Smart Home Security & Privacy
Guest lecture CY 2550 – Dennis Giese
The S in IoT stands for Security.
And the P stands for Privacy.
About me

- PhD student at Northeastern University, USA
  - Working with Prof. Guevara Noubir@Khoury
- Physics and IT Security at TU Darmstadt, Germany

- Worked for >6 Years for Deutsche Telekom/T-Systems CERT
- Interests: Reverse engineering of interesting devices
  - IoT, Smart Locks
  - Physical Locks ;)}
My research

• Years ago:
  – Electronic locks
  – Printer security
  – IP phones

• Now: Smart home devices
  – Multiple vendors (e.g. Samsung, Wink, Vorwerk/Neato)
  – Xiaomi since ~April 2017, completely black box approach
  – Current focus: Amazon Echo & Alexa
Outline

• Part 1: Smart Home Security & Privacy
  (short break)
• Part 2: Security Analysis of the Xiaomi IoT Ecosystem

Disclaimers:
For me IoT and Smart Home is the same
  – I reverse engineer and hack devices
Smart Home Security & Privacy Outline

• Motivation
• IoT devices from a security perspective
• Reverse Engineering Intro
• Privacy Implications
MOTIVATION
Why reverse IoT?

- (Find and exploit bugs to hack other people)
- Detach devices from the vendor
- Enhance functionality
  - Add new features
  - Localization (e.g. Sound files)
  - Defeat Geo blocking
- Supporting other researchers
Mon(IoT)or Lab@NEU
How I started

May 2017
Mi Band 2
Vacuum Robot Gen 1

June 2017
Lumi Smart Home Gateway
+ Sensors

July 2017
Yeelink Lightbulbs (Color+White)
Yeelink LED Strip
How I continued

Yeelink Desk lamp
Philips Eyecare Desk lamp
Xiaomi Wi-Fi router

Yeelink/Philips Ceiling Lights
Philips Smart LED Bulb

Vacuum Robot Gen 2
Yeelink Bedside Lamp
Xiaomi (Ninebot) M365

Lumi Aqara Camera
Yeelink Smart LED Bulb (v2)
Smart Power strip

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
And even more devices
My expensive Hobby

- Today: ~300 devices
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 best path to clean.

Source: Xiaomi advertisement
IOT DEVICES FROM A HACKER PERSPECTIVE
Overview of an IoT Device

- **Firmware**
  - OS
  - Application
  - WiFi/Zigbee

- **Storage**
  - SPI Flash
  - eMMC Flash
  - NAND Flash
  - SD Cards

- **Connectivity**
  - WiFi
  - BT
  - Zigbee

- **Debug Interfaces**
  - UART
  - JTAG/SWD
  - USB

- **SoC**

- **Sensors/Actors**

- **Interfaces**
  - UART
  - JTAG/SWD
  - USB
Overview of an IoT Device
Features and Connectivity

• IoT devices have powerful hardware
  – Multicore CPUs
  – Often based on Linux
  – Very similar to general purpose computers
• IoT devices are connected to other devices and the Internet
  – Smart Home not possible without other devices
  – Most products require Internet connectivity
IoT Hardware: Vacuum Robot

- Quadcore ARM
- 512 Mbyte RAM
- 4 GByte Flash
- Ubuntu OS
IoT Hardware: Smart Speaker

- Quadcore ARM
- 512 Mbyte RAM
- 4 GByte Flash
- Android OS
Let’s talk about Cyber security

• Long time experience and well researched field
  – Lots of security mechanisms in hardware
    • Secure boot, Firmware signatures, Attestation
  • Secure boot, Firmware signatures
  • Best practices applied in industry
    – Security Development Lifecycles
    – Programming guidelines

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
Cybersecurity and IoT

- Cybersecurity is hard
  - Requires knowledge
  - New attacks are developed
  - Third-party code vulnerable

- IoT devices are complex
  - Hardware, Software and Networks
  - More challenges for developers
  - Dependence on internet

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
IoT development cycle

- IoT Vendors/Developers are often lazy
  - Limited development time
  - Fast product development cycles
  - Quality control too expensive
- Assumed development of firmware:
  1. Take SDK/toolchain (e.g. Hi3518)
  2. Modify sample code so that the product runs
  3. If it works: publish firmware ... fix later (or never)
Product support and lifespan

• Development cycle similar to smartphones
  – New products and models every year
  – Product support dropped after 1-2 years
  – Developers can only focus on new products

• Problem: Smart Home devices are used longer
  – Average lifespan of a washing machine: 7-13 years
  – No incentive for customer to replace working device
  – No incentive for vendor to support old devices
How IoT becomes vulnerable

• General problem: Security does not pay
• Customer is not well educated
  – Connects IP cameras directly to the Internet without firewalls
  – Does not change default passwords
  – Is not aware of functionality
• Developer and customer behavior leads to vulnerable devices
  – Example: Mirai Botnet, which abused default credentials
Example: Home Alarm

