HACKFEST

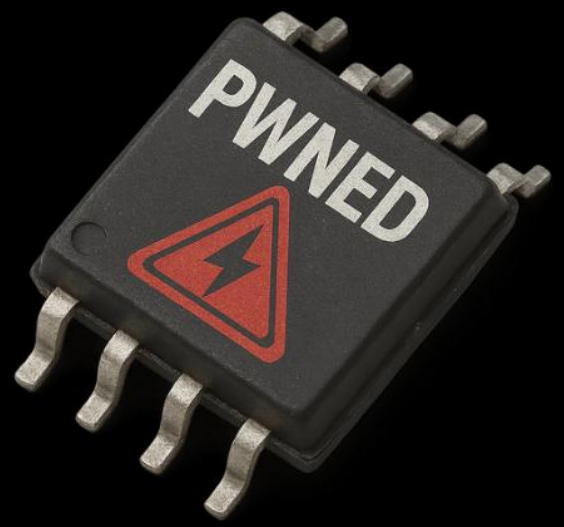
IN THE SHELL

ATTACKING HOME AUTOMATION

CONSUMER-GRADE VULNERABILITIES

Presenter: Elysee Franchuk

MOBIA



About me

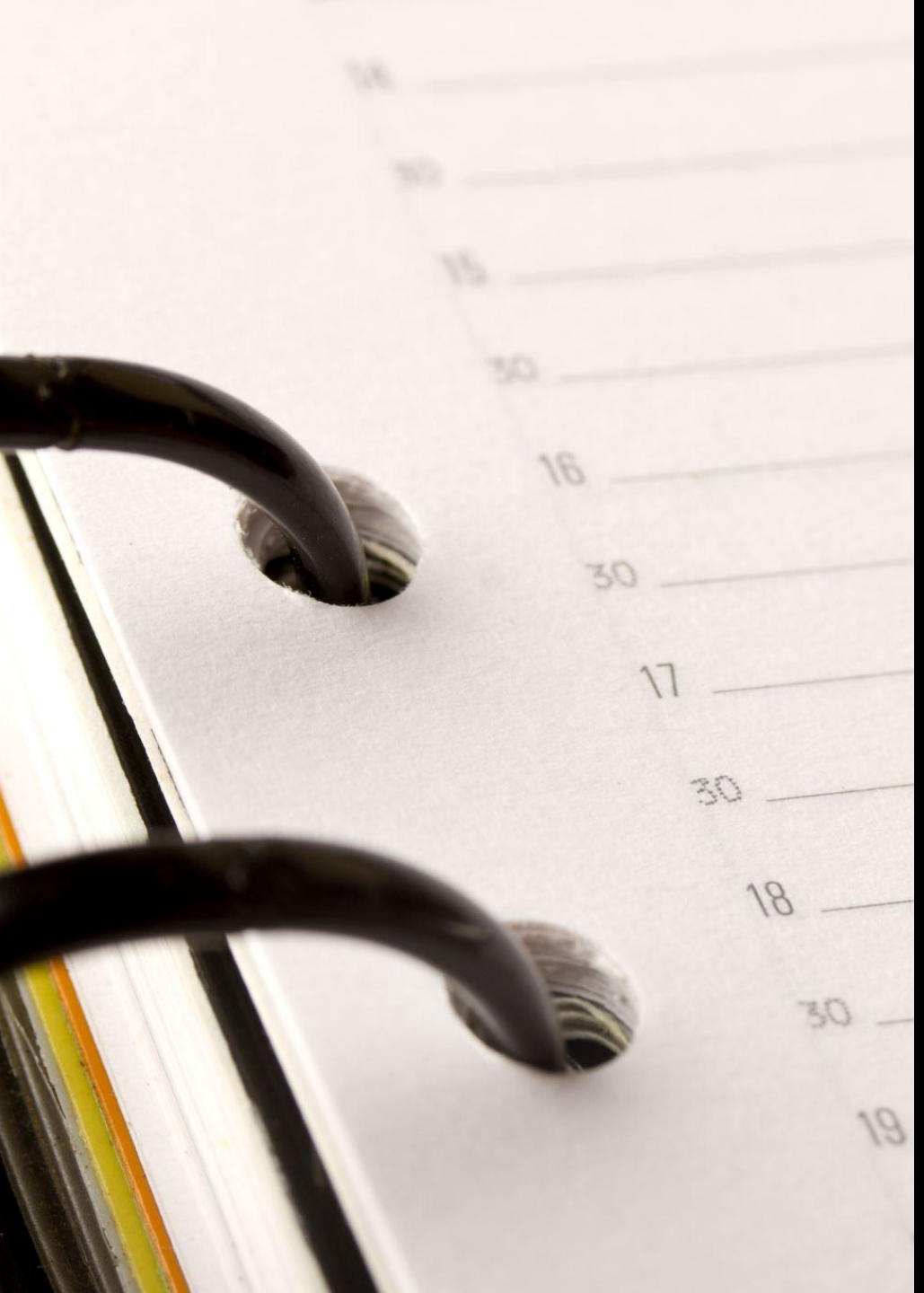
Elysee Franchuk, Penetration Tester @ MOBIA Technology Innovations
(OSCP, OSWE, OSWP, CISSP, GPEN)

Interests:

- Breaking stuff
- Fixing stuff
- Video games
- Spending time with family

Previous Talks:

- BSides Calgary 2025
- BSides Las Vegas 2024

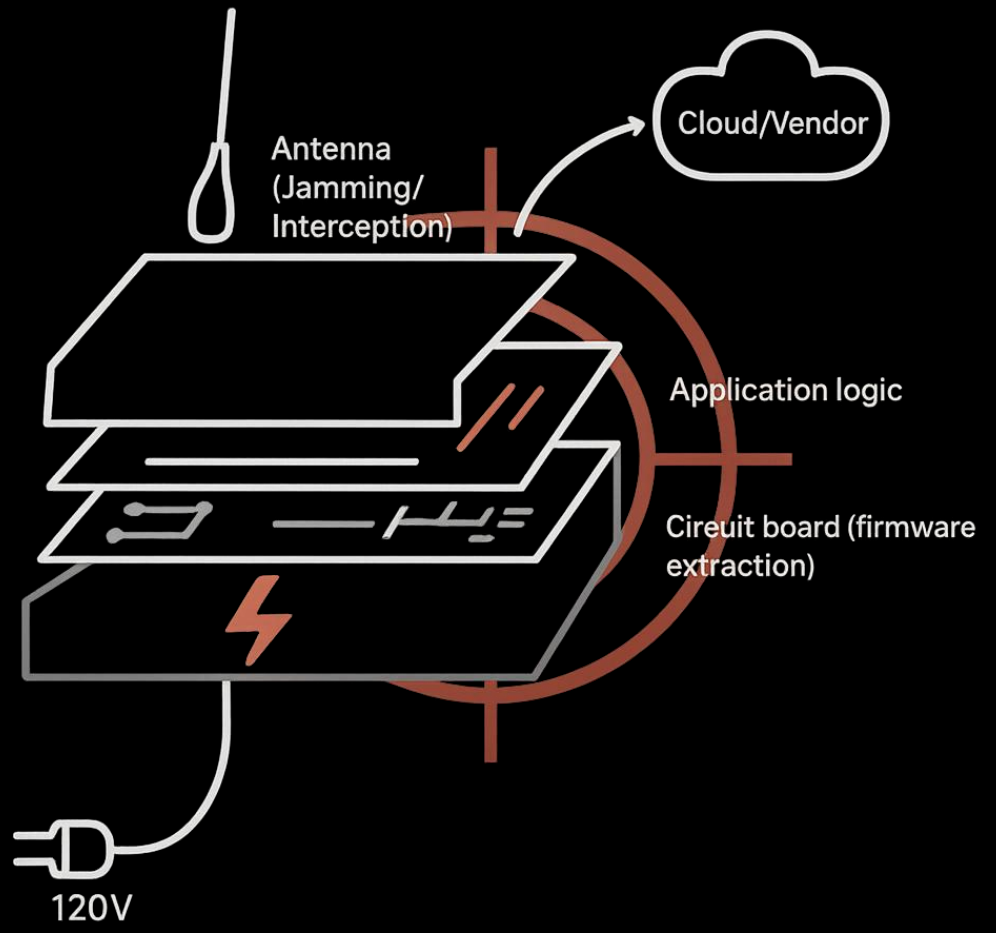


Important Notes:

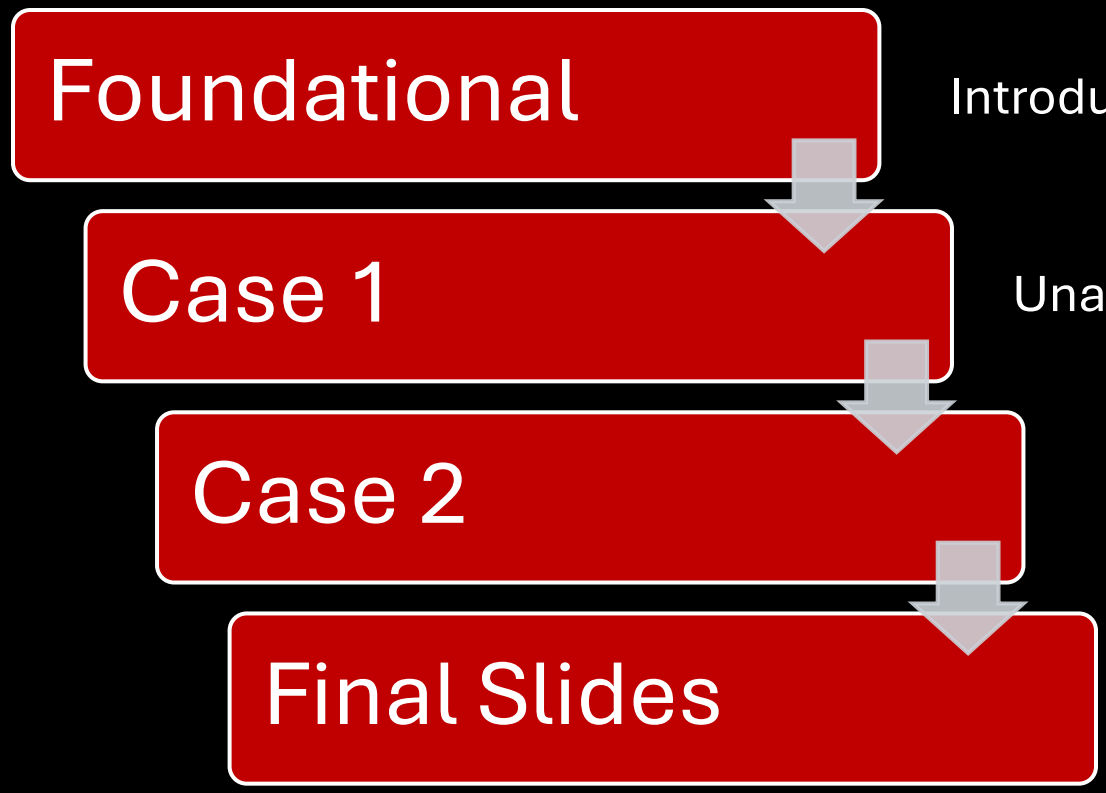
- Some of the findings here are redacted.
  - Version, product, company
  - No zero days are dropped (sorry)

Goals of this talk

- The goal of this talk:
  - Share methodologies
  - Share what threat actors are capable of
  - Showcase findings (both published, and redacted)
  - Spark discussions on IoT / Home Automation security
  - How discussed vulnerabilities can relate to OT/ICS



Schedule



Introduction to IoT infrastructure, systems, and dataflows

Unauthenticated command execution

Extracting sensitive data from firmware

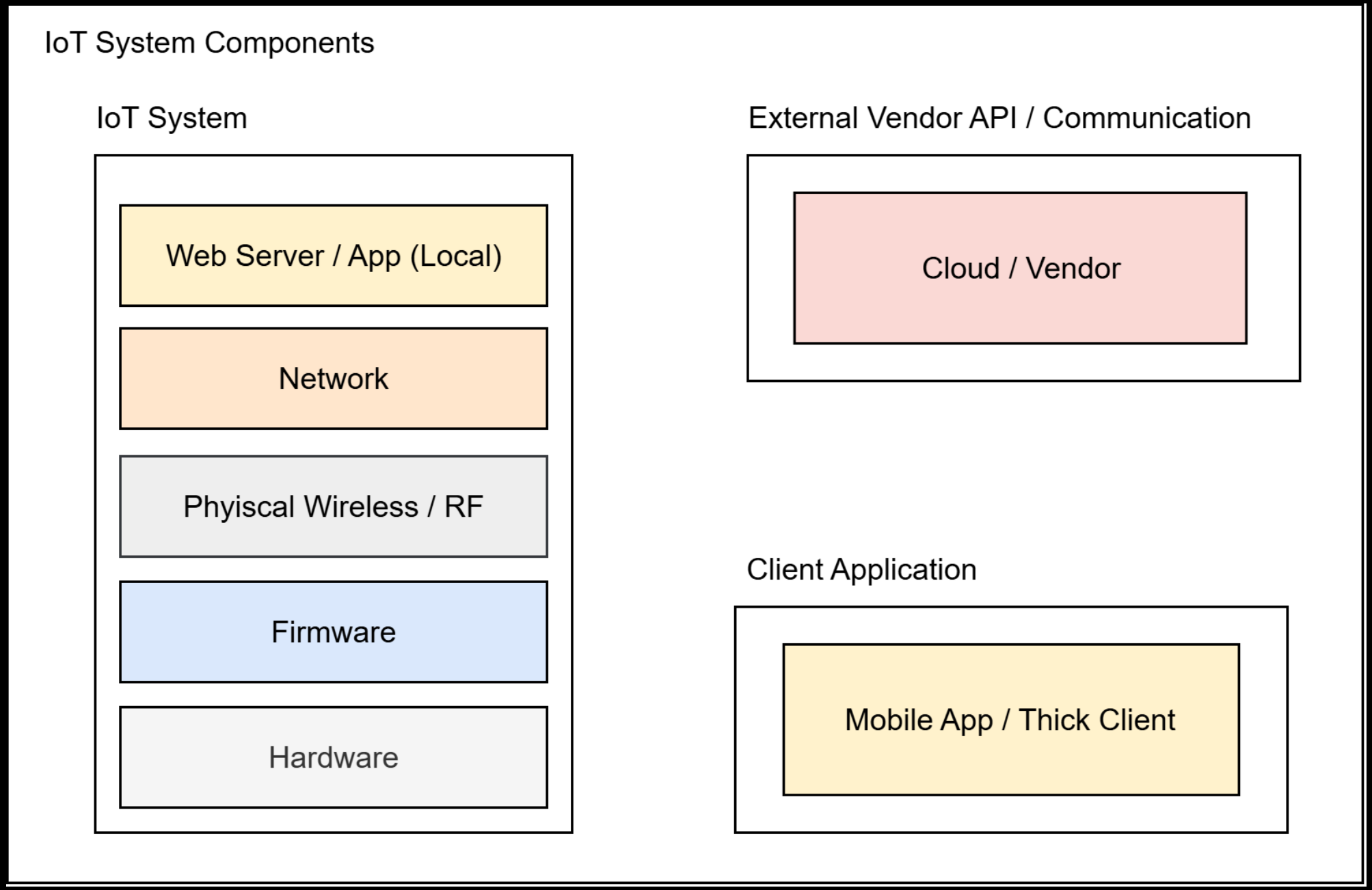
Additional insights, tips on disclosures, questions

