Privacy leaks in smart devices:
Extracting data from used smart home devices
Chaos Communication Camp 2019 – Dennis Giese
Outline

• Motivation
• Data on IoT devices
• Storage on IoT devices
• Reset states of used devices
• Data extraction methods
• Device analysis
About me

• PhD student at Northeastern University, USA
  – Working with Prof. Guevara Noubir@Khoury
• Grad student at TU Darmstadt, Germany
  – Working with Prof. Matthias Hollick@SEEMOO
• Interests: Reverse engineering of interesting devices
  – IoT, Smart Locks
  – Physical Locks ;}
Side notes

• This is not a Xiaomi bashing talk, issues applies to all vendors
• Most methods already well known
• For ethical/legal reasons, I had to censor most of the data
• Use methods on your own risk
• Some technical aspects are simplified
  – NAND flash is more complex, but I simplify there a lot
  – Actual NAND data interpretation/reassembly out of scope
  – Device-specific rooting out of scope
MOTIVATION
Old problem: data on used hard drives

- Traditionally: second hand hard drives contain still data
  - Issue existed forever, increased with platforms like eBay
  - Devices still contain data like: personal information, emails, pictures and other media, sensitive documents
  - Awareness was raised in early 2000’s

Remembrance of data passed: a study of disk sanitization practices
Publisher: IEEE

Abstract:
Many discarded hard drives contain information that is both confidential and recoverable, as the authors’ own experiment shows. The availability of this information is little publicized, but awareness of it will surely spread.

Old problem: data on used hard drives

• Also affected: Multifunction printers, Lab instruments
• Published standard: NIST SP 800-88 (2006)
• Solutions:
  – Wipe hard drives
  – Sell used devices without hard drive
• Remaining problems:
  – Lack of knowledge or awareness
  – Carelessness
  – Broken devices


IT Liquidators [CC BY-SA 3.0]
https://commons.wikimedia.org/wiki/File:Destroyed_Hard_Drive.jpg
Old problem: data on used hard drives

• Problem still exists today
  – Of 159 second-hand HDD’s/SSD’s 66 (42%) still contained sensitive data


Many Used Hard Drives Sold on eBay Still Contain Leftover Data

Data removal company Blancco sponsored a study that analyzed 159 SSD and HDD storage drives purchased on eBay and found that many still contained leftover data from the previous owners.

By Michael Kan  April 26, 2019 4:36PM EST
Smartphones: sensitive data to go

- Phones store much sensitive information:
  - Pictures, Messages, Account credentials, call lists
- Device storages were not encrypted by default
  - Introduced with iOS 8 (2014), Android 6.0 (2015)
- Factory reset was not wiping all the data
  - Paper “Security analysis of android factory resets” (2015): Android < 4.0 does not wipe data correctly
- Addressed with new NIST SP 800-88 Rev. 1 (2014)
IoT is everywhere

• In contrast to smartphones/PC:
  – Smaller or no user interface
  – Data on device not directly accessible
  – Unclear which data is collected in the first place

• Factory resets
  – not fully verifiable
  – Implementation unclear, depends on version and vendor
Motivation for this talk

• During my master thesis @SEEMOO
  – Analysis of security of many IoT devices
  – Goal: root access to devices
  – When factory resetting devices:
    • traces of data were left on the device
    • sometimes all data was still available
Where you find used devices

- eBay, Amazon Warehouse deals
- flea markets
- Trash
- Family and friends
- (In your home)
When smart devices pass secrets to the police

It may crackdown on crime — and privacy, too. That's if German police get powers to seize personal data on smart devices. Germany's discussing plans that are already a reality in the USA.
DATA ON IOT DEVICES
Data on IoT devices

- Data on individual devices depending on device type
- All IoT devices require: Wi-Fi credentials, Cloud credentials, Cloud bindings
- Rule of thumb: The more performance/functions/storage a device has, the more data is available on it
Vacuum cleaners

- Connection log files
- Maps
- Cleaning logs
- User ID
Smart Home Gateways

• Connection log files
• Sensor/actuators bindings
• Sensor/actuators log files
• Key material
• User ID
Cameras

- Cached snapshots/video clips
- Recorded video
- Event logs
- User ID
- Cloud storage credentials
Routers

- DHCP leases (MAC, IP, timestamp)
- Firewall configurations
- Media files
- Logfiles (connection, DNS, filters, etc.)
- Other credentials
Media players

- Connection log files
- Media libraries
- Playlists
- Cache
- Browsing history
- Other credentials/tokens
  - Google Play Store
  - Network shares

