Hacking your Jump Rope or your Coffee Machine

Axelle Apvrille

Insomni'hack, March 2023

Who am I?

Axelle Apvrille

Principal Security Researcher at **Fortinet**, @cryptax

Mobile malware

IoT + Ph0wn CTF

I don't drink coffee...

Hacking smart devices

Prodigio M135

Renpho Smart Jump Rope R-Q001

Hacking BLE devices from different perspectives

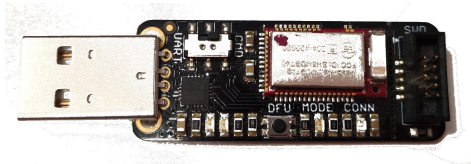
BLE communication

Hack here!

Hack here!

Hack here!

UART, Firmware,
Hardware RE...

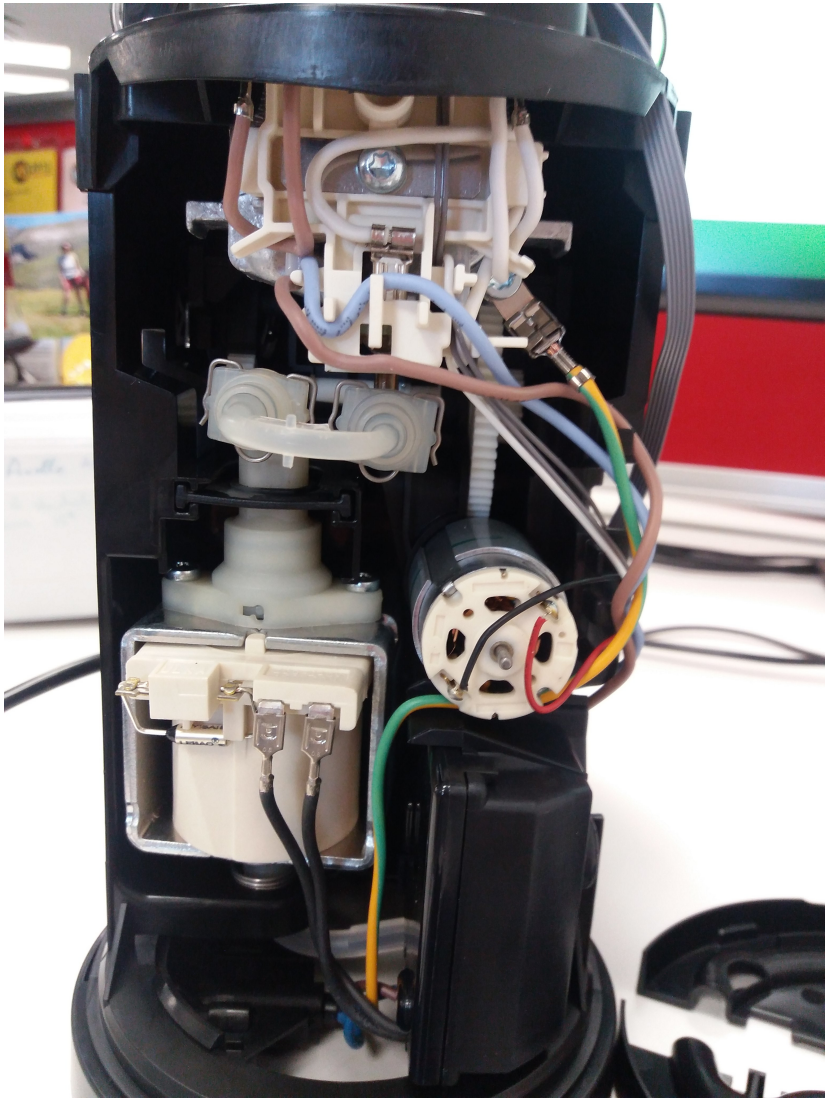


logcat, BLE snif
app RE

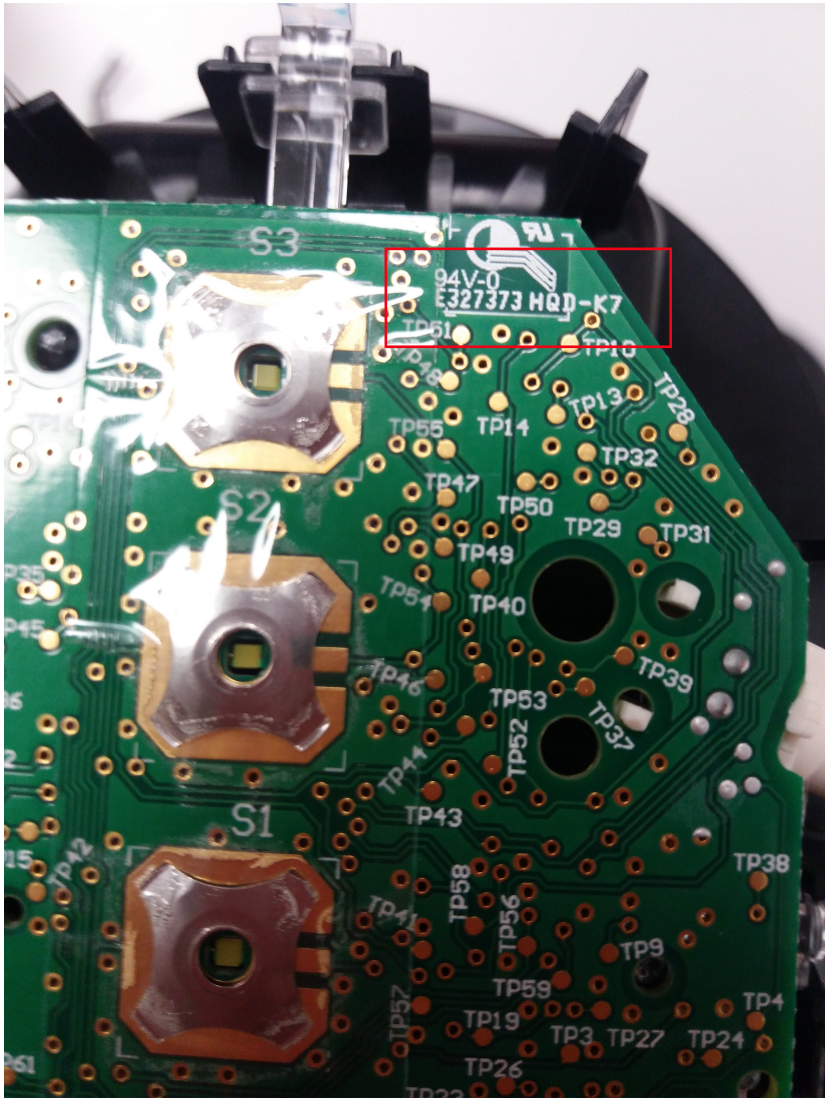
Adafruit BLE Sniffer, Ubertooth...

Hardware reconnaissance by a n00b

Hardware reconnaissance by a n00b



Hardware reconnaissance by a n00b



94V-0 E327373 HQD-K

Separate parts on eBay

ebay Shop by category Search for anything All Categorie

Back to search results | Listed in category: Home & Garden > Major Appliances > Other Major Appliances

Krups Nespresso PCB Card Prodigio Keys XN410 XN411 EN170 EN270 C70 D70 - show original title

Be the first to write a review.

Condition: New

Quantity: 1 3 available / 17 sold

Price: EUR 35.98
Approximately US $38.45

Buy It Now

Add to cart

♡ Add to Watchlist

Returns accepted

Shipping: Free Spedizione internazionale standard. See details
Located in: Palermo, Italy

Delivery: Estimated between Thu, Mar 16 and Tue, Mar 28 to 06410

Returns: 30 day returns. Buyer pays for return shipping. See details

Payments: PayPal VISA

$ Have one to sell? Sell now

Separate parts on eBay

Krups:
XN410T
XN411T
XN410T10/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO
XN410T10/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO
XN410T40/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO
XN410TCH/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO
XN411T10/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO&MILK
XN411T10/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO&MILK
XN411T40/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO&MILK
XN411TCH/FB0 CAFFETTIERA ESPRESSO NESPRESSO PRODIGIO&MILK