Devices

- TP Link Power Strip (KP303)
- Censored Power Outlet Device
- Censored Light Switches
- General IoT Wi-Fi enabled devices

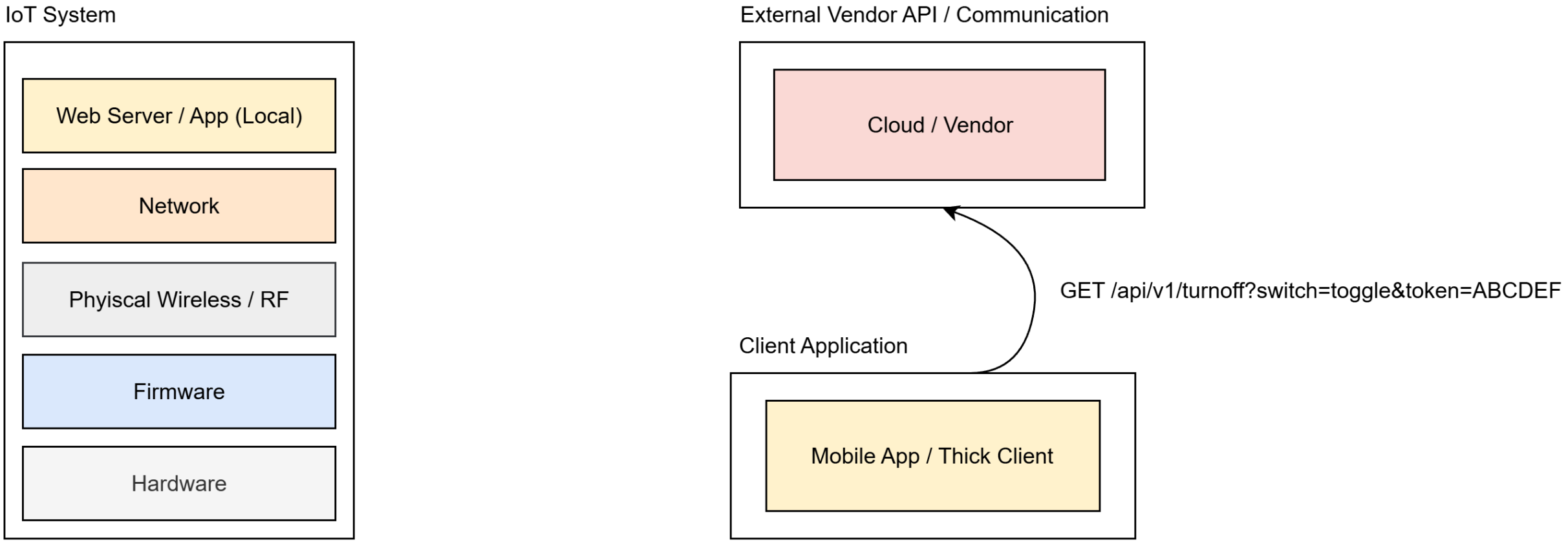
What is Home Automation?

- Lighting
- Climate controls
- Entertainment Systems
  - Televisions
- Access Control / Alarms
- Video
- Washing and Drying Machines
- Vacuum Cleaners
- Kitchen Appliances

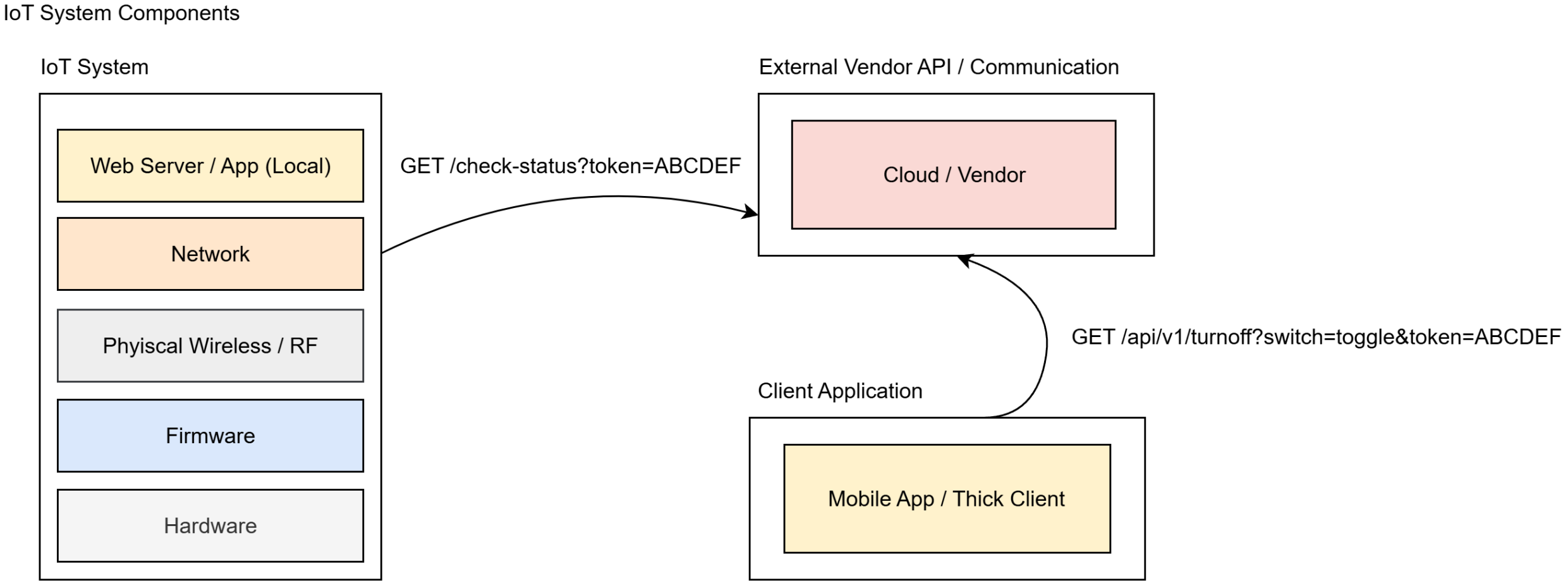

IoT System Components
IoT System
Web Server / App (Local)
Network
Phyiscal Wireless / RF
Firmware
Hardware
External Vendor API / Communication
Cloud / Vendor
Client Application
Mobile App / Thick Client

Basic Command Input via Mobile/Cloud(1)

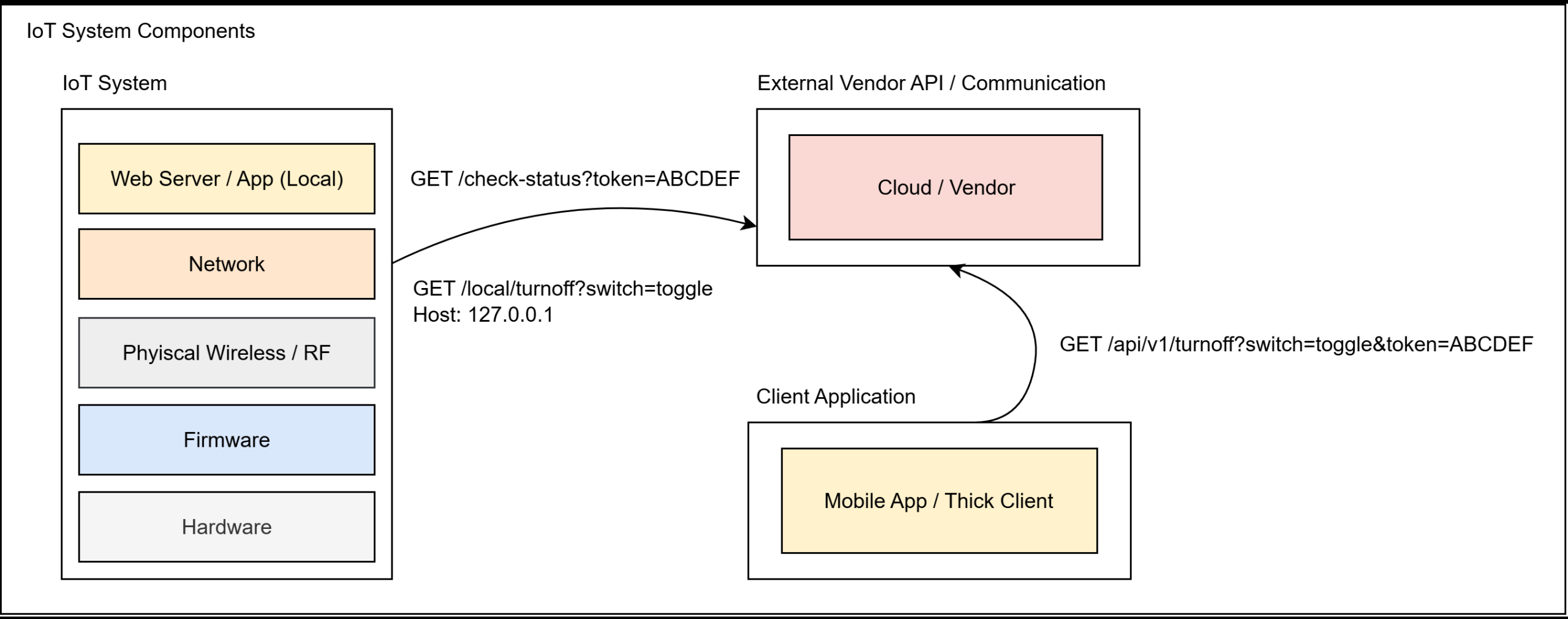
IoT System Components - Example Commands



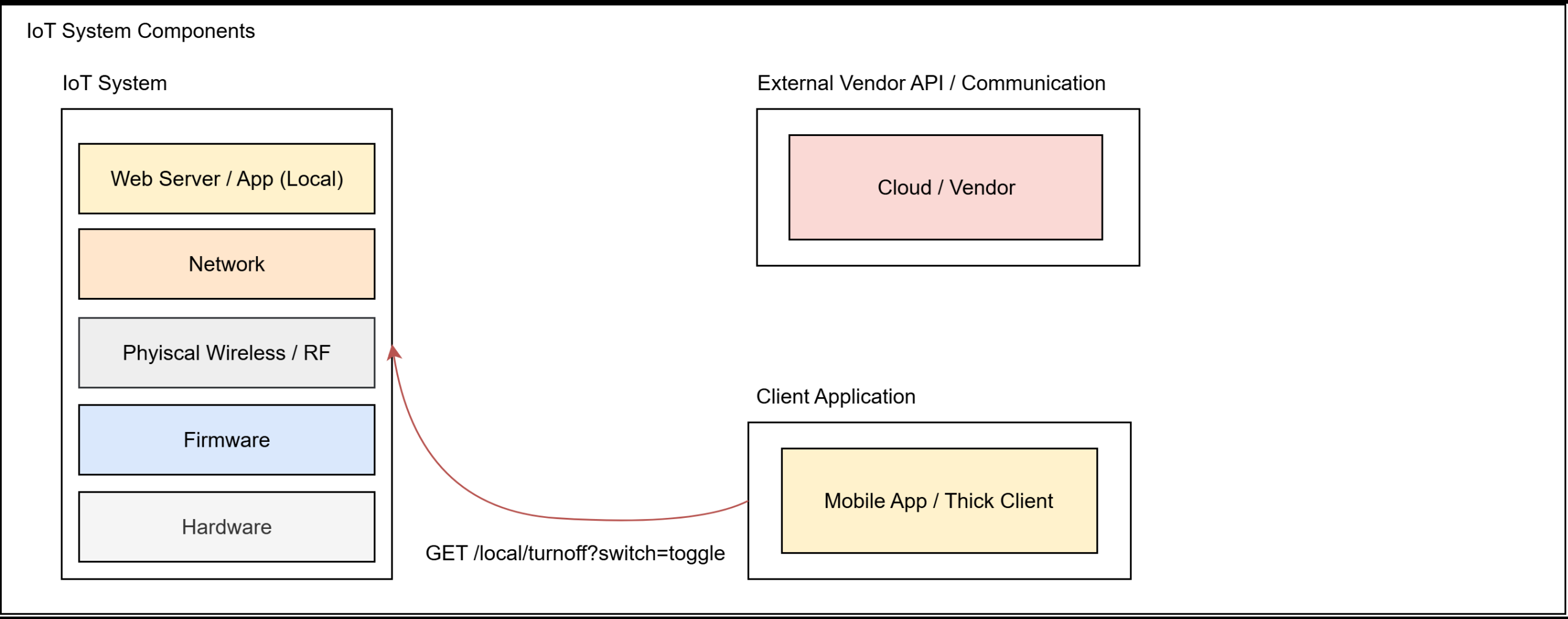
Basic Command Input via Mobile/Cloud(2)



