>19</td>
<td>16</td>
</tr>
<tr>
<td>Windows Mobile</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Linux</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Others</td>
<td>37</td>
<td>29</td>
</tr>
<tr>
<td>Totals</td>
<td>262</td>
<td>258</td>
</tr>
</tbody>
</table>

- **600,000 Apps & installs**
  - 50 app installs per Android device
Recovery Mode

- **A sort of standard booting mode**
  - Holding keys during boot process
    - Ex) volume key + power key + ...

- **Feature**
  - Wiping partitions, install an system application, etc...
  - Not mount userdata partition

- **We use recovery mode for acquisition user data**
  - DFRWS 2011: “Toward a general collection methodology for Android devices”, Vidas
Related Work

- Logical Method of Data Acquisition
- Physical Method of Data Acquisition
- Commercial Tools
Logical Method of Data Acquisition

Content Provider

- **Data sharing interface for application level**
  - Supporting data between different applications

- **Android use a Sandbox mechanism for security**
  - An application can not approach another application data
    - Have to know application’s URI for access application’s data
    - Google make public android default application’s URI
      - Call history, Contacts, SMS/MMS, etc..

- **Have to install an application contain the content provider**
  - For acquisition other application data
  - User data area could be altered
Logical Method of Data Acquisition

ADB (Android debug Bridge) protocol

- File unit, partition

```bash
# ls -l
ls -l
drwxr-xr-x app_0  app_0  2010-07-19 12:14 com.sec.mms
drwxr-xr-x app_1  app_1  2010-07-19 12:12 com.sec.android.app.appinstaller
drwxr-xr-x app_131 app_131 2011-01-02 16:00 com.sec.android.app.dlna
drwxr-xr-x app_2  app_2  2010-07-19 12:12 com.sec.android.app.drmua
drwxr-xr-x app_3  app_3  2010-07-19 12:12 com.svox.pico
drwxr-xr-x radio  radio  2010-07-19 12:12 com.android.stk
```

E:\#adb pull /data/data/com.facebook.katana/databases .\#facebook_file
pull: building file list...
pull: /data/data/com.facebook.katana/databases/webviewCache.db -> .\#facebook_file/webviewCache.db
pull: /data/data/com.facebook.katana/databases/webview.db -> .\#facebook_file/webview.db
pull: /data/data/com.facebook.katana/databases/uploadmanager.db -> .\#facebook_file/uploadmanager.db
pull: /data/data/com.facebook.katana/databases/fb.db -> .\#facebook_file/fb.db
4 files pulled. 0 files skipped.
1375 KB/s (277504 bytes in 0.197s)
Logical Method of Data Acquisition

Rooting

- **Temporary Rooting**
  - Using by exploit
    - psneuter, regeagainstthecase, zergRush, and so on...

- **Full Rooting**
  - Using by custom booting kernel image

- **Possible to get all data from android device**
  - Imaging partitions, copying files
    - Using ADB protocol and DD binaries
    - Faster than JTAG
Chip-off

- **Directly separates flash memory**
  - From embedded device board

- **Possible to get damaged in process of separating Flash memory**
  - Smartphone, Flash memory

- **Necessary to data reconstruction (File System)**
  - Raw data $\rightarrow$ logical data
  - Ext file system
Physical Method of Data Acquisition

JTAG

- Via JTAG debug port
- Can acquisition all flash memory data
  - 0x00 offset – 0xEnd offset
- But..!
  - Take so long time to extract data
    - 1GB / 1hour
  - Have to detect JTAG debug port
    - Some android smartphone don’t have a JTAG debug port
Commercial Tools

- Cellebrite UFED
  - Use an exploit for rooting
    - To data partition
    - So, data partition can be altered

- XRY
  - Data acquisition is possible only when device rooted
Background

- File System of Android Device
- Data Integrity
File System

- **YAFFS2 (Yet Another Flash File System2)**
  - Motorola smartphone

- **RFS (Robust File System)**
  - A few SAMSUNG smartphone
    - Galaxy S, Galaxy Tab

- **ExtX (Extended File System X) : Ext3/4**
  - After Gingerbread (Android 2.3) use Ext4
    - 83.2% of Android devices use Gingerbread or above as of Oct.2012
File System of Android Device

Feature of Ext3/4

- Ext3/4 File System has a journaling function
  - So, unallocated and journal areas are altered when a partition is mounted
    - Data structure for the ExtX Superblock