For ethical reasons I have to skip this device 😞
Toys

- WIFI credentials
- Configuration settings
- Video/audio data
- Usage logs
STORAGE ON IOT DEVICES
Storage on IoT devices

- 2 groups of storage types:
  - Raw flash
    - serial flash (SPI)
      - NAND
      - (NOR)
    - Raw parallel NAND flash
  - Block devices
    - eMMC
    - eMCP
    - (SD cards)

- Choice of storage type affects useable filesystems
Raw NAND flash

- SPI flash: typically sizes < 64MByte
  - Packages: SOP8, WSON8,…
- Raw NAND: typically 128MByte – 4GByte
  - Packages: TSOP-48, TSOP-56, BGA-63
- Cheap and fast storage, but Bit-errors
- Host processor/OS tasks:
  - Wear leveling
  - ECC (sometimes CPU accelerated)
  - Bad-Block management
- Abstraction under Linux
  - MTD subsystem (Memory Technology Devices)
  - Character device -> Block device
Raw NAND flash properties

- organized in blocks and pages
  - To erase data, a whole block needs to be erased
  - Erasing sets all bits to 1
  - Typical block sizes: 16-512 Kbytes
  - Typical page size: 0.5-2 Kbyte
  - Programming works on page level
  - OOB: management + ECC
- Flash contains additional spare blocks
- ECC is computed by Host CPU
  - Sometimes vendor specific computation
Wear-leveling for raw flash

- Problem: individual flash cell has limited writes
  - File-systems like Ext2/3/4 are not wear-leveling aware
  - Many writes can destroy the flash or corrupt the data
- Solution: Flash aware file-systems or additional layer
  - File-System (on partition level only): YAFFS, JFFS/JFFS2
  - Additional layer (on device level): UBI+UBIFS
  - Support of Bad-Block management and Wear leveling in OS
  - Idea:
    - Deleted blocks are not erased, but only marked as such
    - The changed information is copied into a new block
    - Garbage collector may clean up erased blocks if needed
How Wear-leveling works

Logical level
- Data block
- Data block
- Data block

Physical level
- Used block
- Used block
- Bad block
- Used block
- Free block
- Free block

Simplified!
How Wear-leveling works

Logical level
- Data block
- Data block
- Data block

Physical level
- used block
- used block
- bad block
- used block
- free block
- free block

Block to be changed

Simplified!
How Wear-leveling works

Logical level
- Data block
- Data block
- Block to be changed

Physical level
- used block
- used block
- bad block
- used block
- free block
- free block

Simplified!
How Wear-leveling works

Logical level
- Data block
- Data block
- Data block

Physical level
- used block
- used block
- bad block
- used block

Block to be changed

Free block

Simplified!
How Wear-leveling works

Logical level
- Data block
- Data block
- Data block

Physical level
- used block
- used block
- bad block
- used block
- used block
- free block

Block to be changed

Simplified!
How Wear-leveling works

 Logical level
 Data block  Data block  Data block

 Physical level
 used block  used block  bad block  dirty block  used block
   write block  free block

Block to be changed

Simplified!
How Wear-leveling works

Logical level

Data block → Data block → Data block

Physical level

used block → used block → bad block → dirty block → used block → free block

Data still present

write block
Interesting Wear-leveling properties

• Multiple copies of the data may exist
  – Data is not being erased as long as the block is not erased
  – Size of copies usually > 2KByte
  – Data changed regularly exists more often

“History” of changes remains
Recommended material about NAND

- Blackhat USA 2014: “Reverse Engineering Flash Memory for Fun and Benefit” by Jeong Wook (Matt) Oh
  - Intro in the communication protocol
  - Soldering/Unsoldering of NAND flash
  - How-to reverse engineer NAND formats

- “From NAND chip to files” by Jean-Michel Picod
  https://www.j-michel.org/blog/2014/05/27/from-nand-chip-to-files
Sidenote

