Stay fit: Hack a Jump Rope

Axelle Apvrille

Troopers, June 2023

Who am I?

Axelle Apvrille

Principal Security Researcher at **Fortinet**, @cryptax
Mobile malware
IoT + Ph0wn CTF

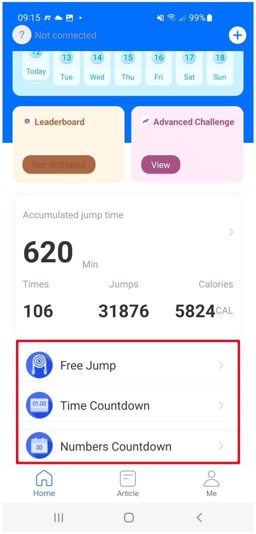
Agenda

- Hack a **Jump Rope**: Understand its **Communication Protocol**
- Create a **CTF** challenge: protect the flag, prevent team cheating...

Renpho Smart Jump Rope R-Q001

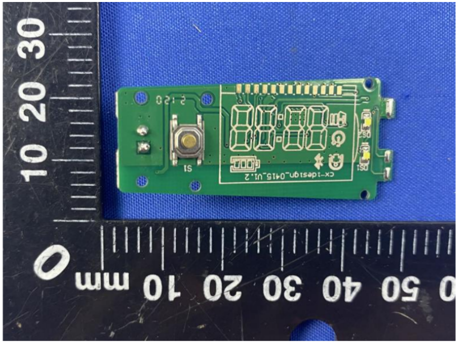
Jump Modes

- Free Jump Mode.
- Time Countdown Mode.
- Numbers Countdown Mode.



Viewing the PCB, without opening the jump rope!

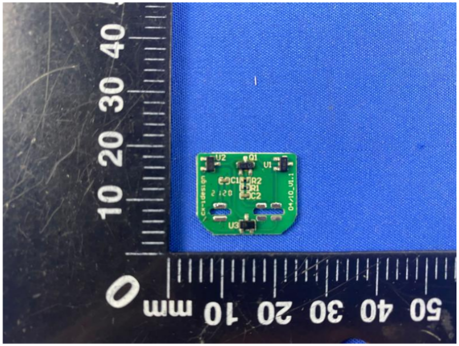
- FCC.io is your friend: `https://fccid.io/2APXU-R-Q001`
- Board 1: Bluetooth antenna, LCD, button, Beken chip
- **Beken BK3432**: Bluetooth 5.0, low consumption, OTA



Source: FCC.io Internal Parts

Viewing the PCB, without opening the jump rope!

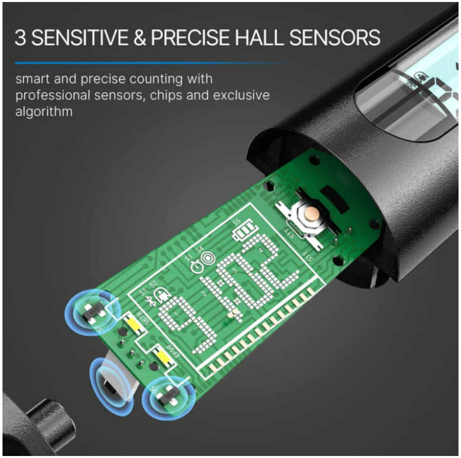
- FCC.io is your friend: `https://fccid.io/2APXU-R-Q001`
- Board 1: Bluetooth antenna, LCD, button, Beken chip
- **Beken BK3432**: Bluetooth 5.0, low consumption, OTA
- Board 2: perpendicular. Resistors, capacitors, 3 HALL sensors



Source: FCC.io Internal Parts

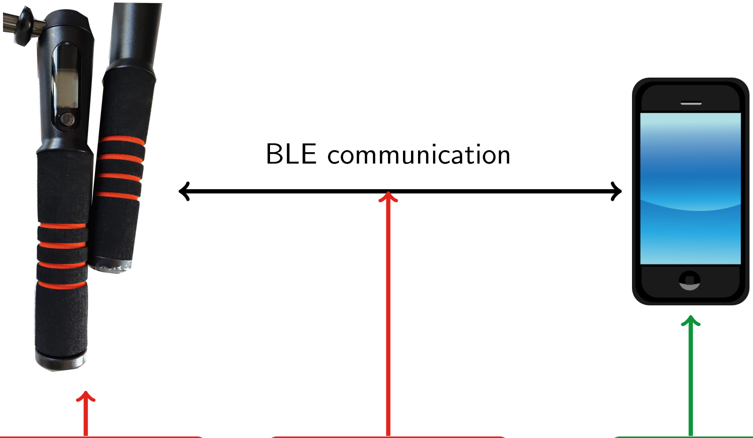
Viewing the PCB, without opening the jump rope!