DeLonghi:
EN170.S
EN270.SAE

Breville:
BEC500XT

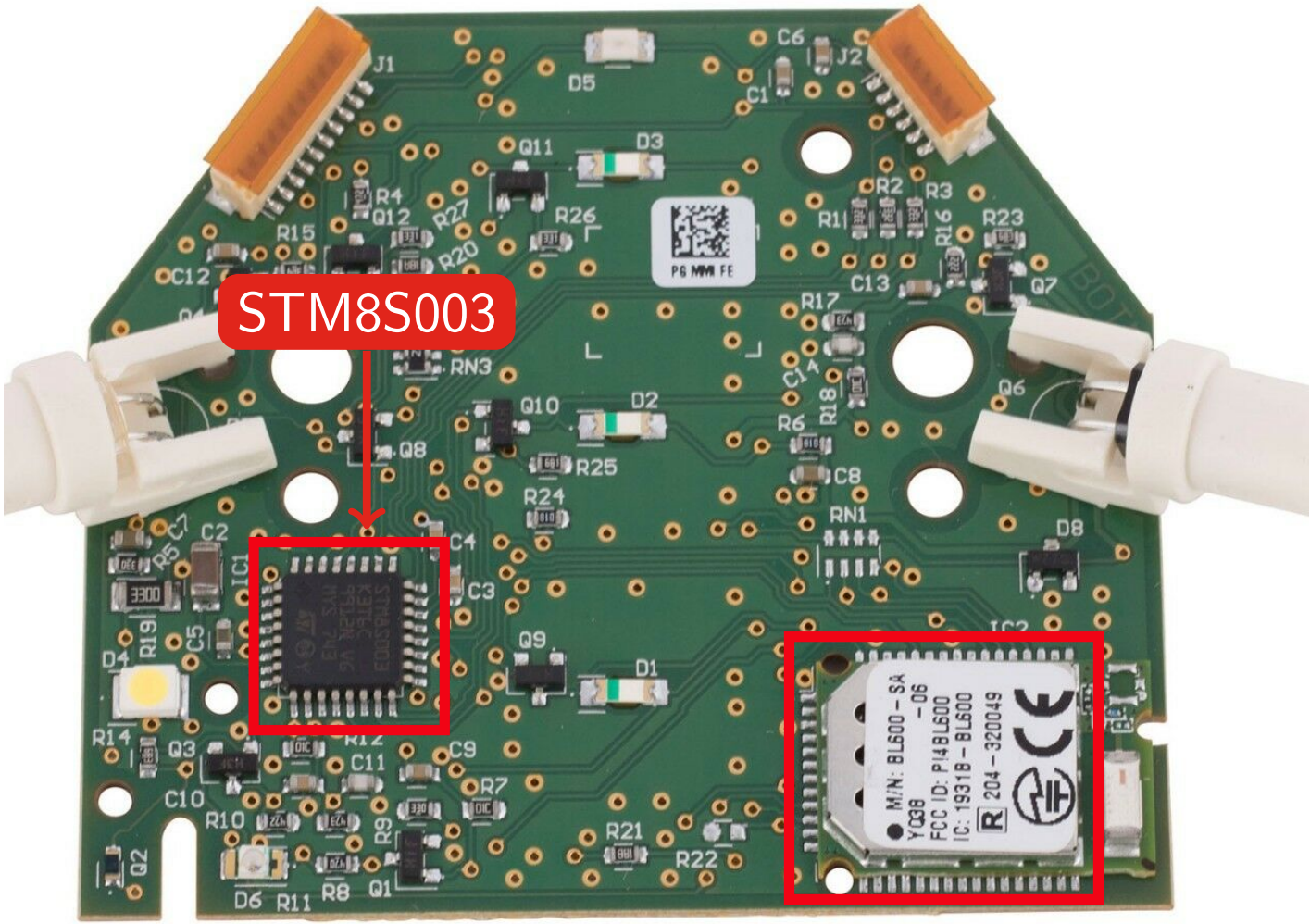
Koenig:
B03157
B03158

Magimix:
11375
11376

Turmix:
TX190
TX290

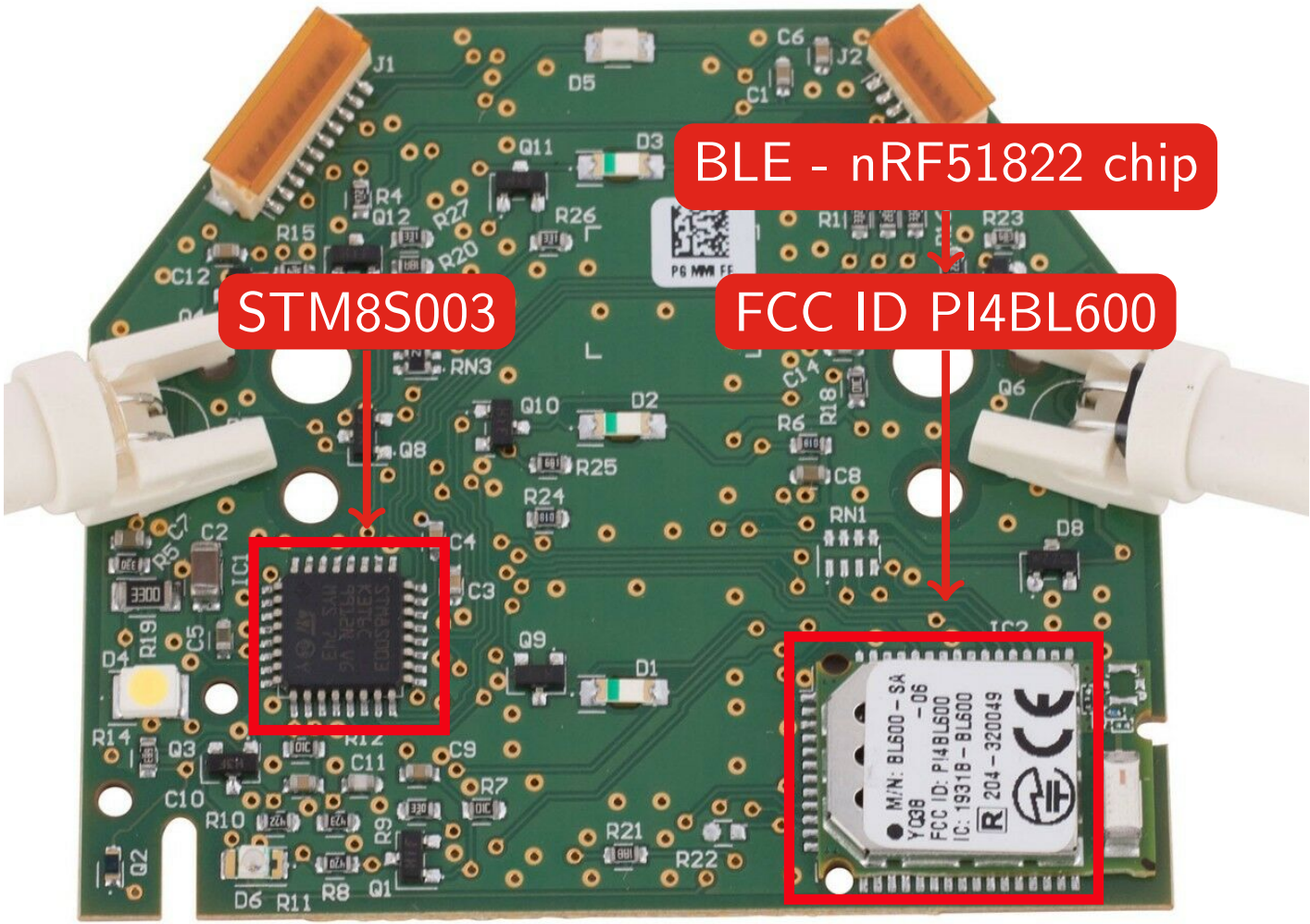
Nespresso:
C75
C75
D70
D75

Hardware components



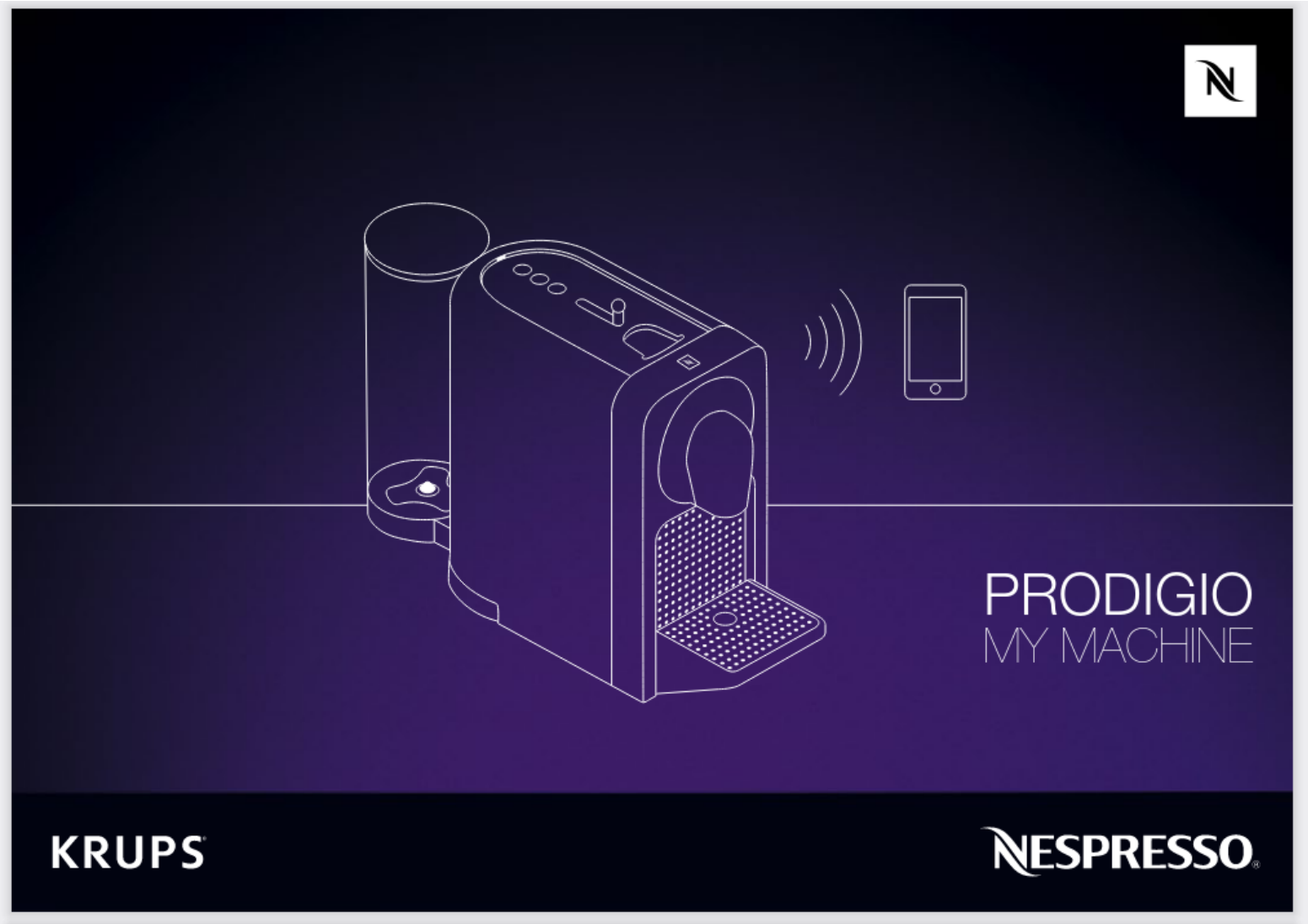
https://www.st.com/resource/en/datasheet/stm8s003f3.pdf

Hardware components



https://www.st.com/resource/en/datasheet/stm8s003f3.pdf
https://fccid.io/PI4BL600/Users-Manual/User-Manual-1937726

Documentation: Instruction Manual

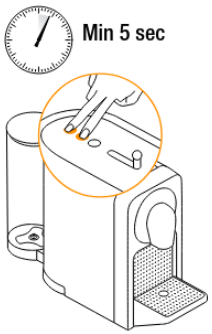


https://www.nespresso.com/shared_res/manuals/prodigio/www_PRODIGIO_C_KRUPS(EN_FR_DE_IT_ES_PT_CZ_HU_NL_GR_PL).pdf

Documentation: Instruction Manual

EN FR

Reset to Factory Settings / Restauration des réglages par défaut



If you reset to factory settings, this will cancel the pairing, reset the capsules stock management and the descaling alert.

Toute restauration des réglages par défaut entraîne l'annulation de l'appairage, ainsi que la réinitialisation de la gestion du stock de capsules et de l'alerte de détartrage.

1 Factory settings are:
- Lungo, Espresso, Ristretto coffee buttons 110ml / 3.7oz, 40ml / 1.35 oz., 25ml / 0.84 oz.
- Automatic OFF mode after 9 minutes.
- The water hardness set by default is hard, which corresponds to around 1000 Espresso cups.

To do it via your machine:
1. Ensure the machine is turned OFF.
2. Press and hold Espresso & Lungo buttons for at least 5 seconds.

All the coffee buttons and LEDs will blink once as confirmation. To unpair your machine, please refer to the «Troubleshooting section».

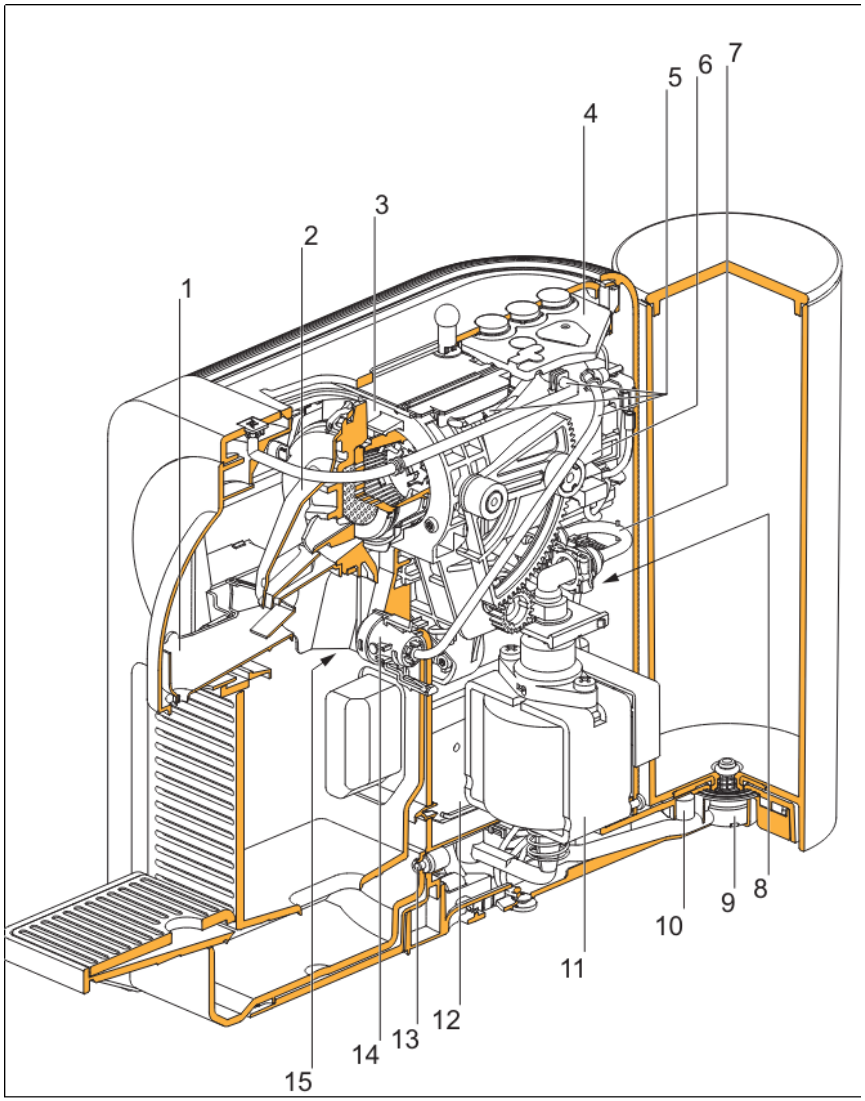
Les réglages par défaut sont les suivants:
- Boutons Lungo (110 ml), Espresso (40 ml) et Ristretto (25 ml).
- Mode de mise hors tension automatique après 9 min de non-utilisation.
- Par défaut, la dureté de l'eau est définie sur «dure», ce qui correspond à environ 1000 tasses Espresso.