- Even device manufacturers are not aware of JFFS2 properties
- Example of leaked developer keys from my DC26 talk

```
0004cc10  e3 b5 3b e8 00 2c 23 20 63 61 74 20 2f 65 74 63 |.;.;. cat /etc|
0004cc20  2f 6d 69 69 6f 2f 6d 75 73 74 20 62 65 20 61 20 |/miio/device.con|
0004cc30  6d 75 73 74 20 62 65 20 61 75 73 69 67 6e 64 0a | must be |
0004cc40  23 20 64 69 64 3d 32 38 3a 6c 69 6e 64 0a iXn.mac=28:6C:07|
0004cc50  3a 32 45 3a XX XX XX XX XX XX XX XX XX XX XX XX | 2E: vendor|
0004cc60  69 58 6e 0a 6d 61 78 20 6c 65 6e 20 32 33 0a | len 23........|
0004cc70  80 02 94 03 00 02 2e 76 65 6e 64 0a 31 31 31 41 0a | =lumi. model ma|
0004cc80  3d 69 66 66 65 20 2f 20 20 20 2f 65 74 63 61 74 20 | x len 23........|
0004cc90  69 6e 0a 6d 61 78 20 6c 65 6e 20 32 33 3d 30 0a | camera.aq1.p2p_i|
000cca0  3a 32 45 3a XX XX XX XX XX XX XX XX XX XX XX XX XX | d=A,...011A....|
```

Chaos Communication Camp 2019 – Dennis Giese
Block devices

• Also known as managed NAND
• Standards: eMMC 4.x, 5.x
• eMMC:
  – flash with integrated controller
  – Packages: FBGA-153
• eMCP
  – like eMMC, but with on-chip DRAM
  – Advantage: RAM + flash on one chip
  – Packages: FBGA-162, 221
• Under Linux: normal block storage device, supports Ext2/3/4
• Integrated wear-leveling, ECC, bad-block management by FTL
Access to deleted data

- eMMC controller does not allow raw access as for raw NAND
- eMMC/eMCP use raw NAND internally
  - Bypassing of the eMMC controller and direct attachment to NAND possible
  - Challenge: the data format of the eMMC controller
- Recommended talk: “eMMC CHIPS. DATA RECOVERY BEYOND CONTROLLER” by Rusolut
  - Summary: Even is eMMC is deleted, data is still present on internal NAND flash

RESET STATES OF USED DEVICES
Reset states of used devices

- Reset states mostly depends on previous owner
  - No reset at all
    - Device still contains all data and configuration
    - Most probable cause: missing knowledge how to reset or device broken
  - Wi-Fi configuration reset
    - Device may contain data, but is in un-provisioned state
    - Many devices offer only a Wi-Fi reset, e.g. initiated by button
  - Device wipe
    - All data has been wiped, device is factory state (e.g. firmware)
    - There might be still traces of data
    - Not all devices support this
Wi-Fi reset vs. Device wipe

- Some devices support both
  - Wi-Fi reset is usually marked by special button
  - Device wipe is available via the App or via button combination
- Idea of Wi-Fi reset
  - Device can be reconnected quickly to a new Wi-Fi SSID
  - No long duration to reset
  - Most settings remain
- Device wipe: Should erase all user data and restore factory OS
DATA EXTRACTION METHODS
Data extraction methods

• Idea: extract all available data
• Methods:
  – Via software using root access
  – Dumping flash contents without de-soldering (ISP)
  – Dumping flash contents with de-soldering (Chip-Off)
Software methods

• For many IoT devices there are public rooting methods
  – Installation of custom firmware
  – Access via USB or UART
  – Boot via external media (e.g. SD card)
• Dumping flash contents using dd
  – dd is not flash aware, that is helpful in our use-case
  – Extraction of the data via external media, SSH or netcat
• Works good for JFFS2/UBIFS
• Disadvantage: low-level access on flash might be limited
Dumping flash without de-soldering

• Works mainly for SPI and eMMC flash
• Some devices allow In-system programming (ISP)
  – Flash can be accessed via test pins
  – Processor must not interfere
• Advantage: reduced risk of destroying hardware
• Disadvantage: requires knowledge of test pins or PCB traces
Dumping flash with de-soldering

• Works for all flash chips

• De-soldering:
  – Preheating of the whole PCB recommended
  – For accessible pins (e.g. TSOP): create low temperature alloy
  – For BGA chips: Hot air, IR or reflow soldering station required

• Disadvantage for BGA chips: re-soldering requires re-balling
• Raw NAND: requires the use of adapters due to pin count
Tools for SPI flash

- Any device which supports bit-banging on GPIOs
  - Raspberry Pi, Arduino, Bus pirate, etc.
- My favorite: Flashcat USB
Tools for eMMMC flash

- (Some eMMMC chips require a lower voltage!)
- Exploitee.rs eMMMC adapters (~10USD)
  - Connection via SD card reader
  - Difficulty:
    - requires soldering with microscope or good eyes
    - May not be able to access all partitions

https://www.exploitee.rs/index.php/Exploitee.rs_Low_Voltage_e-MMC_Adapter
Tools for eMMC flash

