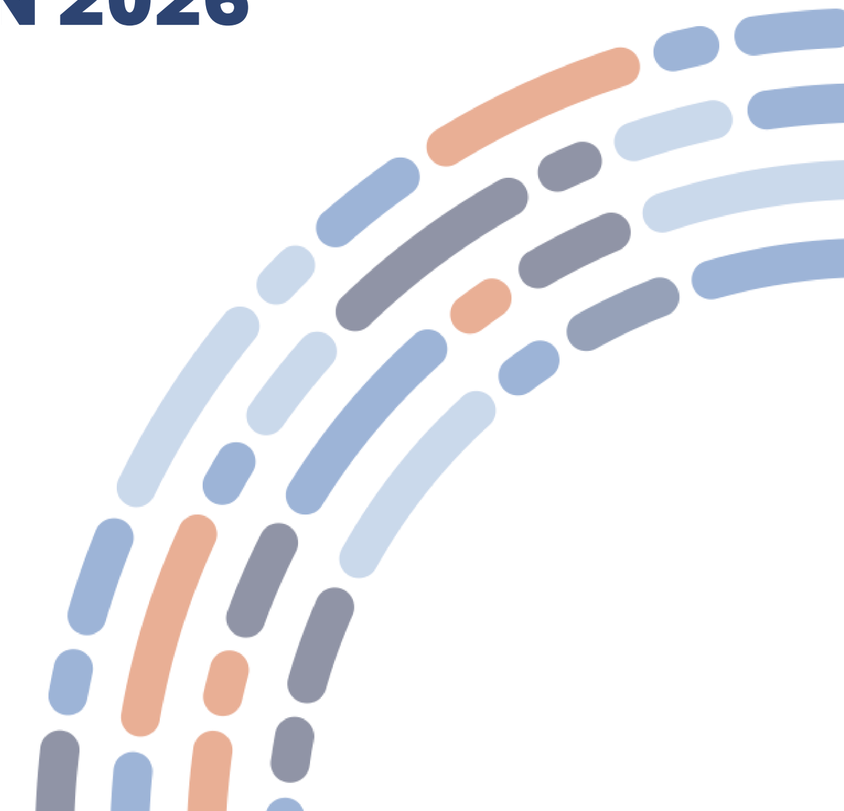
PWNING BLUETOOTH DEVICES IN UNEXPECTED WAYS

DAMIEN CAUQUIL - BRUCON 2026



Quarkslab

WHO AM I?

Damien Cauquil

- Security Engineer at Quarkslab
- Bluetooth Low Energy addict since 2016
- Maintainer of **WHAD** 😄
  https://whad.io/

WHAT'S THIS TALK ABOUT?

- Unexpected/weird behaviors I discovered in some BLE devices
- Some Bluetooth Low Energy features you may not be aware of
- How I thought I found a critical vulnerability in most SoCs (with a final twist)

INTRODUCTION

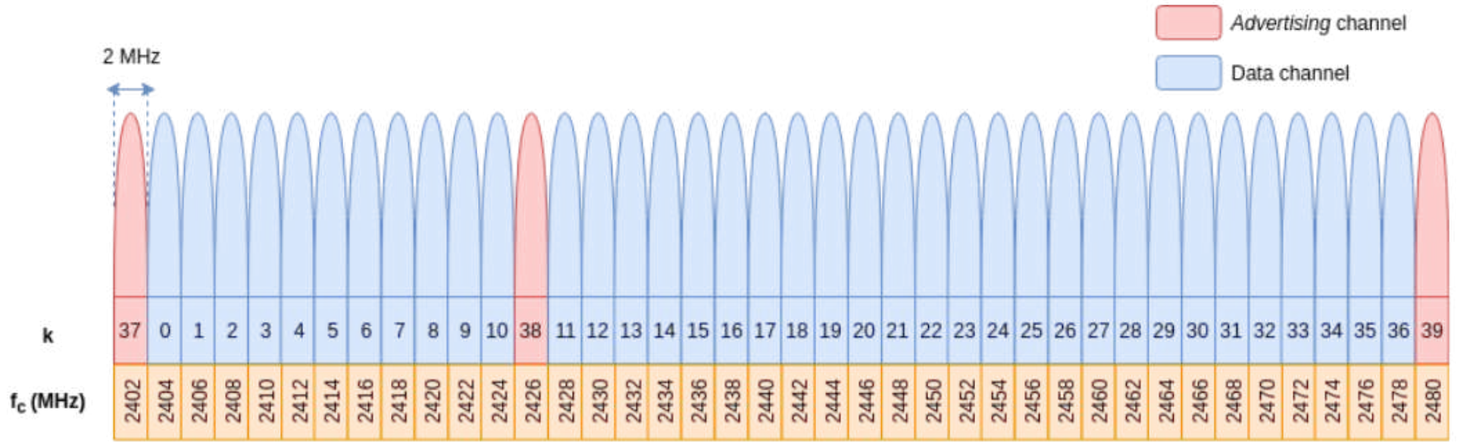
Bluetooth Low Energy (BLE) explained, short short version

