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 ;)

Northeastern University
Khoury College of Computer Sciences

TECHNISCHE UNIVERSITÄT DARMSTADT

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

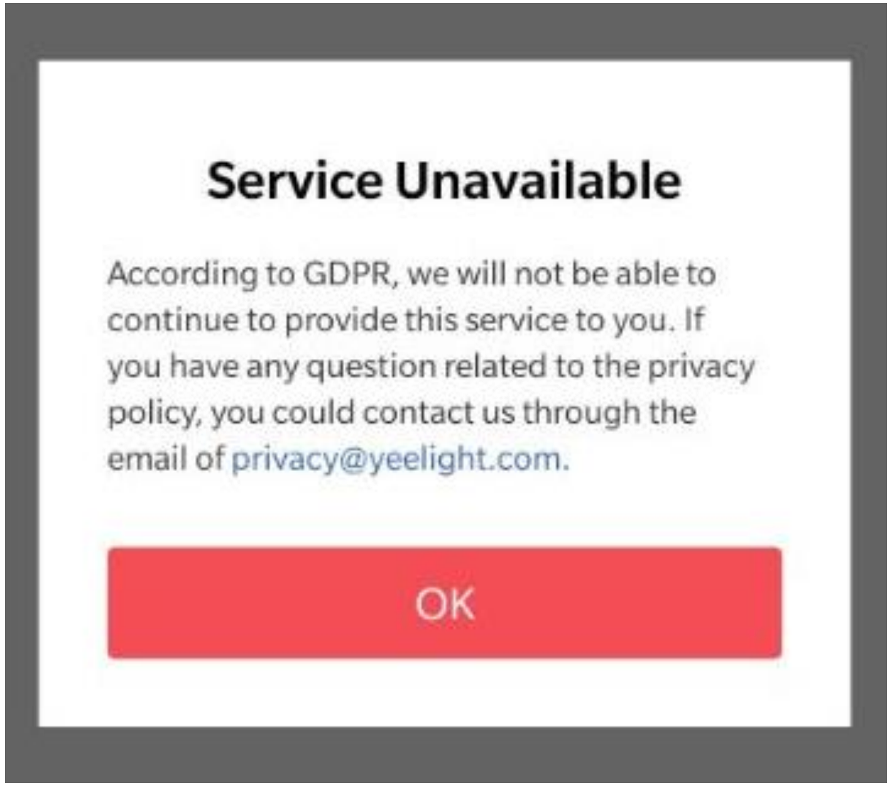
Smart Home Security & Privacy Outline

- 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

https://moniotrlab.ccis.neu.edu/

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

Research in cooperation with Daniel Wegemer

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

And even more devices

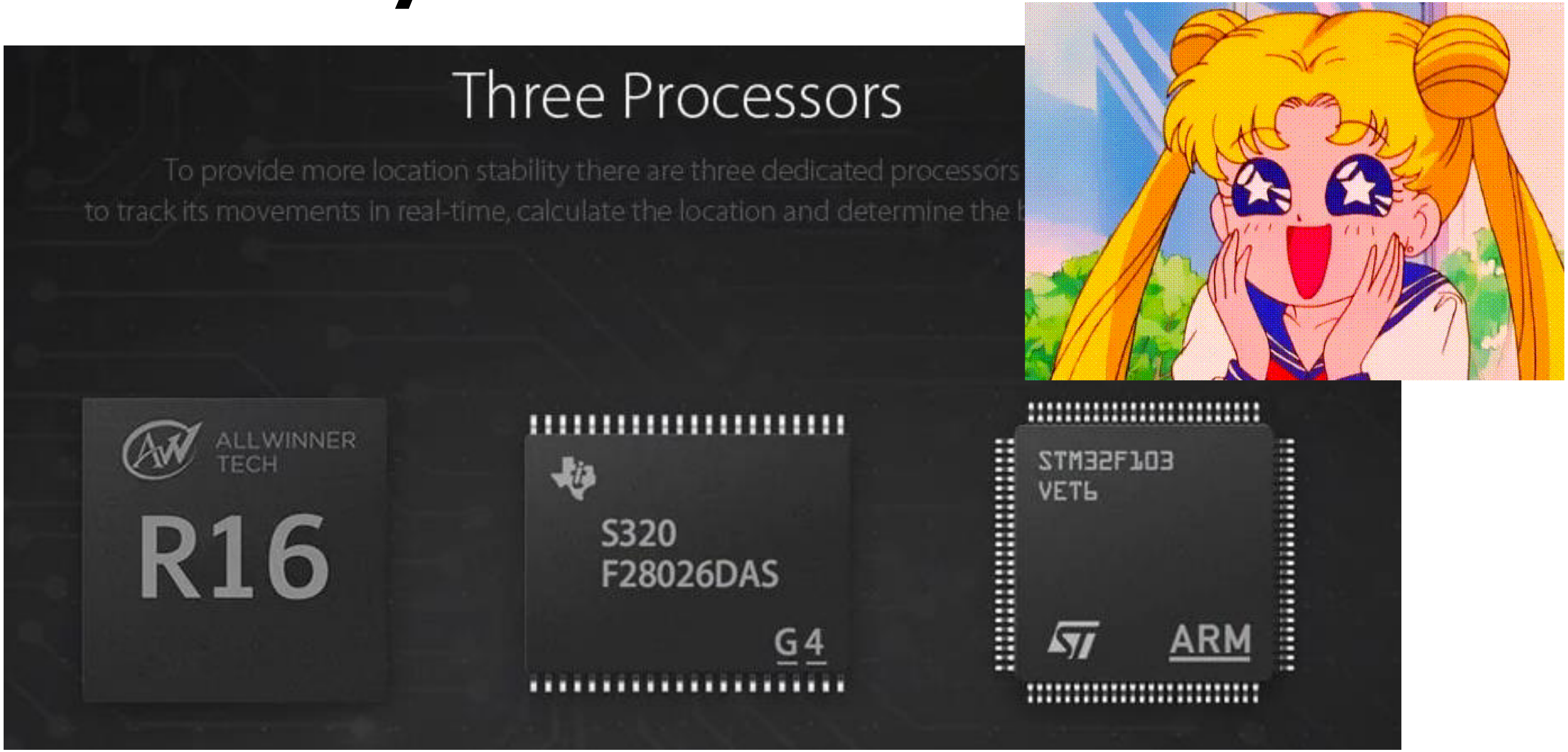





My expensive Hobby

- Today: ~300 devices

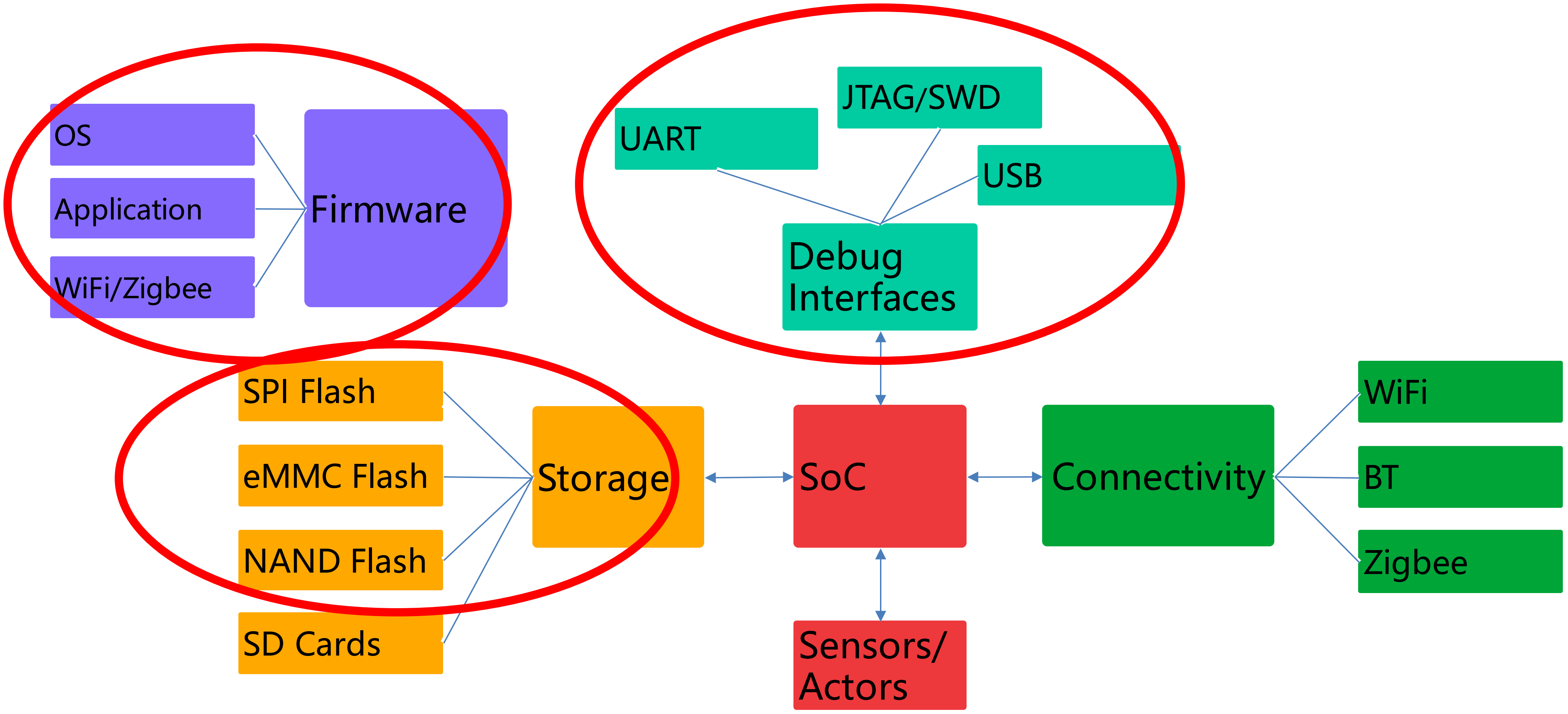
Why Vacuum Robots?



Source: Xiaomi advertisment

IOT DEVICES FROM A HACKER PERSPECTIVE

Overview of an IoT Device



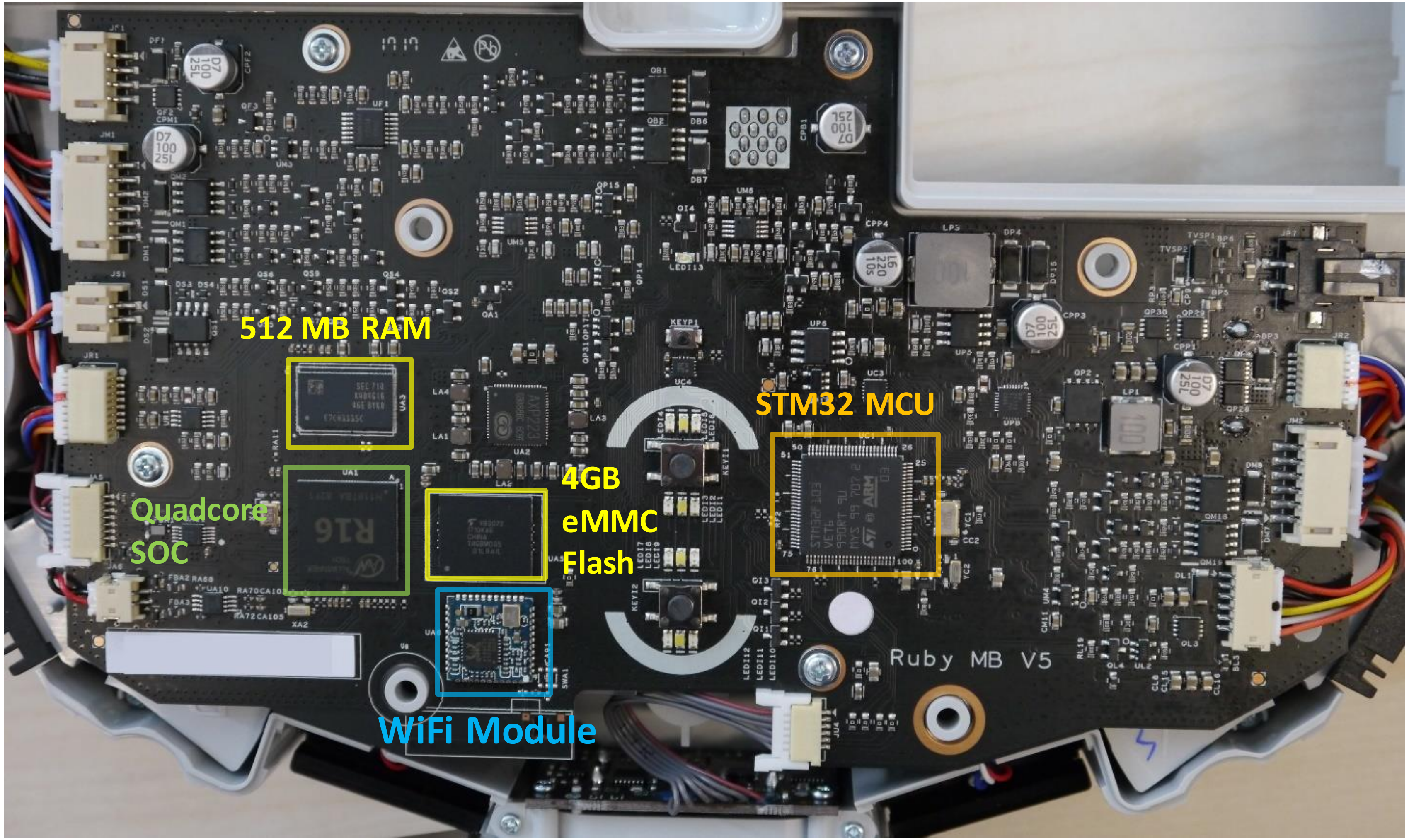
Overview of an IoT Device

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

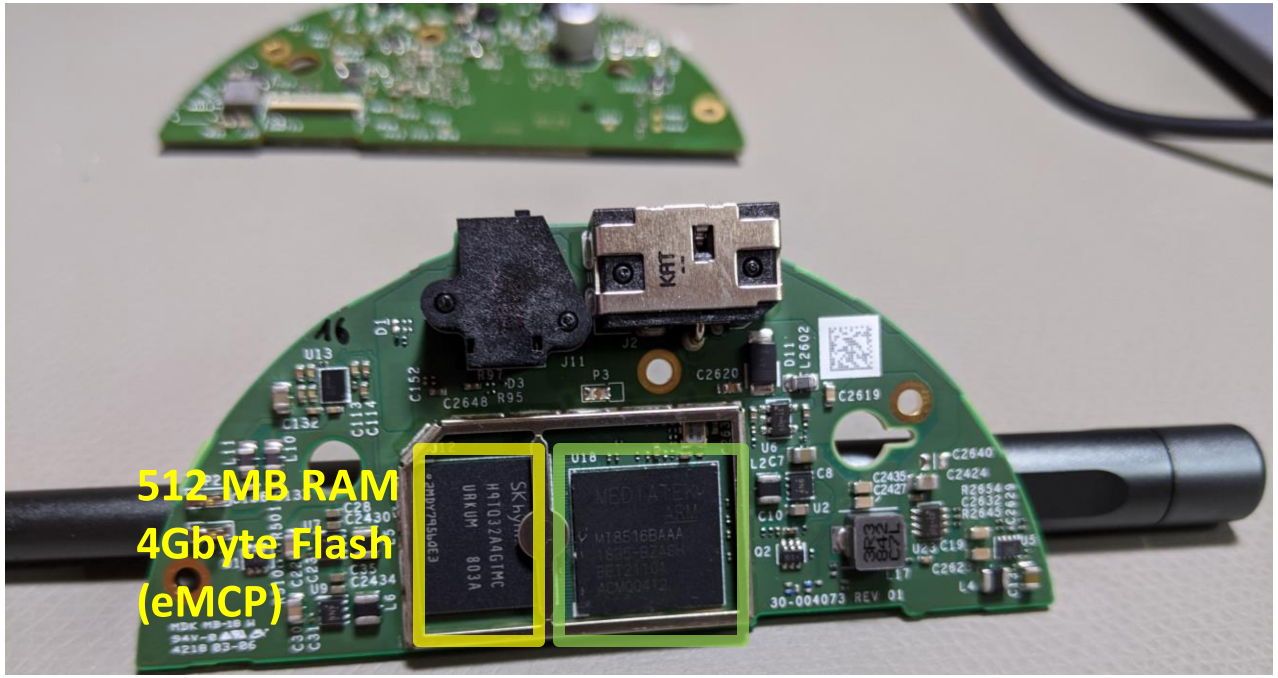
IoT Hardware: Vacuum Robot

- Quadcore ARM
- 512 Mbyte RAM
- 4 GByte Flash
- Ubuntu OS



IoT Hardware: Smart Speaker

- Quadcore ARM
- 512 Mbyte RAM
- 4 GByte Flash
- Android OS

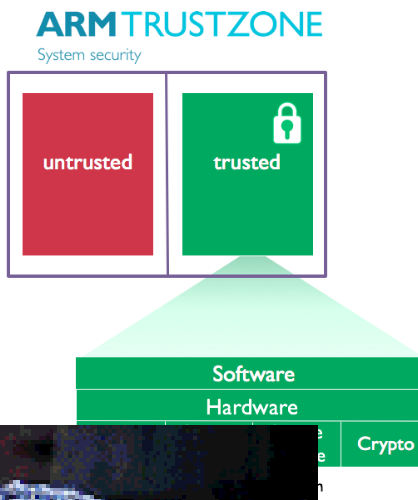


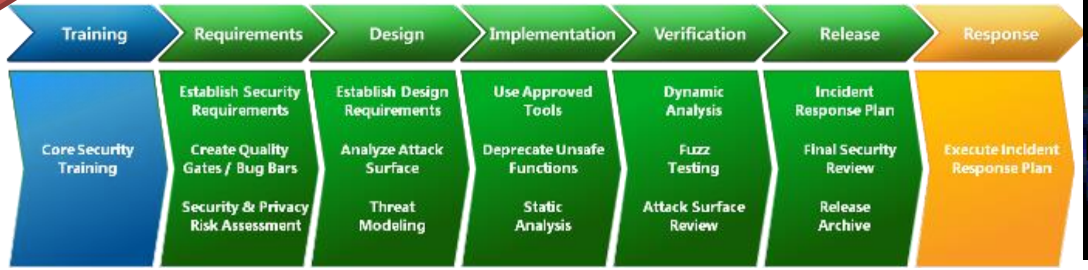
Quadcore SOC

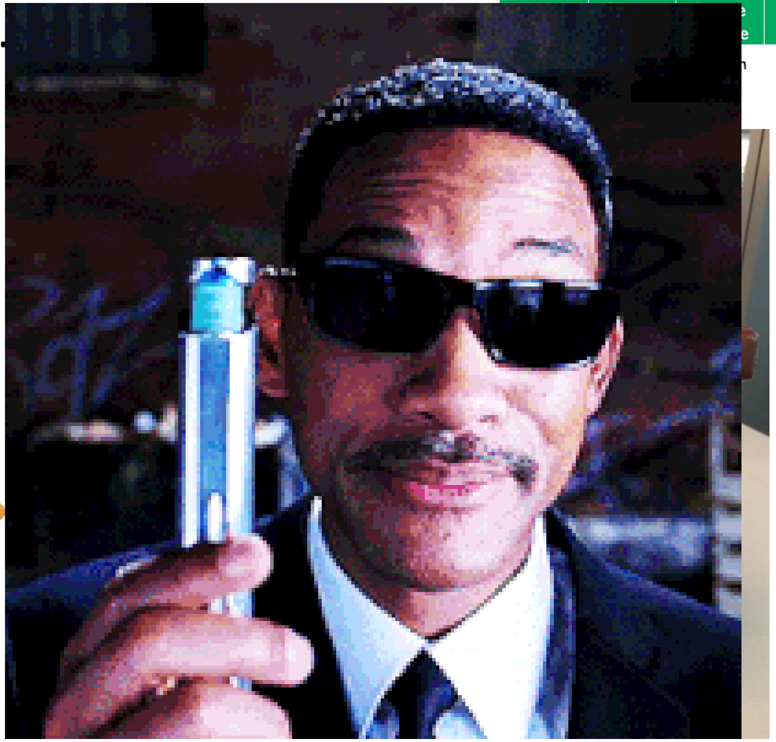
Let’s talk about Cyber secu

- Long time experience and well ... ield
  – Lots of security mecha ... rdware
    - Secure boot ... gnatures, A
- Best practices ... in industry
  – Secu ... pment Lifecycles
  – ... ing guidelines