• UFI Box Lite with BGA sockets (~75USD)
  – Simple connection to BGA chips
  – Supports dual voltage chips
  – Disadvantages:
    • Needs many tries to find correct position for BGA
    • Original software for Windows questionable (detected as Malware)
Tools for raw NAND

• High pin count makes soldering difficult
• Requires some sort of NAND controller
  – See also: “Reverse Engineering Flash Memory for Fun and Benefit”
Tools for raw NAND

- Flashcat USB with adapters
  - Supports all kind of raw flash chips
  - However: does not interpret proprietary ECC/OOB data
Tools for raw NAND

• Evaluation boards
  – Idea: soldering the flash on a board with similar SOC and OS
  – Enables read and writes for supported flash chips
  – Disadvantage: Boards are not always available
Analyzing the dumps

• Binwalk
• Hex editor
• raw NAND dumps:
  – Dumpflash by Jeong Wook (Matt) Oh  https://github.com/ohjeongwook/DumpFlash
  – Nand-dump-tool by Jean-Michel Picod  https://bitbucket.org/jmichel/tools
  – Problem: exotic OOB sizes and ECC data
• UBI images: UBIFS Dumper  https://github.com/nlitsme/ubidump/blob/master/README.md
• JFFS2 images: Jefferson  https://github.com/sviehb/jefferson
Methods used

• Disassemble devices and dump flash
• Powering on devices and root devices (if possible)
• Connecting devices to the App
• Using devices and reset them
• Compare available data before and after reset
Ecovacs DEEBOT 900

- Brought 2019, factory reset by previous owner
- Platform:
  - OS: Linux
  - SOC: Rockchip RV1108
  - Flash: Toshiba NAND in TSOP48 (128MByte)
  - RAM: 128 Mbyte DDR3
- Approach:
  - Dumping NAND flash
  - Connecting over UART
Ecovacs verification of factory reset

• After unsoldering and dumping flash:
  – Confirmation that device has been factory reset
  – Fragments of Logfiles, Keys, Maps and Wi-Fi credentials found
  – Problem: Unknown OOB method

• UART on the board is read-only, however found information:
  – Rockchip specific NAND driver with custom wear-leveling
  – Root partition uses SquashFS, Data partition uses EXT4
Ecovacs: extraction of credentials

- Factory credentials

- Previous Owner
## Ecovacs: network connection logs

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019-05-12A</td>
<td>05:36:14</td>
<td>Login</td>
<td>User logged in to Ecovacs network</td>
</tr>
<tr>
<td>2019-05-12A</td>
<td>05:36:15</td>
<td>Logout</td>
<td>User logged out from Ecovacs network</td>
</tr>
<tr>
<td>2019-05-12A</td>
<td>05:36:16</td>
<td>Event</td>
<td>Event triggered by user activities</td>
</tr>
<tr>
<td>2019-05-12A</td>
<td>05:36:17</td>
<td>Alert</td>
<td>Alert message for network connectivity issue</td>
</tr>
<tr>
<td>2019-05-12A</td>
<td>05:36:18</td>
<td>Error</td>
<td>Network error detected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sat 2019-05-12A 10:51:** Connection with Ecovacs network successful and stable.
Ecovacs: locating former owner

- Google’s Geolocation API useful
  - Input: 2 MAC addresses and signal strength
  - Output: Location coordinates with accuracy rating
- Device contained only one BSSID in the log files 😞
Ecovacs: locating former owner

- After querying both SSIDs in wigle.net
Ecovacs: summary

- Most of the user-data still exists on device
  - XMPP network logs, Maps, Credentials
- However: due to custom Map format reassembly difficult
- After resetting the device 3 times: Data fragments still readable
- Interesting aspect: factory logs were still stored
- Previous owner could be tracked
- Good news: App did not leak previous maps
- Very similar results with VIOMI Vacuum Robot V2
Xiaomi/Rockrobo Mi Vacuum Robot

• From 2018, unclear condition of device
• Platform:
  – OS: Ubuntu 14.04
  – SOC: Allwinner R16
  – Flash: eMMC (4GByte)
  – RAM: 512 Mbyte DDR3
• Approach:
  – Dumping partitions via UART
  – Connect device to cloud account
Mi Vacuum Robot data extraction