Pour restaurer les réglages par défaut à partir de votre machine:
1. Vérifiez que la machine est éteinte.
2. Appuyez simultanément sur les boutons Espresso et Lungo pendant au moins 5 secondes.

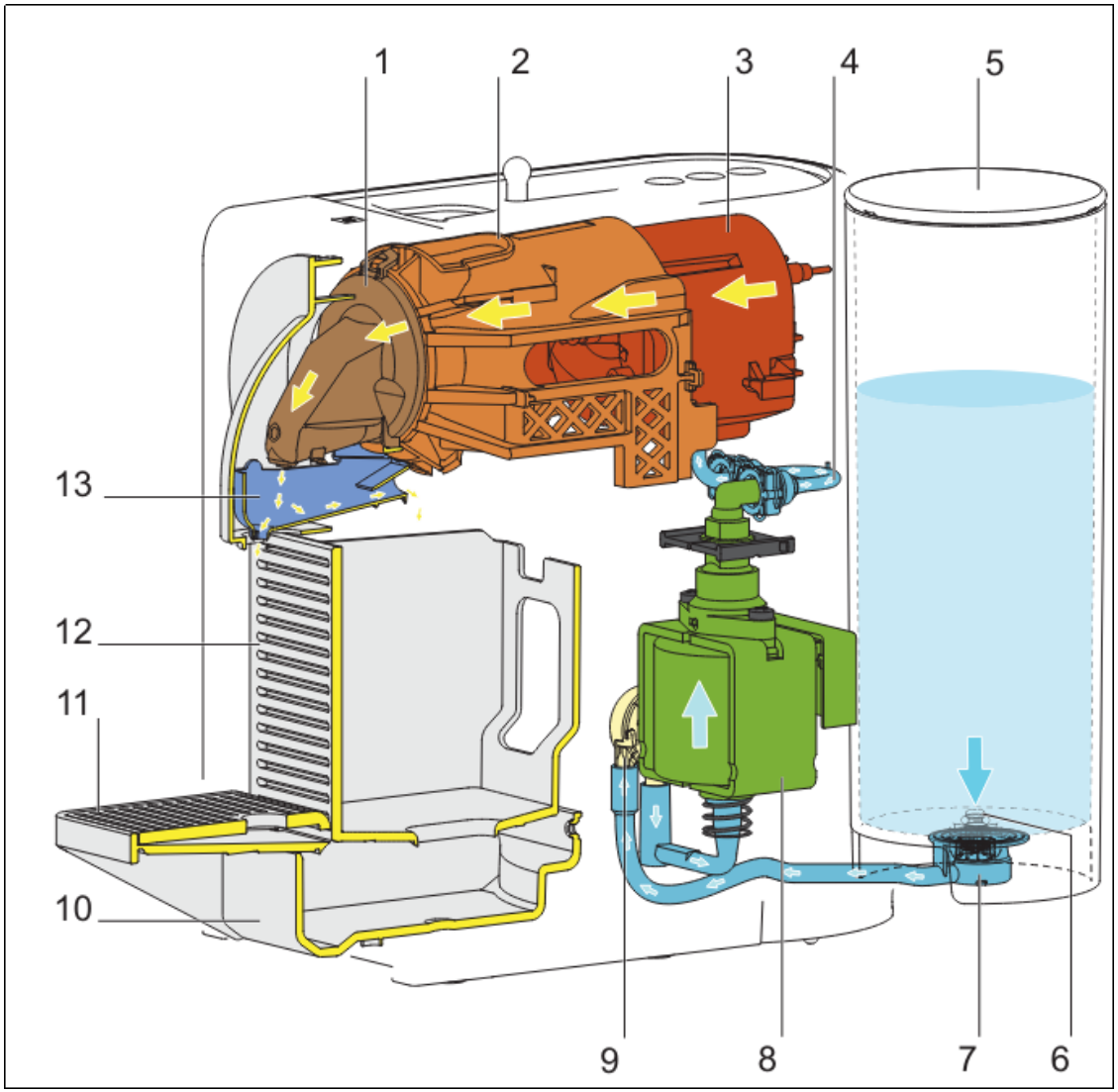
Tous les boutons et LEDs s'allumeront une fois pour confirmer le changement. Pour annuler l'appairage de votre machine, consultez la section «Dépannage».

https://www.nespresso.com/shared_res/manuals/prodigio/www_PRODIGIO_C_KRUPS(EN_FR_DE_IT_ES_PT_CZ_HU_NL_GR_PL).pdf

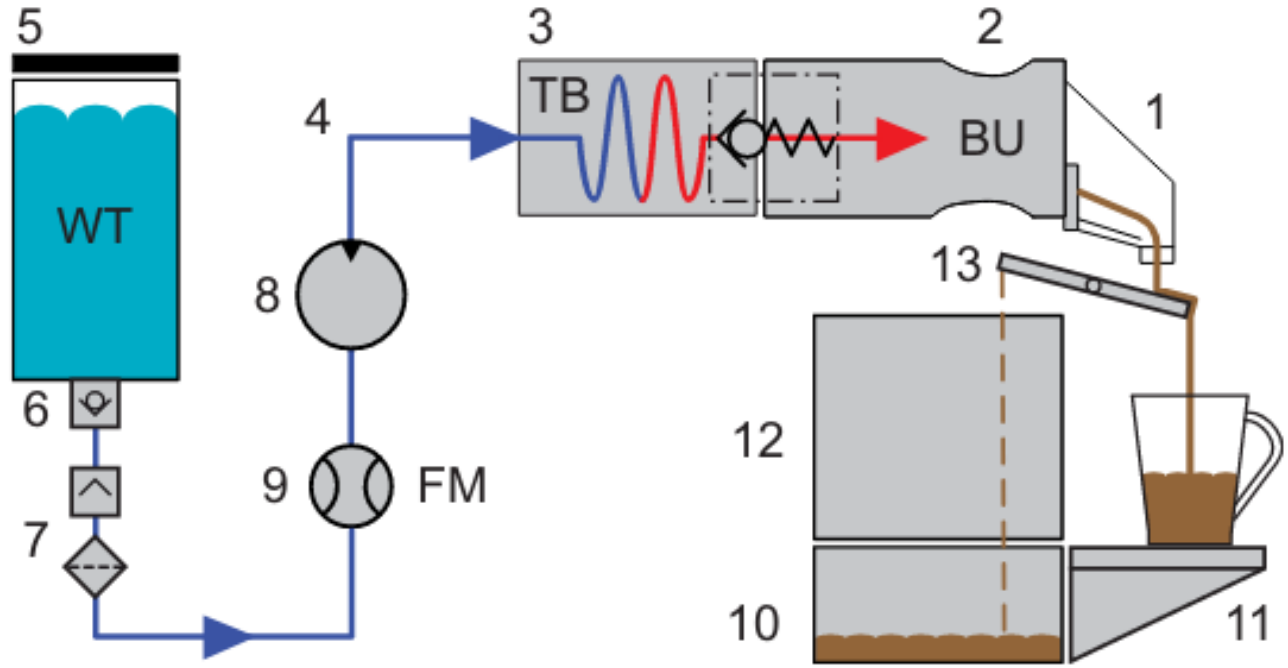
Service Manual - https://m.buyspares.co.uk/images/mediator/2752/c75%20service%20manual.pdf



Service Manual - https://m.buyspares.co.uk/images/mediator/2752/c75%20service%20manual.pdf



Service Manual - https://m.buyspares.co.uk/images/mediator/2752/c75%20service%20manual.pdf



1) Coffee outlet
2) Brewing unit
3) Thermoblock
4) High pressure connector
5) Water tank
6) Water tank valve
7) Water tank connector with filter
8) Pump
9) Flowmeter
10) Drip tray
11) Cup support
12) Used capsule container
13) Drop stop

Hardware: what have we learned?

- Uses a **BLE** chip. (I already knew that...)
- The **PCB** is used for several different coffee makers
- How to **reset to factory settings**
- How to **repair** the device

Hacking BLE devices from different perspectives

Basic: ok.
Advanced:
not for me

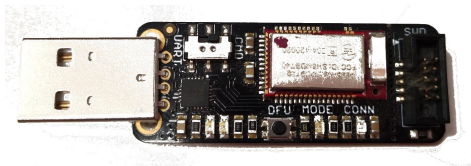
BLE communication

Hack here!

Hack here!

Hack here!

UART, Firmware,
Hardware RE...

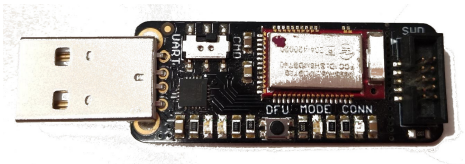


logcat, BLE snif
app RE

Adafruit BLE Sniffer, Ubertooth...

Sniff BLE

- Ubertooth
- Adafruit LE sniffer
- Old tools, but still maintained
- Flash firmware, add wireshark plugin etc
- All BLE layers, not only ATT




```
  6473 40.953235    dd:3a:e8:08:b1:a4  Broadcast          LE LL     40 ADV_NONCONN_IND
   625 4.003341     df:35:35:16:54:bf  Broadcast          LE LL     63 ADV_IND
   215 1.406840     df:e5:25:16:54:a7  Broadcast          LE LL     63 ADV_IND
     1 0.000000     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
     2 0.000734     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    22 0.122296     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    23 0.123039     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    43 0.237836     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    44 0.238539     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    61 0.332541     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    81 0.445887     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
    82 0.446291     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND[Malformed Packet]
    95 0.544839     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   112 0.659030     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   113 0.659405     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   127 0.758428     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   128 0.758765     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   171 1.090423     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   172 1.091232     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
   173 1.092037     df:e5:35:16:54:a7  Broadcast          LE LL     63 ADV_IND
```

```
▸ Frame 625: 63 bytes on wire (504 bits), 63 bytes captured (504 bits) on interface /tmp/wireshark_extcap_-dev-ttyUSB0-3.2_20230313163336_oEeykj, id 0
▸ Nordic BLE Sniffer
▾ Bluetooth Low Energy Link Layer
    Access Address: 0x8e89bed6
  ▸ Packet Header: 0x2500 (PDU Type: ADV_IND, ChSel: #1, TxAdd: Public)
    Advertising Address: df:35:35:16:54:bf (df:35:35:16:54:bf)
  ▾ Advertising Data
    ▸ Flags
    ▾ Unknown
        Length: 11
        Type: Unknown (0xdf)
      ▸ Data: 030101dfe5351654a601
    ▾ Unknown
        Length: 15
        Type: Unknown (0xef)
      ▸ Data: 53454e5048cb6c524e50452d5231
  ▸ CRC: 0x803797
```

```
0000  c7 06 38 01 22 26 06 0a  00 26 56 00 00 64 04 00   ··8·"&·· ·&V··d··
0010  00 d6 be 89 8e 00 25 bf  54 16 35 35 df 02 01 36   ····· %· T·55···6
0020  0b df 03 01 01 df e5 35  16 54 a6 01 0f ef 53 45   ·····5 ·T····SE
0030  4e 50 48 cb 6c 52 4e 50  45 2d 52 31 01 ec e9      NPH·lRNP E-R1···
```

https://github.com/greatscottgadgets/ubertooth

https://www.adafruit.com/product/2269

Hacking BLE devices from different perspectives

BLE communication

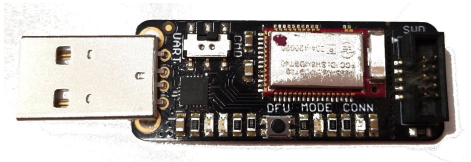
Not for me

Not that efficient (for me)

Hack here!