Basic Command Input via Mobile/Cloud(3)

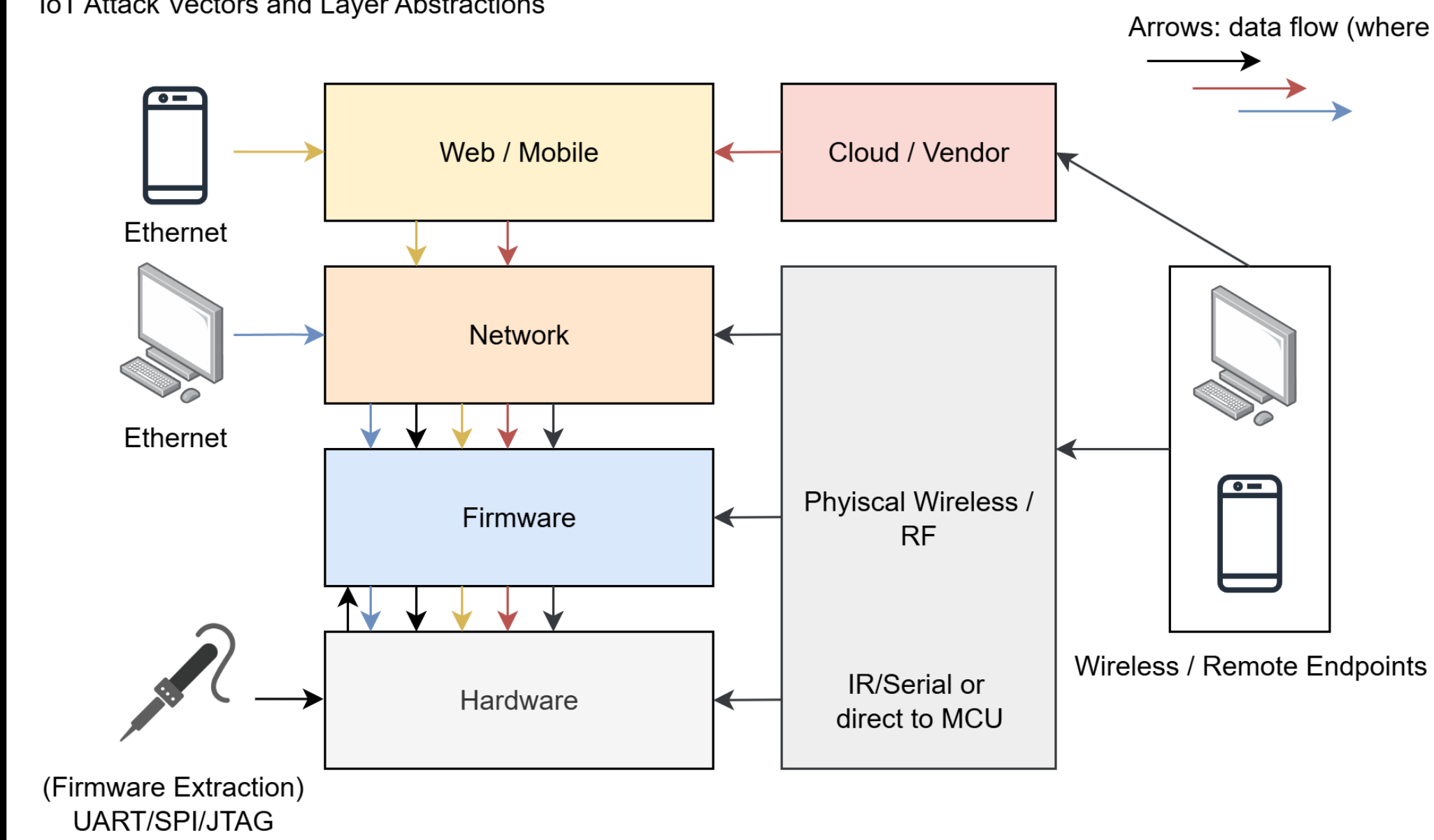


Basic Command Input via Local Communication



Question: is this good, or bad?

IoT Attack Vectors and Layer Abstractions



Communication Protocols

- RF (Radio Frequencies)
  - 433 MHz
  - Zigbee
  - BLE
- Wi-Fi
  - Monitor mode – Outbound requests
    - Beacon Frames
    - Probe Requests
    - Deauthentication
    - Association
  - Managed Mode
    - TCP/UDP
    - Application Protocols / Data
- Network / Ethernet / LAN (Layer 3 / Layer 2)
  - TCP/UDP
  - Application Protocols / Data
- Application Protocols
  - HTTP/S
  - MQTT
  - CoAP
  - Websockets
- Hardware Protocols
  - UART
  - RS-232
  - I^2C
  - SPI

- Jamming Attacks
- Relay Attacks

- Evil Twin / Rogue AP
- Deauthentication Attack
- Key Interception

- Sniffing / Interception
- Exhaustion Attacks
- Attacking Network Services
- Command Injection

- OWASP 10
- DoS (Denial of Service)
- Command Injection
- Interaction via Cloud

- Dumping / Overwriting Firmware
- Low Level Shell Interaction

This is not the OSI Layers.

The lower layer you're targeting the more access you need to a system

Check out OWASP ITSG for a better image.

Couple Notes on Attack Vectors

- Intercepting Communication – Client to Vendor
- Real-world attacks sometimes do not line up one-to-one with CVSS

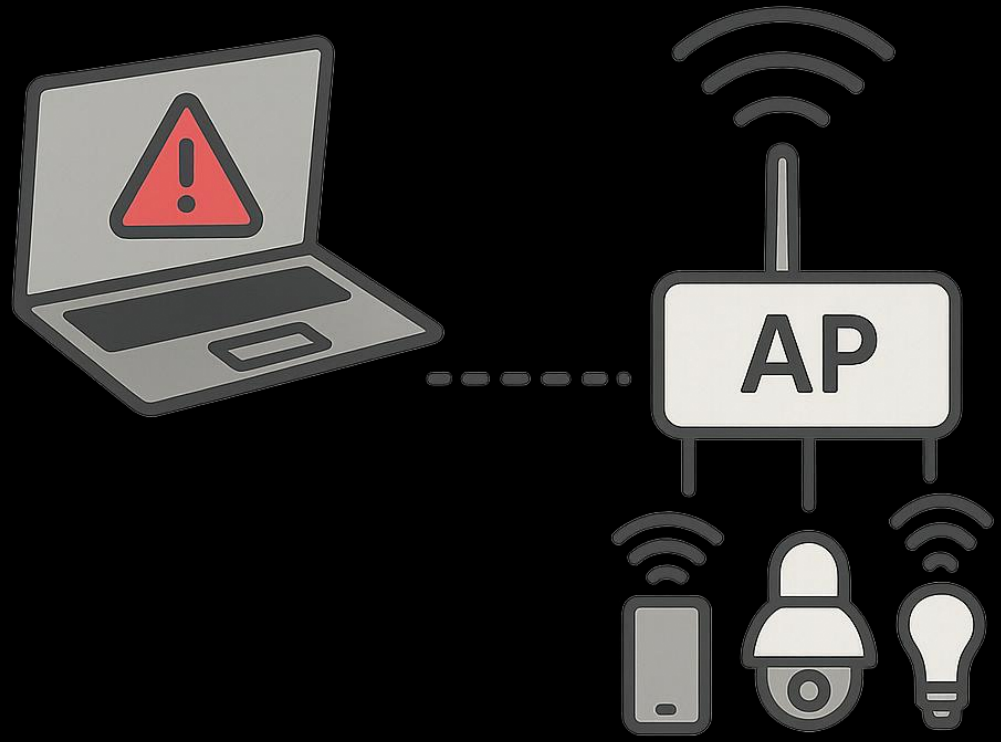
Attack Vector (AV)*			
Network (AV:N)	Adjacent Network (AV:A)	Local (AV:L)	Physical (AV:P)

Low Effort Attacks - Levels of Jamming:

- Dauthentication / disassociation
- Evil Twin
- Various DoS attacks (someone is just trying to knock it off the AP)
- 802.11 jamming*

Attacking Clients vs. APs

Attacking APs vs. Clients



Case 1: Performing Unauthenticated Command Execution

Censored Power Outlet Device

Unauthenticated Commands with Kasa

python-kasa

pypi package 0.10.2 | CI passing | codecov 93% | docs passing

python-kasa is a Python library to control TPLink's smart home devices (plugs, wall switches, power strips, and bulbs).

This is a voluntary, community-driven effort and is not affiliated, sponsored, or endorsed by TPLink.

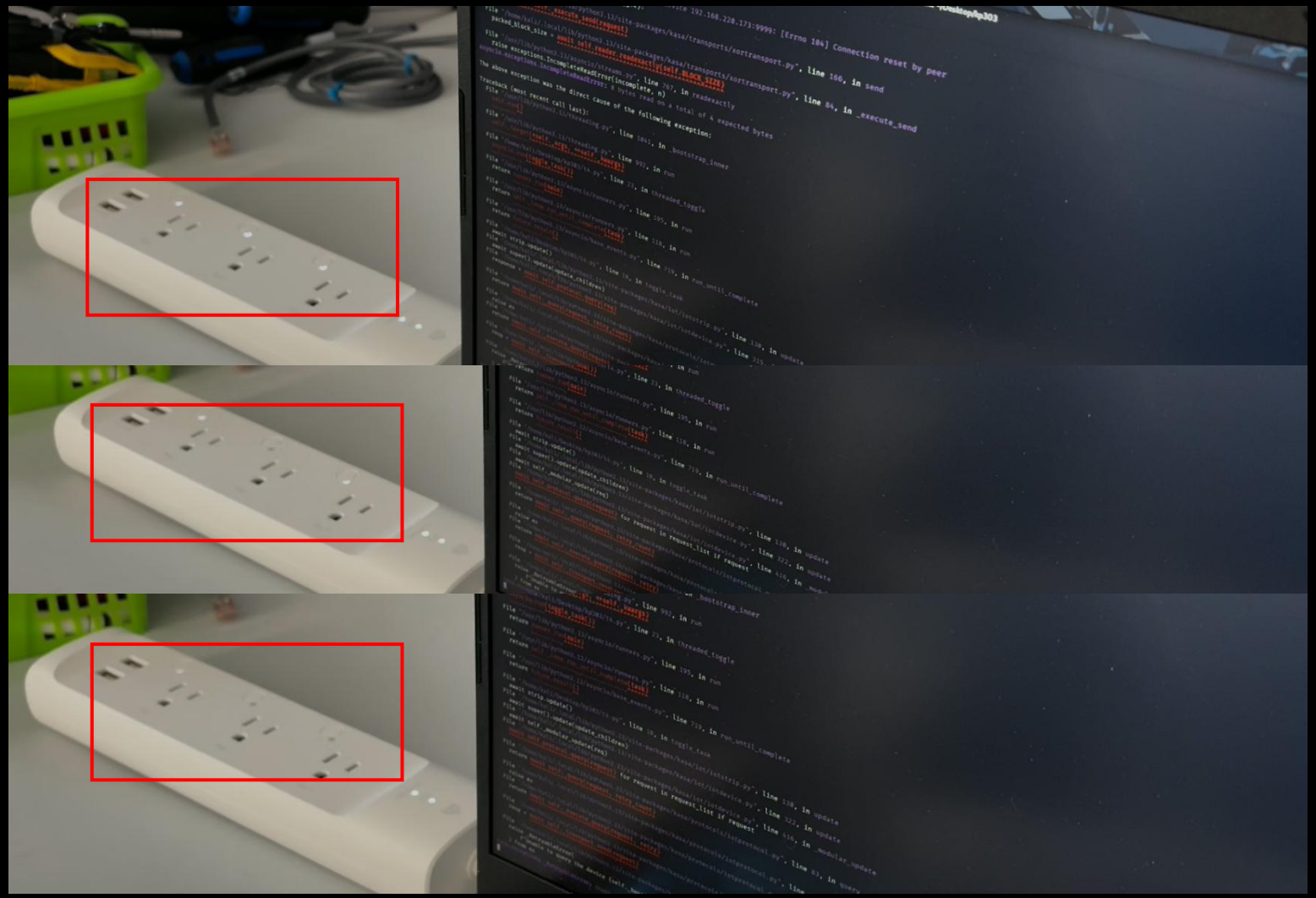
Contributions in any form (adding missing features, reporting issues, fixing or triaging existing ones, improving the documentation, or device donations) are more than welcome!

```
┌──(kali㉿kali)-[~/Desktop/kp303]
└─$ kasa
No host name given, trying discovery..
Discovering devices on 255.255.255.255 for 10 seconds
== - KP303 ==
Host: 192.168.220.173
Port: 9999
Device state: True
Time:         2025-04-25 17:32:28-06:00 (tz: MST7MDT)
Hardware:     2.0 (US)
Firmware:     1.0.9 Build 240131 Rel.141407
MAC (rssi):   DC:62:79:88:56:9D (-47)
```

```
┌──(kali㉿kali)-[~/Desktop/kp303]
└─$ kasa --host 192.168.220.173 device off
Discovering device 192.168.220.173 for 10 seconds
Turning off
```

Demo

Screenshots Demo



Good ole intended functionality (its not)

- How to verify something is intended?
  - Compare against products / protocols
  - Does functionality align with a CWE?
  - Without authentication – device takeover was possible

Questions I asked during testing:

- Should rate limitation be on engaging/disengaging relays?
- Is there controls to detect overheating?
- What would happen if it was connected to another device, such as a 3D Printer?