Does not apply for IoT







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

heise online heise+

IT Wissen Mobiles Security Developer Entertainment Netzpolitik

TOPTHEMEN: NASA BITCOIN AMAZON CORONA WINDOWS 10 E-AUTO

Security 〉7-Tage-News 〉04/2013 〉**Vaillant-Heizungen mit Sicherheits-Leck**

Vaillant-Heizungen mit Sicherheits-Leck

Die Heizungsanlage ecoPower 1.0 kann man über das Internet steuern – allerdings auch dann, wenn man dazu gar nicht berechtigt ist. Ein Angreifer könnte die Anlage dadurch potenziell dauerhaft beschädigen. Kunden sollen jetzt den Netzwerkstecker ziehen.

Lesezeit: **1 Min.** 319

15.04.2013 13:00 Uhr | **Security**

Von Ronald Eikenberg

Die Vaillant-Heizungsanlagen des Typs ecoPower 1.0 enthalten ein hochkritisches Sicherheitsloch, durch das ein Angreifer die Anlage über das Internet ausschalten und potenziell beschädigen kann. In einem Informationsschreiben rät der Hersteller seinen Kunden daher zu einem drastischen Schritt: Sie sollen den Netzwerkstecker ziehen und auf den Besuch eines Servicetechnikers warten.

Bei den für Ein- und Zweifamilienhäuser ausgelegten ecoPower-Anlagen handelt es sich um sogenannte Nano-Blockheizkraftwerke, die aus Gas nicht nur Wärme, sondern gleichzeitig auch Strom

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

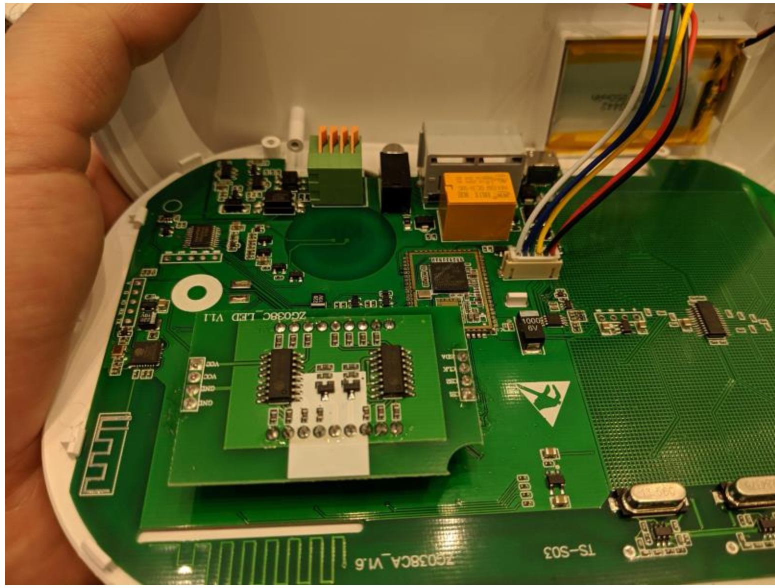
- Development cycle similar to smartphones
  - New products and models every year
  - Product support dropped after 1-2 years
  - Developers can only focus on new products
- Problem: Smart Home devices are used longer
  - Average lifespan of a washing machine: 7-13 years
  - No incentive for customer 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

Risk of Malicious IoT devices

- What can possibly go wrong?

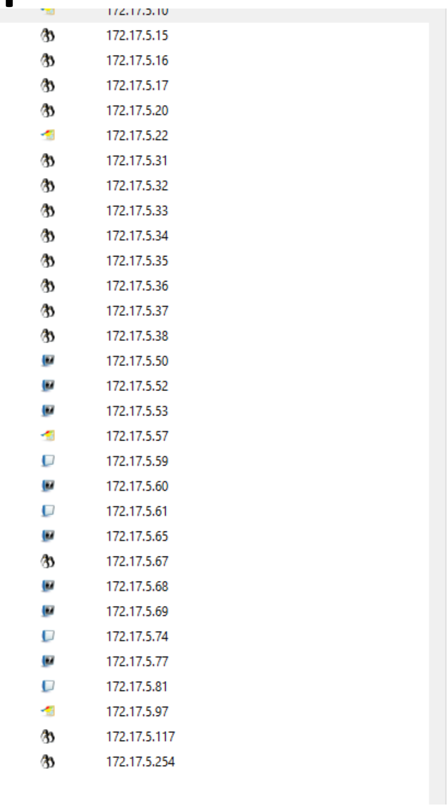
  Missing network separation

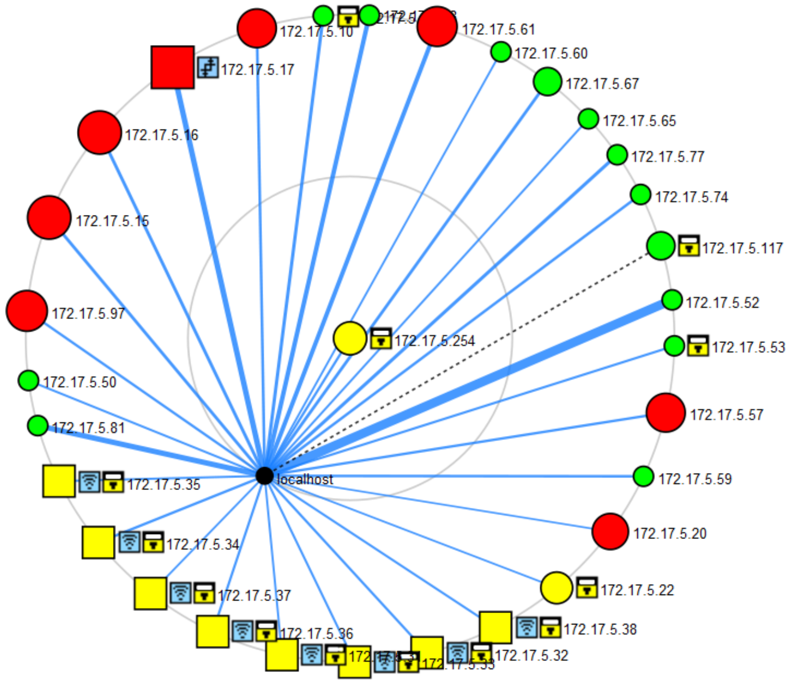
  + default passwords

  + Vulnerable IoT devices

  = Lots of fun for attackers

  = Hard time for you!