Hack here!

UART, Hardware RE...



Adafruit BLE Sniffer, Ubertooth...

Hacking BLE devices from different perspectives

BLE communication

Not for me

Not that efficient (for me)

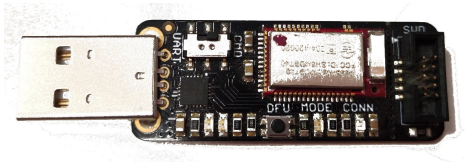
My favorite solution

Hack here!

Hack here!

Hack here!

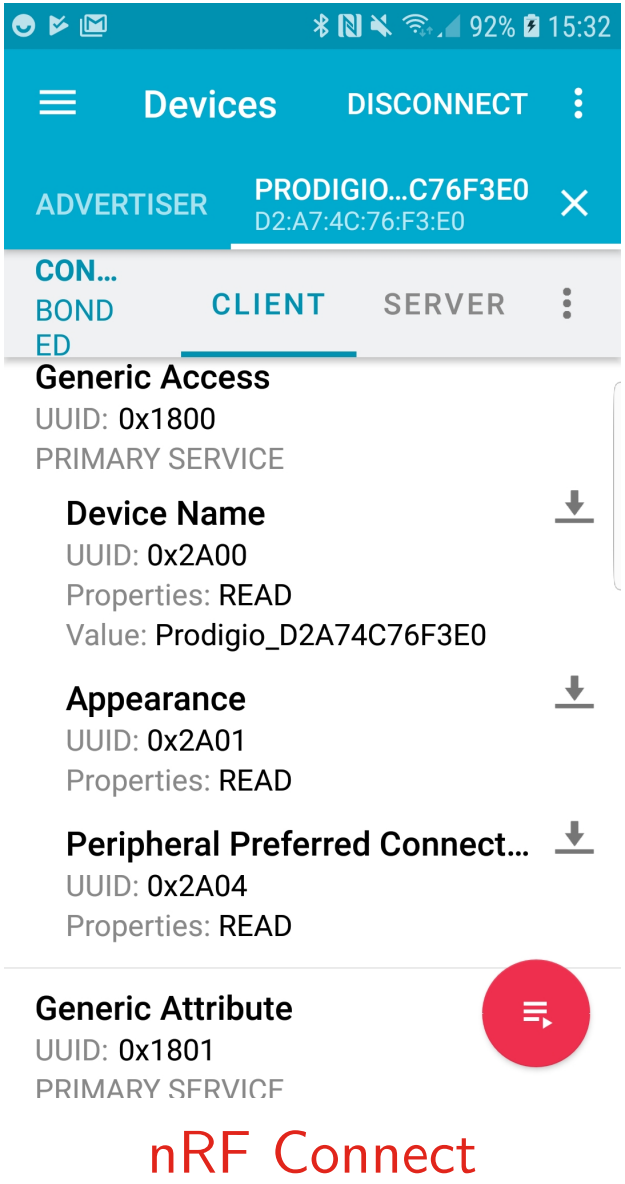
UART, Hardware RE...



logcat, BLE snif
app RE

Adafruit BLE Sniffer, Ubertooth...

BLE enumeration



nRF Connect

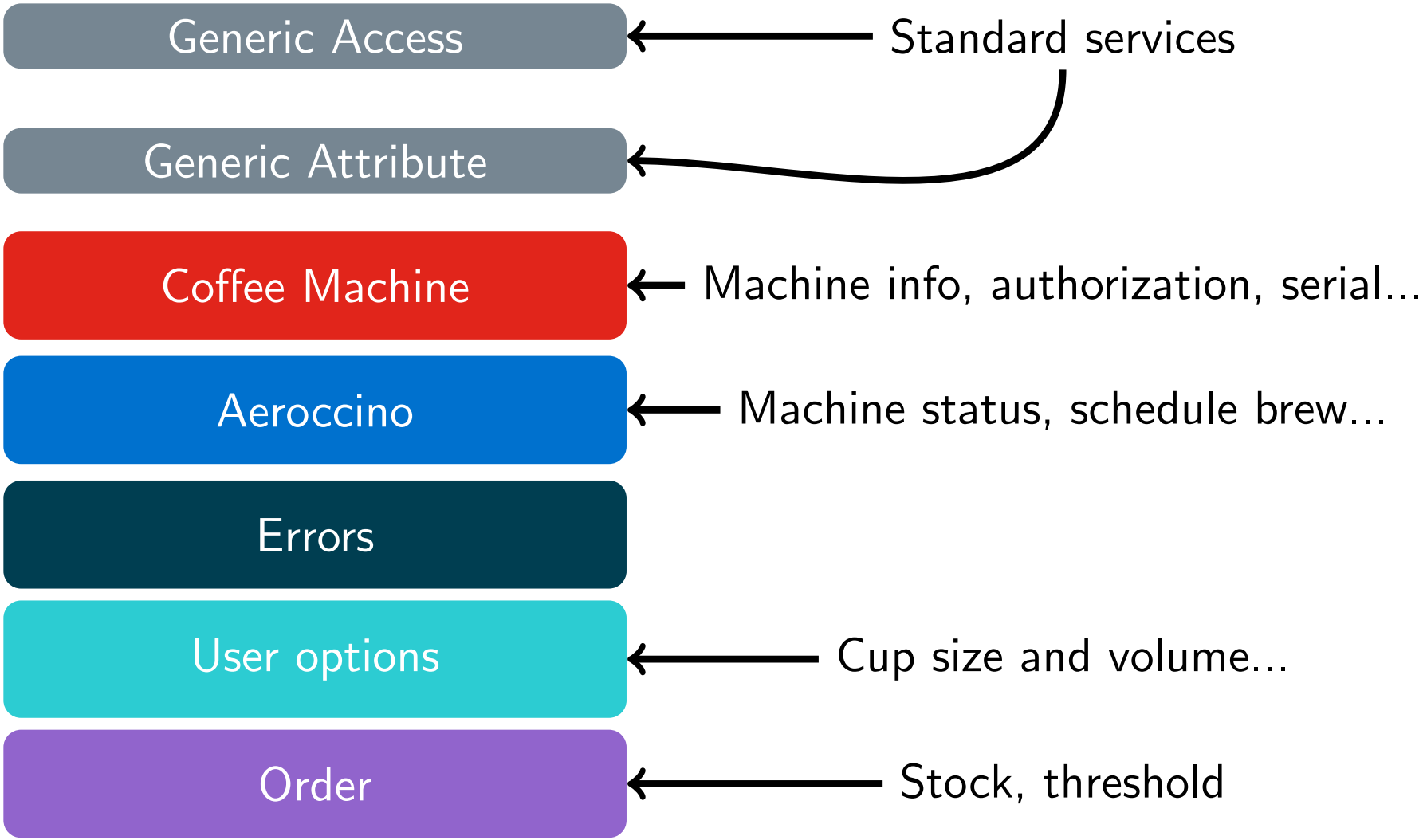
- **Bleah**: compact enumeration, deprecated
- **Gatttool**: deprecated but good
- **Mirage**: interesting for some things
- **Bluetoothctl**: excellent (but talks too much)

```
<< ble_connect|ble_discover >>~~> set ble_connect1.CONNECTION_TYPE random
<< ble_connect|ble_discover >>~~> run
[INFO] Trying to connect to : d2:a7:4c:76:f3:e0 (type : random)
[INFO] Updating connection handle : 72
[SUCCESS] Connected on device : d2:a7:4c:76:f3:e0
[INFO] Services discovery ...
Services
```

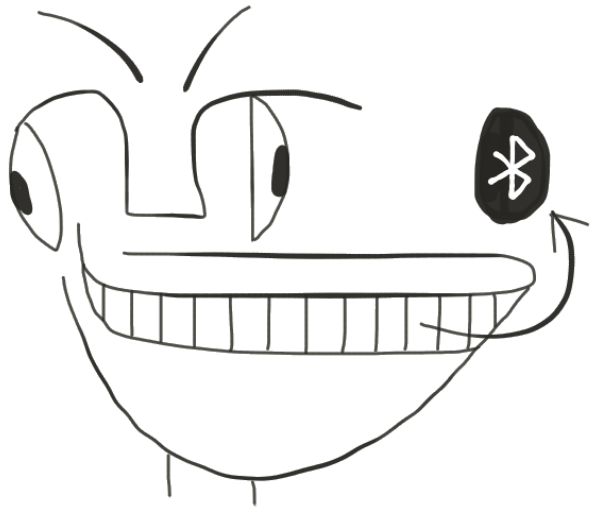
Start Handle	End Handle	UUID16	UUID128	Name
0x0001	0x0007	0x1800	00001800000010008000000805f9b34fb	Generic Access
0x0008	0x000b	0x1801	00001801000010008000000805f9b34fb	Generic Attribute
0x000c	0x0019		06aa1910f22a11e39daa0002a5d5c51b	
0x001a	0x0027		06aa1920f22a11e39daa0002a5d5c51b	
0x0028	0x002d		06aa1930f22a11e39daa0002a5d5c51b	
0x002e	0x0039		06aa1940f22a11e39daa0002a5d5c51b	
0x003a	0xffff		06aa1950f22a11e39daa0002a5d5c51b	

https://github.com/evilsocket/bleah
http://homepages.laas.fr/rcayre/mirage-documentation/index.html

Smart coffee services



Smart Coffee: we need authentication



```
sudo gatttool -b d2:a7:4c:76:f3:e0 -I -t random
[d2:a7:4c:76:f3:e0][LE]> connect
Attempting to connect to d2:a7:4c:76:f3:e0
Connection successful
[d2:a7:4c:76:f3:e0][LE]> char-read-hnd 0x000e
Error: Characteristic value/descriptor read failed:
↪  Attribute requires authentication before
↪  read/write
```

How to request Security Mode 1, Level 3

- Level 1. No encryption.
- Level 2. Unauthenticated encryption.
- Level 3. Authenticated encryption

see https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html

gatttool (deprecated)

```
gatttool -b d2:a7:4c:76:f3:e0 -I -t random
                        --sec-level=high
```

bluetoothctl

```
[bluetooth]# connect D2:A7:4C:76:F3:E0
[Prodigio_D2A74C76F3E0]# pair D2:A7:4C:76:F3:E0
Attempting to pair with D2:A7:4C:76:F3:E0
[CHG] Device D2:A7:4C:76:F3:E0 Paired: yes
Pairing successful
```

BLE authorization

```
[d2:a7:4c:76:f3:e0][LE]> char-read-hnd 0x001c
Error: Characteristic value/descriptor read failed: Attribute requires
↪  authorization before read/write
```

Authentication is not Authorization

Authentication

- *Act of proving an assertion, e.g. identity of a user/computer*
- Done during pairing

Authorization

- *"Is device X allowed to access/use service Y?"*
- **Rare** for BLE
- Implementation of authorization to be done by device.

How do I get authorization?