- Chinese Manufacturer
- Network traffic not encrypted
- Only security: Serial number
  - Devices can be hijacked remotely
- Insecure Wireless protocol
- Easily bypassable
- Firmware vulnerable
- Still: many positive Reviews
Risk of Malicious IoT devices

- Visited Xiaomi Stores in Taiwan
- Lots of IoT devices accessible
  - Demo Phones with App
  - Sharing function possible
  - Devices rootable
Risk of Malicious IoT devices

- What can possibly go wrong?
  - Missing network separation
  - + default passwords
  - + Vulnerable IoT devices
  - = Lots of fun for attackers
  - = Hard time for you!
Fixed by Xiaomi
IOT REVERSE ENGINEERING
Tools

• Very few tools required:
  – Raspberry Pi
  – Soldering iron/Hot air soldering station
  – Multimeter
• Nice to have:
  – Reflow oven
  – Microscope
• If everything fails:
  – $$$ microprobing station
Raspberry Pi

• Very universal tool:
  – JTAG (using OpenOCD)
  – SPI Flash (using Flashrom)
  – UART
  – Mounting of flash images
• Same architecture (ARM) like many IoT devices
Debug Interfaces

- UART
  - serial console output of firmware
  - Interaction with bootloader
- USB/ USB DRD/ ADB
  - access to OS
  - Interaction with bootloader
  - download firmware on device
  - potential boot source
Chip Debugging

• JTAG / SWD
  – Integrated in most ICs
  – Allows debugging of:
    • Registers, memory contents, instructions
    – Used for initial firmware provisioning
  • Useful for us:
    – learning memory layout, dumping firmware
    – extraction of secret keys
Problems with warranty seals?
Example

- Smart Home Gateway JTAG and UART

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

*UART
IOT AND PRIVACY
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
Smart Home Gateways

- Connection log files
- Sensor/actuators bindings
- Sensor/actuators log files
- Key material
  - Z-Wave Keys
  - Cloud bindings
Cameras

- Cached snapshots/video clips
- Recorded video
- Event logs
- Cloud storage credentials
Media players

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

• Many devices do not encrypt traffic at all
  – Especially applies to IP Cameras
  – Passwords can be transmitted in the clear
• Many vendors implement SSL incorrectly
  – SSL certificates not checked
  – Man-in-the-middle (MITM) possible
• Connections to foreign servers
• Servers might be poorly secured
Telemetry

- Nearly all devices transmit telemetry data
  - Independently of control data
  - Most of the times not known to the user / hidden in the ToS
- Servers often in US or China
- Problematic in combination with non encrypted traffic
- Example:
  - Usage data of heating system
  - Logfiles of Smart Home Hubs
  - Smart TVs (which program was viewed, etc.)
  - Washing machines (wash cycles, diagnostic data)
Recovering Data From Devices

Experiment:

1. Disassemble devices and dump flash
2. Powering on devices and root devices (if possible)
3. Connecting devices to the App
4. Using devices and reset them
5. Compare available data before and after reset
Xiaomi/Rockrobo Mi Vacuum Robot

- Used device
- From 2018, unclear condition of device
- 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
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
  - 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
Current research: Amazon Echo Dot*

*Paper under submission
Current research: Amazon Echo Dot*

• Purchased 86 used devices
  – Working, broken and refurbished devices
• “Privacy preserving” forensics
• Result:
  – All (13) broken devices were still provisioned
  – 61% of working devices still provisioned

*Paper under submission
Used IoT Devices conclusions

• 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 has 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
SUMMARY
Summary

• IoT devices are powerful and complex
  – Development is expensive
  – Likelihood of vulnerabilities is high

• Product lifecycle and lifespan differ
  – Product lifecycle similar to smart phones
  – Lack of updates after a short time, Products remain vulnerable

• Secure products do not mean more profit
  – Security low priority for vendors
  – Customer does not want to pay for it

• IoT devices may contain and collect lot of data

• Insecure Servers risk private data
End of Part 1: Questions?
Security Analysis of the Xiaomi IoT Ecosystem Outline

• Introduction
• Methodology
• Analysis of Mi Home App
• Analysis of Devices
• Discussion
• Conclusion
Outline

➢ Introduction
• Methodology
• Analysis of Mi Home App
• Analysis of Devices
• Discussion
• Conclusion
Introduction
The Xiaomi Ecosystem

• Xiaomi mostly known for Smartphones (4th worldwide)
• They claim to have the biggest IoT ecosystem worldwide
  – 310 Million Devices, 2400 different models (January 2021)
• Different Vendors, one ecosystem
  – named „Mijia“
  – Same communication protocol
  – Different technologies supported
  – Implementation differs from manufacturer to manufacturer
    • Software quality very different
    • Custom features added to firmware
Introduction

Products
Introduction

Communication relations

- BLE
- Wi-Fi
- Cloud Protocol (Wi-Fi)
- Xiaomi Cloud

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
Introduction

Different vendors in one ecosystem

- ~2400 different models supported (Wi-Fi + Zigbee + BLE)
- Depending on selected server location
  - Mainland China
  - Singapore (Worldwide)
  - Russia
  - US
  - Germany (Europe)
  - India
- models might be region-blocked

Values estimated, Mi Home 5.6, Mainland China Server
Introduction

Motivation

• IoT devices have high impact in the daily life
  – Smart home devices gain more importance and are common
  – Devices have much computation power
  – IoT means that devices are connected to the Internet
  – Devices may collect much private data
  – However: User cannot inspect functionality of the device
• Xiaomi Ecosystem is a good target for security analysis
  – Due to market share impact on many customers
  – Many different implementations can have security vulnerabilities
  – Same protocol makes knowledge transferable to many devices
    • Mijia SDK is shared for all the devices
Introduction

Goals

• Research question: How secure is the implementation of the ecosystem of the IoT market leader Xiaomi?