Fixed by Xiaomi


D.LAB
小米十周年纪念金章
XIAOMI 10th ANNIVERSARY
2010-2020

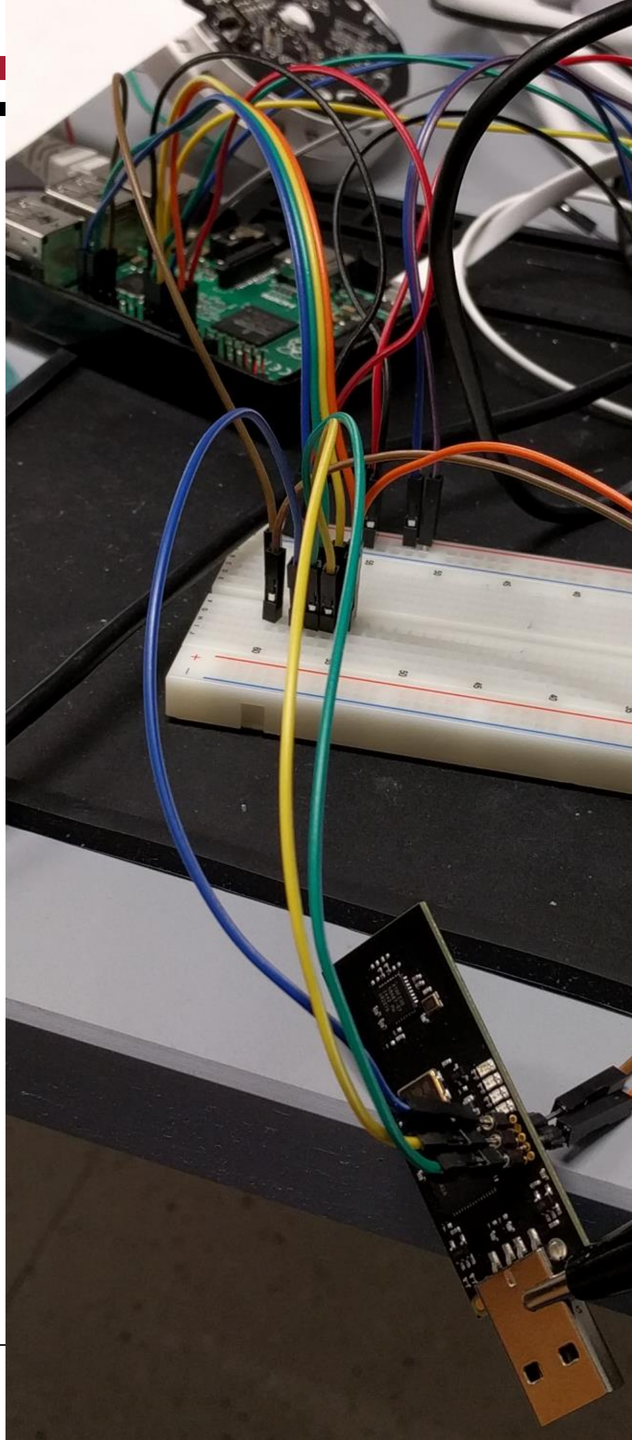
IOT REVERSE ENGINEERING

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

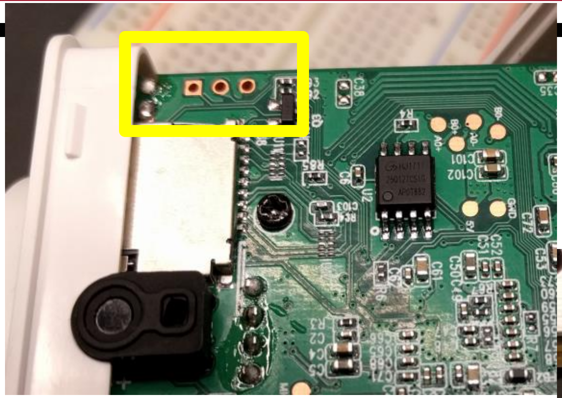
Raspberry Pi

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

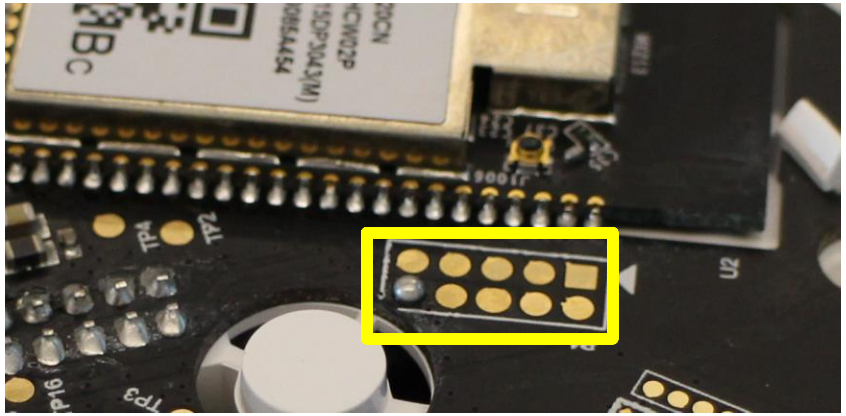


Debug Interfaces

- 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

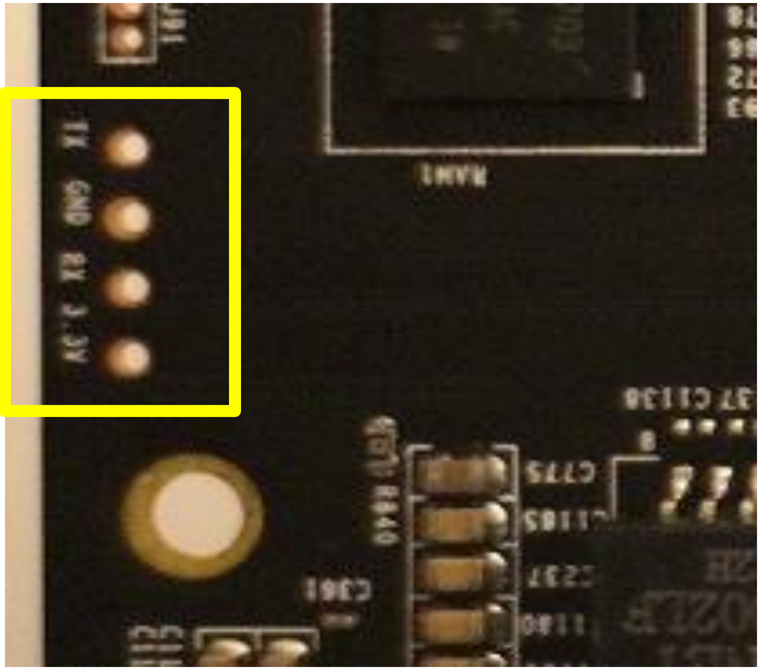


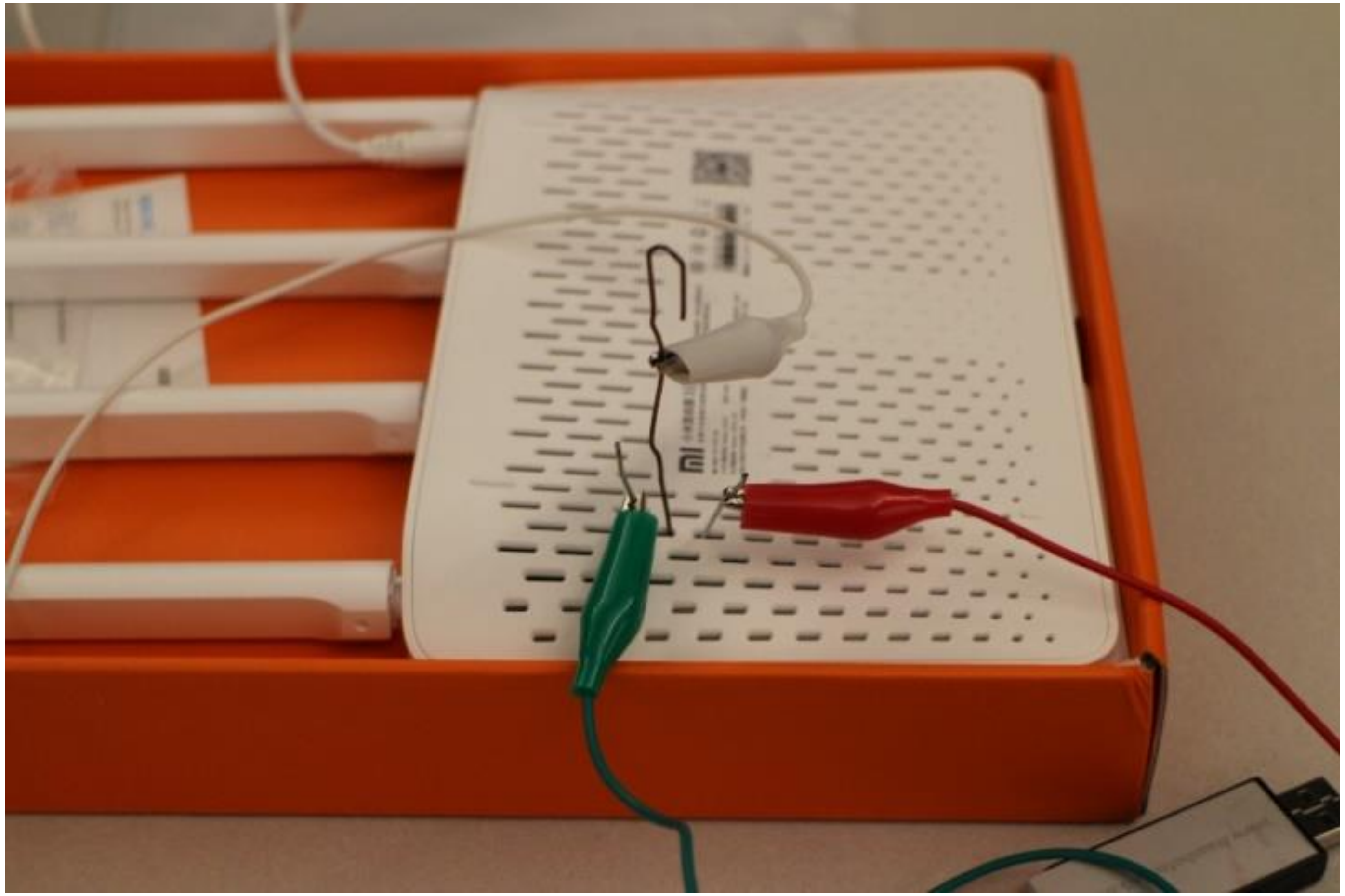
Chip Debugging



- 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

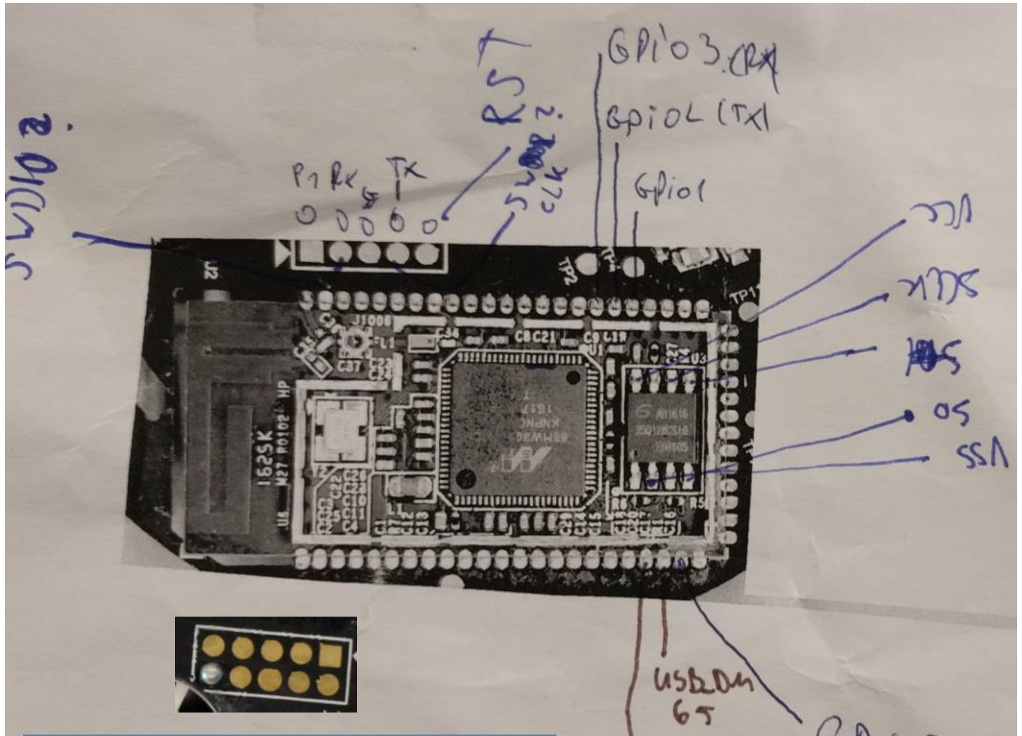
Problems with warranty seals?





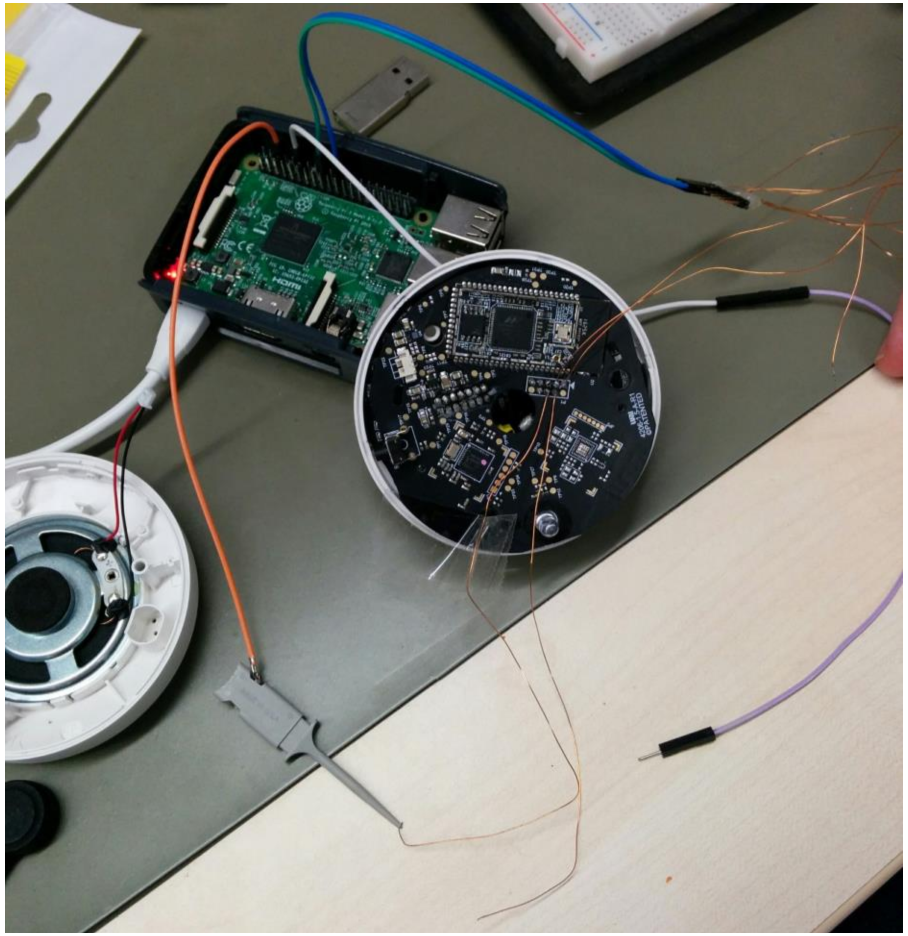
Example

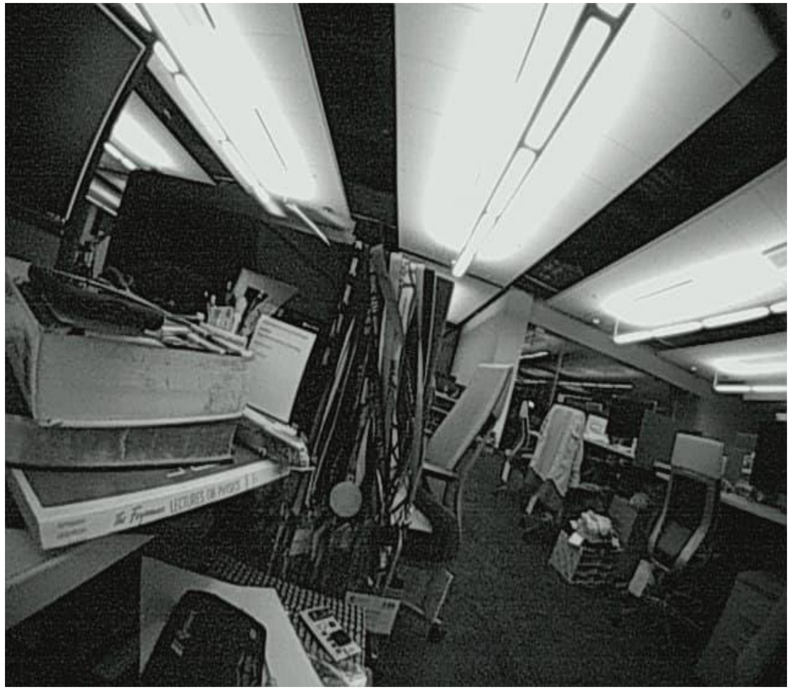
- Smart Home Gateway JTAG and UART



			SDCLK	SDIO
RST	TX*	GND	RX*	

*UART





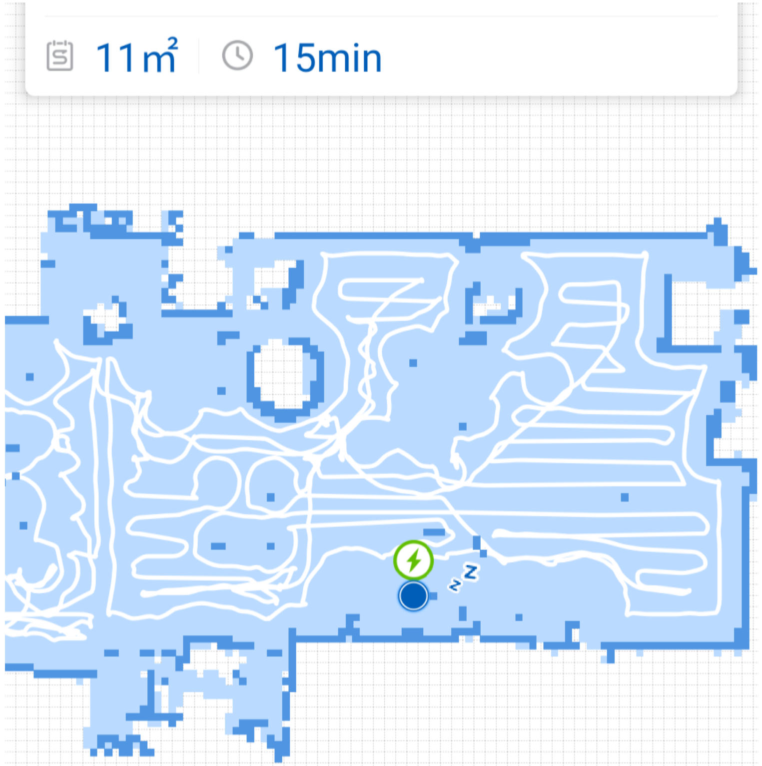
IOT AND PRIVACY

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)