Sniff for authorization ON the smartphone

BLE communication

Enable Bluetooth sniffing

- Bluetooth HCI snoop log: https://www.bluetooth.com/blog/debugging-bluetooth-with-an-android-app
- Reboot
- Retrieve packet capture for inspection: `adb pull /sdcard/btsnoop_hci.log`

Where is the authorization?

```
:e0 (Prodigio… ATT     16 Sent Read By Group Type Request, GATT Primary Service Decl
:7c (Galaxy S… ATT     23 Rcvd Read By Group Type Response, Attribute List Length: 2
:e0 (Prodigio… ATT     16 Sent Read By Group Type Request, GATT Primary Service Decl
:7c (Galaxy S… ATT     31 Rcvd Read By Group Type Response, Attribute List Length: 1
:e0 (Prodigio… ATT     16 Sent Read By Group Type Request, GATT Primary Service Decl
:7c (Galaxy S… ATT     31 Rcvd Read By Group Type Response, Attribute List Length: 1
:e0 (Prodigio… ATT     16 Sent Read By Group Type Request, GATT Primary Service Decl
:7c (Galaxy S… ATT     31 Rcvd Read By Group Type Response, Attribute List Length: 1
:e0 (Prodigio… ATT     16 Sent Read By Group Type Request, GATT Primary Service Decl
:7c (Galaxy S… ATT     31 Rcvd Read By Group Type Response, Attribute List Length: 1
:e0 (Prodigio… ATT     16 Sent Read By Group Type Request, GATT Primary Service Decl
:7c (Galaxy S… ATT     31 Rcvd Read By Group Type Response, Attribute List Length: 1
:e0 (Prodigio… ATT     16 Sent Read By Type Request, GATT Include Declaration, Handl
```

- Look for **ATT** protocol
- First, many enumeration packets

Where is the authorization?

```
. SamsungE_b9:e9:7c (Galaxy S… ATT     15 Rcvd Find Information Response, Handle: 0x003d (Unknown: Unk…
. d2:a7:4c:76:f3:e0 (Prodigio… ATT     14 Sent Find Information Request, Handles: 0x0044..0xffff
. SamsungE_b9:e9:7c (Galaxy S… ATT     15 Rcvd Find Information Response, Handle: 0x0044 (Unknown: Unk…
. d2:a7:4c:76:f3:e0 (Prodigio… ATT     14 Sent Find Information Request, Handles: 0x0045..0xffff
. SamsungE_b9:e9:7c (Galaxy S… ATT     14 Rcvd Error Response - Attribute Not Found, Handle: 0x0045 (U…
. d2:a7:4c:76:f3:e0 (Prodigio… ATT     12 Sent Exchange MTU Request, Client Rx MTU: 185
. SamsungE_b9:e9:7c (Galaxy S… ATT     12 Rcvd Exchange MTU Response, Server Rx MTU: 23
. d2:a7:4c:76:f3:e0 (Prodigio… ATT     12 Sent Read Request, Handle: 0x0016 (Unknown: Unknown)
. SamsungE_b9:e9:7c (Galaxy S… ATT     11 Rcvd Read Response, Handle: 0x0016 (Unknown: Unknown)
. d2:a7:4c:76:f3:e0 (Prodigio… ATT     20 Sent Write Request, Handle: 0x0014 (Unknown: Unknown)
. SamsungE_b9:e9:7c (Galaxy S… ATT     10 Rcvd Write Response, Handle: 0x0014 (Unknown: Unknown)
```

- Look for **ATT** protocol
- First, many enumeration packets
- Then, a *Read Request* on handle `0x16`

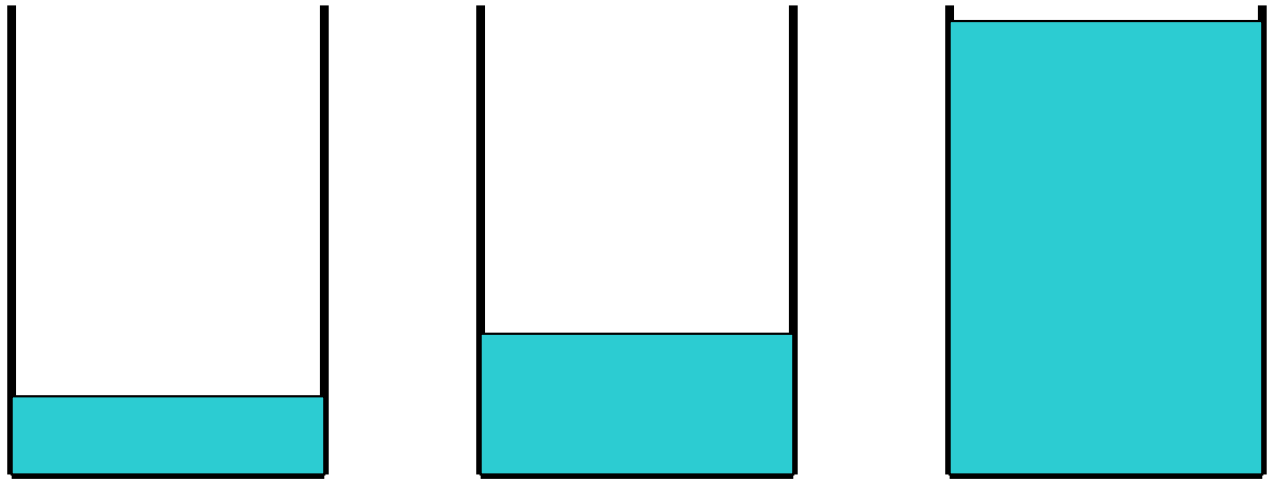
Where is the authorization?

```
▸ Frame 1952: 20 bytes on wire (160 bits), 20 bytes captured (160 bits)
▸ Bluetooth
▸ Bluetooth HCI H4
▸ Bluetooth HCI ACL Packet
▸ Bluetooth L2CAP Protocol
▾ Bluetooth Attribute Protocol
    ▸ Opcode: Write Request (0x12)
    ▸ Handle: 0x0014 (Unknown: Unknown)
      Value: 8418ffdaf230af08
      [Response in Frame: 1953]
```

```
0000  02 40 00 0f 00 0b 00 04  00 12 14 00 84 18 ff da   ·@······ ········
0010  f2 30 af 08                                       ·0··
```

- Look for **ATT** protocol
- First, many enumeration packets
- Then, a *Read Request* on handle `0x16`
- and a `Write Request` on handle `0x14`

We want a 90mL cup

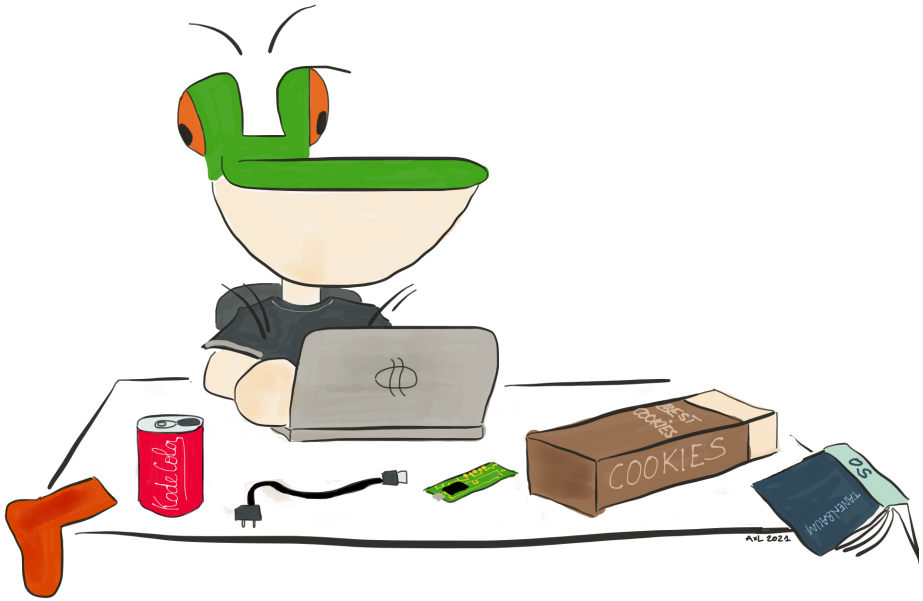


Ristretto 25mL Espresso 40mL Lungo 110mL

We want **90mL in 1 cup**

$25 + 25 + 40 = 90$ but that's **3 cups**

Cup size in the code




```
CupSizeVolume.RISTRETTO_VOLUME_RANGE = Range.open(15, 30);
CupSizeVolume.ESPRESSO_VOLUME_RANGE = Range.open(30, 70);
CupSizeVolume.ESPRESSO_VTP2_VOLUME_RANGE = Range.open(20, 70);
CupSizeVolume.LUNGO_VOLUME_RANGE = Range.open(70, 130);
CupSizeVolume.AMERICANO_COFFEE_VOLUME_RANGE = Range.open(15, 110);
```

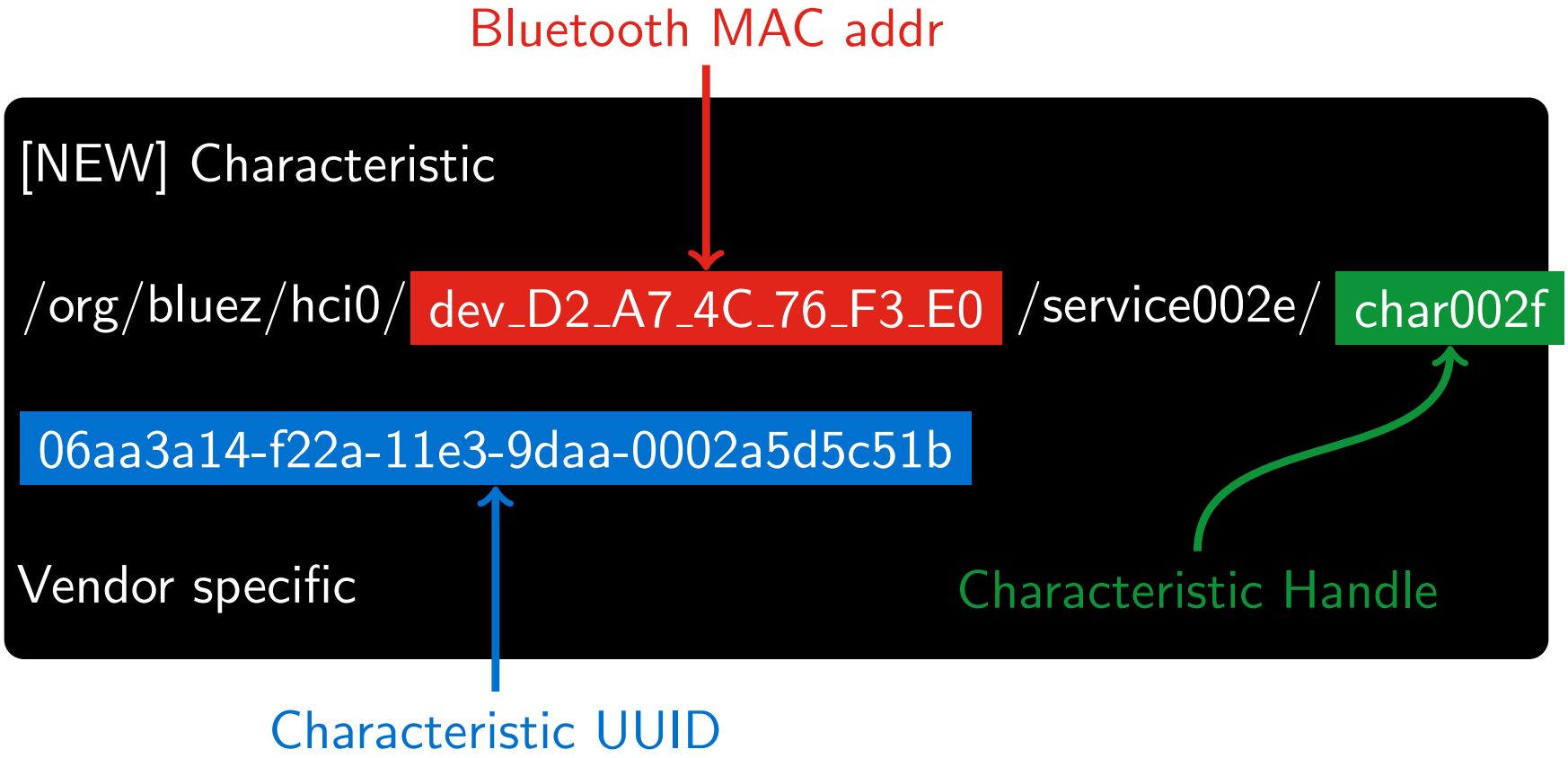
We can **customize** Lungo volume to 90mL