BLE INTRODUCES LOW POWER

- Exchange data at **regular interval** (7.5ms to 4s)
- Uses **frequency hopping**
- **Re-uses components** used in Bluetooth Classic

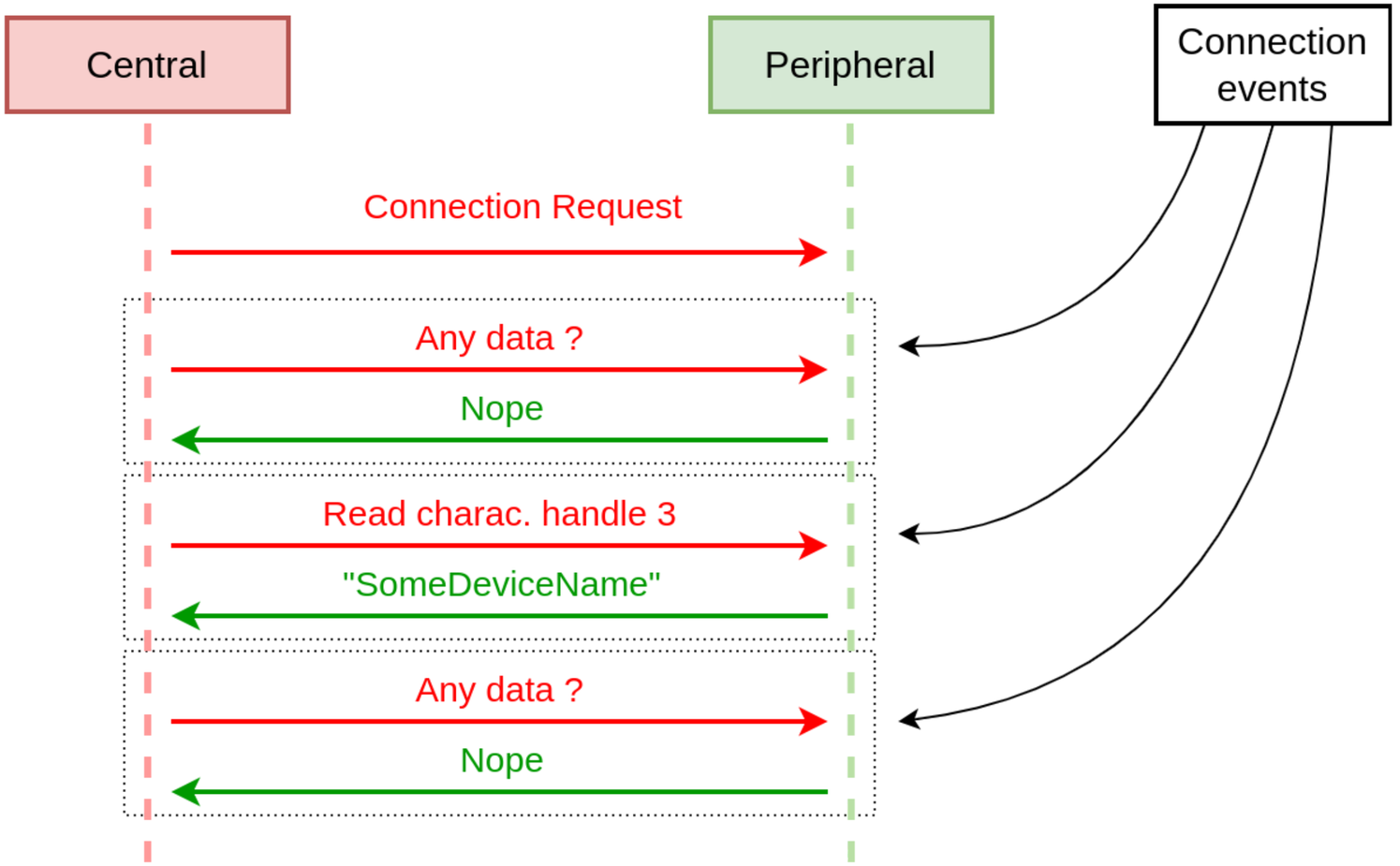
ADVERTISING PDUS (BROADCAST)