<table>
<thead>
<tr>
<th>Byte Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-15</td>
<td>Number of unallocated blocks</td>
</tr>
<tr>
<td>16-19</td>
<td>Number of unallocated inodes</td>
</tr>
<tr>
<td>24-27</td>
<td>Block Size</td>
</tr>
<tr>
<td>44-47</td>
<td>Last mount time</td>
</tr>
<tr>
<td>48-51</td>
<td>Last written time</td>
</tr>
<tr>
<td>52-53</td>
<td>Current mount count</td>
</tr>
<tr>
<td>88-89</td>
<td>Size of each inode structure</td>
</tr>
<tr>
<td>208-223</td>
<td>Journal ID</td>
</tr>
<tr>
<td>224-227</td>
<td>Journal inode</td>
</tr>
</tbody>
</table>
Data Integrity

Mount as read-only and Turn off the Android Device

- Data Integrity is most important for Digital Forensics

- Have to prevent alteration
  - Partition mount as read-only
    - Unallocated area and metadata(supercell) are not altered
  - Turned off the Android device
    - Prevent partition mounting from booting process
Process of User Data Acquisition

- Prepare the Custom Recovery Mode Image (CRMI) (1/6)
- Boot the Device for Flashing (2/6)
- Flash the CRMI to Boot Partition of the Device (3/6)
- User Data Acquisition (4/6)
- Return to Former State (5/6)
- Restoring a Device to Its Original State (6/6)
A Study of User Data Integrity During Acquisition of Android Devices

Process of User Data Acquisition

- Include a busybox binary, enable adb service, has root authority and mount rootfs partition as read/write mode.
- The partition(s) has(have) to mount as read only mode.
- Using adb & dd(cat) or busybox nanddump(if destination partition is using yaffs2) & busybox netcat
- Using adb push and dd(cat) or busybox nanddump (if destination partition is using yaffs2)
1. Checking the model name of the target device

2. Check whether or not the ready Custom Recovery Mode Image (CRMI) for the model
   ➔ move next step

3. Make a CRMI for model of the device
Prepare the Custom Recovery Mode Image (CRMI) (1/6)

Make a CRMI for Data Integrity

- Edit a recovery partition ramdisk (modify init.rc, default.prop, adbd file...)
  - Enable root authority and ADB protocol
  - User data partition unmounts
Prepare the Custom Recovery Mode Image (CRMI) (1/6)

Make a CRMI for Data Integrity

- **Edit a recovery partition ramdisk (modify init.rc, default.prop, adbd file...)**
  - Delete the not related files to booting (resource...)
    - CRMI size should equal the size of the boot partition
      - CRMI should be used for the boot partition
    - Mostly, the size of recovery and boot partition are the same
    - But, some of Android device’s recovery partition is bigger than boot partition

<table>
<thead>
<tr>
<th>Device</th>
<th>Size(Kb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boot Partition</td>
</tr>
<tr>
<td>Droid (A855)</td>
<td>3584</td>
</tr>
<tr>
<td>Galaxy S2 (SHW-M250S)</td>
<td>8192</td>
</tr>
<tr>
<td>Galaxy Nexus (SHW-M420K)</td>
<td>8192</td>
</tr>
<tr>
<td>Galaxy Note (SHV-E160S)</td>
<td>10240</td>
</tr>
<tr>
<td>Galaxy S3 (SHV-E210S)</td>
<td>8192</td>
</tr>
<tr>
<td>Galaxy Note 2 (SHV-E250S)</td>
<td>8192</td>
</tr>
<tr>
<td>Vega LTE (IM-A800S)</td>
<td>10240</td>
</tr>
</tbody>
</table>
1. **Boot the device in flashing mode for flash the CRMI**
   - Use ADB protocol
     ex) adb reboot download
   
   ❖ **Method for entering flashing mode**
     varies for each model
     ex) power key + volume down key + home key

```plaintext
Warning!!
A custom OS can cause critical problems in phone and installed applications.
If you want to download a custom OS, press the volume up key, Otherwise, press the volume down key to cancel.
Volume up : Continue
Volume down : Cancel (restart phone)
```

```plaintext
Downloading...
Do not turn off target!!
```

Start
Check model of target device

Is there a custom recovery mode image\(^1\) for the model of the device that was already made?