Required CVE Record Information

CNA: TP-Link Systems Inc. –

Published: 2025-08-25 **Updated:** 2025-08-25
Title: Unauthenticated Protocol Commands On TP-Link KP303

Description

The TP-Link KP303 Smartplug can be issued unauthenticated protocol commands that may cause unintended power-off condition and potential information leak. This issue affects TP-Link KP303 (US) Smartplug: before 1.1.0.

CVSS 1 Total

Learn more

Score	Severity	Version	Vector String
8.7	HIGH	4.0	CVSS:4.0/AV:A/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:H/SC:N/SI:N/SA:N

What the Patch Prevents

- This fix from TP Link basically prevents unauthenticated command execution regardless if attackers:
  - take over the AP
  - Attack the client directly.
- Even if the device is on an isolated guest or IoT subnet, this fix hardens it.
- The TP Link device can still be issued commands, but your cloud credentials are needed now.

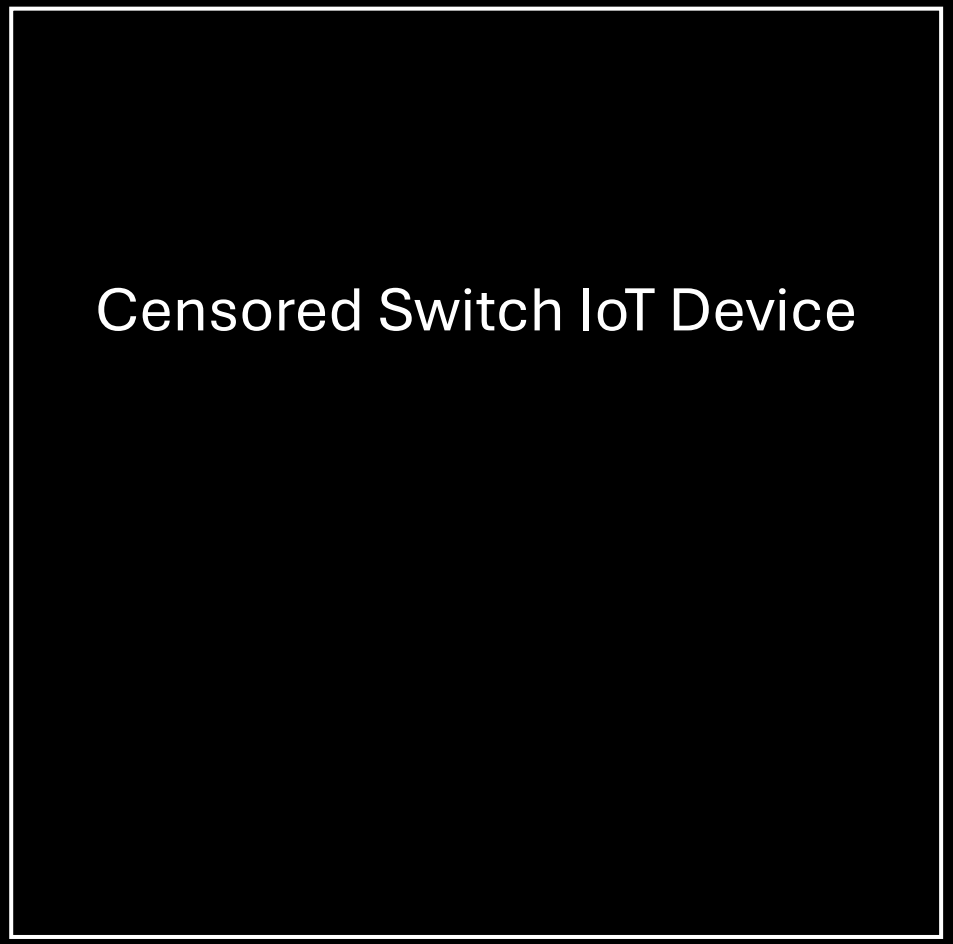
Additional findings

- This fix patch also fixed the unauthenticated email scraping:

```
┌──(kali㉿kali)-[~/Desktop/kp303]
└─$ python3 protocol_test.py
/home/kali/Desktop/kp303/protocol_test.py:4: DeprecationWarning: SmartStrip is deprecated, use IotStrip from package
ce.connect() to support new protocols
  from kasa import SmartStrip
{'cnCloud': {'get_info': {'username': 'elyseemafranchuk@gmail.com', 'server': 'n-devs.tplinkcloud.com', 'binded': 1,
cspStatus': 1, 'fwDlPage': '', 'tcspInfo': '', 'fwNotifyType': -1, 'err_code': 0}}}

┌──(kali㉿kali)-[~/Desktop/kp303]
└─$
```

- Another device that connected directly to your wall was vulnerable to this… the patch was supposed to be deployed in September…

Case 2: Challenges of Hardware Hacking:



Inconsistent and Changing PCBs:

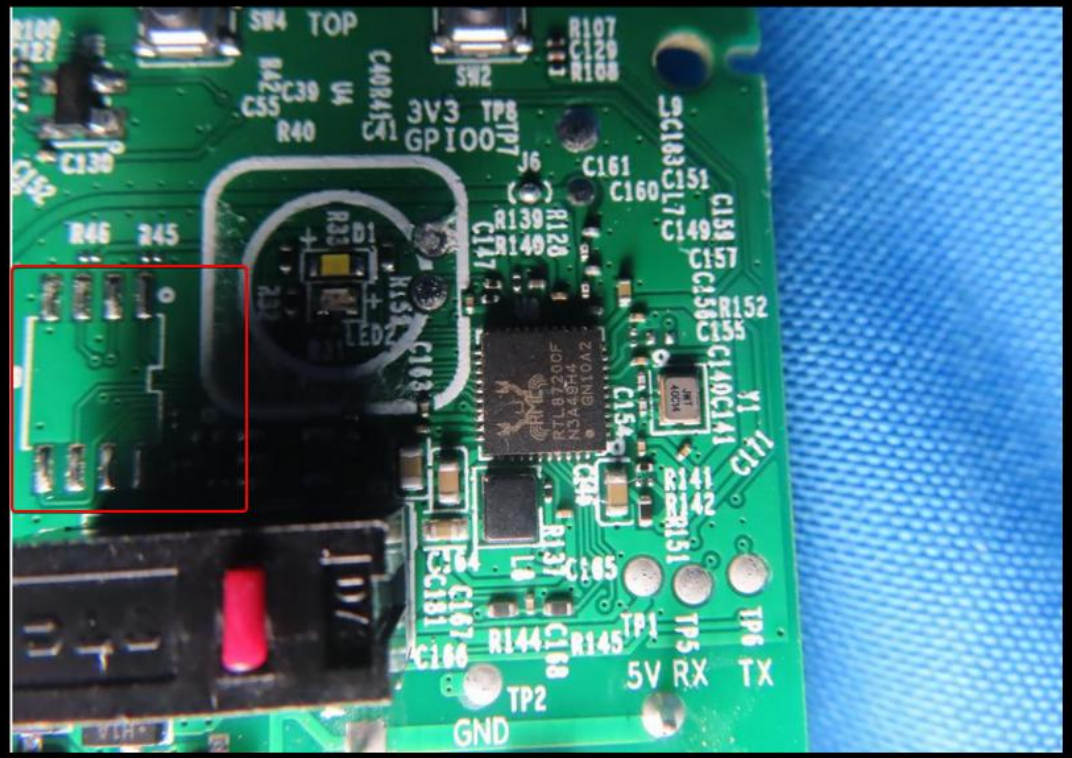
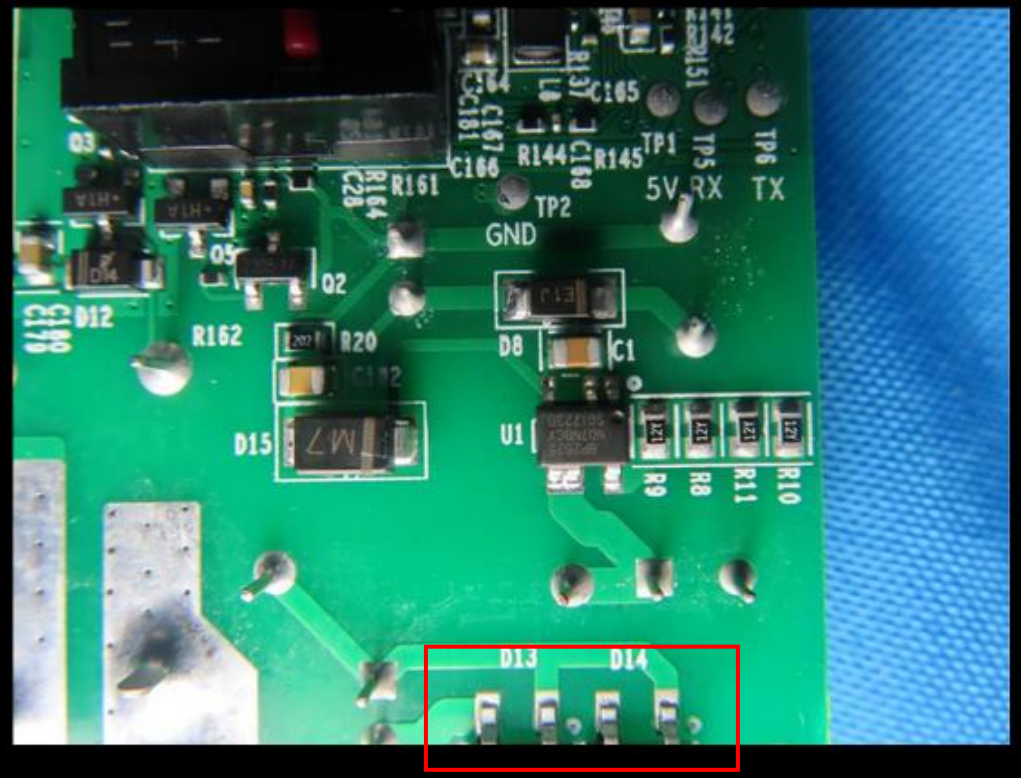
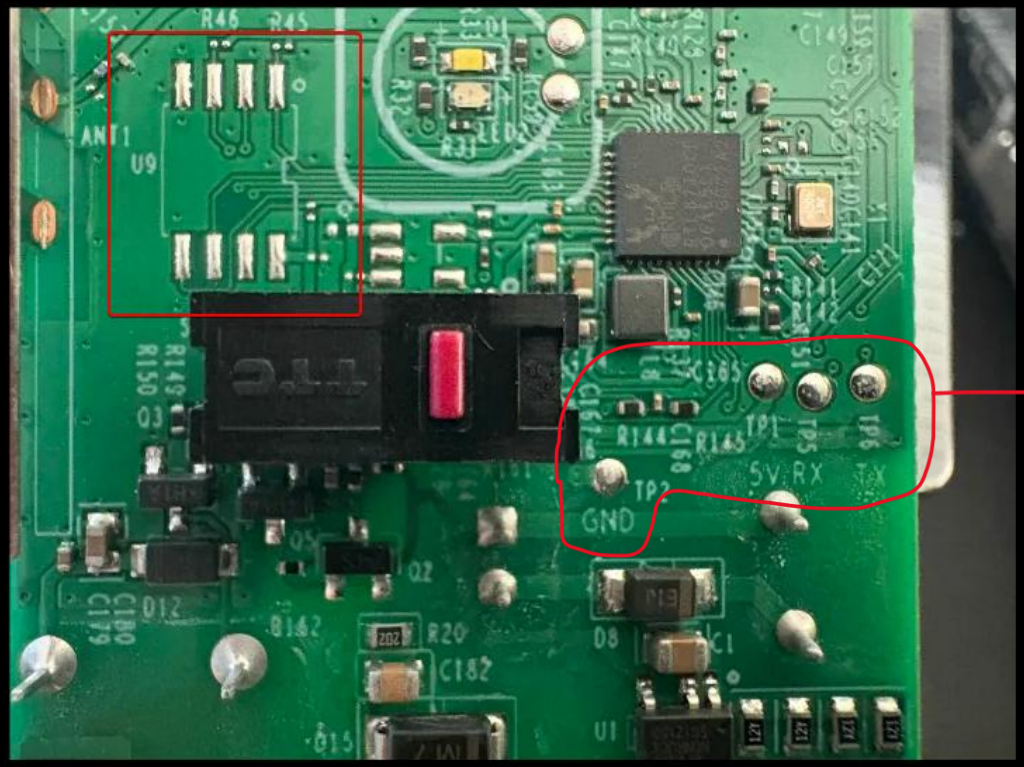


Image shows Internal PCB with missing BIOS chip.