Which service?

```
CupSizeOperations.SERVICE_UUID =
↪  CupSizeOperations.UUID_TEMPLATE.evaluate(0x1940L);
UUID uUID0 = CupSizeOperations.UUID_TEMPLATE.evaluate(0x3A14L);
CupSizeOperations.WRITE_CUPE_SIZE_TARGET_CHARACTERISTIC_DESCRIPTION =
↪  new CharacteristicDescription(CupSizeOperations.SERVICE_UUID,
↪  uUID0);
UUID uUID1 = CupSizeOperations.UUID_TEMPLATE.evaluate(0x3A24L);
CupSizeOperations.VOLUME_CHARACTERISTIC_DESCRIPTION = new
↪  CharacteristicDescription(CupSizeOperations.SERVICE_UUID, uUID1);
```

- Service: 06aa**1940**-f22a-11e3-9daa-0002a5d5c51b. **User options** service.
- Cup size characteristic: **0x3A14**
- Cup volume characteristic: **0x3A24**

Which characteristic handle? bluetoothctl



- Long UUID: `06aa`**`3a14`**`-f22a-11e3-9daa-0002a5d5c51b`
- Short UUID: `3a14`
- Handle: `0x002f`
- With *gatttool*, use value handle = characteristic handle + 1

Cup size packet format

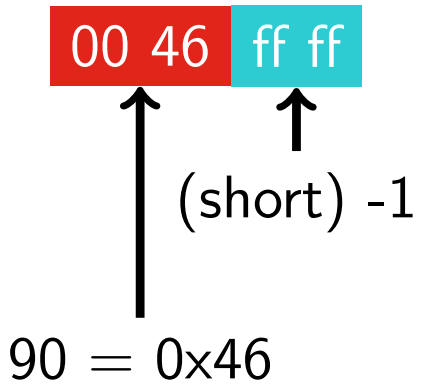
```
private ByteBuffer getCupSizekindData(int coffeeType) {
    ByteBuffer buf = ByteBuffer.allocate(2);
    buf.put(ByteConversion.toByteBuffer(((short)coffeeType)));
    return buf;
}

ByteBuffer getCupSizeKindByteBuffer(CupSizeType cupSizeType) {
    switch(cupSizeType) {
      /* ... */
        case LUNGO: {
            return this.getCupSizekindData(2);
        } } }
```

Send **00 02** to Cup Size characteristic (handle=0x0030)

Cup volume packet format

```
ByteBuffer getCupSizeVolumeData(int volume) {
    ByteBuffer byteBuffer0 = ByteBuffer.allocate(4);
    byteBuffer0.put(ByteConversion.toByteBuffer(((short)volume)));
    byteBuffer0.put(ByteConversion.toByteBuffer(((short)-1)));
    return byteBuffer0;
}
```



Brew a coffee

03 05 07 04 Delay Seconds (4 bytes) Coffee Type (2 bytes)

- Brew Lungo Now: 03 05 07 04 00 00 00 00 00 02
- Brew Ristretto in 5 mintutes: 03 05 07 04 00 00 01 2c 00 00
- Get Hot Water now: 03 05 07 04 00 00 00 00 00 04
- Cancel: 03 06 01 02

Service / Characteristic

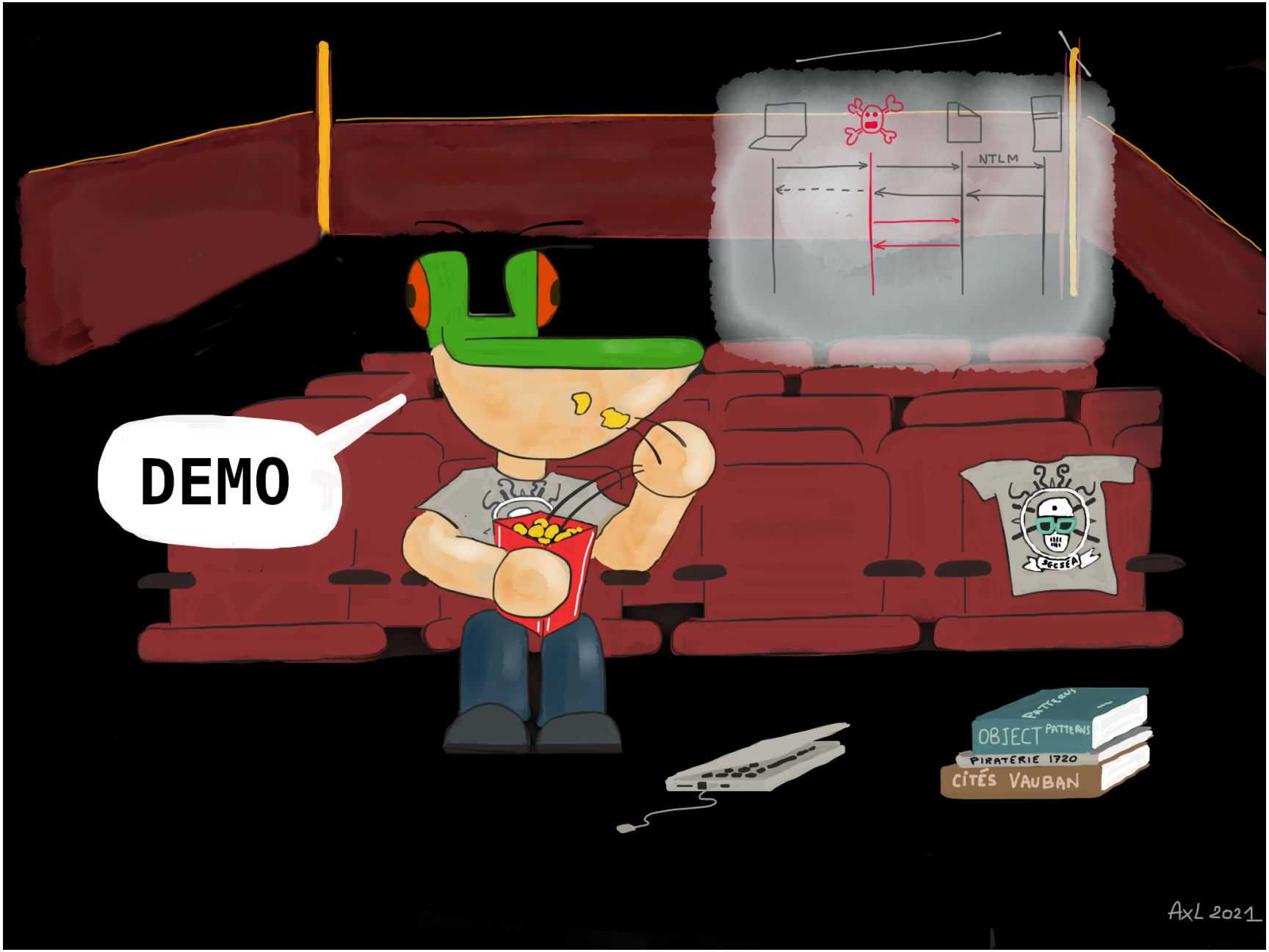
- Aeroccino Service: 06aa**1920**-f22a-11e3-9daa-0002a5d5c51b
- *Brew Key* characteristic: 0x3A42 (value handle: 0x0024)
- Brew response: 0x3A52 (value handle: 0x0026) e.g brewing, slider opened...

90mL coffee: summary

1. Pair + Authorize (handle 0x0014)
2. Customize Lungo Cup (handle=0x0030): `00 02`
3. Customize Volume (handle=0x0032): `00 46 ff ff`
4. Brew (handle=0x0024): `03 05 07 04 00 00 00 00 00 02`

BLE Packets are sent with Write Command, specifying the target handle. Tool: bluetoothctl, gatttool...

Demo


DEMO
NTLM
OBJECT PATTERNS
PIRATERIE 1720
CITÉS VAUBAN
AxL 2021

Recipes!

01 16 08 Recipe ID (2 bytes)

List of ingredients:

- 01 Coffee volume (2 bytes) in mL
- 02 Water volume (2 bytes) in mL
- ...
- Finish: 00 00 00

Much more...

- Top slider status
- Specify temperature of coffee
- Water hardness
- Standby delay
- Factory reset (by BLE command)
- Nb of coffee capsules in stock
- Get error messages: missing water, slider opened, no coffee, invalid command...

https://github.com/cryptax/talks/blob/master/BSidesMunich-2020/nespresso-techinfo.md

(updated March 2023)

Only 1 person can make coffee?!!

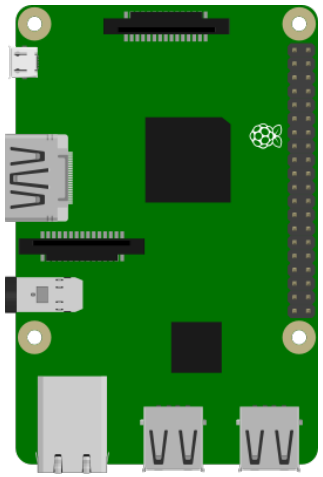
Bluetooth
Low Energy

Due to Bluetooth pairing and authorization: **single client**!
No WiFi, no Ethernet

Sharing the coffee maker

BLE

Single BLE user

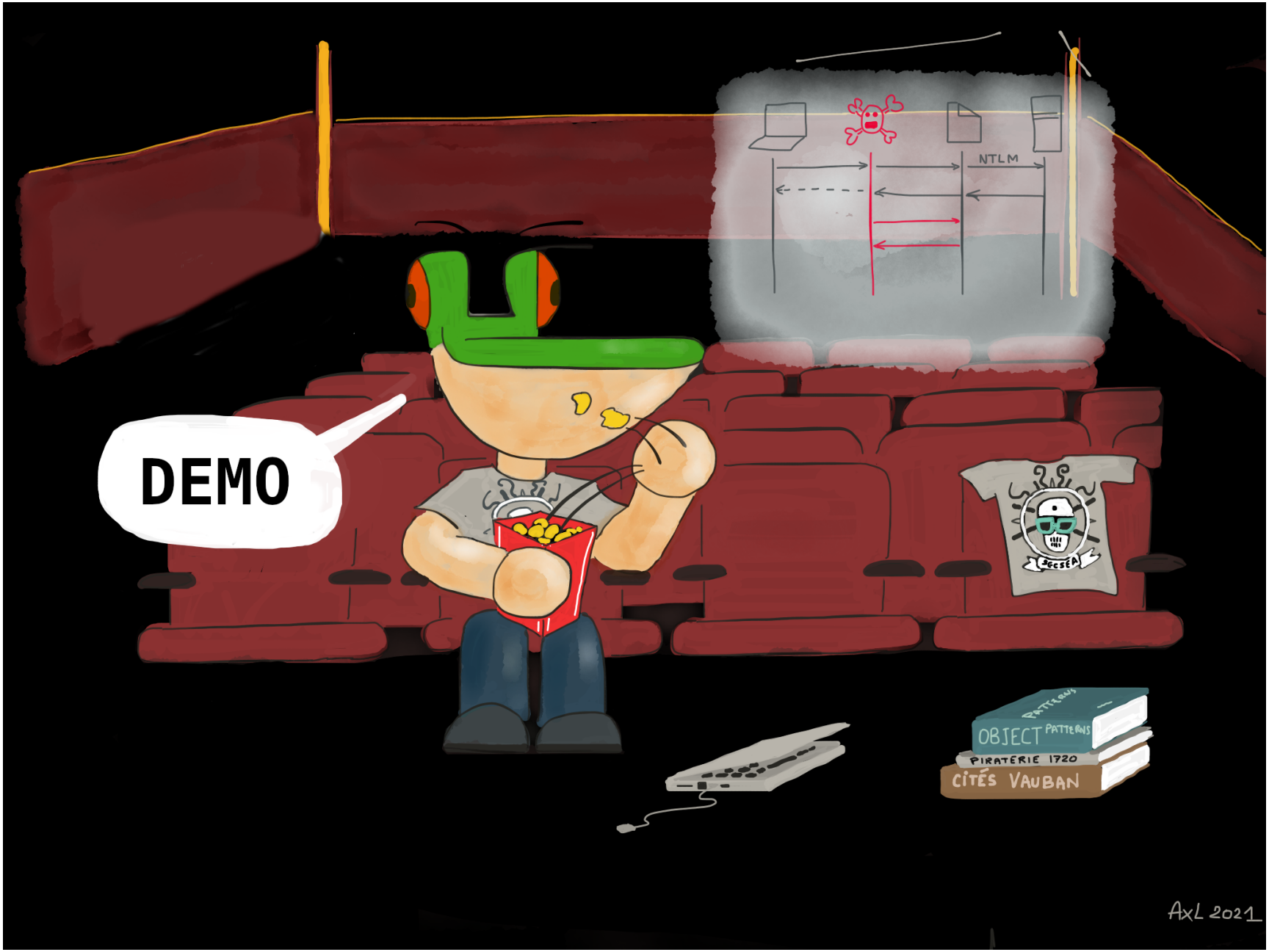


Webpresso website

User requests

https://github.com/cryptax/webpresso/

Demo


DEMO
NTLM
SECSEA
OBJECT PATTERNS
PIRATERIE 1720
CITÉS VAUBAN
AxL 2021

Hacking a Jump Rope

BLE communication

Hack here!