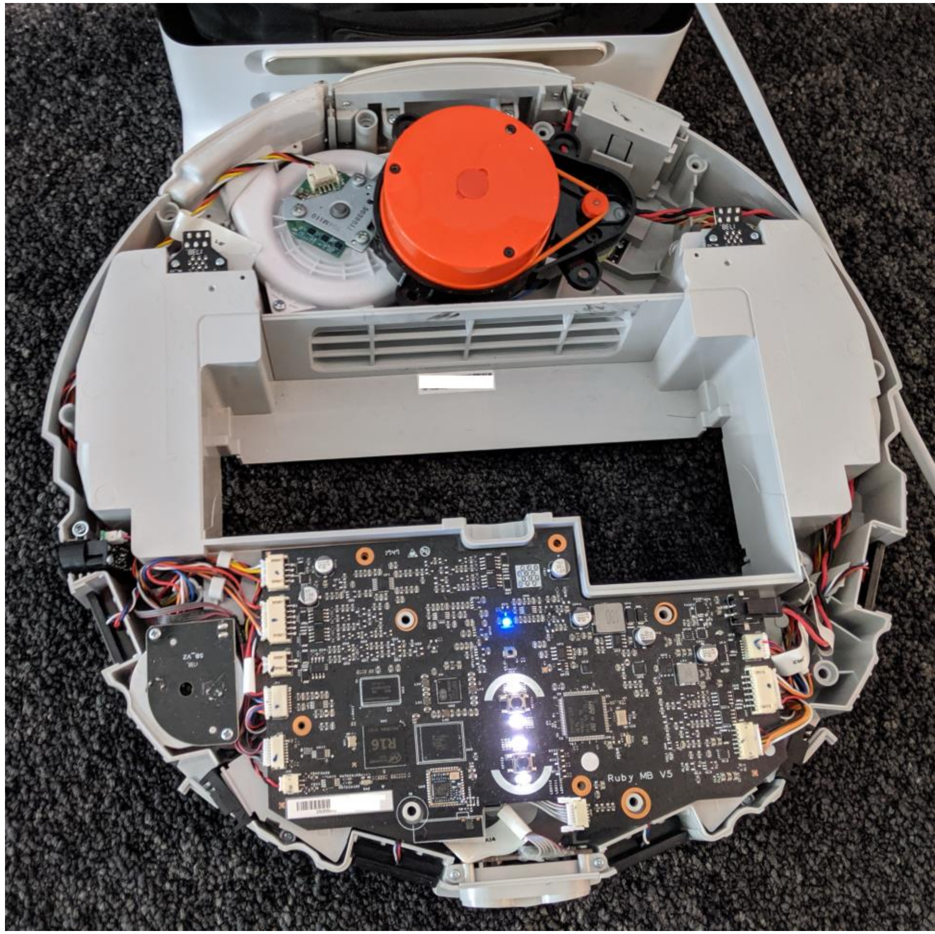
Recovering Data From Devices

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

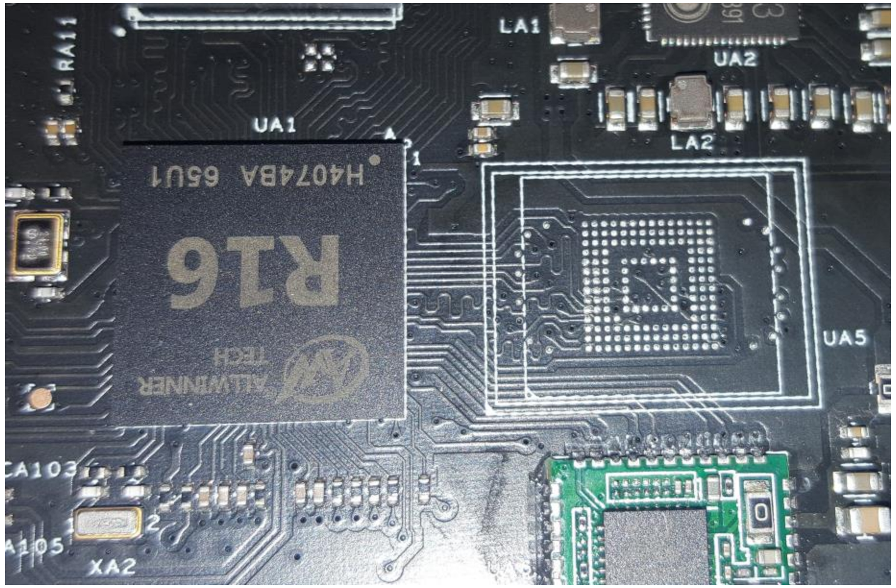
Xiaomi/Rockrobo Mi Vacuum Robot

- 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



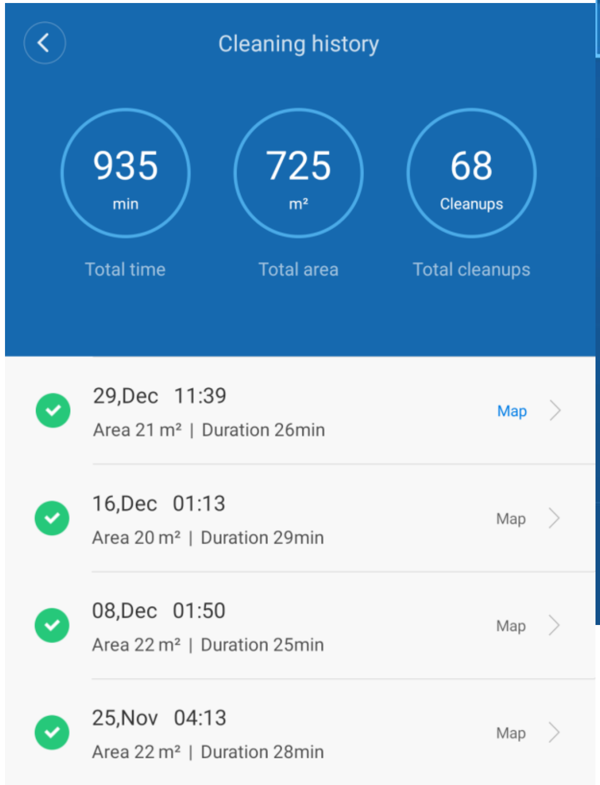
https://github.com/dgiese/dustcloud

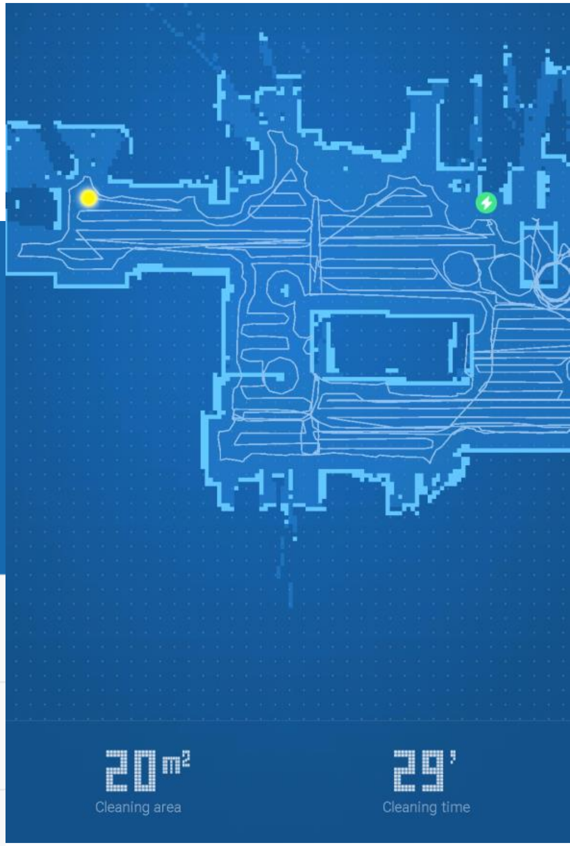
Mi Vacuum Robot reset methods

- 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

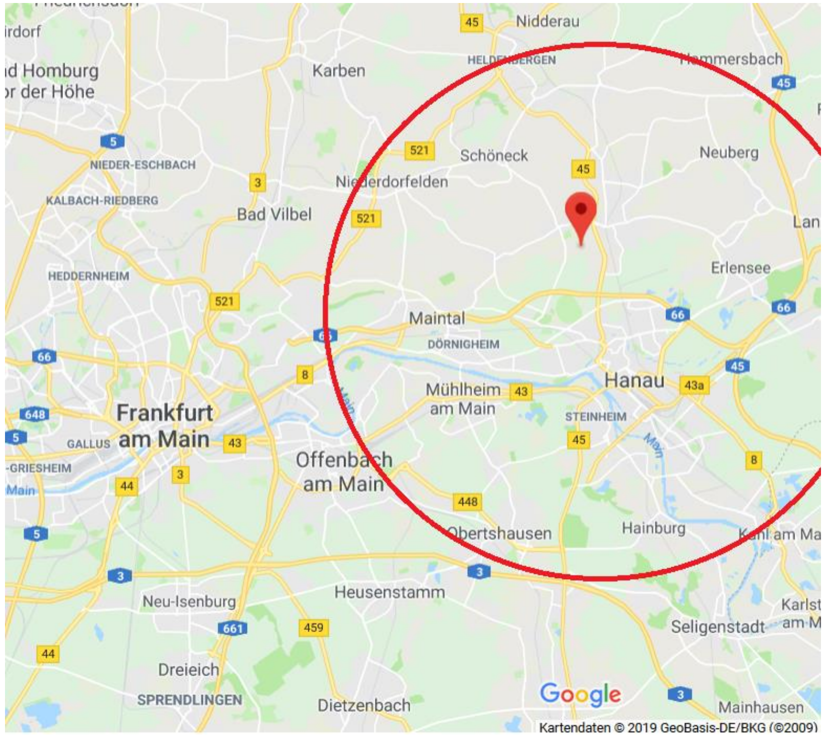




Mi Vacuum Robot: locating former owner

- 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

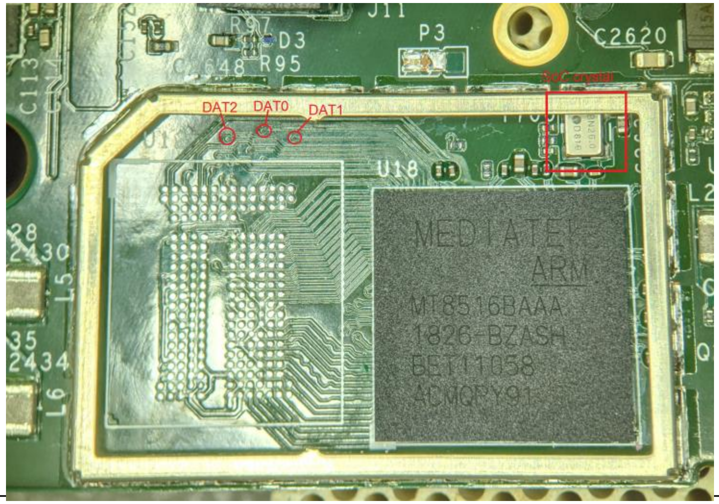

```
{
  "location": {
    "lat": 50.17[redacted],
    "lng": 8.90[redacted]
  },
  "accuracy": 20
}
```

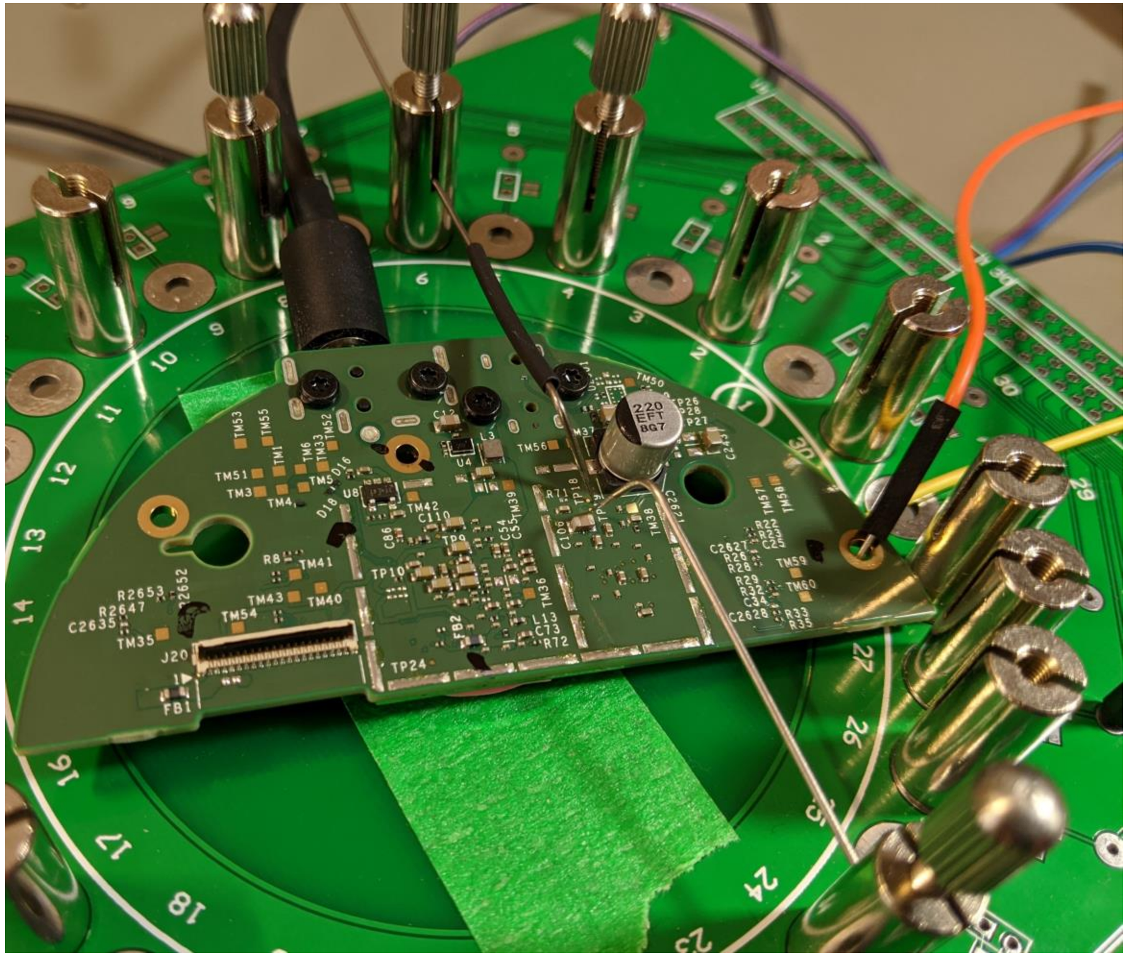



Source: Google Maps

Current research: Amazon Echo Dot*

*Paper under submission





Current research: Amazon Echo Dot*

*Paper under submission

- 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

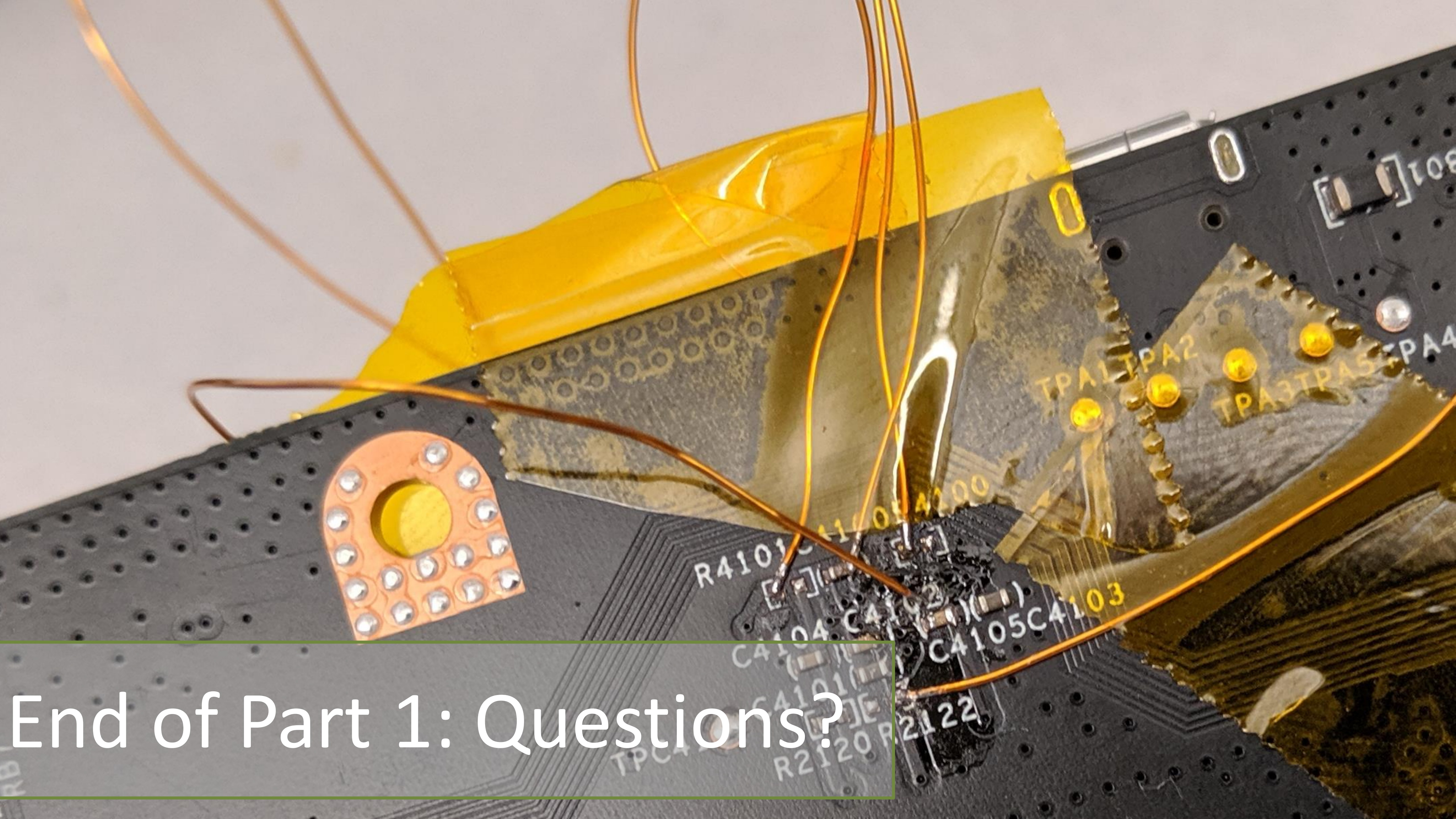
Used IoT Devices conclusions

- 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

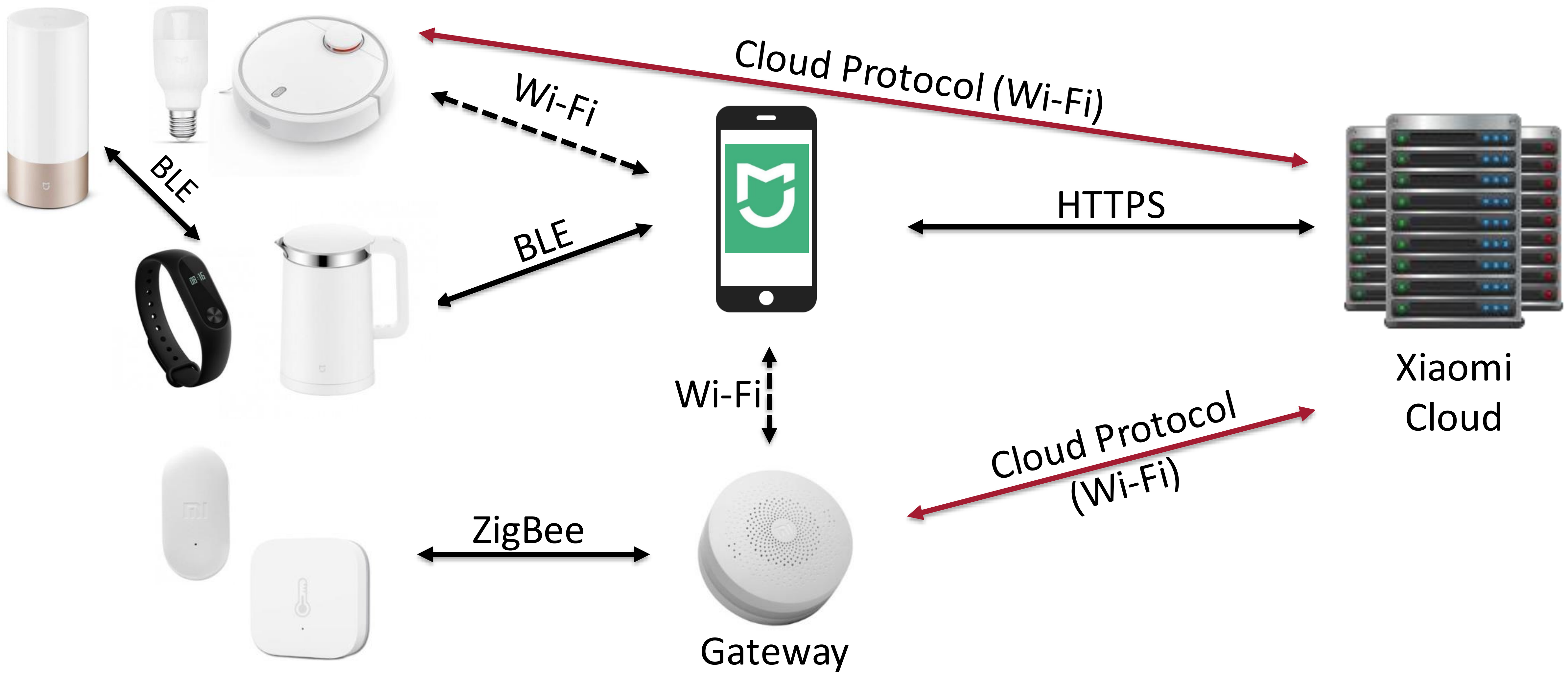
- ➢ Introduction
- Methodology
- Analysis of Mi Home App
- Analysis of Devices
- Discussion
- Conclusion