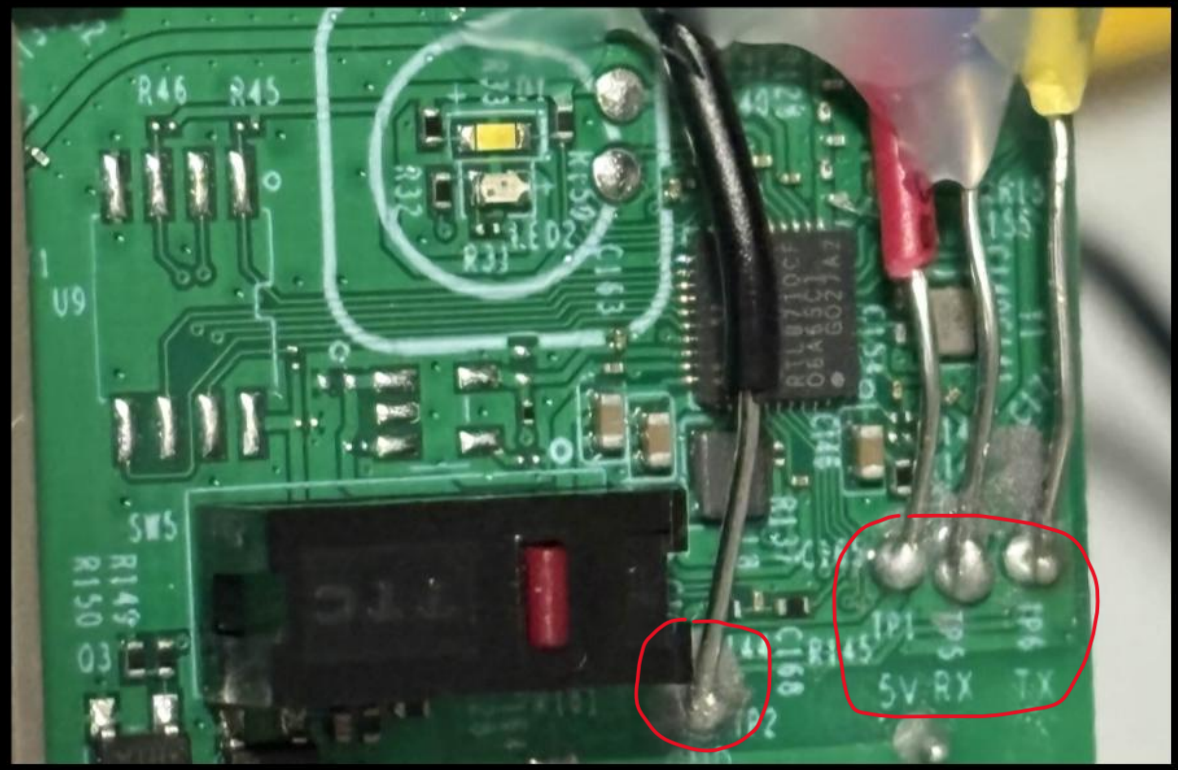


While another from the same FCC document has BIOS chips in the frame.

Inconsistent and Changing PCBs:

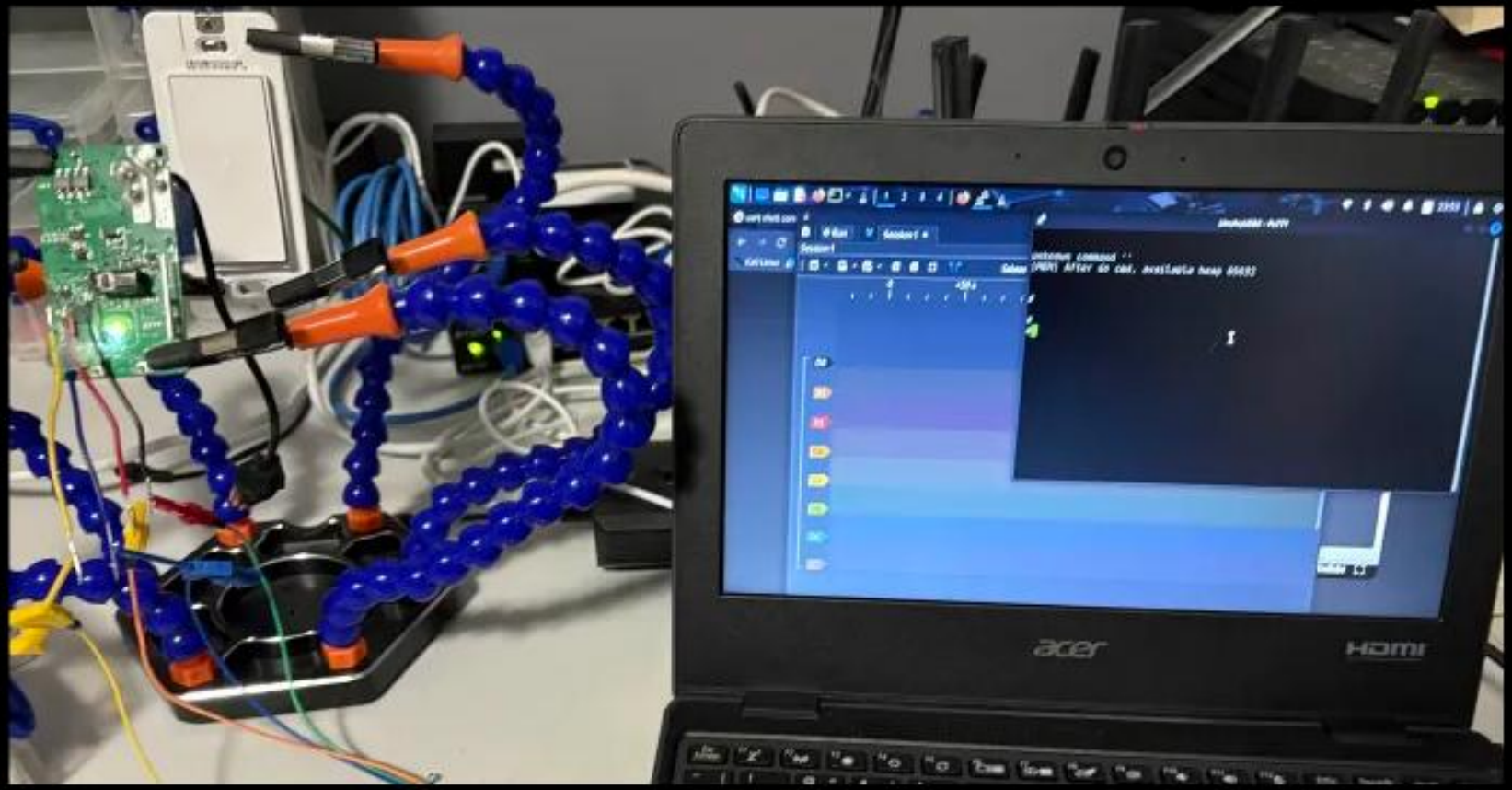


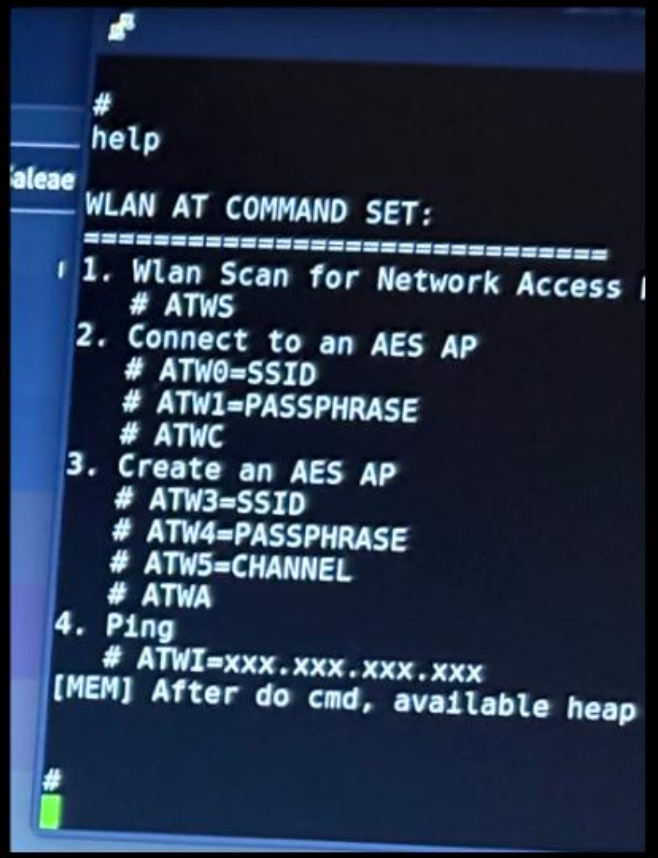
My device did not have this 8-pin BIOS chip.



But it did have UART pins labeled GND, 5V, RX, TX

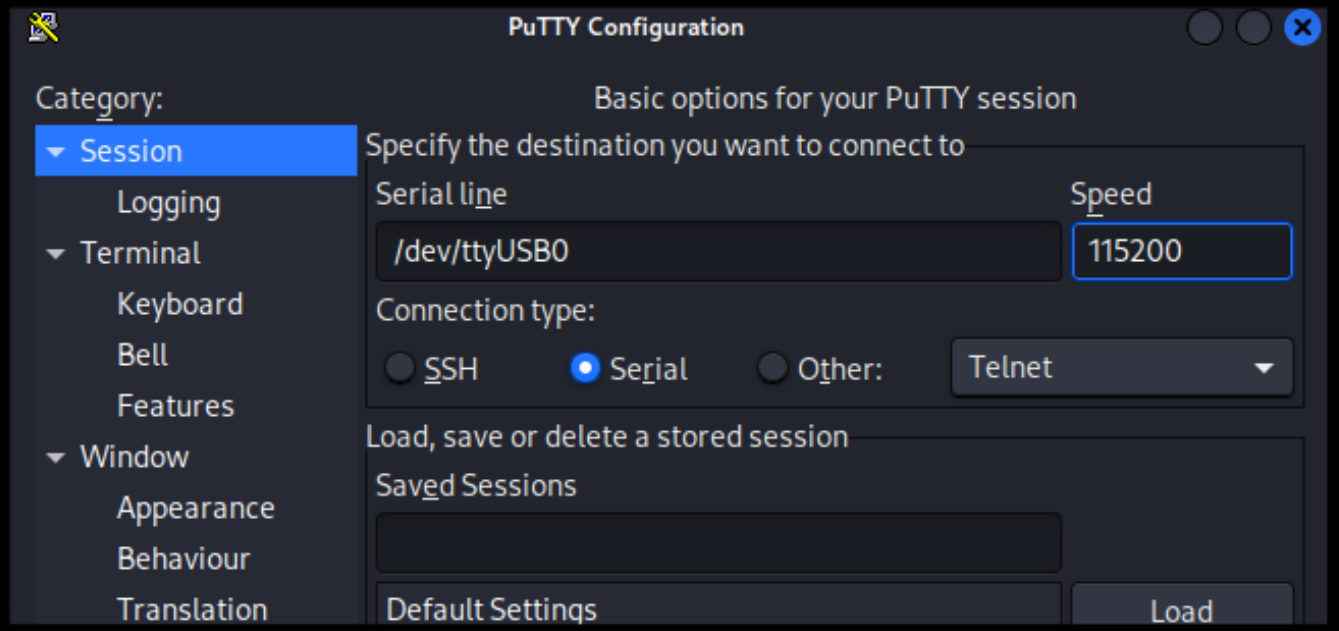
Achieving a UART Shell:

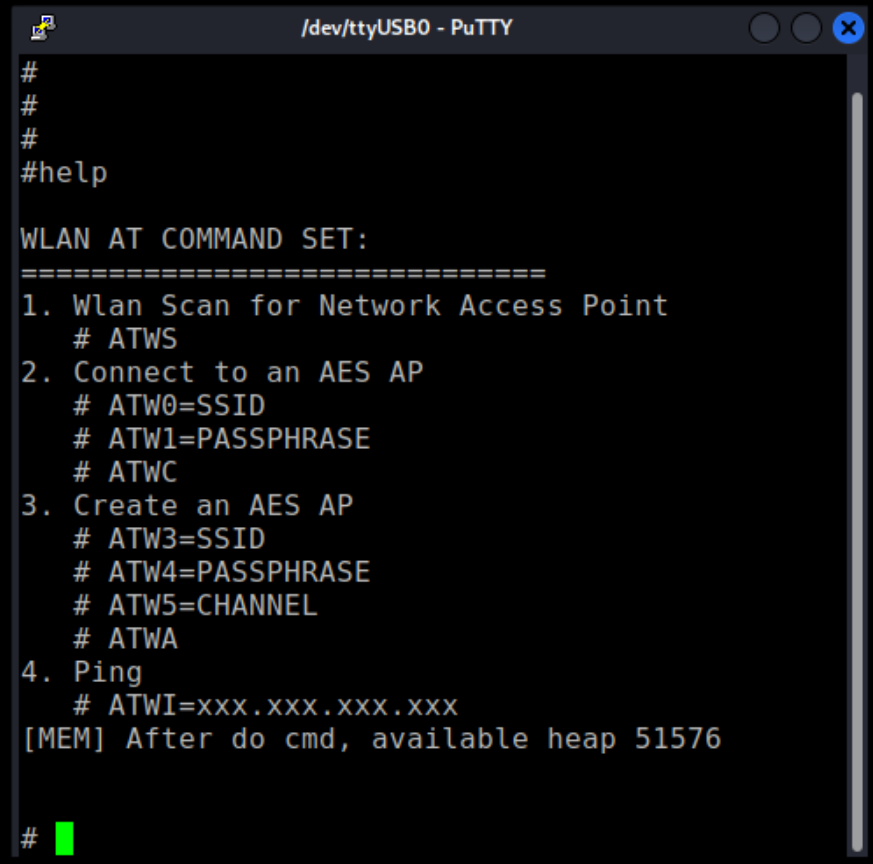




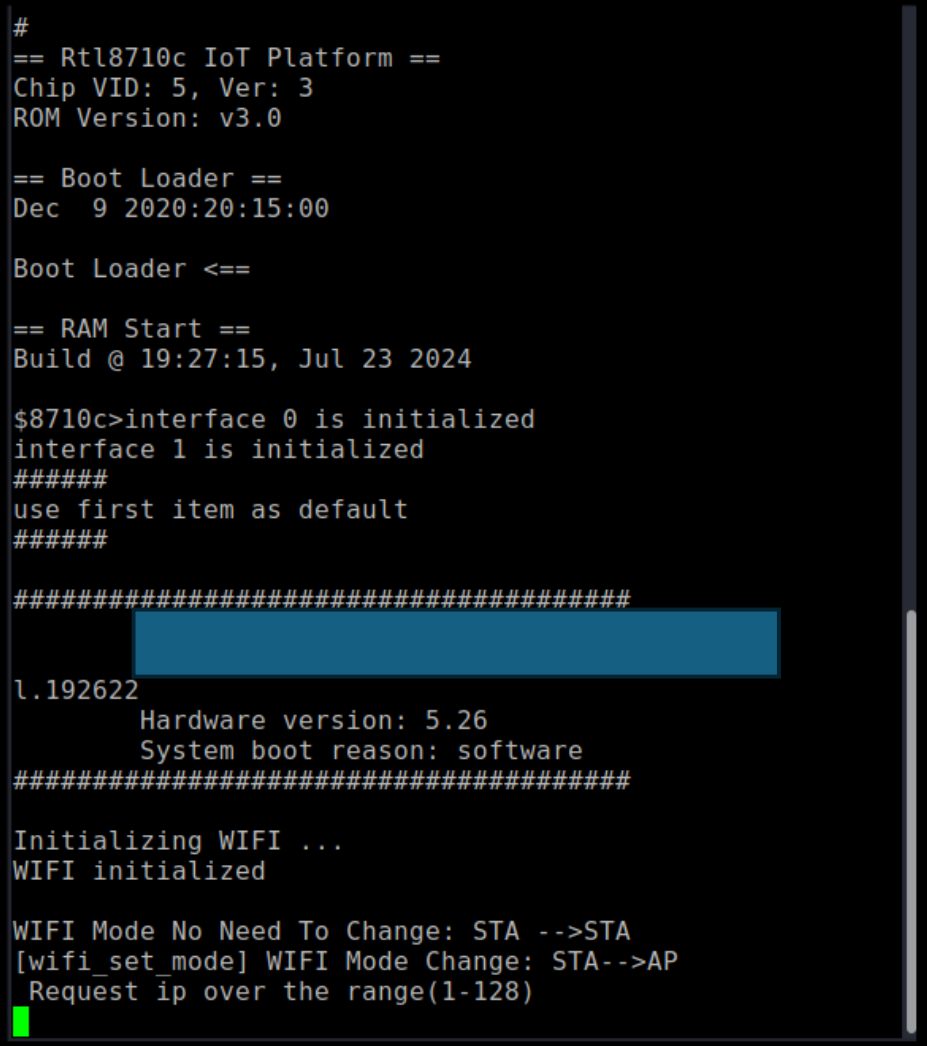
It was possible to get a limited UART shell with Putty. The actual interface had some available options, to connect to an SSID, create an SSID, ping an IP, and scan for wireless APs.

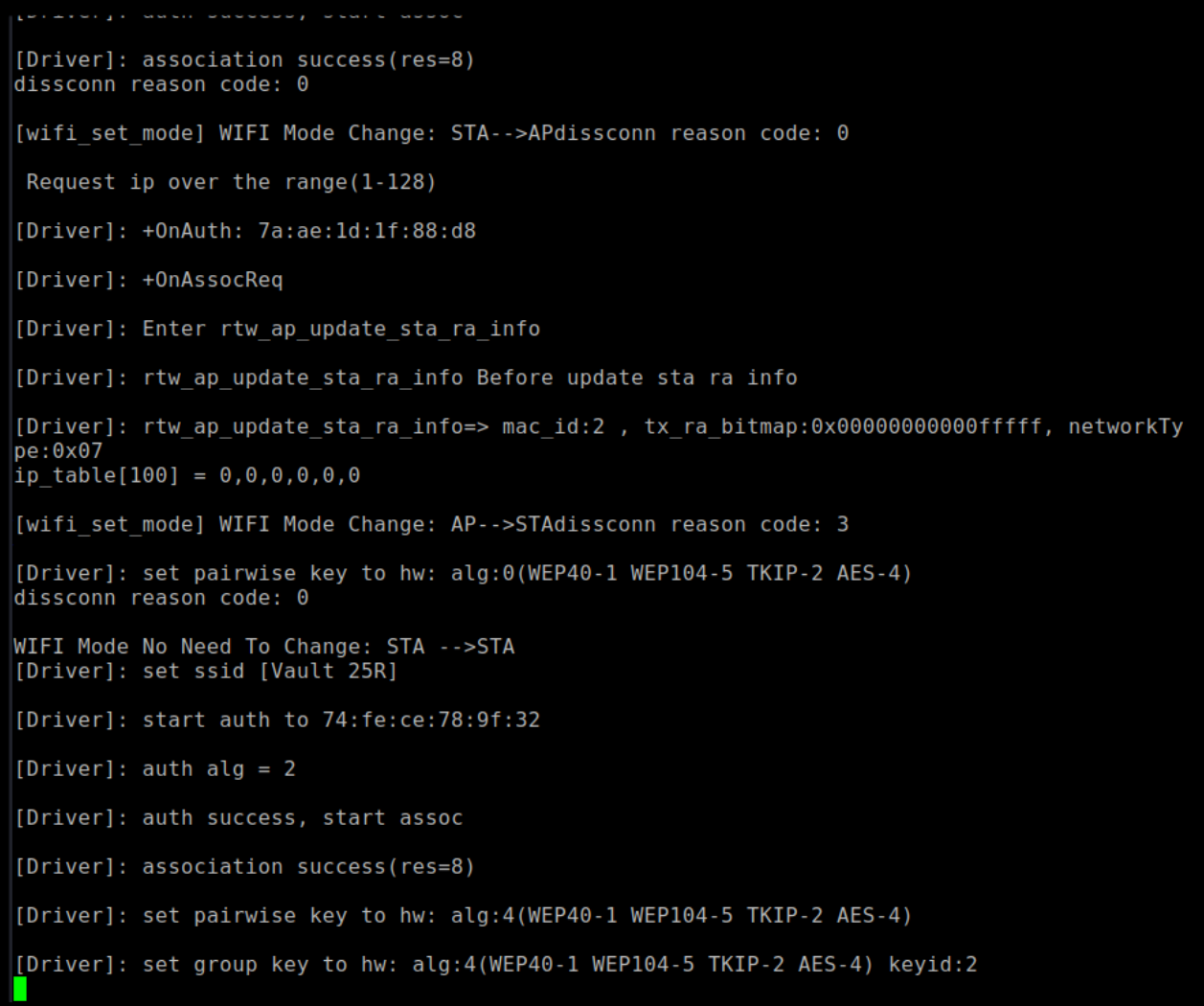
Achieving a UART Shell:





Greater Debugging Visibility:





It was possible to get a limited UART shell with Putty. The actual interface had some available options, to connect to an SSID, create an SSID, ping an IP, and scan for wireless APs.

What Updating Did:

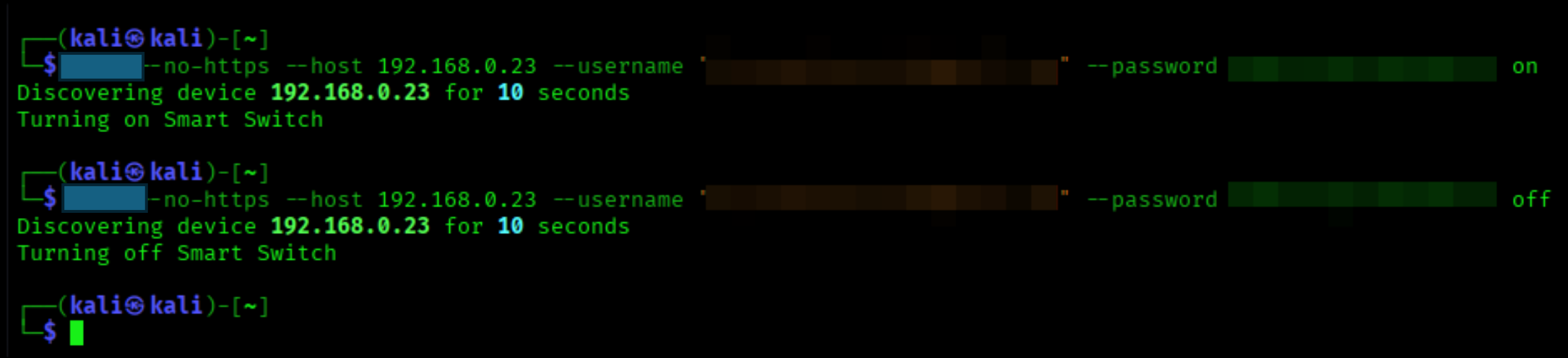
```
# ATWS

unknown command 'ATWS'
[MEM] After do cmd, available heap 51576


#
```

```
# ATW0=TEST123

unknown command 'ATW0=TEST123'
[MEM] After do cmd, available heap 71224


#
```

```
# ATW3=AAA

unknown command 'ATW3=AAA'
[MEM] After do cmd, available heap 50152


# ATW4=AAA

unknown command 'ATW4=AAA'
[MEM] After do cmd, available heap 50152


# ATW5

unknown command 'ATW5'
[MEM] After do cmd, available heap 50152
```

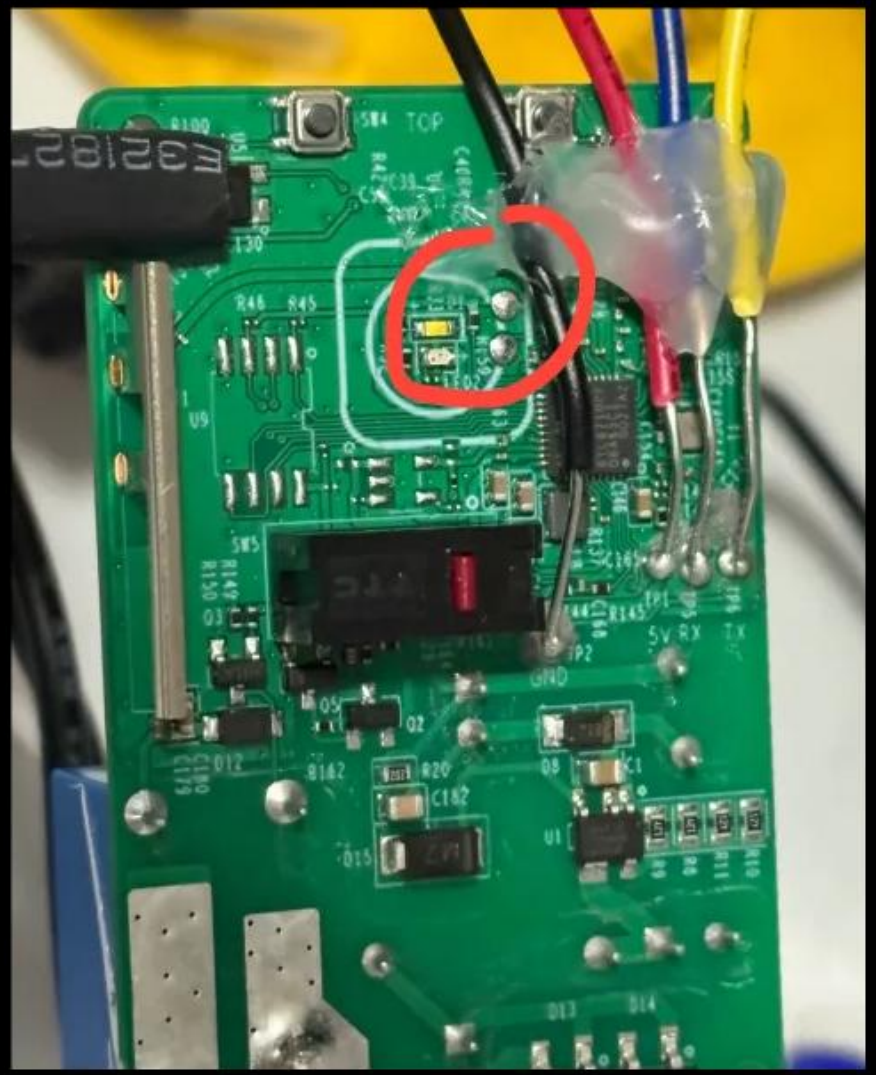
After connecting the device to my account through the mobile app, the firmware was issued an update, and the UART commands did not work at all, even after unregistering the device.