- FCC.io is your friend: `https://fccid.io/2APXU-R-Q001`
- Board 1: Bluetooth antenna, LCD, button, Beken chip
- **Beken BK3432**: Bluetooth 5.0, low consumption, OTA
- Board 2: perpendicular. Resistors, capacitors, 3 HALL sensors


3 SENSITIVE & PRECISE HALL SENSORS
smart and precise counting with professional sensors, chips and exclusive algorithm

Source: `https://renpho.com/collections/fitness/products/smart-jump-rope-1`

Hacking a Jump Rope



You can hack here

UART, Firmware, Hardware RE...

You can hack here

Adafruit BLE Sniffer, Ubertooth...

I hack here!

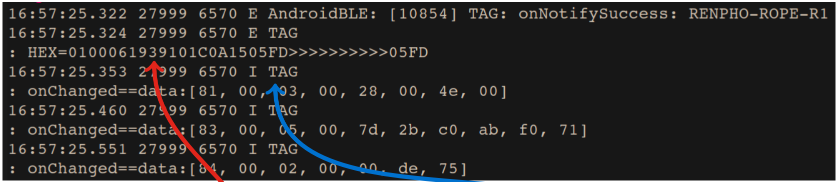
reverse app, logcat, BLE snif

Hacking IoT: different cases

Device	**Reverse engineering method**
Magimix Smart coffee machine	BLE HCI snoop on the phone, app
Beam toothbrush	Bluefruit, app
ReconJet smart glasses	Android, app
Freestyle Libre glucose sensor	Firmware, app
Renpho Jump Rope	Android **Logcat**, app