• Subgoals:
  – Analyze and understand functionality
  – Find potential vulnerabilities
  – Analyze the impact on the users privacy
  – Enable users to take control over their own devices

• Focus: ARM based devices with Wi-Fi
Outline

✓ Introduction
➢ Methodology
  • Analysis of Mi Home App
  • Analysis of Devices
  • Discussion
  • Conclusion
Methodology

Approaches: App

Wi-Fi

BLE

HTTPS

Xiaomi Cloud

Mi Home App
Reverse engineering & analysis
App Reverse Engineering

- Idea: Understand interaction between app and phone, and app and cloud
- Advantage: device data is displayed inside the app -> app needs to know how to interpret it
- Methods:
  - Disassembly: Jadx (APK to Java)
  - Modification: Apktool (APK to smallicode, rebuilding)
  - Monitoring: Logcat (monitoring Android log files)
  - Interception: Xposed framework (modifying flows while execution)
Methodology

How we stay undetected?

• Multiple smartphones/tablets
  – Different Xiaomi accounts
  – Different server location
  – Spoofed GPS coordinates

• Wi-Fi Network
  – Separate Wi-Fi access points
  – VPNs to Hong Kong, China
  – TOR

• No mixture between different accounts and devices
Methodology

Approaches: Devices

HTTPS

Cloud Protocol (Wi-Fi)

Wi-Fi

Xiaomi Cloud

Device reverse engineering

ZigBee

Cloud Protocol (Wi-Fi)

Wi-Fi
Methodology

Device Reverse Engineering

• Idea: Understand function and design of devices (physical hardware)
• Advantage: Data can be obtained directly from the device, transport encryption can be avoided
• Methods (Workflow):
  – Retrieving firmware before purchasing
  – Disassembly of the device and PCB analysis
  – Identification of platform and components
  – Desoldering flash and dumping contents
  – Network traffic analysis
  – Obtaining root access
  – Verify collected user information on devices
Methodology

Device Procurement and Selection

- ARM based devices mit Wi-Fi
- Multiple devices for each model
  - One reference
  - One to disassemble and root
- Selection by usefulness and size
  - No fridges, washing machines, ...

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
Methodology

Comparison

App

• App can be downloaded for free
• Requires Cloud interaction -> legal issues
• Information can be obtained for a large number of models
• Analysis reveals vulnerabilities in cloud APIs
• Vulnerabilities can be fixed by the cloud provider easily

Devices

• Requires procurement of devices
• Any attack can be done (even destructive ones)
• Information is valid for a specific set of models
• Analysis reveals vulnerabilities on devices
• Vulnerabilities can be fixed by firmware updates from the vendor, which requires user interaction
Comparison

**App**
- App can be downloaded for free
- Requires Cloud interaction -> legal issues
- Information can be obtained for a large number of models
- Analysis reveals vulnerabilities in cloud APIs
- Vulnerabilities can be fixed by the cloud provider easily

**Devices**
- Requires procurement of devices
- Any attack can be done (even destructive ones)
- Information is valid for a specific set of models
- Analysis reveals vulnerabilities on devices
- Vulnerabilities can be fixed by firmware updates from the vendor, which requires user interaction

Preferred method
Outline

✓ Introduction
✓ Methodology
➢ Analysis of Mi Home App
  • Analysis of Devices
  • Discussion
  • Conclusion
Analysis of App

Mi Home App (Android)

• App partially obfuscated, usage of native libraries
• Device specific functions: provided by Plugins (APK or JS-Bundles)
• Communication to cloud:
  – Authentication via OAuth
  – Layered encryption
    • Outside: HTTPS
    • Inside: AES using a session key
  – Message format: JSON RPC

➢ Contribution: PHP implementation of App to Cloud API
Analysis of App

App Structure

Core functionality:
- Device add, delete
- Device list
- Device Updates via Cloud
- Plugin Management
- Permission Management

Device specific functionality:
- Device commands
- Map download and presentation

Example Endpoints:
/home/device_list {“ssid”: “…”, bssid: “…”}
/home/multi_checkversion {"dids":[]} 
/home/devupgrade {"did":"..."}
/device/devicepass {"did":"..."}

Endpoints:
/home/rpc/#did# {“method”: ...}

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
Analysis of App

Example of intercepted cloud api call

• REQ: api.io.mi.com/home/device_list method:POST params:[]

• RES:

  {"message":"ok","result":{"list":[{"did":"659812bc...zzz","name":"Mi PlugMini","localip":"192.1"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}
Analysis of App

Example of intercepted cloud api call

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

Source: Openstreetmaps

Works only on “Mainland China”
Analysis of App

App handling of user permission

• Plugin determines permission based on flags

```
"adminFlag":1,"shareFlag":0,"permitLevel":16
```

- User is owner of device

- Device is not shared

- Privilege level (device dependent)

  - User can update firmware, set settings, share device, etc
Analysis of App

App handling of user permission

• Plugin determines permission based on flags

```
"adminFlag":0,"shareFlag":1,"permitLevel":4, "uid": 123
```

User not owner of device

Device is shared

Privilege level (device dependent)

– User can only view device, other options are not visible
Analysis of App

App to Device via Cloud RPC

• Cloud Server is not aware of allowed commands via RPC
  – APP/Plugin can send arbitrary commands to device
  – Commands not filtered
  – Privileges not checked by Cloud

• Attack vector: push firmware updates
  – Works also for shared devices and without privileges

➢ Contribution: found attack vector useful feature for further device analysis

Please do not distribute this information yet
Analysis of App

Device management

- App retrieves JSON file with all supported devices
  - List acts as a whitelist
  - List depends on region and permission
- Devices detected via Wi-Fi SSID format
- Required for device provisioning: Wi-Fi credentials, UserID, Token

➢ Contribution:
  - List for collecting information about new devices and features
  - Collection of historic information (2017-2019: 3600 devices)
  - Add devices to unsupported regions
Outline

✓ Introduction
✓ Methodology
✓ Analysis of Mi Home App
➢ Analysis of Devices
• Discussion
• Conclusion
Analysis of Devices
Deeper Look at Communication relations

Robot intern

compass \textup{uart}_lds \textup{uart}_mcu

\begin{itemize}
\item player
0.0.0.0:6665
\item wifimgr
\item RoboController
\item AppProxy
\end{itemize}

Miio client
\begin{itemize}
\item (local):54322 (tcp)
\item 0.0.0.0:54321 (udp)
\end{itemize}

\texttt{device.conf}: DID, Key, MAC

\texttt{*.fds.api.xiaomi.com} (https)

\texttt{*.api.io.mi.com} (https)

\begin{itemize}
\item ot.io.mi.com:80(tcp)
\item ot.io.mi.com:8053(udp)
\end{itemize}

\text{AES encrypted}

\text{Android/ iPhone App}

\begin{itemize}
\item IPC
\item plain json (tcp)
\item enc(key) json (tcp/udp)
\item enc(token) json (udp)
\end{itemize}
Analysis of Devices

Device to Cloud Communication

• DeviceID
  – Unique per device

• Keys
  – Cloud key (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 Wi-Fi)
Analysis of Devices

Network Setup

IoT Device under test
IoT Device under test
IoT Device under test

WLAN Router
WLAN Interface
Management Interface
Monitoring Computer

Packet capture
Firewall
WAN Interface

Internet
Firmware retrieval

- Dumping Flash memory
  - JTAG, SWD or desolder Flash
  - Helpful tool: Raspberry Pi with OpenOCD and flashrom
- Intercepting traffic while Firmware Update
  - It is advised to actually block the Update
    - Sneaky: If DNS fails then direct IP is used
      - If SSL is used: so far a fake certificate worked 😊
      - Goal: Retrieve special URL for Firmware update
Analysis of Devices

Firmware downloads

- Filenames not easy guessable
- CDN is using URL authentication

http://cdn.cnbj0.fds.api.mi-img.com/miio-fw/

063df95bd538a9cfa22c7c8664XXXXXX_upd_lumi.gateway.v3.bin?

GalaxyAccessKeyId=5721718111234&Expires=1539055099000&Signature=KtIxawkpAdggz3IEuu6ygXXXXX==&uniqRequestId=21234123
Analysis of Devices

How to get Firmwares?

• Problem: Retrieving Firmware is difficult
  – Need of owning the device

• Easy solution: Impersonating devices
  – Model ID initially not fixed in cloud backend -> we can modify it (per region)
  – On rooted device:
    • change model, modify version number to “0.0.0”
    • trigger firmware update from smart phone app
    • Get authenticated firmware URL 😊

➢ Contribution: collection of firmware versions over a long time (2018-2019: 870)
  – Sharing with other researcher for development of open source implementations
# Analysis of Devices