Advertiser



ADVERTISING DATA RECORDS (TLV)

CONNECTING TO A BLE DEVICE



PROTOCOL STACK

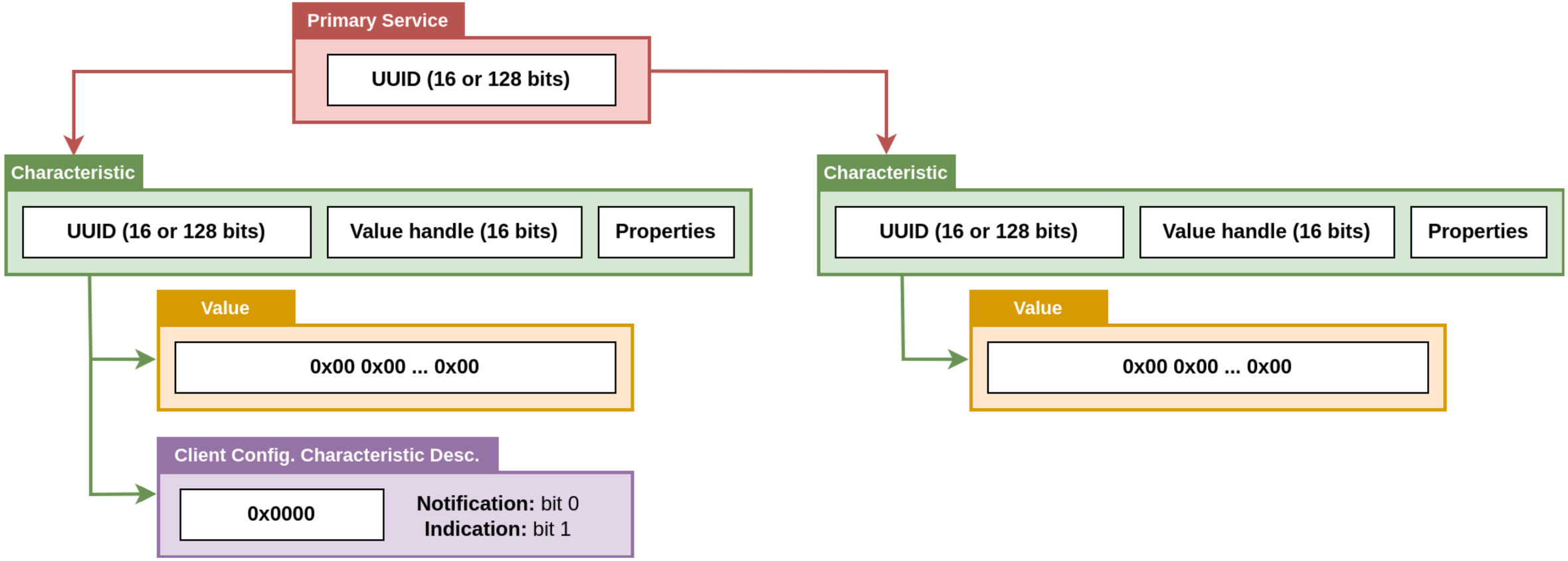
GAP	**G**eneric **A**ttribute **P**rofile
GATT	**G**eneric **ATT**ribute **P**rotocol
ATT	**ATT**ribute **P**rotocol
L2CAP	Handles **fragmentation** and **reassembly**
Link-Layer	Exchange data over the air

EXAMPLE PDU

ATTRIBUTE PROTOCOL

- Each attribute has a **type UUID** and a **16-bit handle**
- Defines some key attributes: **Primary service**, **Characteristic**, ...
- Defines a set of **operations** and **procedures**