- No
  Make a custom recovery mode image\(^2\) for model of the device
- Yes

Boot the Device for Flashing

Flash the custom recovery mode image to boot partition of the device

Reboot device

Select a method of user data acquisition

File(s)
Partition(s)

Copy file(s)/folder(s) using adb pull

Mount\(^3\) (a) user data partition(s)

Partition(s) imaging\(^4\)
1. Flash the CRMI to boot partition
   ➔ Use Odin program

2. Reboot device
   ➔ Device is booted recovery mode using CRMI after flashing is finished
   ➔ Acquire root authority
User Data Acquisition (4/6)

1. Select a method of user data acquisition
   ➔ Partitions / Files

2. If select a file acquisition
   • Mount a user data partition (read-only)
   • Copy files in data partition using ADB protocol
     ex) adb pull /data/data/com.android/databases/xxx.db

3. If select a partition acquisition
   • Imaging data/SDcard partition
   • Use DD, NC binary
1. Check the original firmware version of the device
   ➔ /system/build.prop file

2. Overwriting the original boot image to boot partition by using DD binary
   ➔ copy the boot partition to the device by using ADB push

<table>
<thead>
<tr>
<th>Device</th>
<th>Boot block name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Droid (A855)</td>
<td>/dev/block/mtdblock5</td>
</tr>
<tr>
<td>Galaxy S2 (SHV-M250S)</td>
<td>/dev/block/mmcblk0p5</td>
</tr>
<tr>
<td>Galaxy Nexus (SHV-M420K)</td>
<td>/dev/block/mmcblk0p7</td>
</tr>
<tr>
<td>Galaxy Note (SHV-E160S)</td>
<td>/dev/block/mmcblk0p8</td>
</tr>
<tr>
<td>Galaxy S3 (SHV-E210S)</td>
<td>/dev/block/mmcblk0p5</td>
</tr>
<tr>
<td>Galaxy Note 2 (SHV-E250S)</td>
<td>/dev/block/mmcblk0p8</td>
</tr>
<tr>
<td>Vega LTE (IM-A800S)</td>
<td>/dev/block/mmcblk0p8</td>
</tr>
</tbody>
</table>
1. Disconnect the USB Cable

2. Battery remove in order to prevent data modification
   - Sequence is very important (disconnect the USB cable → remove the battery)
   - Certain devices (ex: Galaxy S2) mount the data partition by only cable power

A Study of User Data Integrity During Acquisition of Android Devices

DFRC (Digital Forensic Research Center) Korea University
Android Extractor
Introduce an Android Extractor tool
Experiment using Android Extractor

- Experiment Method
- Experiment Result
## Experiment Method

### Test user data integrity during the acquisition process

#### Select seven different Android devices for test

<table>
<thead>
<tr>
<th>Device</th>
<th>JTAG to CRMI</th>
<th>CRMI to CRMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Droid (A855)</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Vega LTE (IM-A800S)</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Galaxy S2 (SHW-M250S)</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Galaxy Nexus (SHW-M420K)</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Galaxy S3 (SHV-E210S)</td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Galaxy Note (SHV-E160S)</td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Galaxy Note 2 (SHV-E250S)</td>
<td></td>
<td>✔</td>
</tr>
</tbody>
</table>

#### Repeating the process multiple(5) times

- Start (i=5)
- Select a Device to acquire data
- Is it possible to acquire data using JTAG
- Yes: Acquire user data partition using JTAG ($A_i$)
  - $i = i - 1$)
  - Compare hash values of all acquired data
  - Yes: $i = 0$) Finish
  - No: $i = 0$)
- No: Acquire user data partition using CRMI ($B_i$)
- Compare hash values of all acquired data
- Yes: $i = 0$) Finish
- No: $i = 0$)
Experiment Result

- Confirm the hash value are same in all observations
  - JTAG to CRMI
  - CRMI to CRMI

- Suggested data acquisition method preserves the integrity of the data
  - JTAG also preserve integrity
Demonstration
Conclusion
Conclusion

- This study explained a method of preserving integrity at the time of user data acquisition
  - by using the previously studied Recovery Mode

- Result of the experiment
  - Method of user data acquisition using CRMI preserve integrity
    - JTAG also preserve integrity of user data
  - Faster than JTAG
Conclusion

- In order to return to the former state after flashing the CRMI and acquiring data
  - Need an original boot partition from the firmware version
  - If you do not have an original boot partition
    - Flashing the CRMI to recovery partition instead boot partition

- There are several limitations, but..
  - The CRMI method is more efficient
    - compared to other existing methods of forensically sound data acquisition from Android devices.
http://forensic.korea.ac.kr