## Collection of firmwares and device info

```plaintext
modelname: roborock.vacuum.s5
pid: 0
feat_bt_gateway: 0
feat_mesh_gateway: 0
hasBT: -1
hasWiFi: 1
has5GWiFi: 0
hasZigbee: -1
OS: Ubuntu 14.04
RAM: 512MByte
FLASH: 4GByte eMMC
SOC: Allwinner R16
MCU: STM32F103VCT6
SOC-ARCH: ARM Cortex-A7 (4x)
MCU-ARCH: ARM Cortex-M3
WiFi-Chipset: RTL8189ETV
FW-Format: dd image AES encrypted (ccrypt, key: rockrobo)

<table>
<thead>
<tr>
<th>Region</th>
<th>cn</th>
<th>de</th>
<th>ru</th>
<th>sg</th>
<th>us</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>MD5</th>
<th>Filename</th>
<th>Version</th>
<th>Datetime</th>
<th>Regions</th>
</tr>
</thead>
<tbody>
<tr>
<td>app</td>
<td>9e2c0809cebe892e60e6723b30d760f6</td>
<td>v11_001768.fullos.pkg</td>
<td>3.3.9_001768</td>
<td>2019-03-27 11:57:00</td>
<td>cn,de,ru,sg,us</td>
</tr>
<tr>
<td>app</td>
<td>e76f406b6717d9b7ea1cebeb48f3a8</td>
<td>v11_001720.fullos.pkg</td>
<td>3.3.9_001720</td>
<td>2019-05-23 02:23:00</td>
<td>de,sg,us</td>
</tr>
<tr>
<td>app</td>
<td>3d04e386856129a0c0a9508e40e577b7</td>
<td>v11_001864.fullos.lnm09e8u2.pkg</td>
<td>3.3.9_001864</td>
<td>2019-05-31 05:51:00</td>
<td>cn,de,ru,sg</td>
</tr>
<tr>
<td>aplugin</td>
<td>771d4cfe186aae8757a27e12d04d88</td>
<td>com.roborock.rubys.app_2019061715280736461.zip</td>
<td>188</td>
<td>2019-06-17 07:28:00</td>
<td>de,sg,us</td>
</tr>
<tr>
<td>aplugin</td>
<td>e596b0f89b58fecfbd26b829849d4</td>
<td>com.roborock.rubys.app_20190614482451501.zip</td>
<td>191</td>
<td>2019-06-24 06:48:00</td>
<td>cn</td>
</tr>
</tbody>
</table>
```

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
## Analysis of Devices

### Devices under test

- **21 models selected for test**
  - Different regions
  - Different versions

<table>
<thead>
<tr>
<th>Device name</th>
<th>Region</th>
<th>Mijia model</th>
<th>Vendor</th>
<th>Release</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aqara Gateway (Homekit)</td>
<td>CN</td>
<td>lumi.gateway.aqhm01</td>
<td>Lumi</td>
<td>Q2 2018</td>
<td>50</td>
</tr>
<tr>
<td>Aqara Gateway (Homekit)</td>
<td>US</td>
<td>lumi.gateway.aqhm02</td>
<td>Lumi</td>
<td>Q1 2019</td>
<td>50</td>
</tr>
<tr>
<td>Aqara Smart Home Gateway</td>
<td>TW</td>
<td>lumi.gateway.mitw01</td>
<td>Lumi</td>
<td>Q1 2018</td>
<td>35</td>
</tr>
<tr>
<td>Aqara Smart IP Camera</td>
<td>CN</td>
<td>lumi.camera.aq1</td>
<td>Lumi</td>
<td>Q4 2017</td>
<td>35</td>
</tr>
<tr>
<td>Lumi Smart Home Gateway</td>
<td>CN</td>
<td>lumi.gateway.v3</td>
<td>Lumi</td>
<td>Q3 2016</td>
<td>30</td>
</tr>
<tr>
<td>Philips Ceiling Lamp</td>
<td>CN</td>
<td>philips.light.ceiling</td>
<td>Yeelight</td>
<td>Q2 2017</td>
<td>70</td>
</tr>
<tr>
<td>Roborock S50</td>
<td>EU</td>
<td>roborock.vacuum.s5</td>
<td>Roborock</td>
<td>Q1 2018</td>
<td>400</td>
</tr>
<tr>
<td>Roborock S50</td>
<td>CN</td>
<td>roborock.vacuum.s5</td>
<td>Roborock</td>
<td>Q4 2017</td>
<td>350</td>
</tr>
<tr>
<td>Roborock T61</td>
<td>CN</td>
<td>roborock.vacuum.t6</td>
<td>Roborock</td>
<td>Q1 2019</td>
<td>450</td>
</tr>
<tr>
<td>Roborock S61</td>
<td>EU</td>
<td>roborock.vacuum.s6</td>
<td>Roborock</td>
<td>Q1 2019</td>
<td>550</td>
</tr>
<tr>
<td>Xiaomi Mi Vacuum Robot</td>
<td>CN</td>
<td>rockrobo.vacuum.v1</td>
<td>Roborock</td>
<td>Q4 2016</td>
<td>280</td>
</tr>
<tr>
<td>Xiaomi Mi WiFi Speaker</td>
<td>CN</td>
<td>xiaomi.wifispeaker.v1</td>
<td>Xiaomi</td>
<td>Q4 2016</td>
<td>85</td>
</tr>
<tr>
<td>Xiaomi WiFi Plug</td>
<td>CN</td>
<td>chuangmi.plugin.m1</td>
<td>Chuangmi</td>
<td>Q2 2016</td>
<td>15</td>
</tr>
<tr>
<td>Yeelink Bedside lamp</td>
<td>CN</td>
<td>yeelink.light.bslamp1</td>
<td>Yeelight</td>
<td>Q4 2017</td>
<td>25</td>
</tr>
<tr>
<td>Yeelink Bedside lamp</td>
<td>TW</td>
<td>yeelink.light.bslamp1</td>
<td>Yeelight</td>
<td>Q1 2018</td>
<td>30</td>
</tr>
<tr>
<td>Yeelink Ceiling Lamp</td>
<td>CN</td>
<td>yeelink.light.ceiling1</td>
<td>Yeelight</td>
<td>Q3 2017</td>
<td>65</td>
</tr>
<tr>
<td>Yeelink Light Color</td>
<td>CN</td>
<td>yeelink.light.color1</td>
<td>Yeelight</td>
<td>Q4 2016</td>
<td>10</td>
</tr>
<tr>
<td>Yeelink Light Mono1</td>
<td>CN</td>
<td>yeelink.light.mono1</td>
<td>Yeelight</td>
<td>Q4 2016</td>
<td>10</td>
</tr>
<tr>
<td>Yeelink Light Strip</td>
<td>CN</td>
<td>yeelink.light.strip1</td>
<td>Yeelight</td>
<td>Q4 2016</td>
<td>15</td>
</tr>
<tr>
<td>Yeelink Smart White Bulb</td>
<td>EU</td>
<td>yeelink.light.ct2</td>
<td>Yeelight</td>
<td>Q2 2018</td>
<td>15</td>
</tr>
<tr>
<td>Yeelink Smart RGB Bulb</td>
<td>EU</td>
<td>yeelink.light.color2</td>
<td>Yeelight</td>
<td>Q2 2018</td>
<td>15</td>
</tr>
</tbody>
</table>
### Analysis of Devices