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

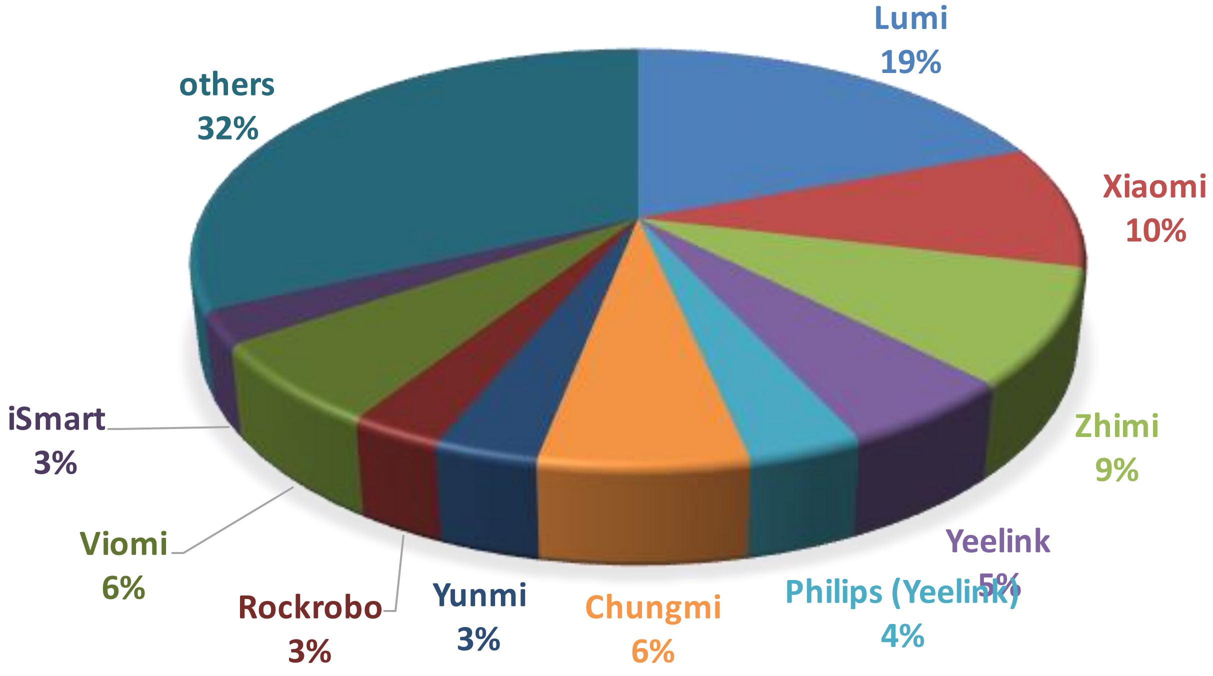
Introduction
Products

Communication relations



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

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

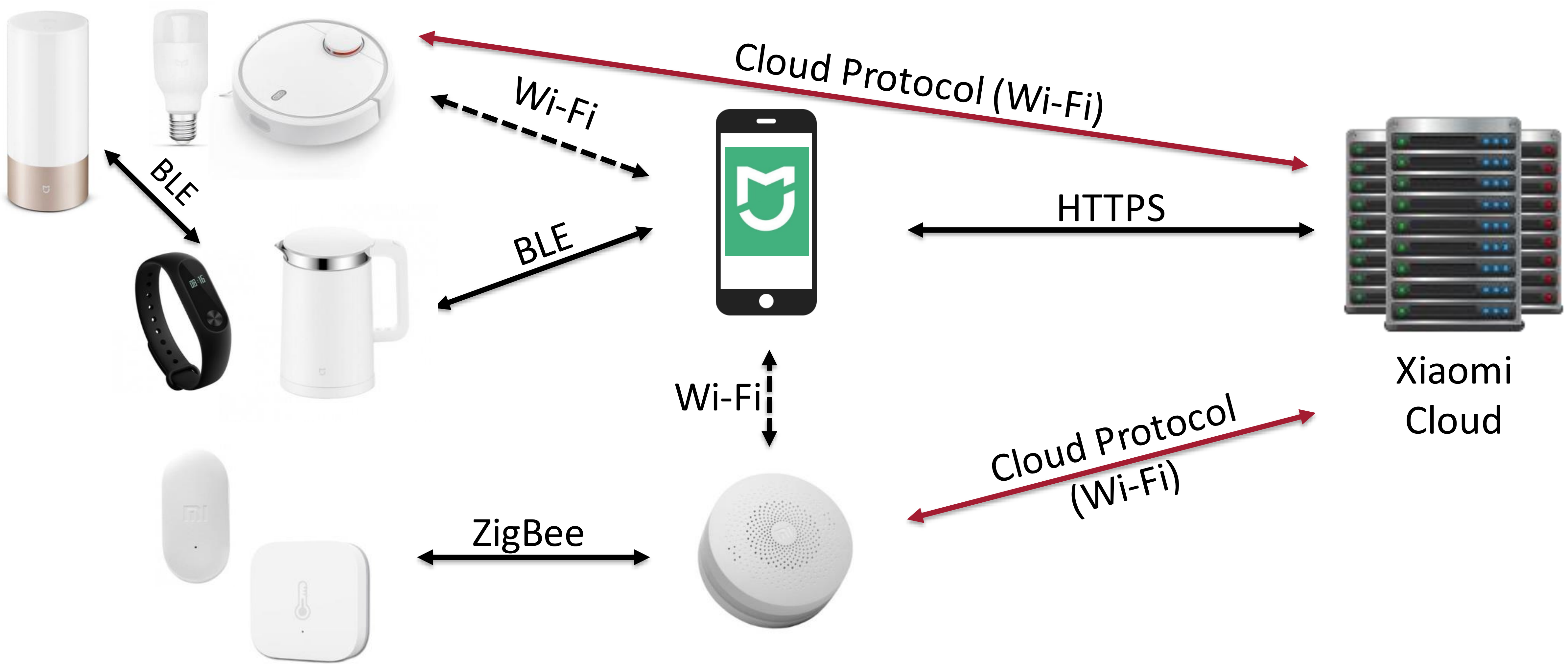
Goals

- 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

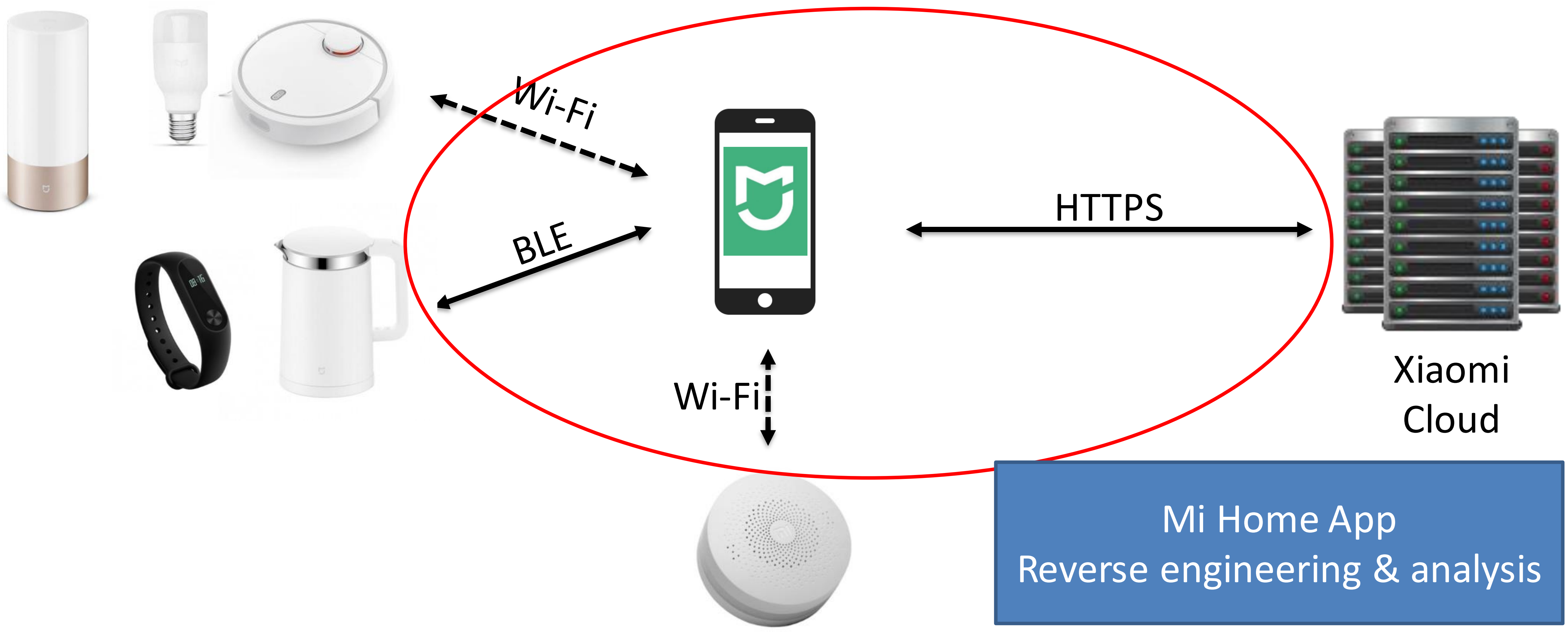
Outline

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

Approaches


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

Approaches: App



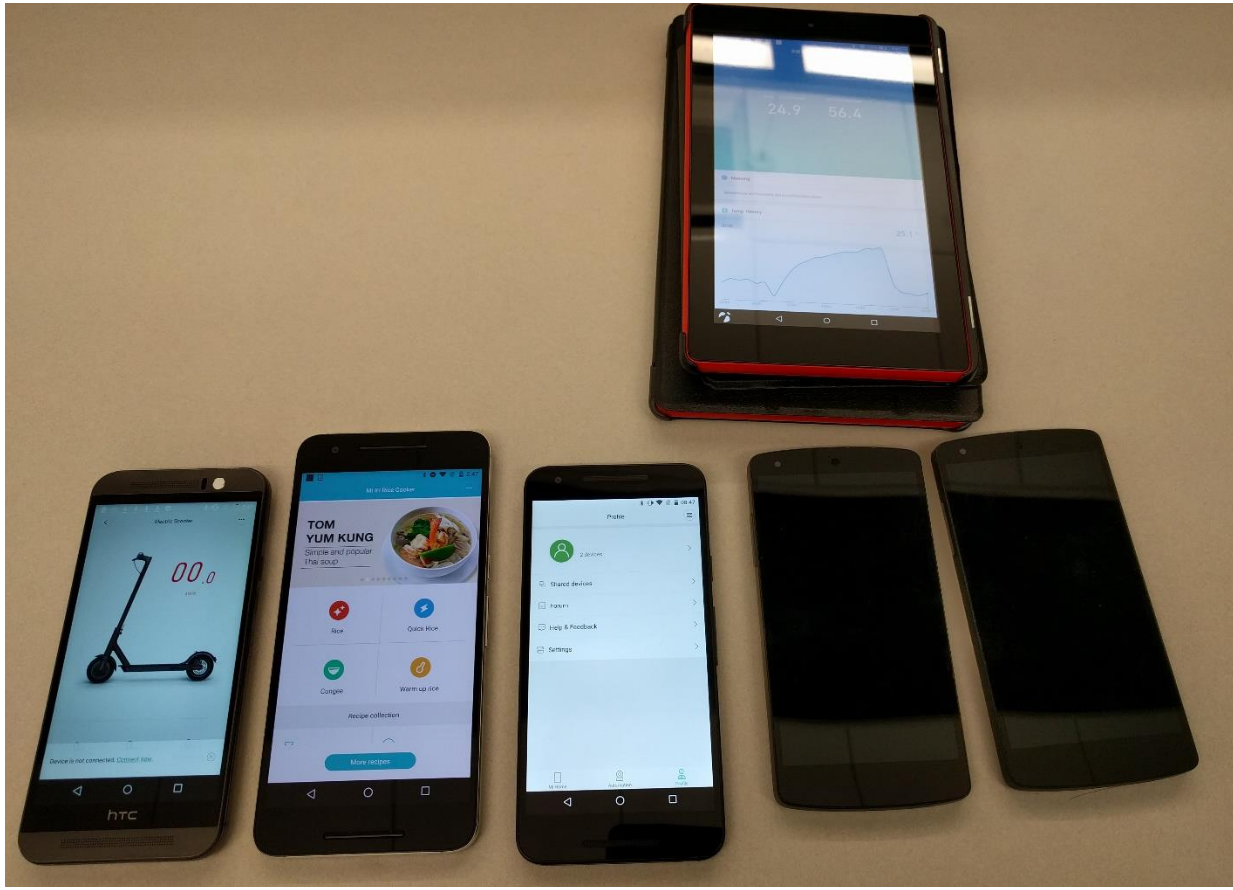
Mi Home App
Reverse engineering & analysis

App Reverse Engineering

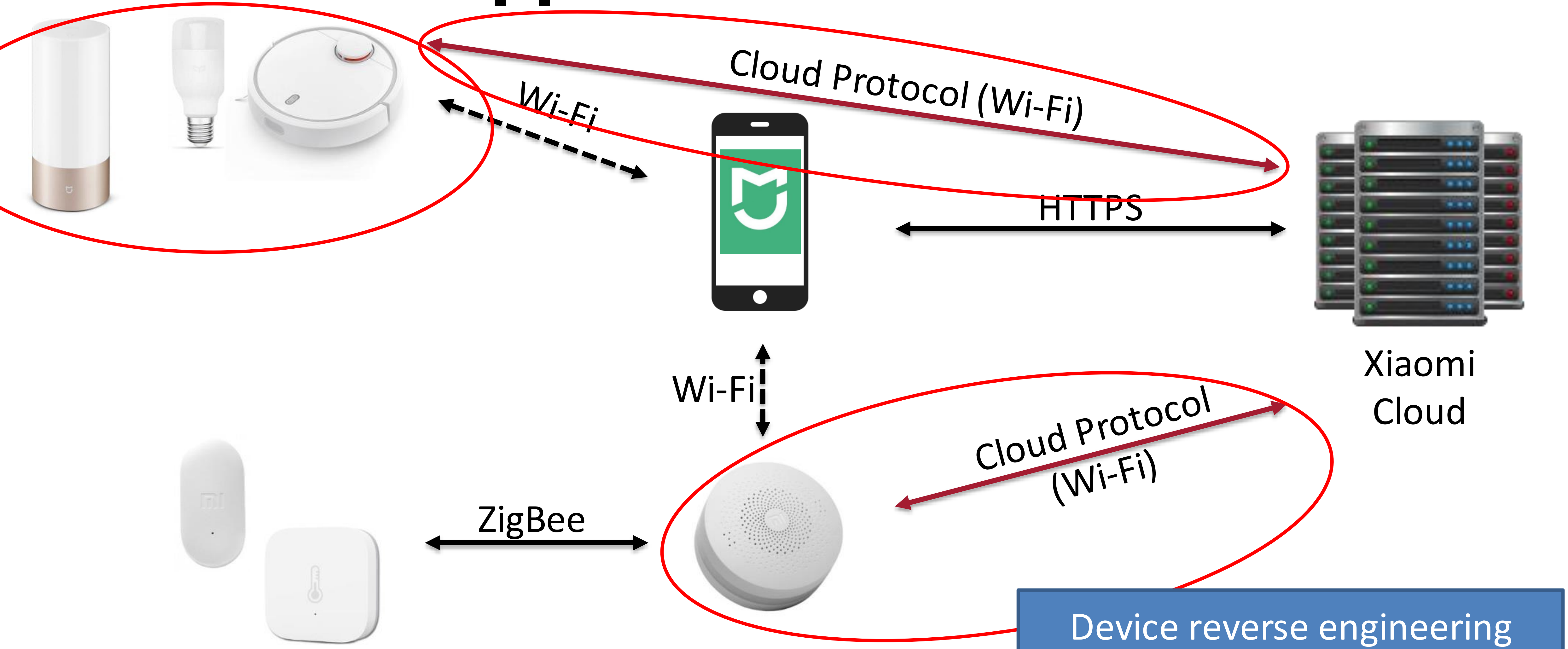
- Idea: Understand interaction between app and phone, and app and cloud
- Advantage: device data is displayed inside the app -> app needs to know how to interpret it
- Methods:
  - Disassembly: Jadx (APK to Java)
  - Modification: Apktool (APK to smalicode, rebuilding)
  - Monitoring: Logcat (monitoring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

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



Approaches: Devices



Device reverse engineering

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

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, … ☹

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

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

App Structure

Mi Home Core App
(Dev: Xiaomi)

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

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

Gen1 Vacuum Plugin
(Dev: Rockrobo)

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

⋮

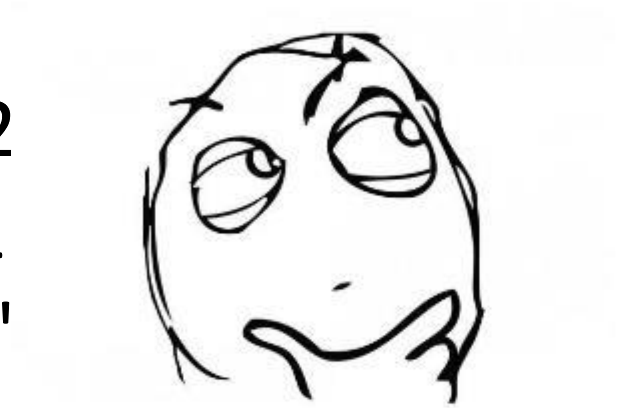
Aqara Cam Plugin
(Dev: Lumi)

Device specific functionality:
- Device commands
- Camera feed view (via P2P)
- Zigbee device control

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

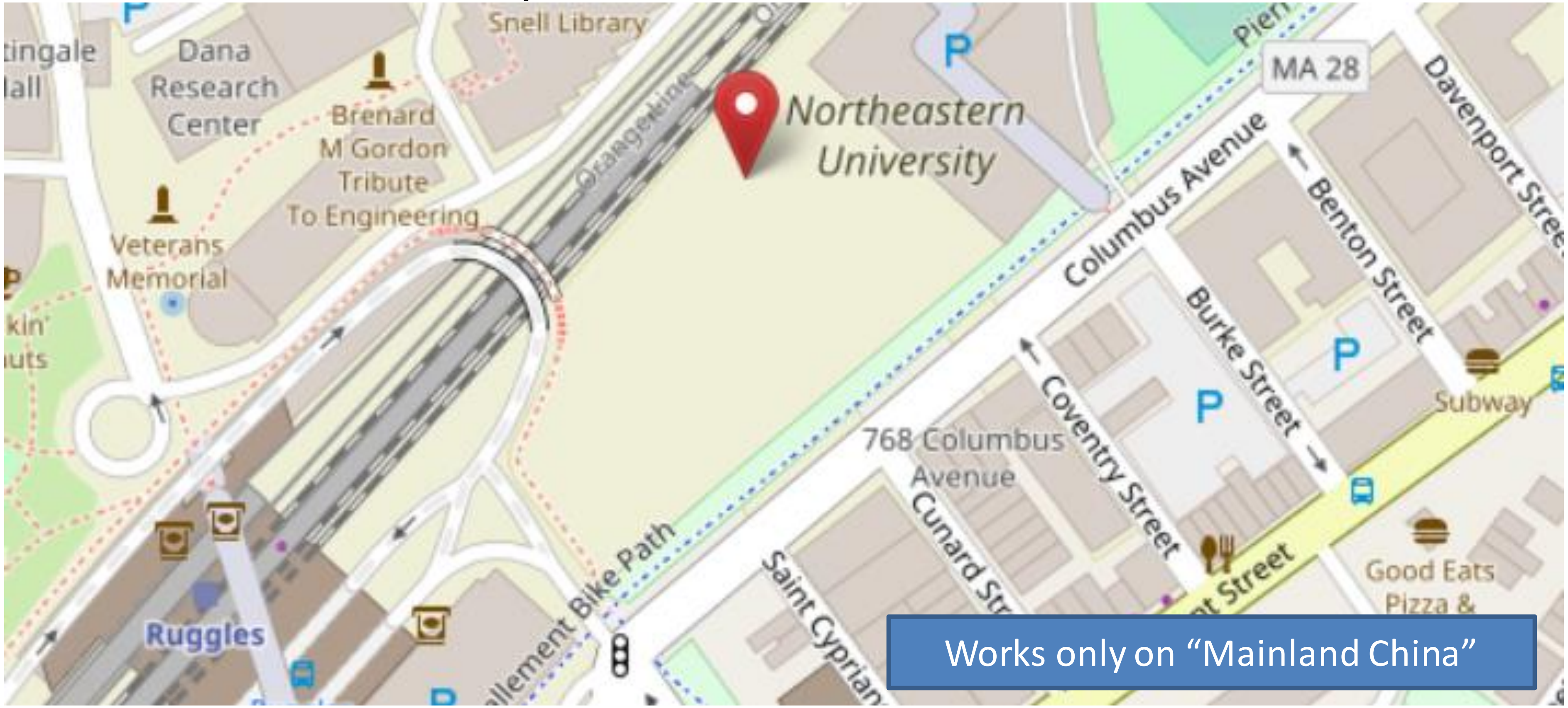
Example of intercepted cloud api call

- REQ: api.io.mi.com/home/device_list method:POST params:[]
- RES:
{"message":"ok","result":{"list":[{"did":"659812
bc...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}