Hack here!

Hack here!

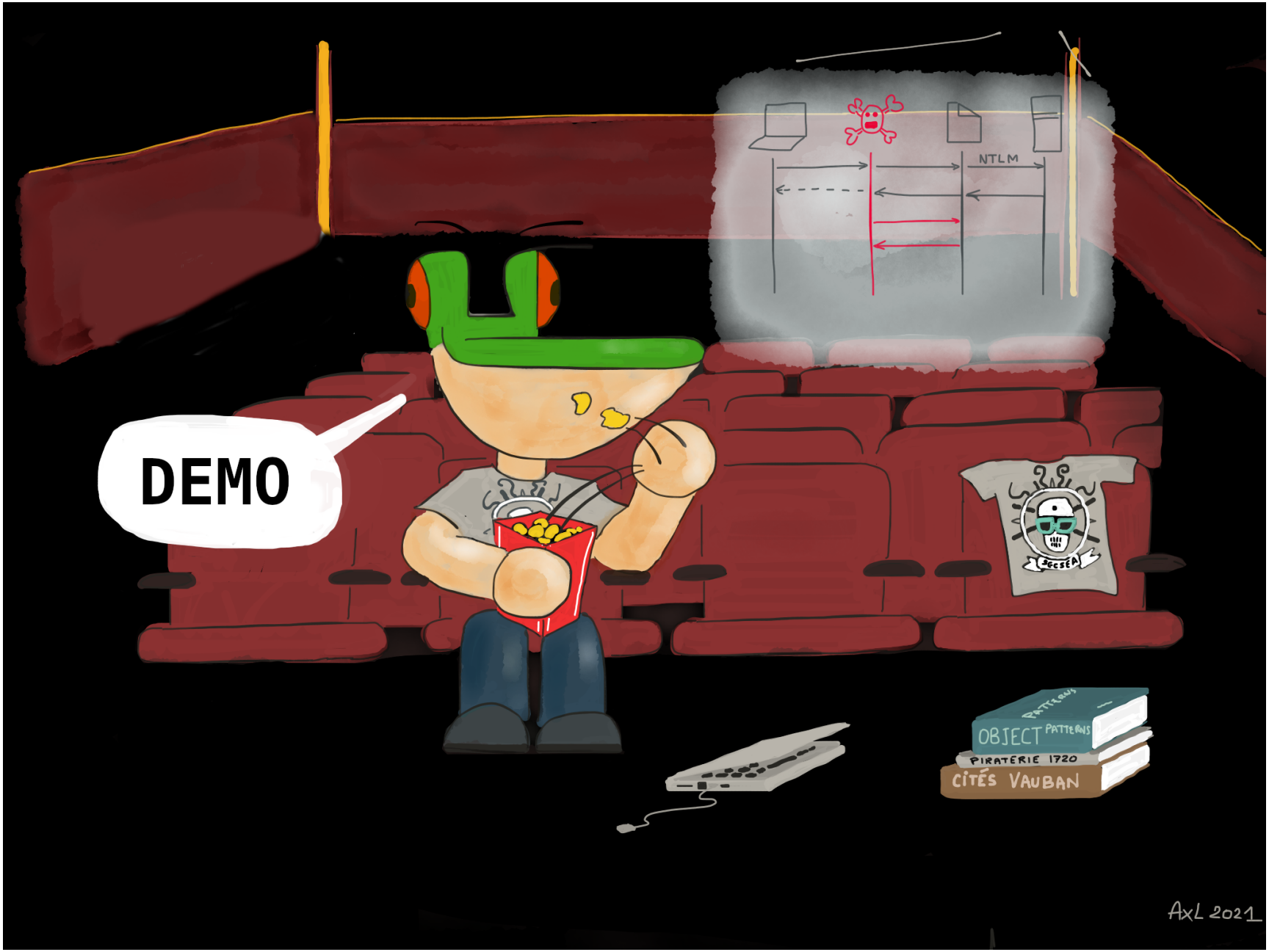
Android Logcat

```
16:57:25.322 27999 6570 E AndroidBLE: [10854] TAG: onNotifySuccess: RENPHO-ROPE-R1
16:57:25.324 27999 6570 E TAG
: HEX=0100061939101C0A1505FD>>>>>>>>>>05FD
16:57:25.353 27999 6570 I TAG
: onChanged==data:[81, 00, 03, 00, 28, 00, 4e, 00]
16:57:25.460 27999 6570 I TAG
: onChanged==data:[83, 00, 05, 00, 7d, 2b, c0, ab, f0, 71]
16:57:25.551 27999 6570 I TAG
: onChanged==data:[84, 00, 02, 00, 00, de, 75]
```

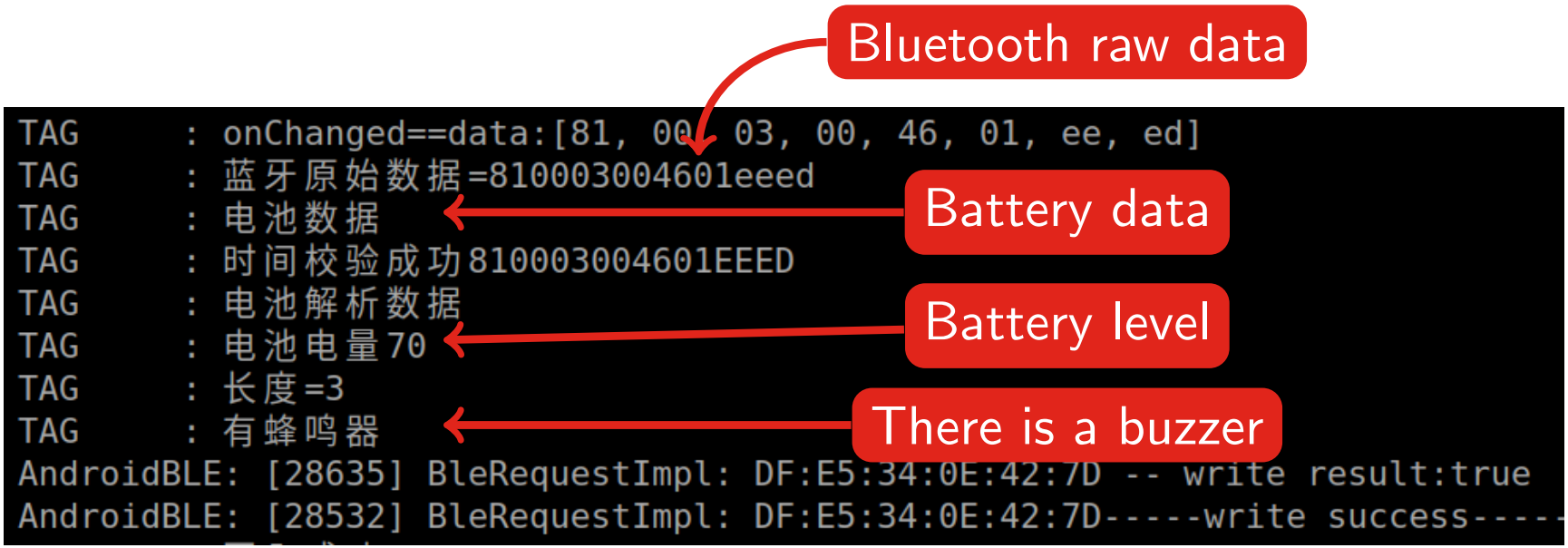
BLE Packet Payload!!!
HEX=...

Application logs
with section **TAG**

Live Demo!!!


DEMO
NTLM
SECSEA
OBJECT PATTERNS
PIRATERIE 1720
CITÉS VAUBAN
AxL 2021

Understanding the logs

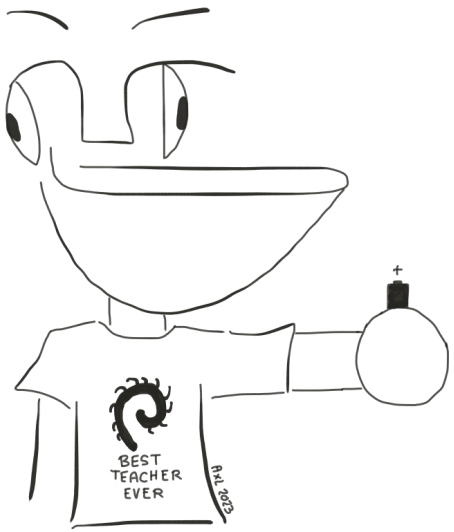


Step 1: Translate

Search in code

Search for Battery Level

```
public final void updateBattery(int v) {
  Log.d("TAG", "CHINESE CHARACTERS" + v);
  BleLiveData.bleBattery.postValue(Integer.valueOf(v));
}
```



- Method is named **updateBattery()** - makes sense
- Provides interesting classes to look into: `BleLiveData`

Search in code

Search for Bluetooth Raw Data

```
private final void parseCommand(byte[] arr_b, BleDevice
↪  bleDevice0) {
  XLog.i(("CHINESE CHARACTERS=" +
↪  HexUtil.formatHexString(arr_b)));
  ThreadUtils.INSTANCE.getSingleThreadExecutor().execute(
  ((Runnable)new BlueLeService.parseCommand.1(this, arr_b,
↪  bleDevice0)));
}
```

- Reverse engineers like **parseCommand()** methods!
- Class: `BlueLeService`

We get the whole picture

Mode	BLE packet	Comment
Numbers Count Down Mode	`02 00 05 81 TT TT TT TT CC CC`	`TT TT TT TT` is the target number of jumps, and `CC CC` is the CRC-16/MODBUS
Time Count Down Mode	`02 00 05 82 TT TT TT TT CC CC`	`TT TT TT TT` is the target number of seconds for the session
Free Jump Mode	`02 00 05 80 00 00 00 00 59 C0`	The CRC is fixed to `59 C0`