#### Devices under test

- 21 models selected for test
  - Different regions
  - Different versions

<table>
<thead>
<tr>
<th>Device name</th>
<th>Region</th>
<th>Mijia model</th>
<th>Vendor</th>
<th>Release</th>
<th>Price (USD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aqara Gateway (Homekit)</td>
<td>CN</td>
<td>lumi.gateway.aqhm01</td>
<td>Lumi</td>
<td>Q2 2018</td>
<td>50</td>
</tr>
<tr>
<td>Aqara Gateway (Homekit)</td>
<td>US</td>
<td>lumi.gateway.aqhm02</td>
<td>Lumi</td>
<td>Q1 2019</td>
<td>50</td>
</tr>
<tr>
<td>Aqara Smart Home Gateway</td>
<td>TW</td>
<td>lumi.gateway.mitw01</td>
<td>Lumi</td>
<td>Q1 2018</td>
<td>35</td>
</tr>
<tr>
<td>Aqara Smart IP Camera</td>
<td>CN</td>
<td>lumi.camera.aq1</td>
<td>Lumi</td>
<td>Q4 2017</td>
<td>35</td>
</tr>
<tr>
<td>Lumi Smart Home Gateway</td>
<td>CN</td>
<td>lumi.gateway.v3</td>
<td>Lumi</td>
<td>Q3 2016</td>
<td>30</td>
</tr>
<tr>
<td>Philips Ceiling Lamp</td>
<td>CN</td>
<td>philips.light.ceiling</td>
<td>Yeelight</td>
<td>Q2 2017</td>
<td>70</td>
</tr>
<tr>
<td>Roborock S50</td>
<td>EU</td>
<td>roborock.vacuum.s5</td>
<td>Roborock</td>
<td>Q1 2018</td>
<td>400</td>
</tr>
<tr>
<td>Roborock S50</td>
<td>CN</td>
<td>roborock.vacuum.s5</td>
<td>Roborock</td>
<td>Q4 2017</td>
<td>350</td>
</tr>
<tr>
<td>Roborock T61</td>
<td>CN</td>
<td>roborock.vacuum.t6</td>
<td>Roborock</td>
<td>Q1 2019</td>
<td>450</td>
</tr>
<tr>
<td>Roborock S61</td>
<td>EU</td>
<td>roborock.vacuum.s6</td>
<td>Roborock</td>
<td>Q1 2019</td>
<td>550</td>
</tr>
<tr>
<td>Xiaomi Mi Vacuum Robot</td>
<td>CN</td>
<td>rockrobo.vacuum.v1</td>
<td>Roborock</td>
<td>Q4 2016</td>
<td>280</td>
</tr>
<tr>
<td>Xiaomi Mi WiFi Speaker</td>
<td>CN</td>
<td>xiaomi.wifispeaker.v1</td>
<td>Xiaomi</td>
<td>Q4 2016</td>
<td>85</td>
</tr>
<tr>
<td>Xiaomi WiFi Plug</td>
<td>CN</td>
<td>chuangmiplug.m1</td>
<td>Chuangmi</td>
<td>Q2 2016</td>
<td>15</td>
</tr>
<tr>
<td>Yeelink Bedside lamp</td>
<td>CN</td>
<td>yeelink.light.bslamp1</td>
<td>Yeelight</td>
<td>Q4 2017</td>
<td>25</td>
</tr>
<tr>
<td>Yeelink Bedside lamp</td>
<td>TW</td>
<td>yeelink.light.bslamp1</td>
<td>Yeelight</td>
<td>Q1 2018</td>
<td>30</td>
</tr>
<tr>
<td>Yeelink Ceiling Lamp</td>
<td>CN</td>
<td>yeelink.light.ceiling1</td>
<td>Yeelight</td>
<td>Q3 2017</td>
<td>65</td>
</tr>
<tr>
<td>Yeelink Light Color</td>
<td>CN</td>
<td>yeelink.light.color1</td>
<td>Yeelight</td>
<td>Q4 2016</td>
<td>10</td>
</tr>
<tr>
<td>Yeelink Light Mono1</td>
<td>CN</td>
<td>yeelink.light.mono1</td>
<td>Yeelight</td>
<td>Q4 2016</td>
<td>10</td>
</tr>
<tr>
<td>Yeelink Light Strip</td>
<td>CN</td>
<td>yeelink.light.strip1</td>
<td>Yeelight</td>
<td>Q4 2016</td>
<td>15</td>
</tr>
<tr>
<td>Yeelink Smart White Bulb</td>
<td>EU</td>
<td>yeelink.light.ct2</td>
<td>Yeelight</td>
<td>Q2 2018</td>
<td>15</td>
</tr>
<tr>
<td>Yeelink Smart RGB Bulb</td>
<td>EU</td>
<td>yeelink.light.color2</td>
<td>Yeelight</td>
<td>Q2 2018</td>
<td>15</td>
</tr>
</tbody>
</table>
Analysis of Devices