Example of intercepted cloud api call

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



Source: Openstreetmaps

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

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

App to Device via Cloud RPC

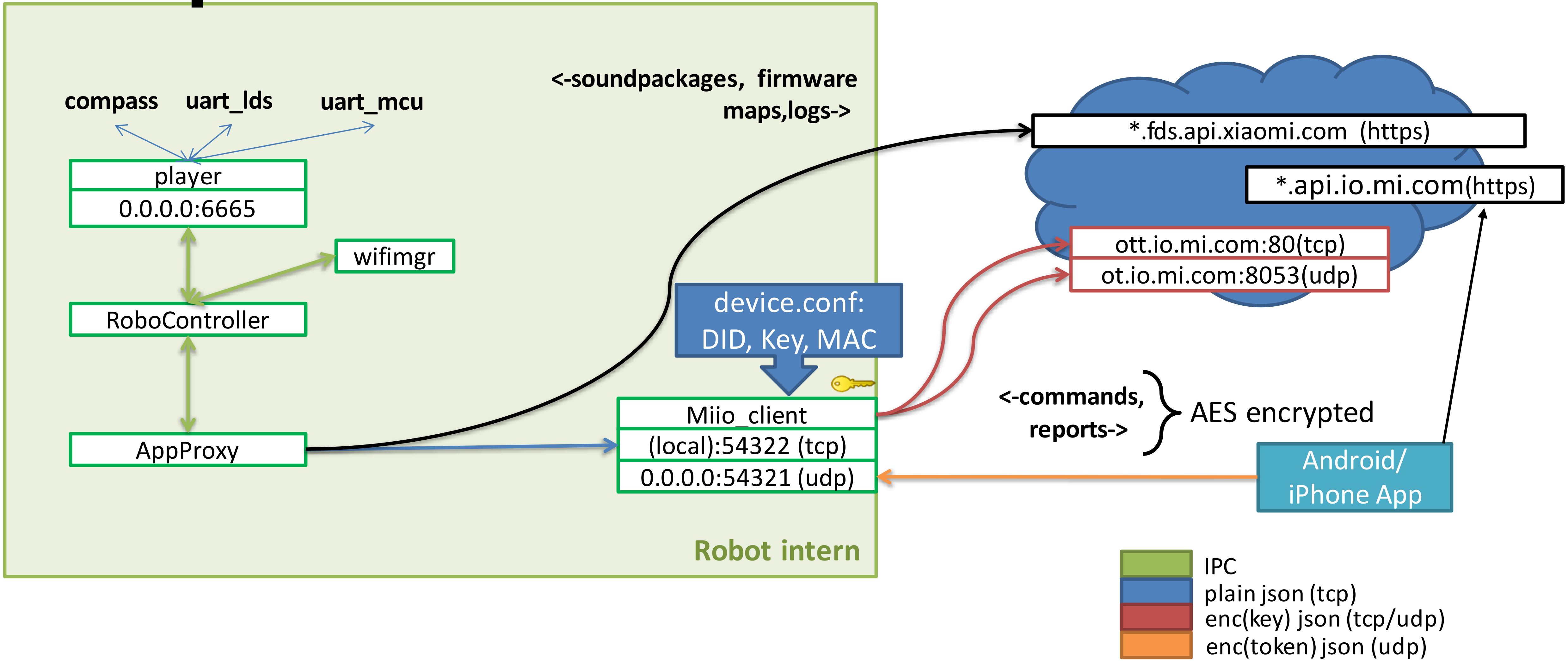
Device management

- 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 (2017-2021: 4300 devices)
  - Add devices to unsupported regions

Outline

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

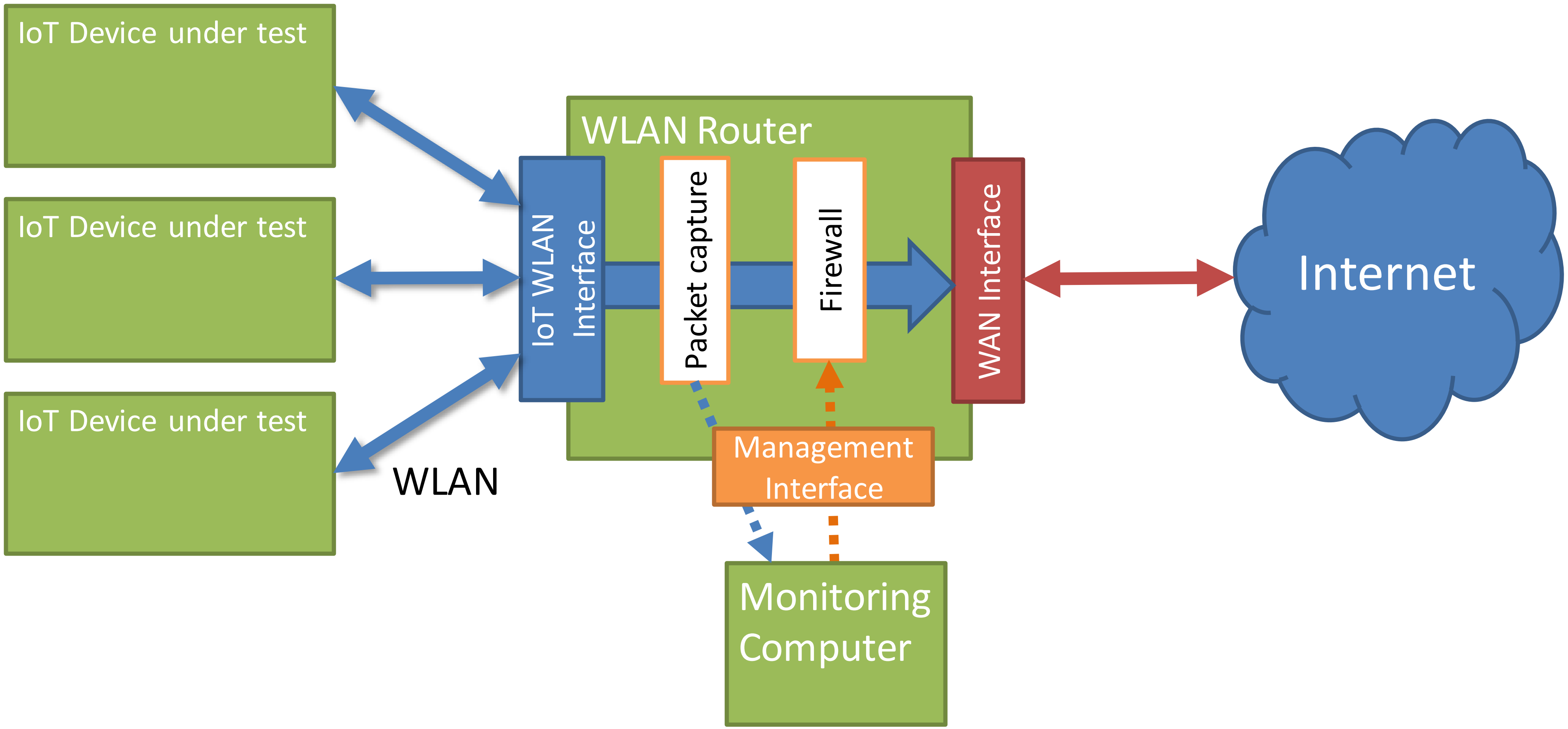
Deeper Look at Communication relations



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)

Network Setup



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

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

MD5

Model

Authentication

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 (2018-2021: 3190)
  - Sharing with other researcher for development of open source implementations

Collection of firmwares and device info

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)

Region	cn	de	ru	sg	us
first seen	2019-03-30	2019-03-30	2019-03-30	2019-03-30	2019-03-30

Type	**MD5**	**Filename**	**Version**	**Datetime**	**Regions**
app	9e2c0809cebc892c60c6723b30d76016	v11_001768.fullos.pkg	3.3.9_001768	2019-03-27 11:57:00	cn,de,ru,sg,us
app	e7c6f4062b6717d9b7ea1cebeb48f3a8	v11_001720.fullos.pkg	3.3.9_001720	2019-05-23 02:23:00	de,sg,us
app	3d04e386856129a0c0a9508c40e577b7	v11_001864.fullos.lmn09e8u2.pkg	3.3.9_001864	2019-05-31 05:51:00	cn,de,ru,sg
aplugin	77a1d4cfc186aaec8757a27e12d04d88	com.roborock.rubys.app_2019061715280736461.zip	188	2019-06-17 07:28:00	de,sg,us
aplugin	e5d96f0f89b5d8fecdfbd26b829849d4	com.roborock.rubys.app_2019062414482451501.zip	191	2019-06-24 06:48:00	cn

Devices under test

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

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

Devices under test

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

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

Mi Vacuum Cleaning Robot (Gen1)