The **application** is a **valuable source of information**

Android Logcat

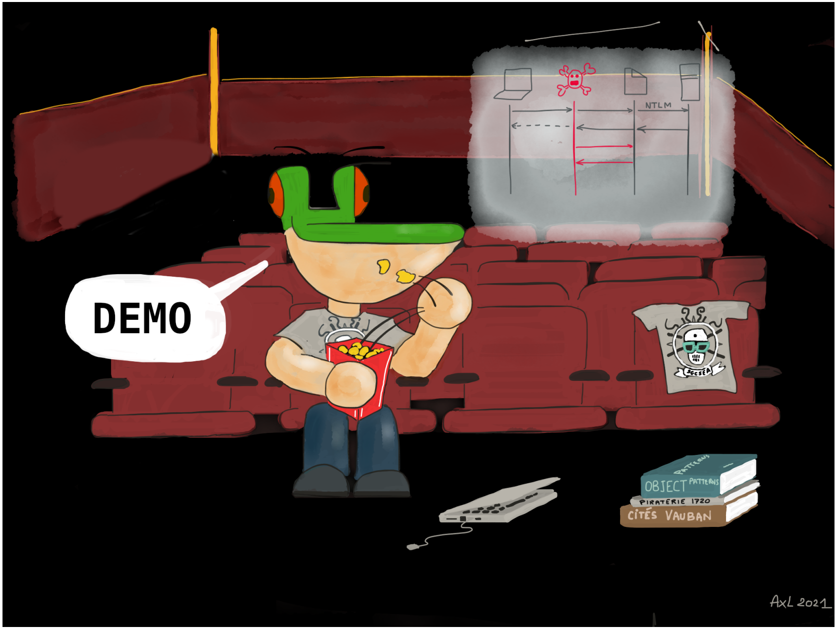


```
16:57:25.322 27999 6570 E AndroidBLE: [10854] TAG: onNotifySuccess: RENPHO-ROPE-R1
16:57:25.324 27999 6570 E TAG
: HEX=0100061939101C0A1505FD>>>>>>>>>>05FD
16:57:25.353 27999 6570 I TAG
: onChanged==data:[81, 00, 03, 00, 28, 00, 4e, 00]
16:57:25.460 27999 6570 I TAG
: onChanged==data:[83, 00, 05,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

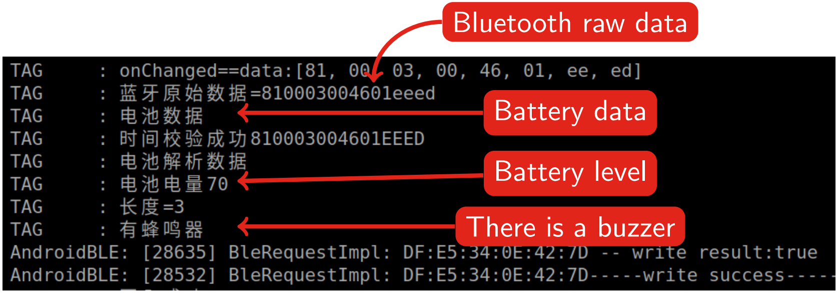
BLE Packet Payload!!!
HEX=...

Application logs with section **TAG**

Live Demo


NTLM
DEMO
SECSEA
PATTERNS
OBJECT PATTERNS
PIRATERIE 1720
CITÉS VAUBAN
AxL 2021

Understanding the logs



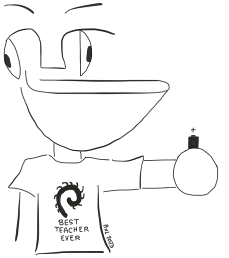
```
TAG      : onChanged==data:[81, 00, 03, 00, 46, 01, ee, ed]
TAG      : 蓝牙原始数据=810003004601eeed
TAG      : 电池数据
TAG      : 时间校验成功810003004601EEED
TAG      : 电池解析数据
TAG      : 电池电量70
TAG      : 长度=3
TAG      : 有蜂鸣器
AndroidBLE: [28635] BleRequestImpl: DF:E5:34:0E:42:7D -- write result:true
AndroidBLE: [28532] BleRequestImpl: DF:E5:34:0E:42:7D-----write success-----
```

Step 1: Translate

Search the code

Where is “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 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`

Jump Rope Commands

Command	BLE packet
Start Free Jump Mode	02 00 05 80 00 00 00 00 59 C0
Start Number Countdown Mode	02 00 05 81 TT TT TT TT CC CC
Start Time Countdown Mode	02 00 05 82 TT TT TT TT CC CC
Cancel Mode	02 00 05 01 00 00 00 00 47 FC
Set Buzzer On	08 00 01 01 14 C2
Set Buzzer Off	08 00 01 00 D4 03
Read Offline Data	04 02 02 00 00 00 74
Clear Offline Data	05 00 01 A5 03 C1
Switch to OTA mode	06 01 01 A5 47 C1
Get Serial Number	03 00 04 00 00 00 00 00 9A 01
...	

- `TT TT TT TT`: target number
- `CC CC`: CRC16/MODBUS

Quick BLE background

Organization of data

- **BLE Characteristic** ≈ entry point to read and/or write data e.g. command characteristic (write).
- Characteristics are referenced by a **UUID**, or a **handle**.
- **Notifications** may be sent when a **characteristic changes**. You need to **request notifications** to receive them.
- **BLE Services** group characteristics

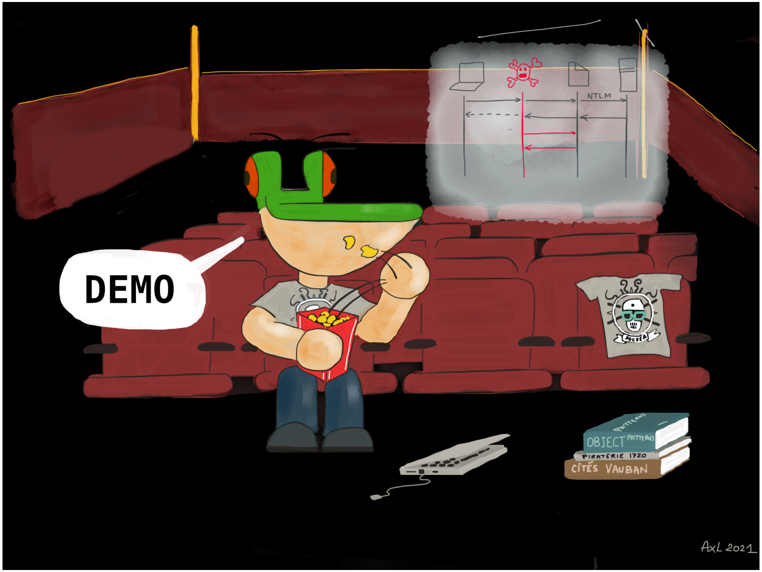
Sending BLE packets

- **Android**. Several apps e.g *nRF Connect*
- **Linux**. `bluetoothctl`. Older: `gatttool`.

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)`

Live Demo



Jump Rope Control Source code

CTF: How can we validate the answer?

$$\begin{tikzcd}\boxed{\text{02 00 05 81 00 00 05 39 DB 3E}} \arrow[r] & \boxed{\text{ph0wn\{beautiful\_flag\}}}\end{tikzcd}$$

CTF: How can we validate the answer?

$$\begin{tikzcd} \text{02 00 05 81 00 00 05 39 DB 3E} \arrow[r] & \text{ph0wn\{beautiful\_flag\}} \end{tikzcd}$$

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

CTF: How can we validate the answer?

$$\begin{tikzcd}\text{02 00 05 81 00 00 05 39 DB 3E} \arrow[r] & \text{ph0wn\{beautiful\_flag\}}\end{tikzcd}$$

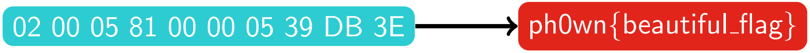
1. **Manual** validation / Demo in front of organizers
2. Validate on a **web** server

CTF: How can we validate the answer?

02 00 05 81 00 00 05 39 DB 3E $\longrightarrow$ 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?

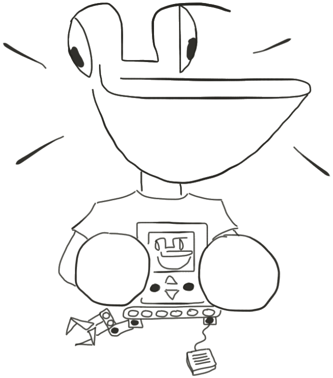


1. **Manual** validation / Demo in front of organizers
2. Validate on a **web** server
3. Validate on the rope itself: need to modify the **firmware**
4. Validate on a **fake** jump rope. Behaves like a jump rope from a BLE point of view, but no rope.

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`

Source code