Mi Vacuum Cleaning Robot (Gen1)

Source: Xiaomi advertisement

Laser Distance Sensor (LDS)
Wall Sensor
Dustbin Sensor
Collision Sensor
Ultrasonic Radar Sensor

Electronic Compass
Cliff Sensor
Gyroscope / Accelerometer
Drop Sensor
Speedmeter
Fan Speed Sensor

Research in cooperation with Daniel Wegemer
Analysis of Devices

Mi Vacuum Cleaning Robot

• Released 2016
• Hardware:
  – Quadcore ARM SOC
  – 512 MB DDR3 RAM
  – 4GB eMMC Flash
• OS: Ubuntu 14.04
• Protections:
  – Firmware encrypted, debug ports require authentication
Analysis of Devices

Frontside layout mainboard

- 512 MB RAM
- R16 SOC
- 4GB eMMC Flash
- WiFi Module
- STM32 MCU
Analysis of Devices

Backside layout mainboard

STM UART (921600 baud)

R16 UART (115200 baud)

LIDAR UART
Analysis of Devices

Gaining Root access

• Approach: Fault injection on eMMC flash to enable BOOTROM
  – Usage of aluminum foil to shortcut data pins under the BGA chip
  – Uploading of custom tool via USB and dumping flash
  – Modification and rewriting flash content

• Analysis of firmware and extraction of keys
  – Usage of IDA Pro to extract firmware encryption keys
  – Developing tools for custom firmware and message decryption

➢ Contribution: First published rooting method, description of functions and hardware, reverse engineered data formats and cloud protocol
  – Current usage of rooted vacuum cleaners > 30000
  – Used by researchers for 5G and Wi-Fi experiments, teaching robotics students
Analysis of Devices
Aluminium fault injection attack

➢ First use of aluminum foil to trigger bootloaders on BGA chips
  – Cheap and simple method
  – Reduced risk in comparison to BGA soldering
Analysis of Devices

Available data on device

• Data
  – Logfiles (syslogs, stats, Wi-Fi credentials)
  – Maps
• Data is uploaded to cloud
• Wi-Fi reset
  – Does not delete data: maps, logs still exist
  – Only Wi-Fi credentials are removed, however still exist in logs
• Factory reset
  – Formats user data partition, but is partially recoverable

➢ Contribution: Documentation of the usage and collection of data

~100 Gbyte writes per Year
Analysis of Devices

Available data on device

• Maps
  – Created by player
  – 1024px * 1024px
  – 1px = 5cm

➢ Contribution: Tools for map interpretation
  – Base for all open source implementations
Analysis of Devices

Custom mod of Gen1

- Custom mod enables usage of bigger software (e.g. ROS)
Analysis of Devices

Roborock S50 (Gen2)

- Released 2018
- Same hardware and software base as Mi Robot Vacuum
  - Improvements in software
  - Supports mopping of floors
  - Small hardware modifications
- Same firmware keys as Gen1
Analysis of Devices

Frontside layout mainboard (Gen2)
Analysis of Devices

Introduced Countermeasures in Gen2

• Encrypting/Obfuscating the log-files and maps
• RRlogd uses AES encryption functions from OpenSSL library
  – Imported as dynamic library
  – Interesting function: EVP_EncryptInit_ex(...) 
  – Ltrace can be used to intercept calls and extract arguments

➢ Contribution: AES128CBC-key: “RoCKR0B0@BEIJING”, documentation of firmware, backporting features to Gen1
Analysis of Devices