• Rooting methods exist
  – Root shell via UART or custom firmware
  – Extraction of data via SSH
• Alternative: removing and dumping of the eMMC flash

https://github.com/dgiese/dustcloud
## eMMC layout

<table>
<thead>
<tr>
<th>Label</th>
<th>Content</th>
<th>Mountpoint</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot-res</td>
<td>bitmaps &amp; some wav files</td>
<td></td>
<td>Ext4</td>
</tr>
<tr>
<td>env</td>
<td>uboot cmd line</td>
<td></td>
<td>Text</td>
</tr>
<tr>
<td>app</td>
<td>device.conf (DID, key, MAC), adb.conf, vinda</td>
<td>/mnt/default/</td>
<td>Ext4</td>
</tr>
<tr>
<td>recovery</td>
<td>fallback copy of OS</td>
<td></td>
<td>Ext4</td>
</tr>
<tr>
<td>system_a</td>
<td>copy of OS (active by default)</td>
<td>/</td>
<td>Ext4</td>
</tr>
<tr>
<td>system_b</td>
<td>copy of OS (passive by default)</td>
<td></td>
<td>Ext4</td>
</tr>
<tr>
<td>Download</td>
<td>temporary unpacked OS update</td>
<td>/mnt/Download</td>
<td>Ext4</td>
</tr>
<tr>
<td>reserve</td>
<td>config + calibration files, blackbox.db</td>
<td>/mnt/reserve/</td>
<td>Ext4</td>
</tr>
<tr>
<td>UDISK</td>
<td>logs, maps, Wi-Fi config, userID</td>
<td>/mnt/data</td>
<td>Ext4</td>
</tr>
</tbody>
</table>

We are interested in this
Mi Vacuum Robot reset methods

• Devices support Wi-Fi reset and Factory reset
• Wi-Fi reset: file with Wi-Fi credentials is deleted
• Factory reset:
  – Requires special procedure, mentioned in the manual
  – OS partitions are restored from Recovery
  – Data partition is formatted, but not wiped
  – Partition with usage data is not erased
Mi Vacuum Robot

- After provisioning of device with new account
  - previous data visible in App
  - Assumption: only Wi-Fi reset
  - Data reuploaded to the Cloud
  - Logfiles locally available
- After factory reset:
  - Maps were not visible anymore
Mi Vacuum Robot: locating former owner

- Log files contained 2 BSSIDs
  - Google Geolocation API returned coordinates
- Wi-Fi credentials reveal part of address
  - Password contains personal data
- User-ID
  - Search via Mi Home App
  - Share device with user to reveal name

Source: Google Maps
Mi Vacuum Robot: summary

• All data was still on the device
• Device was not wiped, instead only Wi-Fi reset
  – Reset button is misleading
  – Correct procedure is documented in the manual
• Previous owner could be tracked due to log files
  – Device creates very verbose log files and stores them locally
Other examples: MiTU Drone

- Children toy, but powerful device
  - OS: Android
  - SOC: Leadcore LC1813
  - Flash: 4GByte eMMC
  - RAM: 512 Mbyte DDR2
  - 2 Cameras
- Access via: Serial, ADB (after root)
- Data: Recorded videos on internal memory
  - Cant be deleted if device is broken
Other examples: Door bells

- Many models same design
  - SOC: HI3518
  - Flash: 8MByte SPI NOR Flash
- All devices use JFFS2/UBIFS
- Wi-Fi credential recoverable
- No video due to SD card
Good factory reset implementations

• Vacuum cleaner robot with usage of Trustzone for key storage
  – User data partitions encrypted using LUKS
  – Key managed by TEE and are device specific
  – Unlocking of configuration and user data at boot
  – At factory reset: deletion of key and recreation of partition
Conclusion

• The device “remembers”
• Secure and correct factory reset difficult to implement
  – Use of raw NAND defeats full wipe
  – There is no way to ensure that a device have been wiped
• Many vendors do not erase all user generated data
  – Usage data remains, Logfiles are not erased
  – Wi-Fi configuration files were overwritten, but information remained in other places
• Also: Missing knowledge from the user
Recommendations

• Do not sell or throw away your device
  – If you expect that it may contain sensitive information
  – If you cannot verify a full wipe
• Physically destroy the flash memory
• Use the device to practice soldering ;)
• Change your Wi-Fi credentials or use a separate IoT Wi-Fi

If its broken: why not break it more? ;)

Hint: Does not prevent leakage, but limits attackers access on your network
Acknowledgements

• Prof. Guevara Noubir (Khoury, Northeastern University)
• Prof. John L. Manferdelli (Khoury, Northeastern University)

Secure Mobile Networking (SEEMOO) Labs and CROSSING S1

http://www.ccs.neu.edu/home/noubir/
https://www.seemoo.informatik.tu-darmstadt.de/
Chaos Communication Camp 2019

Dennis Giese

Questions?

Contact:
See: http://dontvacuum.me
Telegram: https://t.me/dgiese
Twitter: dgi_DE
Meet me in Boston/@DC617