ATTRIBUTE DATABASE



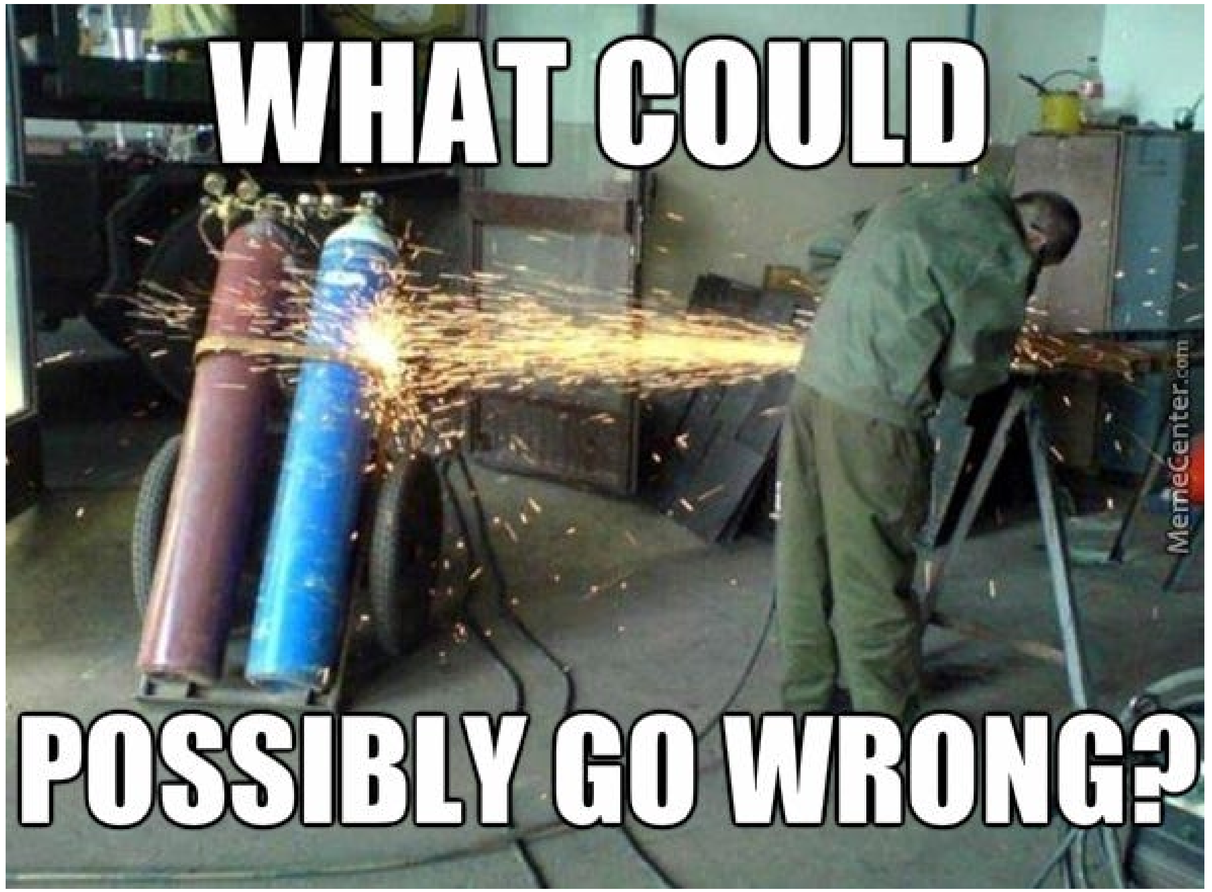
HOW IS DATA EXCHANGED?

- Data is sent by **writing into a characteristic's value**
- Remote device answers by **updating a characteristic's value**
- **Notifications** are used to receive the new value when updated

SECURITY MECHANISMS

- Address randomization for **privacy**
- **Secure Connections** starting from version 4.2
- More **robust channel hopping** algorithm* (CSA #2)

(*) well, not really a security feature but it makes sniffing way harder 😅


WHAT COULD
POSSIBLY GO WRONG?
MemeCenter.com

10+ YEARS OF VULNERABILITIES...

- **Implementation issues**: RCE, DoS, ...
- **Design issues**: security downgrade, MitM, jamming
- **Crypto issues**: weak encryption, key entropy/reuse, ...