Commands with Authentication:



- Don’t recreate the wheel.
  - Use existing tools / libraries / frameworks if exist.
- Inspect data structures, objects, and blobs.
- Fuzz inputs and parameters and check what verbose debug logs give

Jumping PINs:



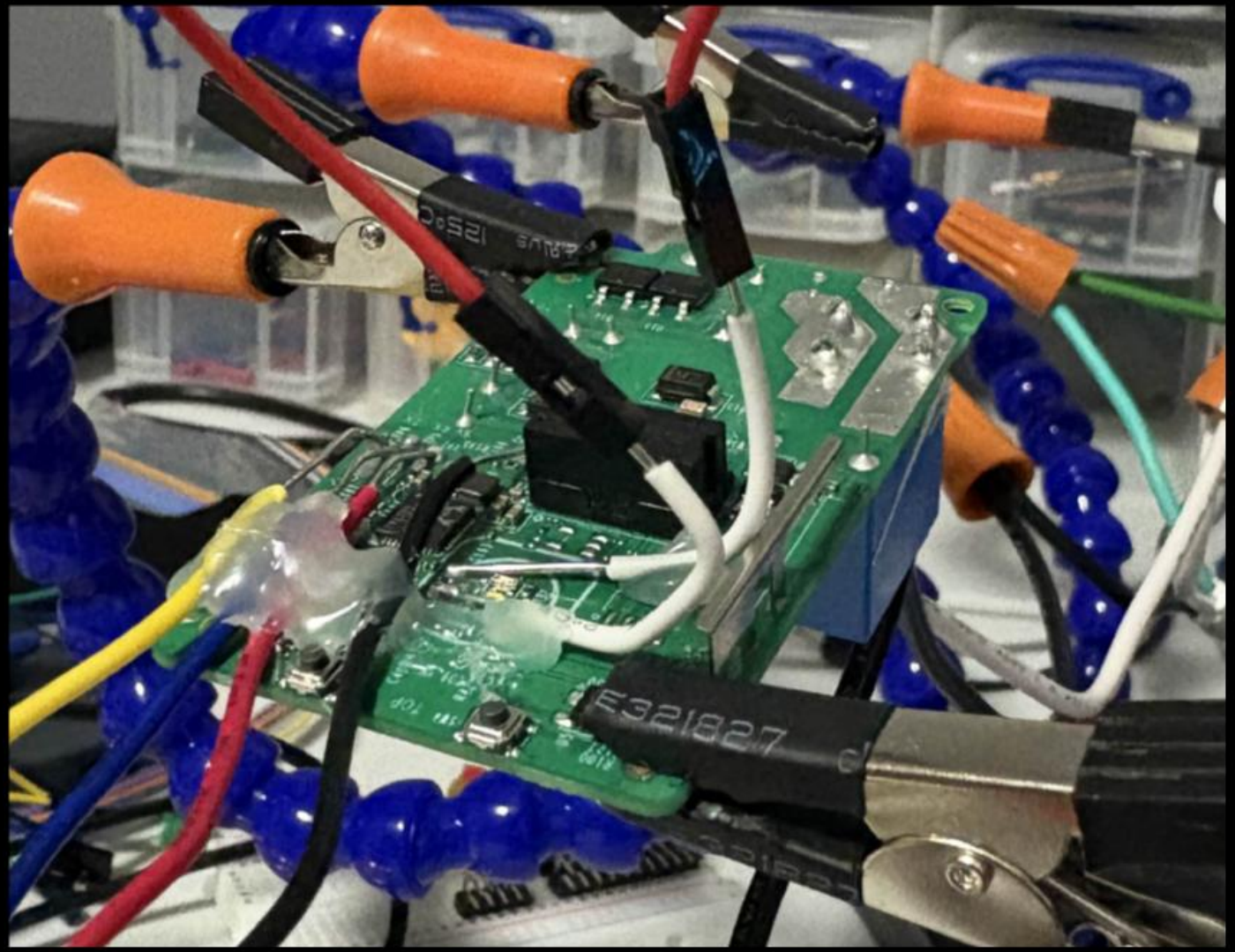
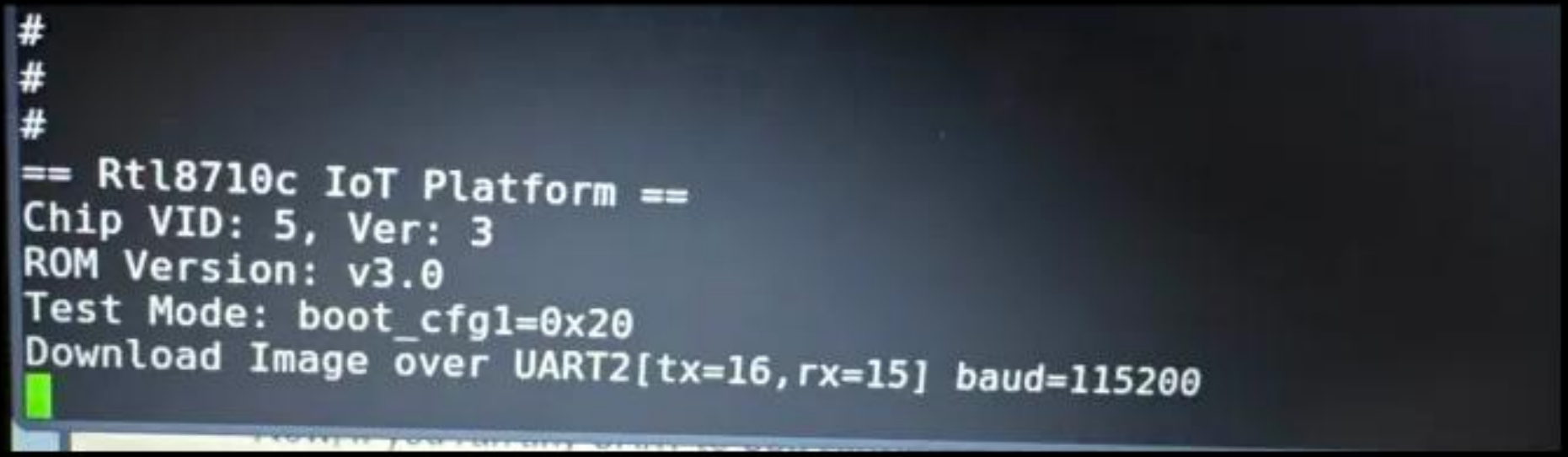
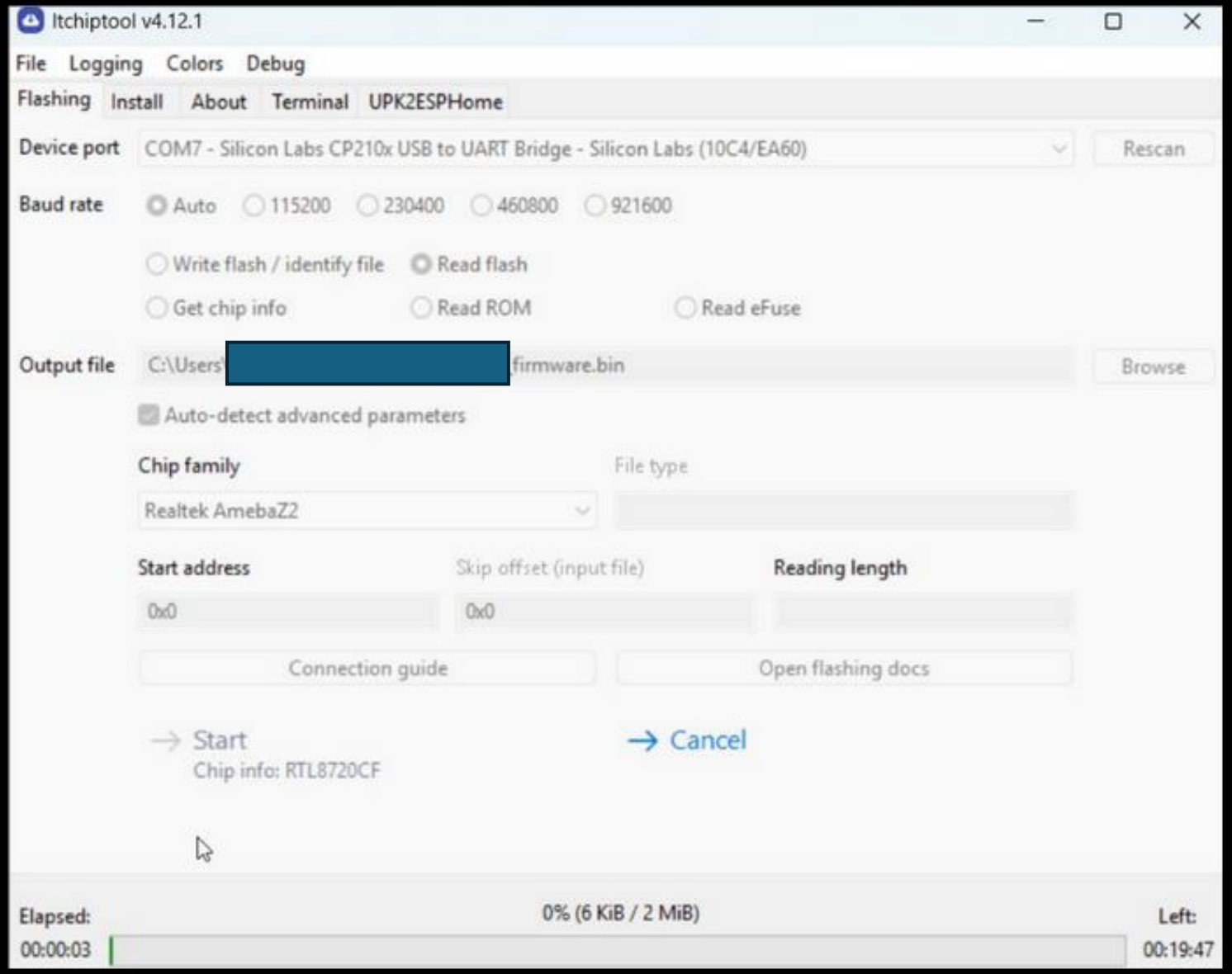


Image Was Available to Download:

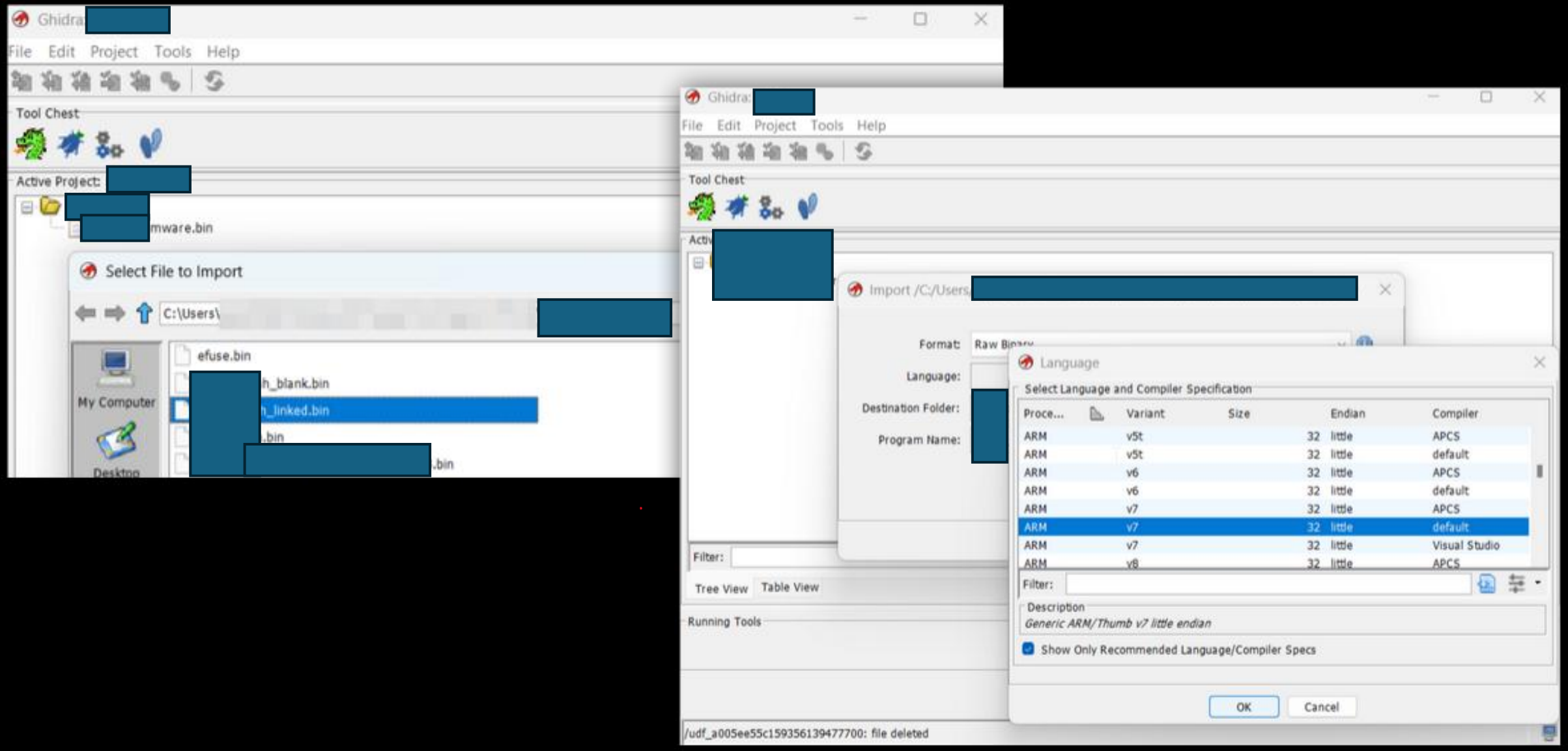
- At this point, it was possible to read flash contents via UART
- Tools that can read this data
  - Itchiptool v4.12.1



Extracting Flash, ROM, and EFuse:

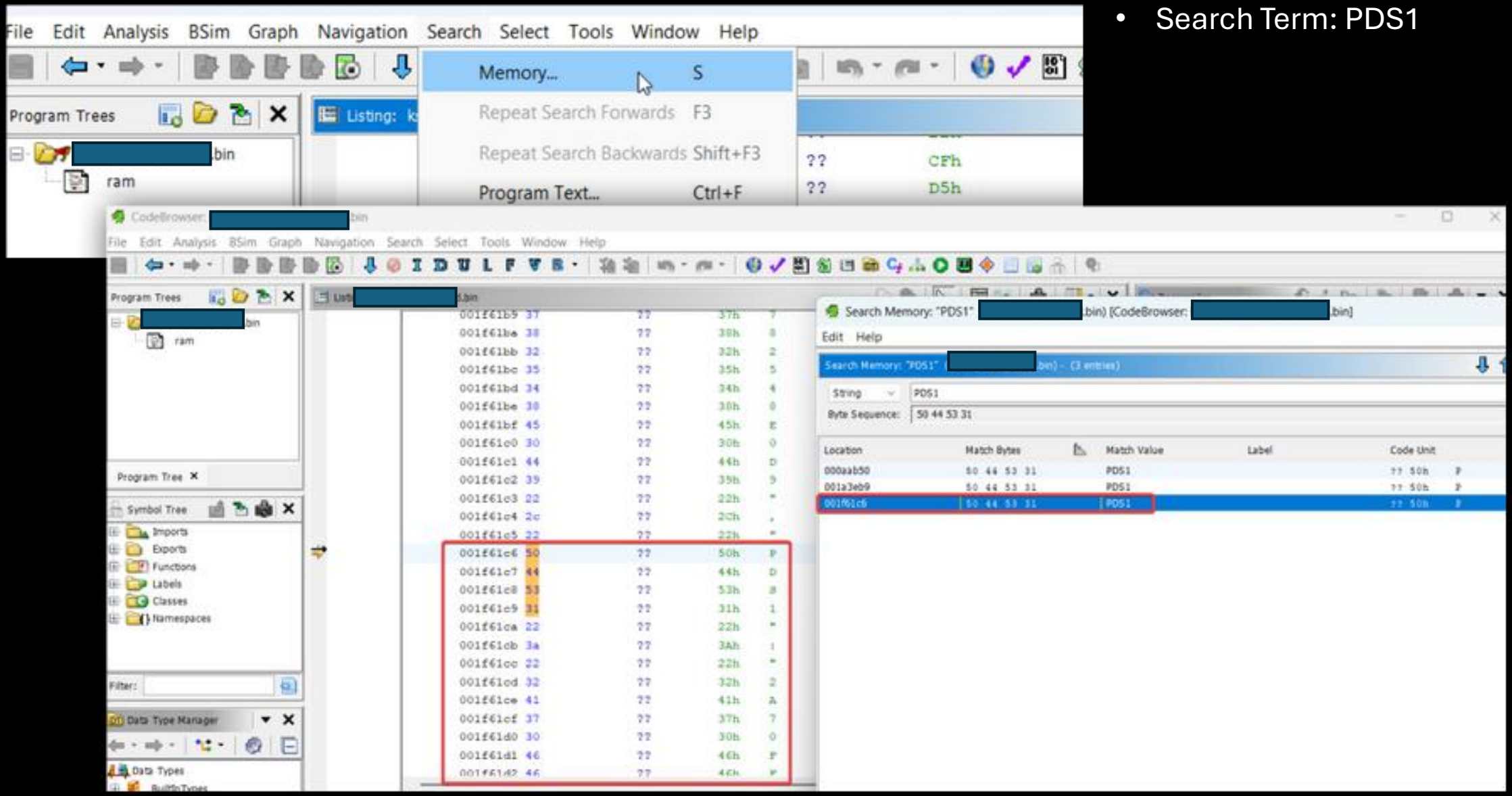


Reverse Engineering with Ghidra (1):

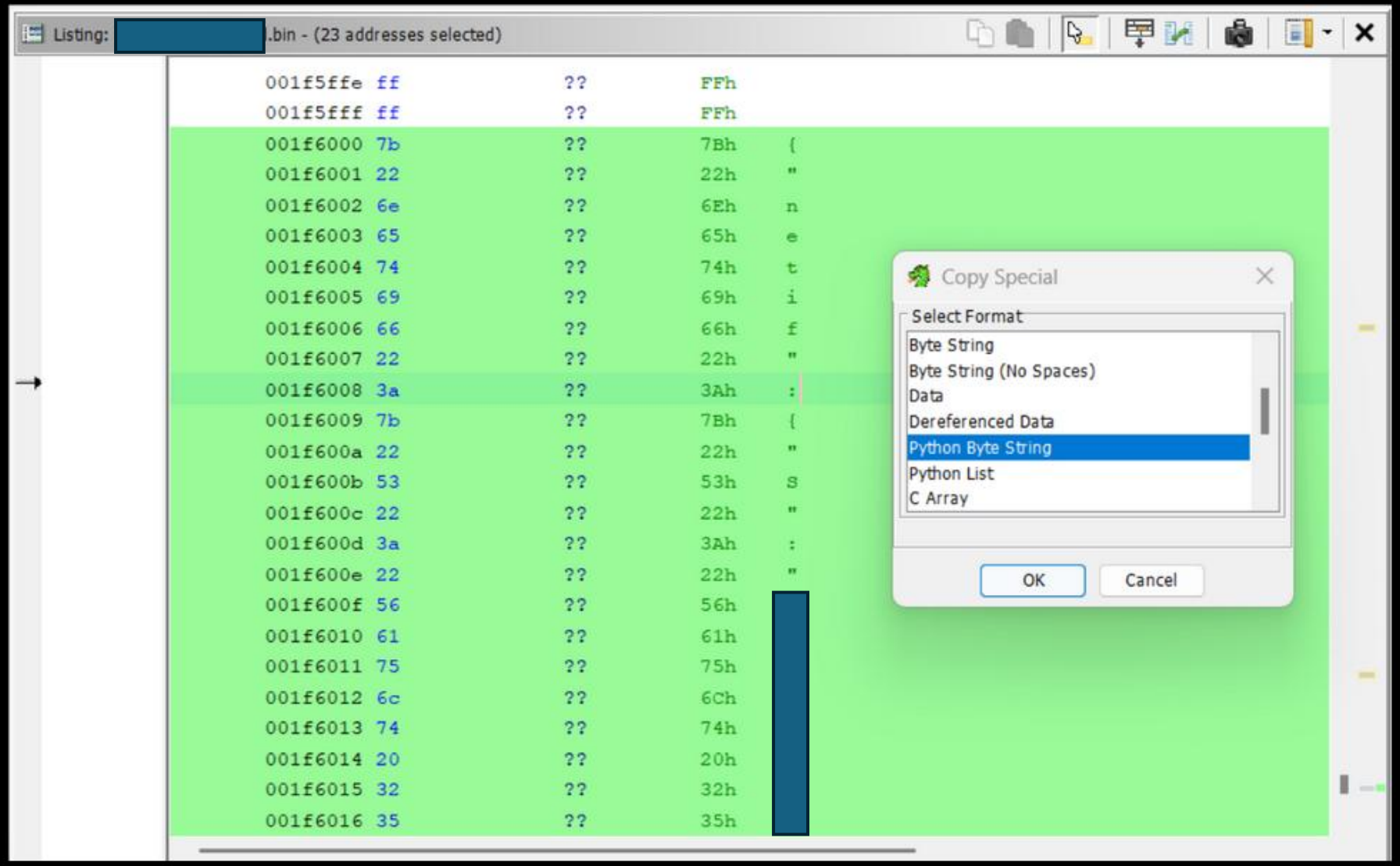


Reverse Engineering with Ghidra (2):

- Search Term: PDS1



Reverse Engineering with Ghidra (3):



Deprecated Hashing Algorithms (SHA1):

```python
import hashlib

username = "[redacted]"
password = "[redacted]"

UNS1 = hashlib.sha1(username.encode()).hexdigest().upper()
PDS1 = hashlib.sha1(password.encode()).hexdigest().upper()

print("UNS1 =", UNS1)
print("PDS1 =", PDS1)
```

```
UNS1 = D313[redacted]E0D9
PDS1 = 2A70[redacted]7769
PS C:\Users\elyse\Desktop\drop>
```

```
"iot_cloud": {
    "NC": false,
    "UN": "[redacted]",
    "PD": "[redacted]",
    "BUN": "x[redacted]==",
    "UNS1": "D313[redacted]E0D9",
    "PDS1": "2A70[redacted]7769",
    "UNM": "4080FADEFDCC379A628FB9A0041D6300",
    "AI": "[redacted]",
    "BC": "",
    "OT": 1756010146,
    "BUNS1": "",
    "BAI": "",
    "BBC": "",
    "RT": 0,
    "ST": "",
    "BI": "",
    "BPS": "",
    "BP": ""
},
```

I compared the JSON that was extracted from the firmware. As it turned out, UNS1 is the hashed email, and PDS1 is the hashed password of the user cloud credentials.
This was confirmed when I used python to hash both my cloud creds, and they matched.

Comparing Network Info with Decompiled Info:

```
┌──(kali㉿kali)-[~]
└─$ [redacted] --no-https --host 192.168.0.23 --username "[redacted]
Discovering device 192.168.0.23 for 10 seconds
== System info ==
{'auto_off_remain_time': 0,
 'auto_off_status': 'off',
 'avatar': 'switch_s500',
 'default_states': {'state': {'on': False}, 'type': 'custom'},
 'device_id': '803B[redacted]5CB5',
 'device_on': False,
 'fw_id': [redacted]
 'fw_ver': [redacted]
 'has_set_location_info': True,
 'hw_id': '6AF6[redacted]F5',
 'hw_ver': '[redacted]',
 'ip': '192.168.0.23',
 'lang': '',
 'latitude': 511486,
 'longitude': -1141869,
 'mac': '[redacted]',
 'model': '[redacted]',
 'nickname': '[redacted]',
 'oem_id': '0[redacted]',
 'on_time': 0,
 'overheat_status': 'normal',
 'region': 'America/Edmonton',
 'rssi': -26,
 'signal_level': 3,
 'smart_switch_state': False,
 'specs': '',
 'ssid': [redacted]
 'time_diff': -420,
 'type': '[redacted]'}

┌──(kali㉿kali)-[~]
└─$
```

1. Sometimes, you can use command line tools and see what unique data there is.

2. Take notes of all identifiers and values, what is missing? Do other locations show additional results?

3. Can you use these strings to perform string search in HTTP/S communication, protocol data, firmware?

4. For reflected inputs, do they repeat on web apps, mobile apps, do they validate / sanitize data?

What else could you find?

- Depending on how the device encrypts, hashes, or stores wireless credentials, extracting the WPA/2/3 or PMK is possible.
- Duplicate private keys are never a good sign.
- PII could be found here.
- More on the vendor side, but APIs, routes, and internal application logic might disclose how data could be handled, and what vendor endpoints allow communication.

Why?

What Attack Vectors are Worthwhile to Attackers?

- High Effort and Low Impact tend to not be as worthwhile.
- “Remote” attacks will likely need to go through the vendor, unless the device has an external WAN facing interface (IP:Port)
- Line of Sight attacks
- Hardware attacks – Discarded or intercepted devices

These Attack Vectors are Worthwhile to Attackers

- Potentially causing an electrical fire with an IoT device?
- The two findings I found with the power strip and outlet could lead to this.

Resources for Further IoT Hacking:

- Check out OWASP Internet of Things
- Practical IoT Hacking by Fotios Chantzis, Ioannis Stais, Paulino Calderon, Evangelos Deirmentzoglou, and Beau Woods
- PIPA course by TCM

Responsible Disclosures, Safe Harbors, and CYA:

(Not a lawyer, not legal advice)

- Pick programs that have disclosure, at the least a security email.
- Companies that had previous CVEs, advisories, patches.
- Many people use aliases, there's pros and cons to this approach
- Have good faith, it won't be diplomatic to threaten release of research, legal action was taken against researchers.
- As an option, Mitre does accept CVEs. Make sure you understand the risks in disclosing something.

Questions? Thank you!

Contact:

Elysee Franchuk

elyseemafranchuk@gmail.com

References:

- Home Automation - https://en.wikipedia.org/wiki/Home_automation
- Key reinstallation Attacks - https://www.tp-link.com/us/support/faq/1970/
- Key reinstallation Attacks, Mathy Vanheof - https://www.krackattacks.com/
- HS200BLE Kasa Smart Light Switch Internal PCB - https://fccid.io/2BCGWHS200BLE/Internal-Photos/10-Internal-Photos-6981763
- NIST CVSS Calculator - https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
- Itchiptool - https://github.com/libretiny-eu/ltchiptool
- Python Kasa - https://github.com/python-kasa/python-kasa
- OWASP IoT - https://owasp.org/www-project-internet-of-things/
- OWASP ISTG - https://github.com/OWASP/owasp-istg
- PIPA TCM - https://certifications.tcm-sec.com/pipa/
- Kettle with PCB Image - https://www.reddit.com/r/todayilearned/comments/1bwlnfp/today_i_learned_the_official_http_error_code_418/
- Iot Television - https://en.wikipedia.org/wiki/File:Samsung_Smart_TV_2012_(E-Series).jpg
- IoT Thermostat - https://en.wikipedia.org/wiki/File:Nest_Diamond_Thermostat.jpg
- Digital Wireless Camera Image - https://en.wikipedia.org/wiki/File:Lorex_digital_wireless_camera.jpg