Roborock S60/T60

• Released 2019
• Same hardware and software base as S50
  – Improvements in software
  – Supports multiple floors
  – Small hardware modifications
• Firmware keys were changed
• Local OTA updates are blocked
• Firmware and configuration is now signed
• Region lock enforced
Analysis of Devices

Roborock S60/T60 UART setup
Analysis of Devices

Roborock S60/T60 UART setup

• Roborock did not fix a vulnerability in U-Boot
  – Root password derivation mechanism remained the same
  – Login over UART possible, however watchdog triggers
  – Watchdog can be disabled in a racing condition

• Firmware is now signed and encrypted
  – Encryption keys and signature public keys obfuscated

➢ Contribution: Extraction of new encryption keys, development of new rooting method, development of automatic tool
## Analysis of Devices

### All results

<table>
<thead>
<tr>
<th>Device name</th>
<th>Debug Interfaces</th>
<th>Firmware</th>
<th>Network</th>
<th>Physical</th>
<th>Data</th>
<th>User data not on device~</th>
<th>Secure unprovisioning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UART</td>
<td>JTAG/SWD</td>
<td>Telnet/SSH</td>
<td>Encrypted</td>
<td>Signed</td>
<td>Verified</td>
<td>HTTPS</td>
</tr>
<tr>
<td>Aqara Gateway (Homekit)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Aqara Smart Home Gateway</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Aqara Smart IP Camera</td>
<td>✓</td>
<td>x</td>
<td></td>
<td></td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Lumi Smart Home Gateway</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Philips Ceiling Lamp</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Roborock S50</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Roborock S6/T61</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Xiaomi Mi Vacuum Robot</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Xiaomi Mi WiFi Speaker</td>
<td>✓</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Xiaomi WiFi Plug</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Bedside lamp</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Ceiling Lamp</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Light Color</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Light Mono1</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Light Strip</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Smart White Bulb</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Yeelink Smart RGB Bulb</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>
Outline

✓ Introduction
✓ Methodology
✓ Analysis of Mi Home App
✓ Analysis of Devices
➢ Discussion
• Conclusion
Discussion

Mi Home App

• Xiaomi puts effort in securing the API and the APP
• Reported vulnerabilities were fixed
• Apps and Plugins are updated on a regular base
• However:
  – Functionality seems more important than security
  – Plugins by vendors introduce new risks
  – Historically grown ecosystem leaves many deprecated APIs
  – Too much trust in the security of the app, missing checks in the cloud
Discussion

Devices

- Xiaomi SDK enables secure communication with the cloud
  - confidentiality, integrity, and availability ensured by design (as long as device specific keys are not leaked)
- Implementations of vendors vary in quality, many contain vulnerabilities
- Vendors try to lock out users and try to restrict devices in a region
- User data is not stored securely, factory resets are not sufficiently done
- Unprovisioned devices are vulnerable due to missing firmware signature and verifications
  - Linux version: cannot detect if a OTA update is pushed from cloud or from local network
  - However: enables user to gain access on their own devices
- Developers lack knowledge of secure implementations of features
- Development time seems to be limited: many firmwares with debug symbols
Outline

✓ Introduction
✓ Methodology
✓ Analysis of Mi Home App
✓ Analysis of Devices
✓ Discussion
➢ Conclusion
Conclusion

Contributions

• Describing Mijia Ecosystem and API
• Analysis and documentation of many different devices
• Development and publication of rooting methods
  – Custom firmwares for all analyzed devices
• Analysis of data usage and life-cycle
• (For Cortex-M based devices: ported Nexmon framework to Marvel and Mediatek based Xiaomi devices)
Conclusion

Key findings

• Legacy API and design in Mi Home App enables unrestricted access
• Missing filtering and permission checking of commands in cloud
• Non Cortex-M devices leave sensitive information after factory reset
• Many devices do not implement HTTPS correctly
• Firmware signatures are rare
• Broken firmware verifications
• All devices have some kind of vulnerabilities
  – Enables user to take control over own device
  – Leaves risk of remote attackers
• In discussions with vendors: missing understanding for risks
Conclusion

Answering the research question

Research question: How secure is the implementation of the ecosystem of the IoT market leader Xiaomi?

- Mijia devices have less interfaces, therefore a smaller attack surface
- Xiaomi puts effort in security and privacy, but there are fundamental issues in the design of the app and APIs
- While the SDK is secure, the additional implementations of vendors introduce vulnerabilities
- Compared to other ecosystems in the same market segment, the implementations are more secure
- Security is often limited by pricing or knowledge constrains
- Rooting methods enable the users to verify security and privacy themselves
IoT is when your AI powered toaster is mining bitcoin to pay off the gambling debts it has with your fridge.

Contact:
See: http://dontvacuum.me
Telegram: https://t.me/dgiese
Twitter: dgi_DE
Email: dgiese@ccs.neu.edu
Short intro

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

(06.04.2021) Smart Home Security & Privacy – Dennis Giese
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

Block to be changed

Data still present

Read block

Write block

Simplified!
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
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