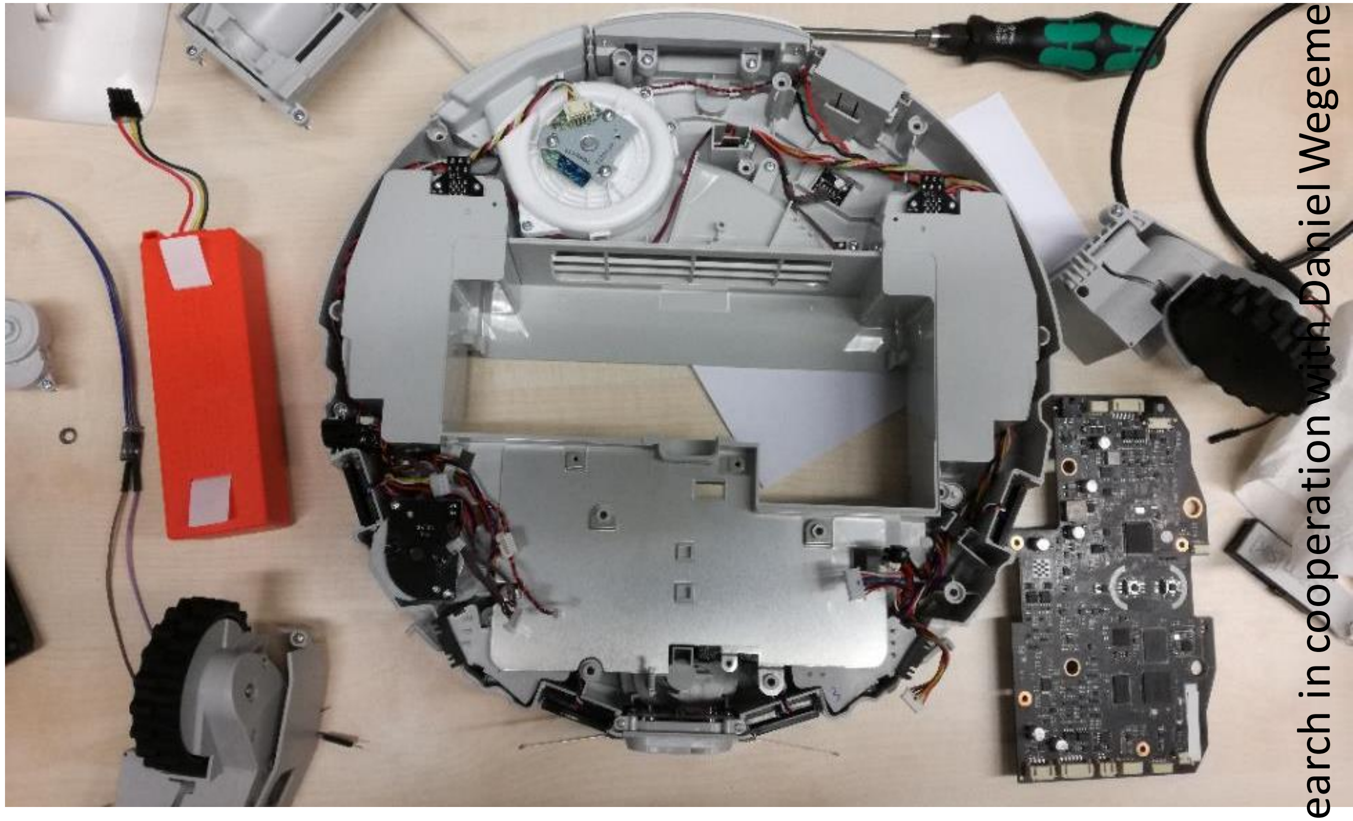
Source: Xiaomi advertisment

Research in cooperation with Daniel Wegemer

Analysis of Devices

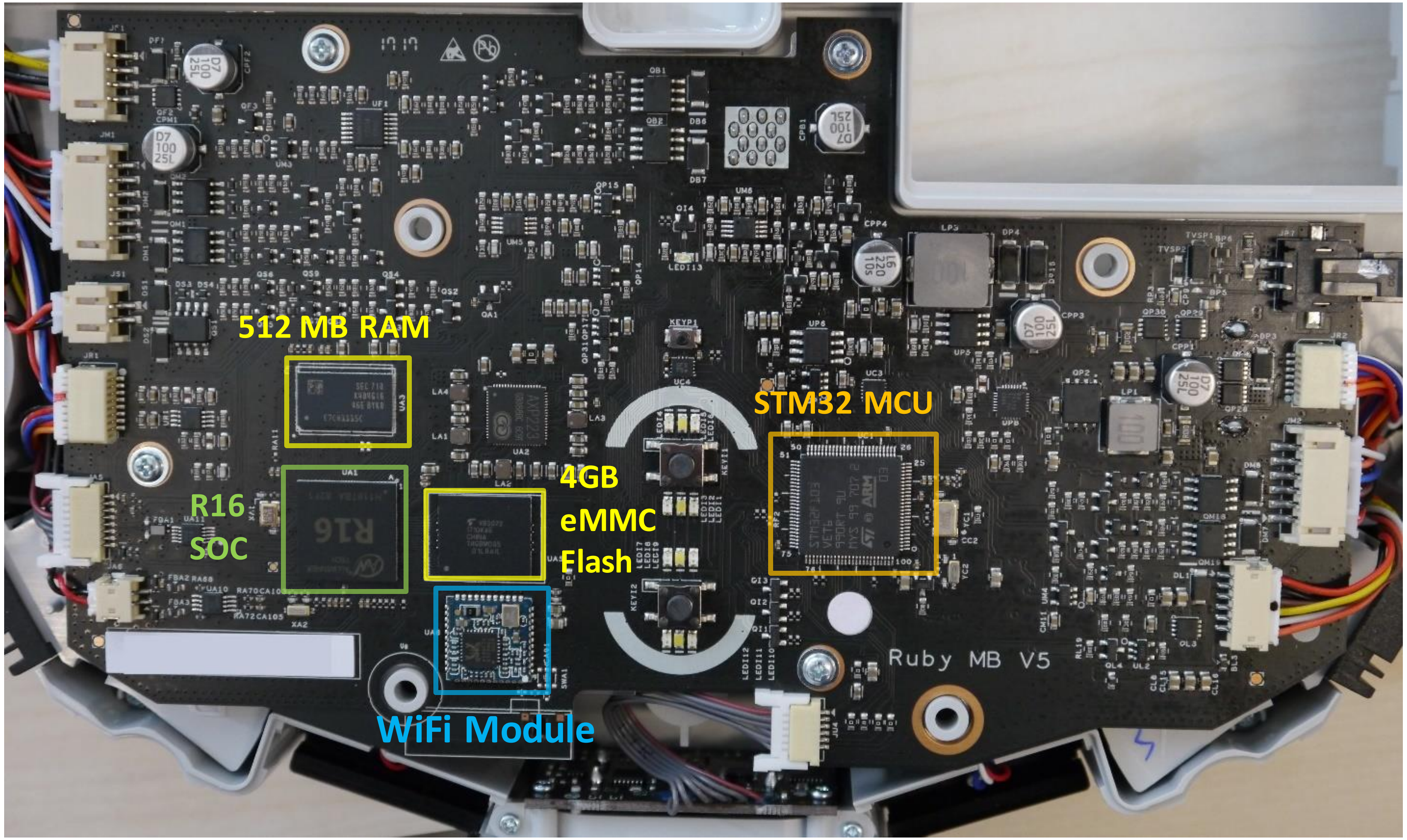
Mi Vacuum Cleaning Robot

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

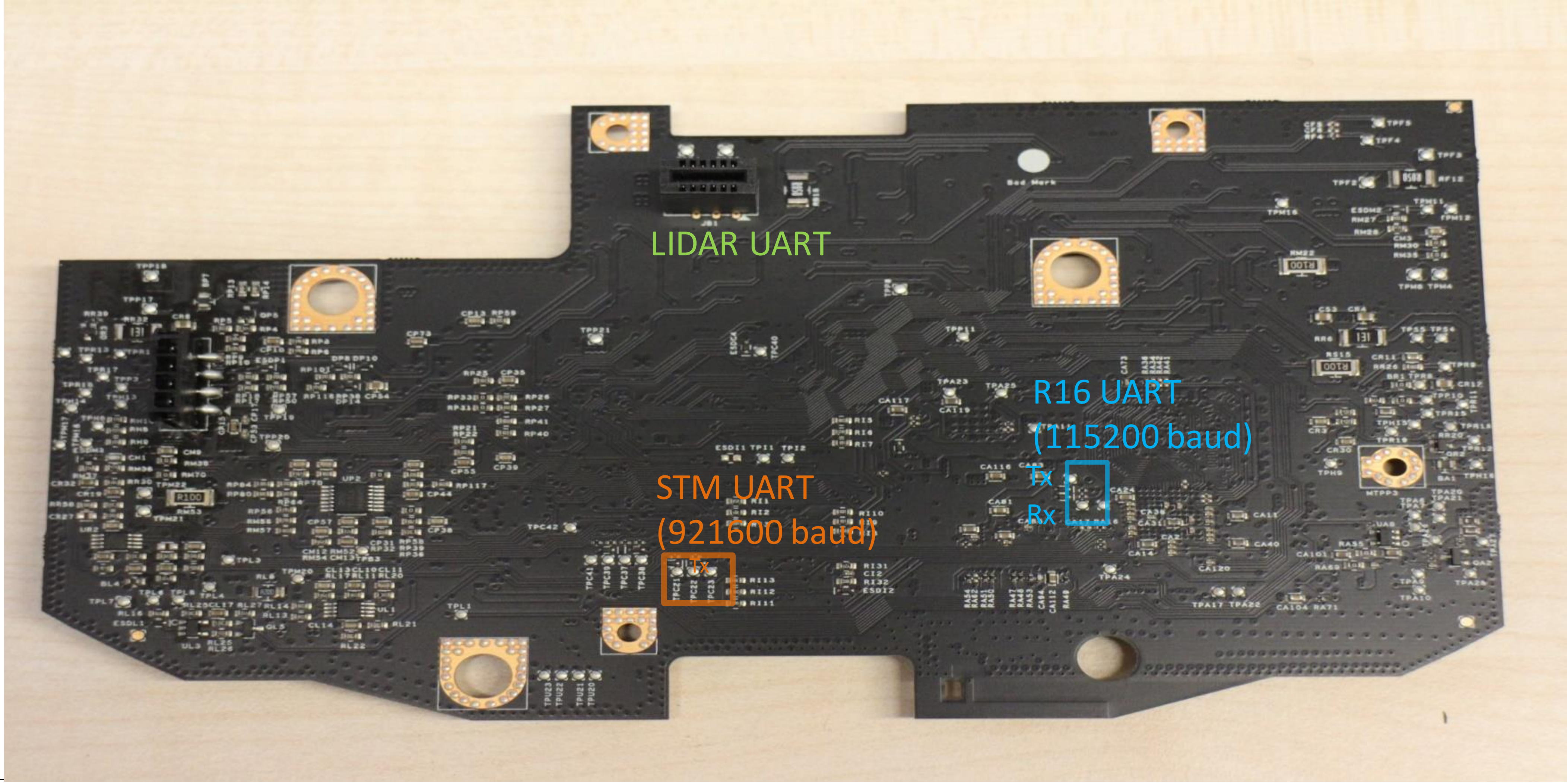


Research in cooperation with Daniel Wegemer

Frontside layout mainboard



Backside layout mainboard



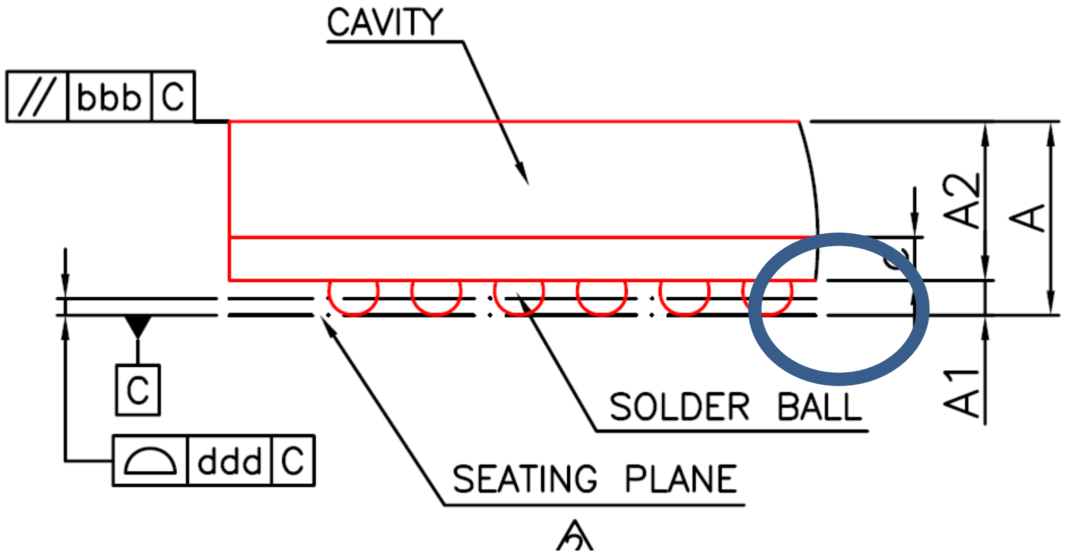
Gaining Root access

- 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 > 40000
  - Used by researchers for 5G and Wi-Fi experiments, teaching robotics students

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



Symbol	Dimension in mm		
	MIN	NOM	MAX
A	---	---	1.29
A1	0.25	0.30	0.35
A2	0.84	0.89	0.94
c	0.32	0.36	0.40
D	13.90	14.00	14.10
E	13.90	14.00	14.10
D1	---	12.80	---
E1	---	12.80	---
e	---	0.80	---
b	0.35	0.40	0.45
aaa		0.15	
bbb		0.10	
ddd		0.10	

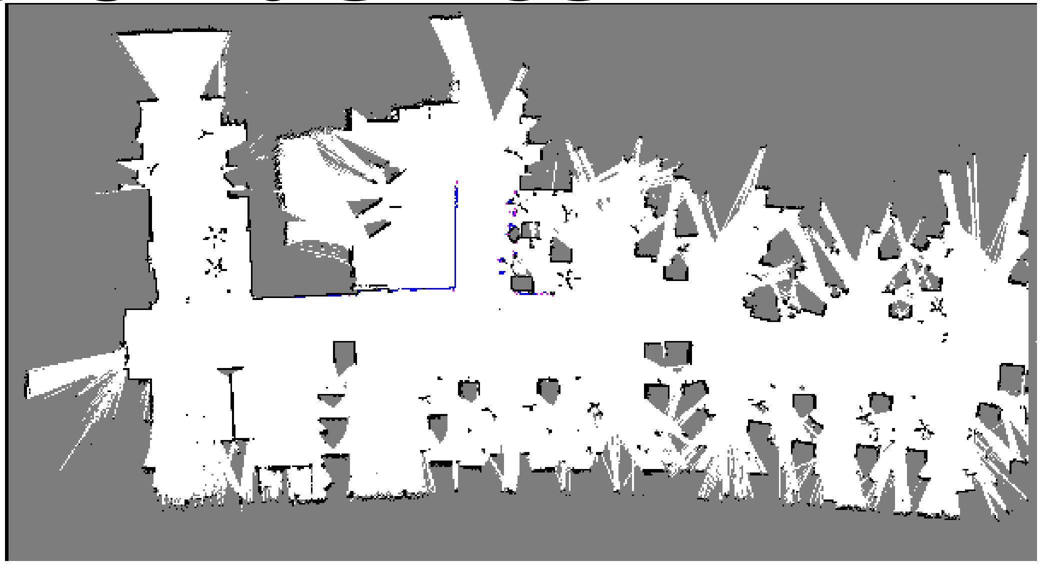
Available data on device

- 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

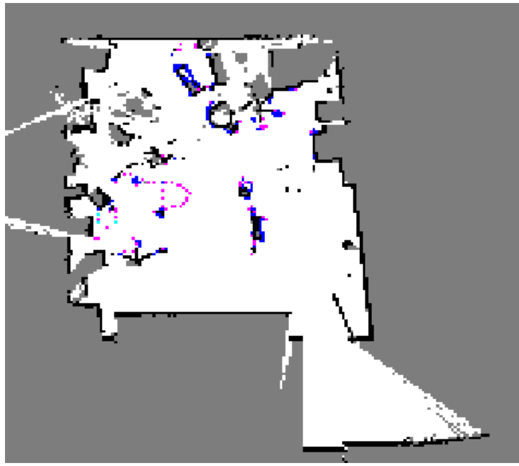
➢ Contribution: Documentation of the usage and collection of data

~100 Gbyte writes per Year

Available data on device



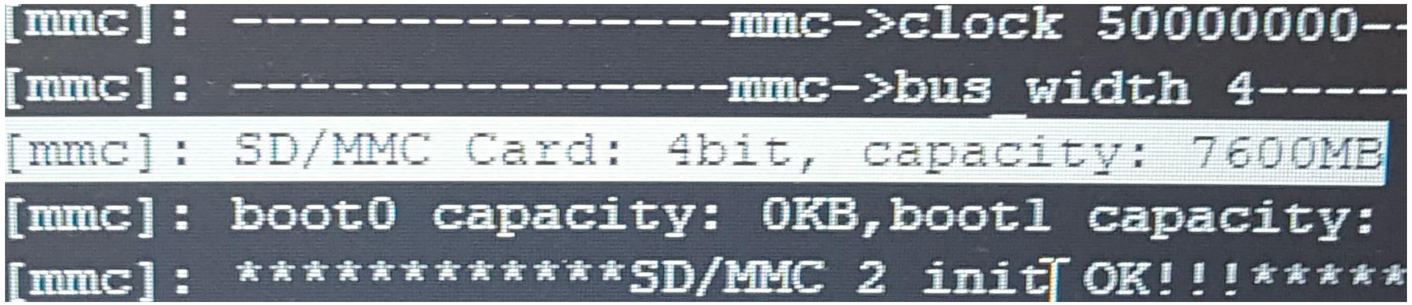
- Maps
  - Created by player
  - 1024px * 1024px
  - 1px = 5cm

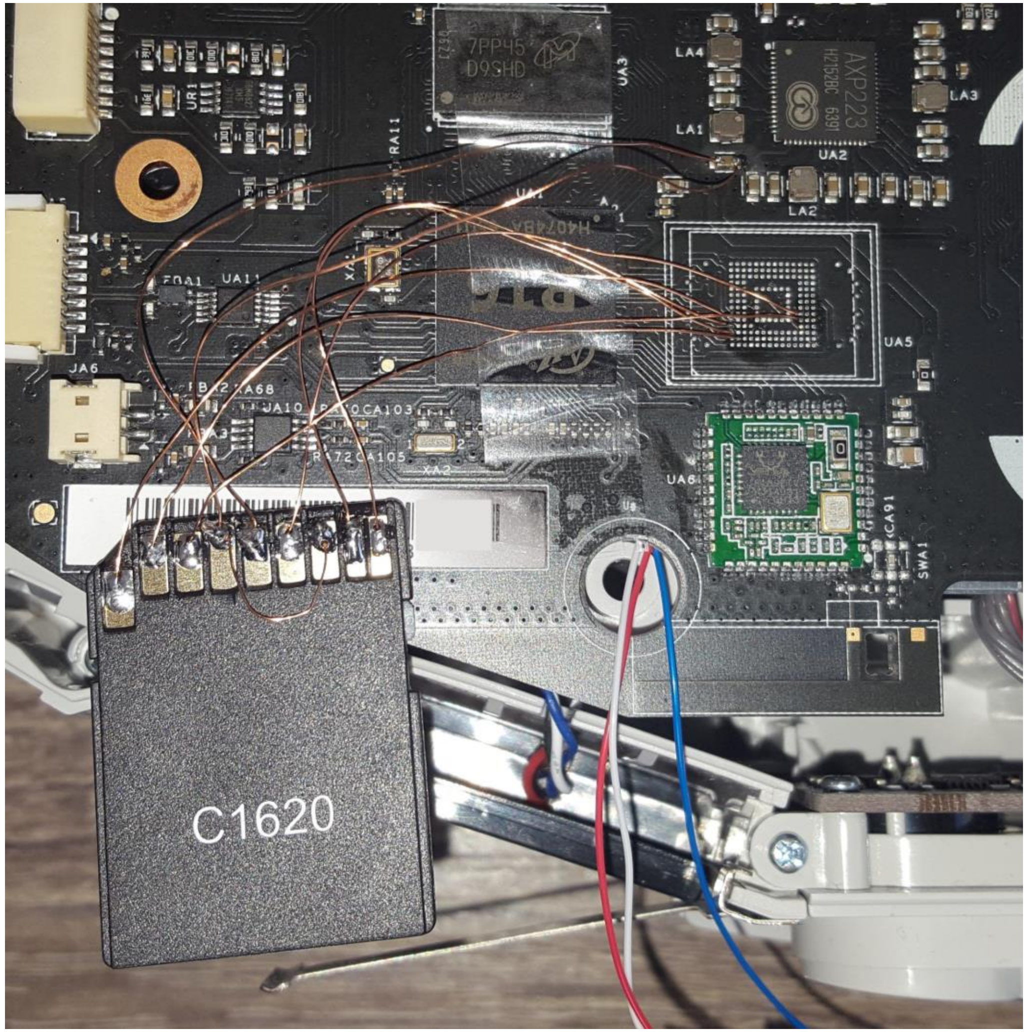


➢ Contribution: Tools for map interpretation
  - Base for all open source implementations

Custom mod of Gen1

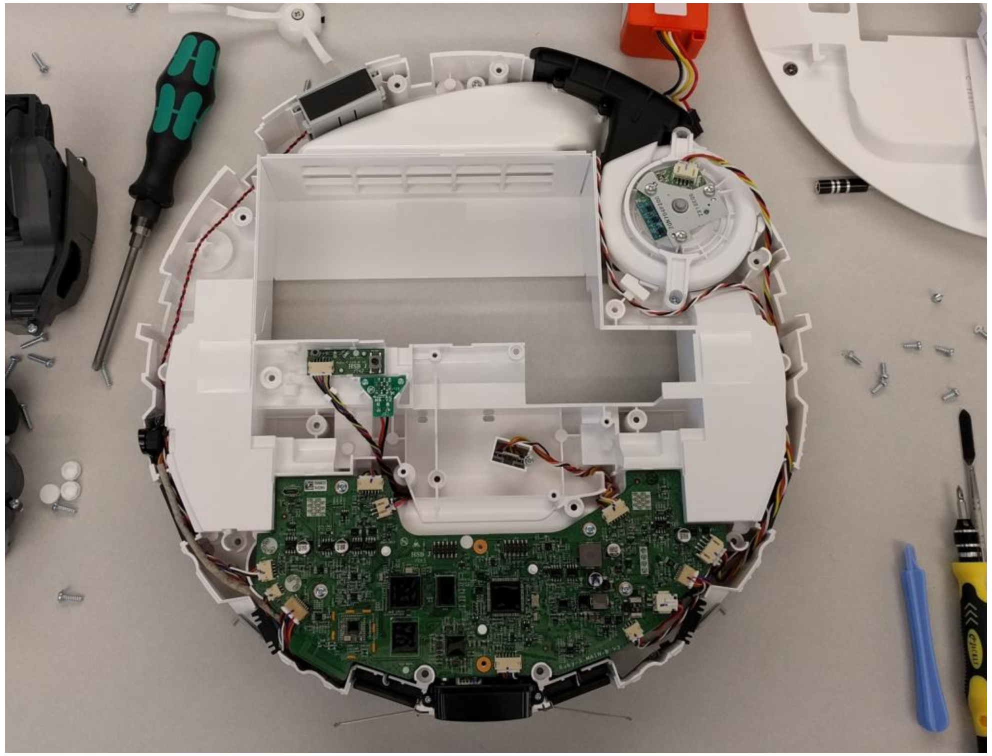
- Custom mod enables usage of bigger software (e.g. ROS)



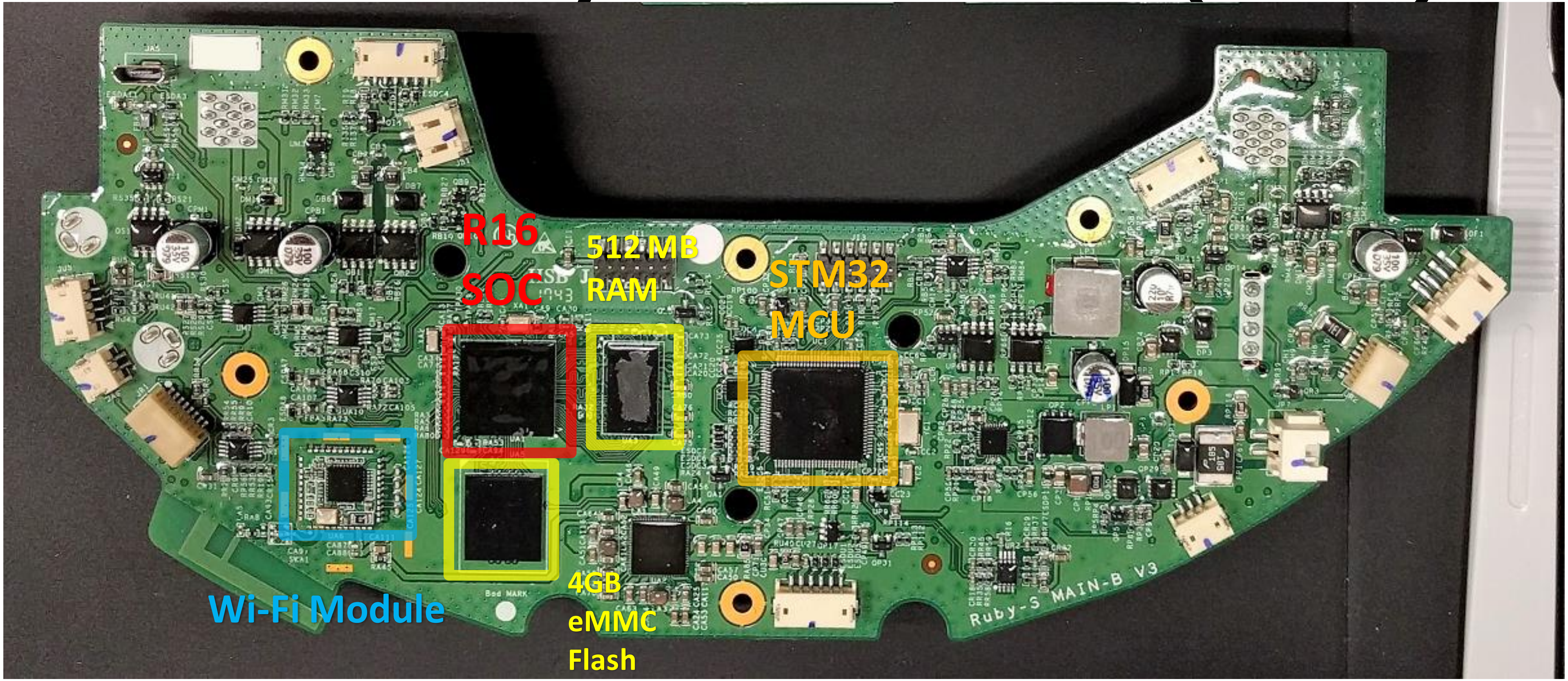


Roborock S50 (Gen2)

- 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



Frontside layout mainboard (Gen2)