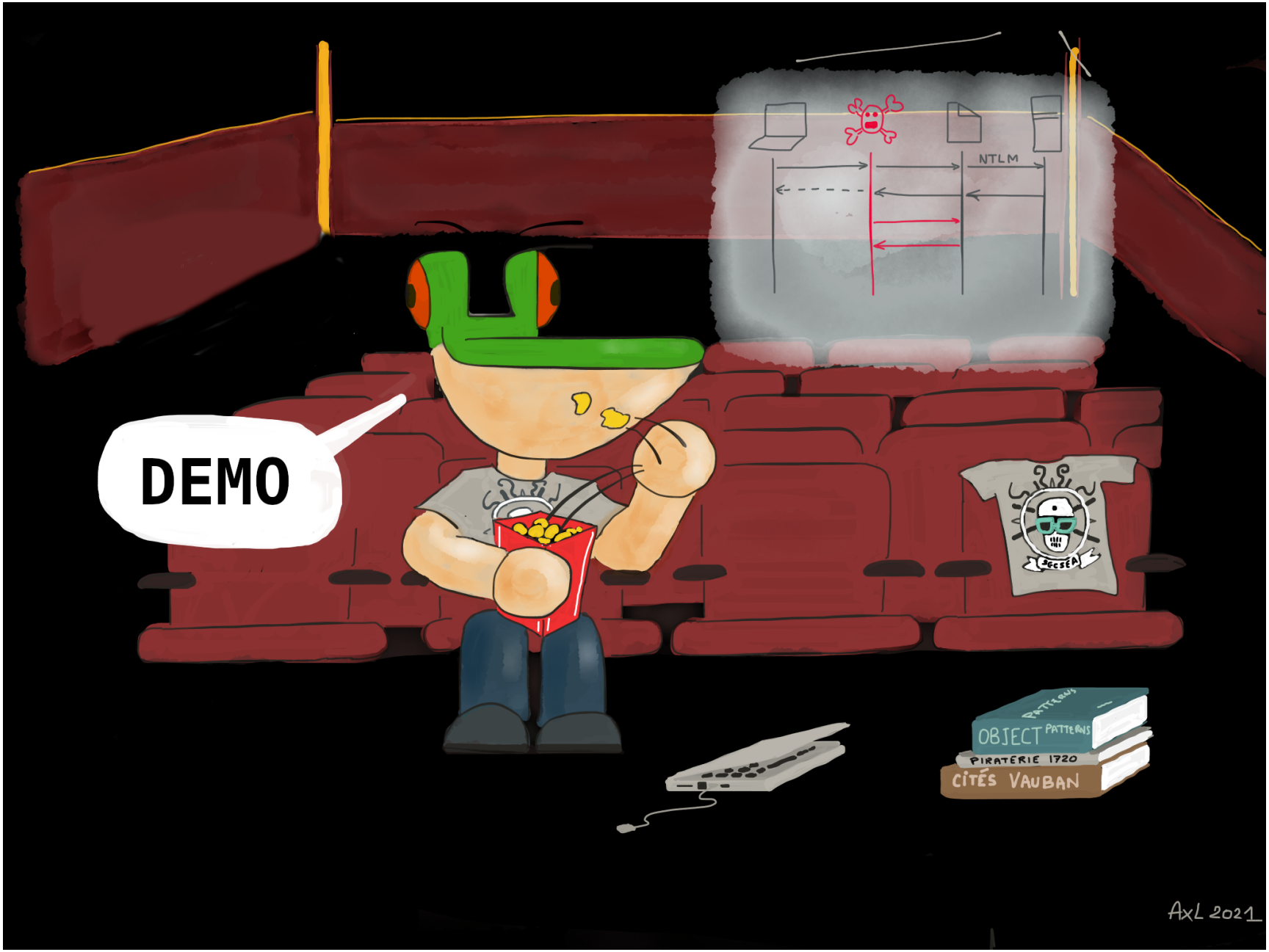
- Reverse **parseCommand()** (and a few others)
- Learn how to start a session, cancel, turn buzzer on/off etc.

Jump Rope Command: Summary

1. Connect to the device
2. Write to UUID `00005302-0000-0041-4c50-574953450000`, handle 0x0010
3. 02 00 05 81 00 00 05 39 DB 3E
   - 0x81 = Number Count Down Mode
   - 0x539 = 1337 target number of jumps
   - 0xDB3E = `CRC16_MODBUS(packet)`

More at https://github.com/cryptax/talks/tree/master/Insomnihack-2023

Live Demo!!!


DEMO
NTLM
SECSEA
OBJECT PATTERNS
PIRATERIE 1720
CITÉS VAUBAN
AxL 2021

CTF: How can we validate the answer?

CTF: How can we validate the answer?

1. **Manual** validation / Demo in front of organizers

CTF: How can we validate the answer?

1. **Manual** validation / Demo in front of organizers
2. Validate on a **web** server

CTF: How can we validate the answer?

02 00 05 81 00 00 05 39 DB 3E → ph0wn{beautiful_flag}

1. **Manual** validation / Demo in front of organizers
2. Validate on a **web** server
3. Validate on the rope itself: need to modify the **firmware**

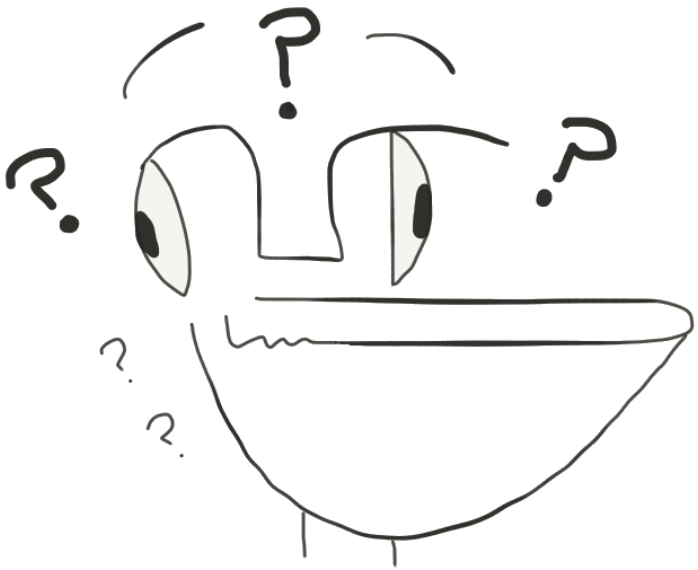
CTF: How can we validate the answer?

02 00 05 81 00 00 05 39 DB 3E → ph0wn{beautiful_flag}

1. **Manual** validation / Demo in front of organizers
2. Validate on a **web** server
3. Validate on the rope itself: need to modify the **firmware**
4. Validate on a **fake** jump rope

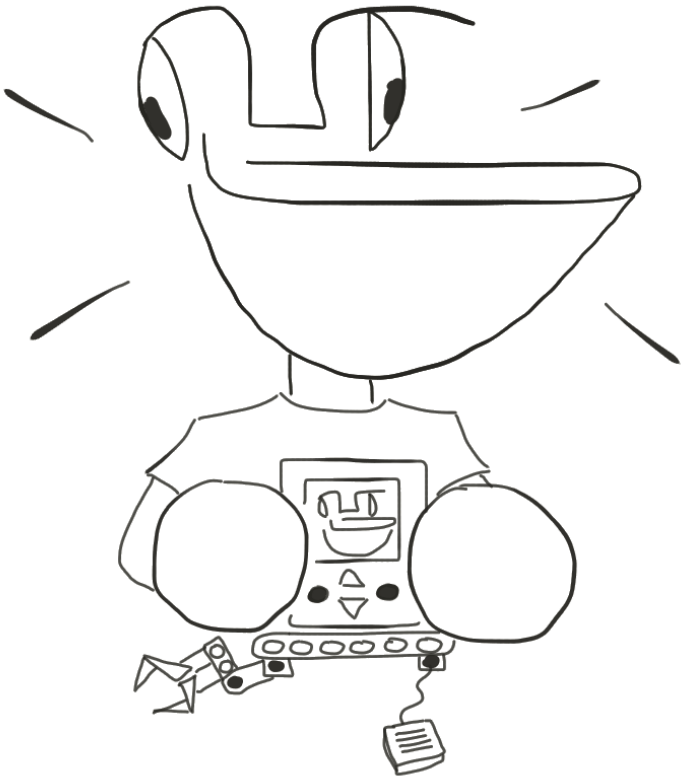
Behaves like a jump rope, from a BLE point of view. But no rope.

I tried, and failed, for weeks



- Turn my laptop into a **BLE peripheral**
- Build issues with obsolete projects
- Bugs or non supported features
- My own bugs, but could not find help

Solution at Hardwear.io CTF



- BLE challenge using a small Arduino-like device
- They shared the code (thanks!)
- Uses BLE from Arduino-ESP32 libraries

https://github.com/espressif/arduino-esp32

Fake Jump Rope Design

- **Same services** and characteristics e.g. same model number etc.
- **Dummy OTA** service: does nothing
- Add a new CTF service **deadbeef**`-ff11-aadd-0000-000100000001`
- Read the **flag** from characteristic `deadbeef-ff11-aadd-0000-000100000002`. After you've sent the correct Jump Command.

Fake Jump Rope Design

- **Same services** and characteristics e.g. same model number etc.
- **Dummy OTA** service: does nothing
- Add a new CTF service **deadbeef**-ff11-aadd-0000-000100000001
- Read the **flag** from characteristic deadbeef-ff11-aadd-0000-000100000002. After you've sent the correct Jump Command.

How to erase the flag between teams?

- Allow a **single** connection at a given time: stop advertising when a client has connected
- **Erase** flag at connection/disconnection

Jump Rope Validation Server

WeMo Lolin32

Advice / Take away

Hackers

- Inspect **Android logs**

Developers

- **Remove debug** logs from product. Keep other logs.

Advice / Take away

Hackers

- Inspect **Android logs**
- **BLE connection** is always **painful**: impossible to connect, unwanted disconnect... Beware in CTFs!

Developers

- **Remove debug** logs from product. Keep other logs.
- Test **quality** of BLE chips. Pairing is a pain to users, avoid it.

Advice / Take away

Hackers

- Inspect **Android logs**
- **BLE connection** is always **painful**: impossible to connect, unwanted disconnect... Beware in CTFs!
- Activate **BLE notifications** when possible.

Developers

- **Remove debug** logs from product. Keep other logs.
- Test **quality** of BLE chips. Pairing is a pain to users, avoid it.
- Use BLE **encryption** if you want to communicate **secrets**

Advice / Take away

Hackers

- Inspect **Android logs**
- **BLE connection** is always **painful**: impossible to connect, unwanted disconnect... Beware in CTFs!
- Activate **BLE notifications** when possible.
- Read public **manuals**: Factory Reset procedure, hints on hardware...

Developers

- **Remove debug** logs from product. Keep other logs.
- Test **quality** of BLE chips. Pairing is a pain to users, avoid it.
- Use BLE **encryption** if you want to communicate **secrets**
- Want to go **Green**? Make your device **repairable** (regular screws, service manual, parts, 3D models...)

Advice / Take away (continued)

Hackers

- Hack from the angle *you* master most.
  Good at hardware? Hardware RE.
  Good at Android? Android RE
  etc.

Developers

- Hackers are **not your enemies**. They can come up with interesting ideas. We mostly hack things we *like*.

Advice / Take away (continued)

Hackers

- Hack from the angle *you* master most.
  Good at hardware? Hardware RE.
  Good at Android? Android RE
  etc.
- Both devices were **positively** hacked

Developers

- Hackers are **not your enemies**. They can come up with interesting ideas. We mostly hack things we *like*.

Advice / Take away (continued)

Hackers

- Hack from the angle *you* master most.
  Good at hardware? Hardware RE.
  Good at Android? Android RE etc.
- Both devices were **positively** hacked
- Unlikely risk of malware infection

Developers

- Hackers are **not your enemies**. They can come up with interesting ideas. We mostly hack things we *like*.
- No easy way to store & execute code on small BLE devices (good). Pay attention to devices which are *directly* connected to Internet + have a few KB storage.

Thanks for your attention!

Twitter: @cryptax
Mastodon: @cryptax@mastodon.social

Thanks to @virtualabs, @CayreRomain, @PagetPhil and *Soudure au beurre*

If you have a cool idea for an IoT challenge, please talk to me!

References

- Tech notes on Smart Coffee: https://github.com/cryptax/talks/blob/master/BSidesMunich-2020/nespresso-techinfo.md
- Webpresso: https://github.com/cryptax/webpresso/
- *Ajoutez le WiFi à une machine à café et rendez vos collègues heureux*, Hackable Magazine, no 33 (in French). https://boutique.ed-diamond.com/numeros-deja-parus/1527-hackable-magazine-33.html
- Tech note on Smart Jump Rope and control program: https://github.com/cryptax/talks/tree/master/Insomnihack-2023
- Ph0wn CTF 2019 Coffee Write Up https://github.com/ph0wn/writeups/blob/master/2019/network/mewantcoffee-1-2-3.md
- Ph0wn CTF 2022 Jump Rope Write Up https://github.com/ph0wn/writeups/blob/master/2022/network/jumprope/solution-cryptax.md
- Jump Rope Validation Server sources https://github.com/ph0wn/writeups/blob/master/2022/network/jumprope/src/jumprope2.ino
- FortiGuard Labs: https://www.fortiguard.com
- Ph0wn CTF: https://ph0wn.org