```
class cmdCallback: public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    std::string value = pCharacteristic->getValue();
    // write your callbacks
  }
}

void setup() {
  // initialize BLE device as a server
  BLEDevice::init(DEVICE_NAME_VALUE);
  BLEServer *pServer = BLEDevice::createServer();
  BLEService *pServiceRenpho =
↪   pServer->createService(SVC_RENPHOFIT_UUID);
  pCharRenpho =
↪   pServiceRenpho->createCharacteristic(CHARAC_RENPHOFIT_WRITE_UUID,
↪   BLECharacteristic::PROPERTY_WRITE);
  pCharRenpho->setCallbacks(new cmdCallback());
  pServiceRenpho->start();
  pAdvertising = pServer->getAdvertising();
  // ...
}
```

Design of the Fake Jump Rope

WeMo Lolin32

- **Same services** and characteristics e.g. same model number etc.
- Dummy **OTA** service: **does nothing**
- Add a **CTF** service and characteristic to **read** the flag

Show flag only after correct command

1. By default, flag characteristic is empty
2. Check command callback value
3. If correct, put flag in its characteristic

Protect flag, prevent cheating!

How can we prevent this?

- **Team A does the good work**
- Flag is available
- **Team B steals** the flag

Solution

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

Deployment notes

- There are **150** participants
- I would not recommend using a single BLE fake rope: you **always** need a **backup in CTFs!**
- Deployed **3 fake ropes**
- A few teams experienced a few BLE connection issues, but nothing major. All 3 devices worked until the end.
- **2 teams** solved the challenge
- 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`

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 CTF challenge, please talk to me!