Introduced Countermeasures in Gen2

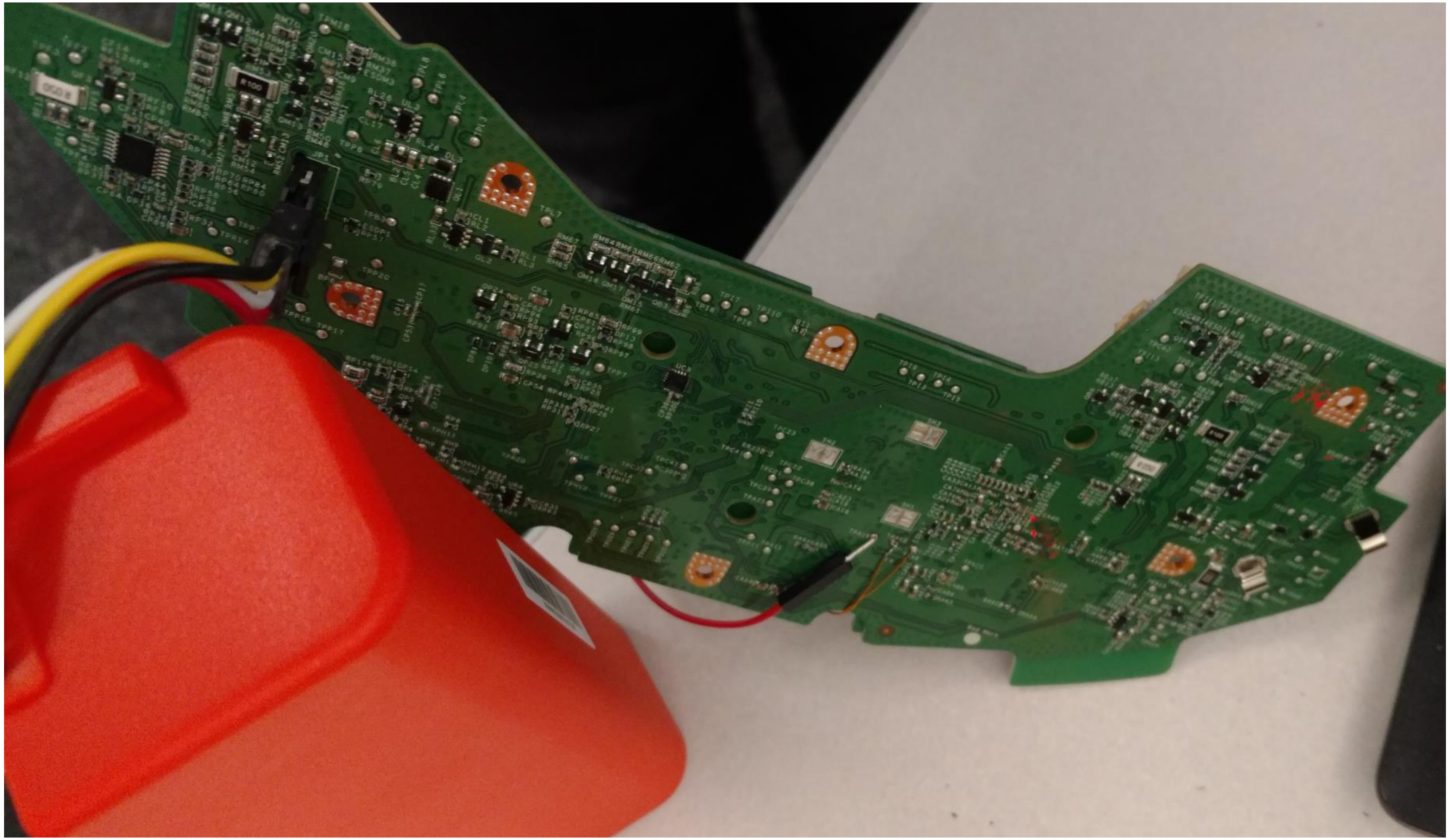
- Encrypting/Obfuscating the log-files and maps
- RRlogd uses AES encryption functions from OpenSSL library
  - Imported as dynamic library
  - Interesting function: EVP_EncryptInit_ex(...)
  - Ltrace can be used to intercept calls and extract arguments

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

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

Roborock S60/T60 UART setup



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

	Debug Interfaces			Firmware			Network		Physical		Data	
Device name	UART	JTAG/SWD	Telnet/SSH	Encrypted	Signed	Verified	HTTPS	Certificate checked	Tamper resistant	Tamper evident	User data not on device~	secure unprovisioning
Aqara Gateway (Homekit)	✓		✗	✗	✗	✓	✓	✓	✗	✓	✗	✗
Aqara Smart Home Gateway	✓		✗	✗	✗	✓	✓	✓	✗	✓	✗	✗
Aqara Smart IP Camera	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
Lumi Smart Home Gateway	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
Philips Ceiling Lamp	✓	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓	✓
Roborock S50	✓		✓	✓	✗	✓	✓	✓	✗	✓	✗	✗
Roborock S6/T61	✓		✓	✓	✓	✓	✓	✓	✗	✓	✗	✓
Xiaomi Mi Vacuum Robot	✓		✓	✓	✗	✓	✓	✓	✗	✓	✗	✗
Xiaomi Mi WiFi Speaker	✓		✗	✗	✗	✓	✓	✓	✗	✗	✗	✗
Xiaomi WiFi Plug	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Yeelink Bedside lamp	✓	✓	✗	✗	✗	✗	✓	✗	✓	✓	✓	✓
Yeelink Ceiling Lamp	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓
Yeelink Light Color	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Yeelink Light Mono1	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Yeelink Light Strip	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
Yeelink Smart White Bulb	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓
Yeelink Smart RGB Bulb	✓	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓

Outline

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

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

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

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

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

Questions?

IoT is when your AI powered toaster is mining bitcoin to pay off the gambling debts it has with your fridge.

Contact:

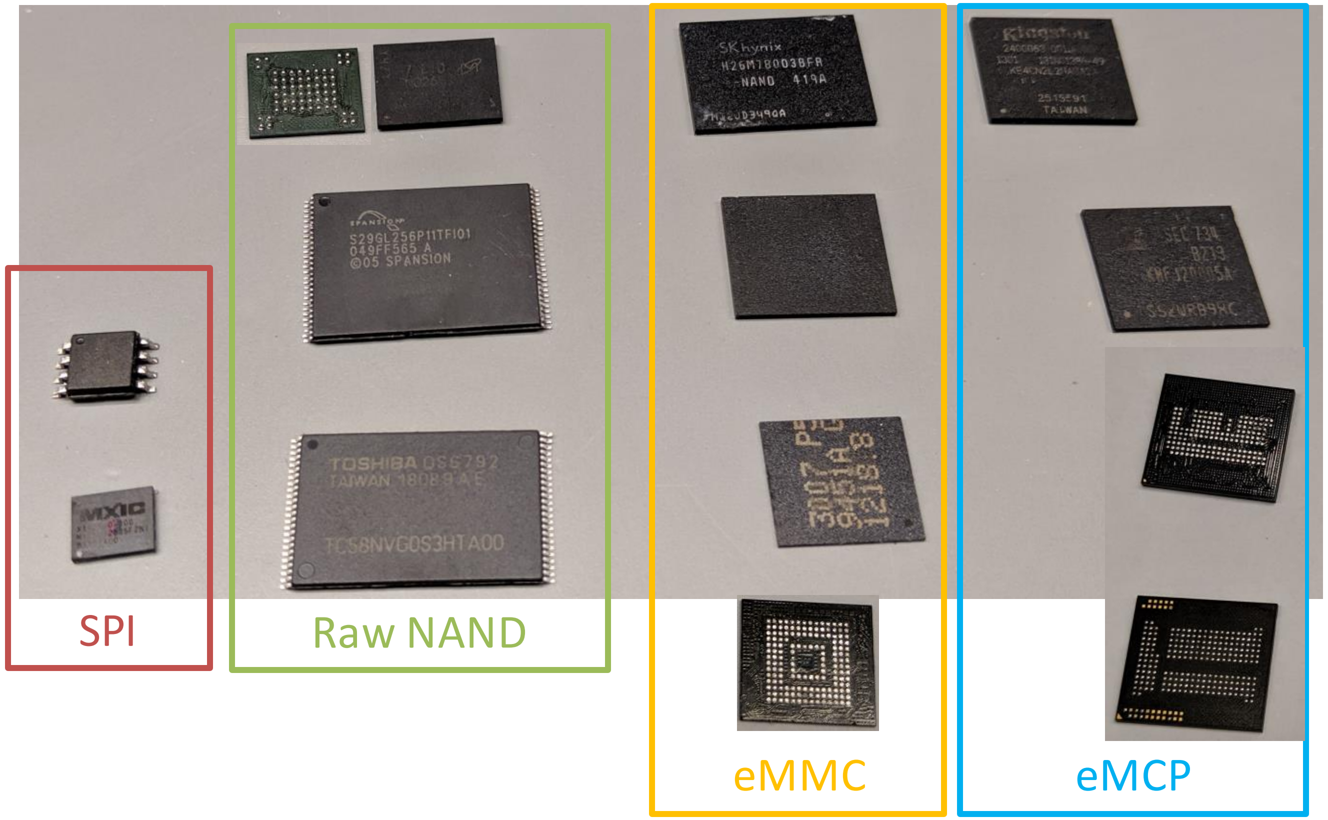
See: http://dontvacuum.me
Telegram: https://t.me/dgiese
Twitter: dgi_DE
Email: dgiese@ccs.neu.edu

Short intro

STORAGE ON IOT DEVICES

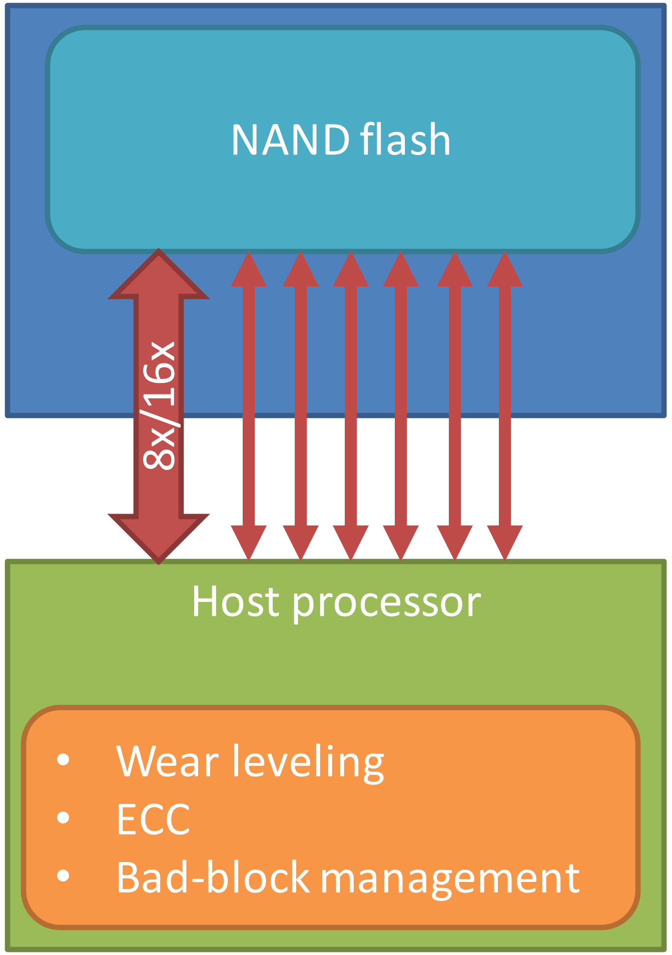
Storage on IoT devices

- 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



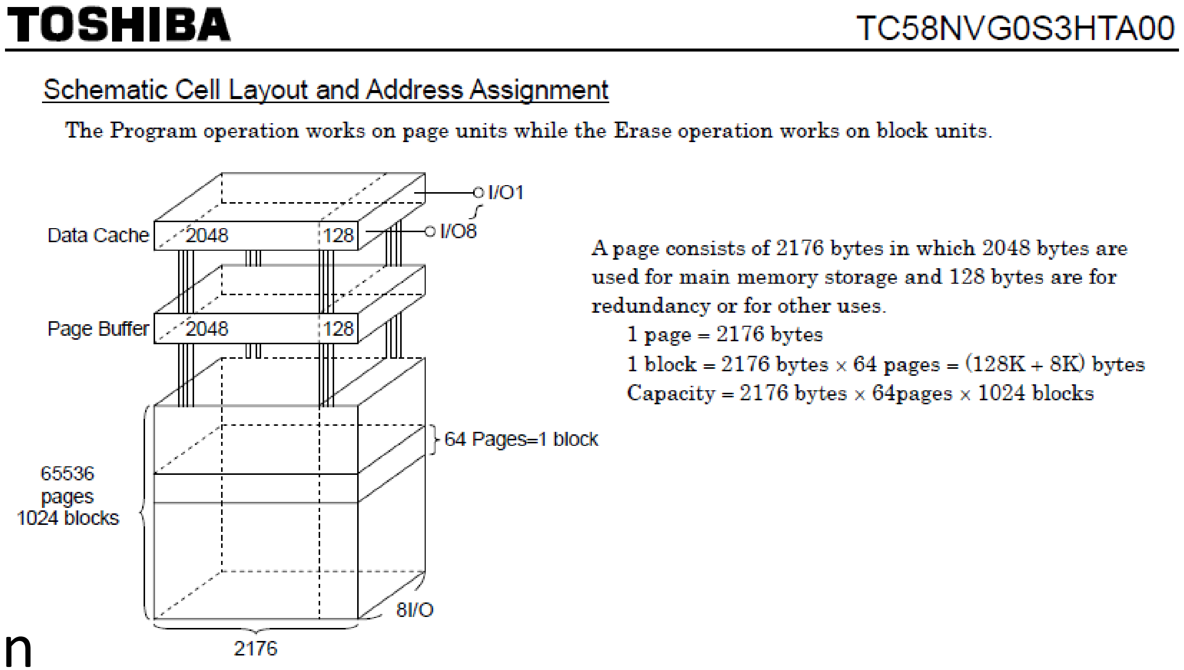
Raw NAND flash

- 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



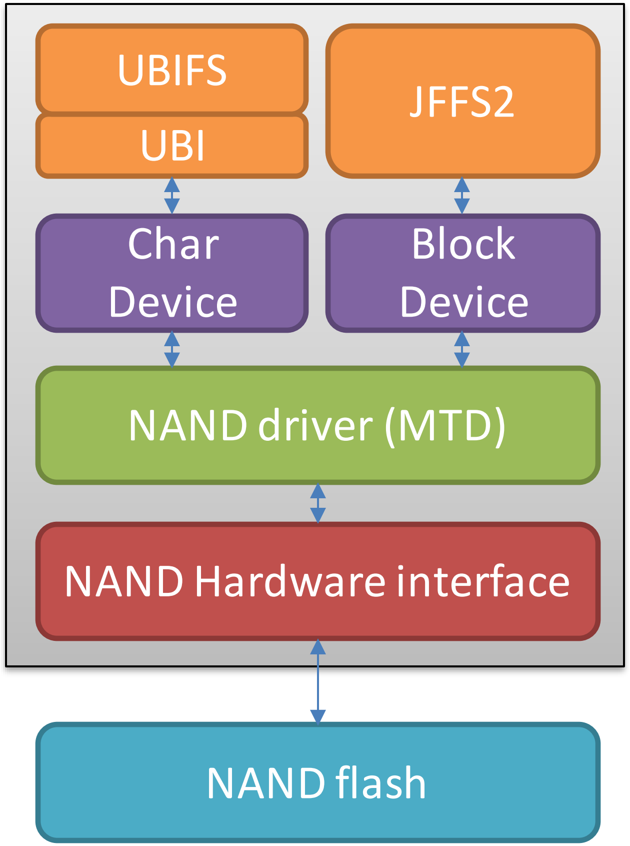
Raw NAND flash properties

- 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

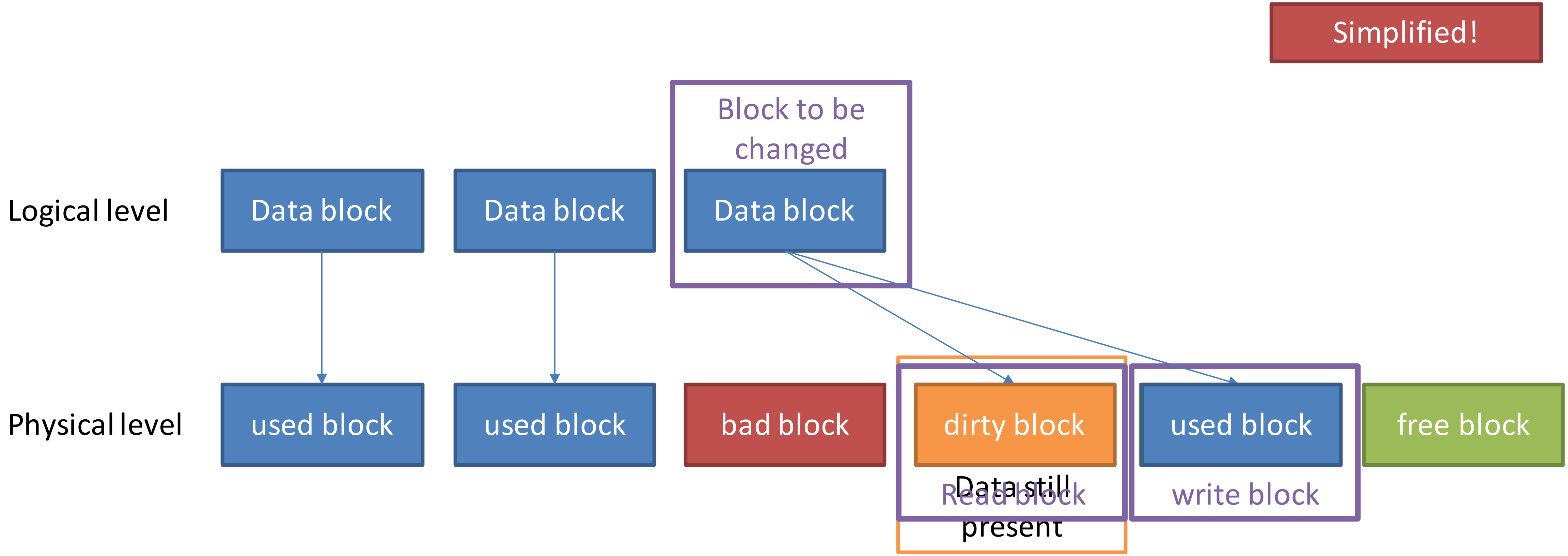


Wear-leveling for raw flash

- 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



How Wear-leveling works



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

    https://www.blackhat.com/docs/us-14/materials/us-14-Oh-Reverse-Engineering-Flash-Memory-For-Fun-And-Benefit-WP.pdf
    https://www.blackhat.com/docs/us-14/materials/us-14-Oh-Reverse-Engineering-Flash-Memory-For-Fun-And-Benefit.pdf
- “From NAND chip to files” by Jean-Michel Picod

  https://www.j-michel.org/blog/2014/05/27/from-nand-chip-to-files