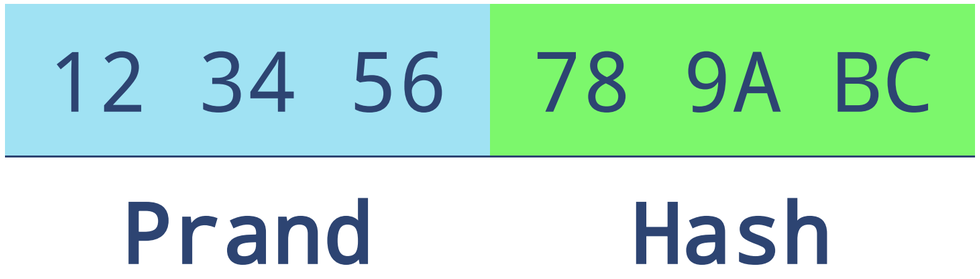
… AND MAYBE MORE!

LET'S TALK ABOUT PRIVACY

ADDRESS RANDOMIZATION

- **Non-Resolvable Private Address**:
  randomized every X minutes to avoid tracking

- **Resolvable Private Address**:
  Address contains a challenge/response (bonding required)



DEFEATING RANDOMIZATION

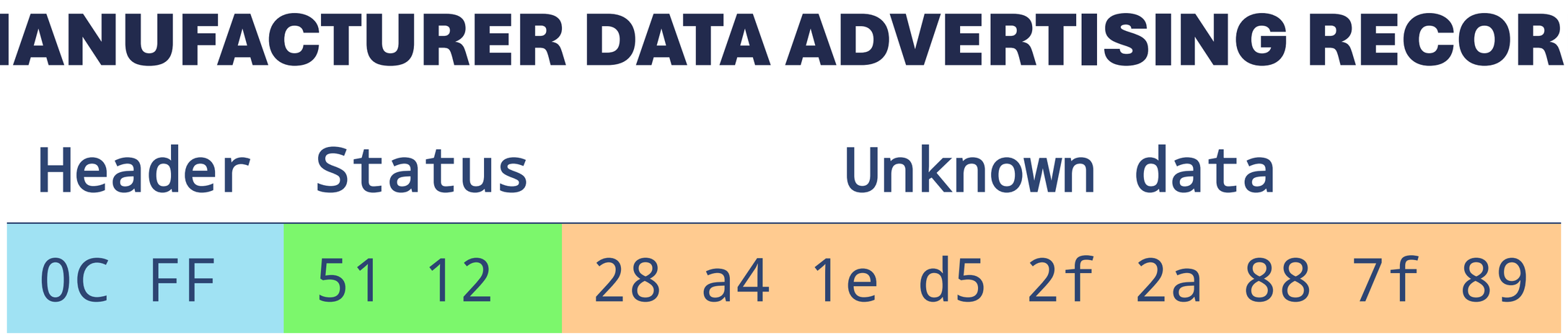
- Identifying a unique ID in its advertising records
- Identifying a unique ID in its exposed characteristics

THE BOSE NC-700 CASE

NC-700 CHARACTERISTICS

- Bluetooth Classic and Bluetooth LE **at the same time**
- Smartphone connects over **Bluetooth Classic for audio**
- Can **control** the headset behavior **over Bluetooth LE**

MANUFACTURER DATA ADVERTISING RECORD



CUSTOM RESOLVABLE PRIVATE ADDRESS

- Bose created its **own Resolvable Private Address** mechanism
- Hardware still **uses Bluetooth's NRPA feature** as well
- This weird mix **makes it weaker!**

Time elapsed	BD Addr.	Unique ID
+ 0s	47:e4:8d:3c:db:a4	a823f268c430
+ 234s	c8:7b:23:58:53:15	a823f268c430
+ 296s	c8:7b:23:58:53:15	28a41ed52f2a887f89

TRACKING CAN BE TRIVIAL

```
[-085 dBm] [PUB] e4:a6:34:4f:f6:df name:"BRC1H 4F:F6:DF"
[-084 dBm] [PUB] e4:a6:34:50:74:47 name:"BRC1H 50:74:47"
[-088 dBm] [PUB] e4:a6:34:50:9d:b3 name:"BRC1H 50:9D:B3"
[-088 dBm] [PUB] e4:a6:34:4f:fc:56 name:"BRC1H 4F:FC:56"
[-091 dBm] [PUB] a4:c1:38:01:5e:ef name:"BLT_015EEF"
[-090 dBm] [RND] 07:72:0d:e4:7c:3a name:"LE-Étoile noire"
```

FINGERPRINTING DEVICES

PASSIVE FINGERPRINTING

- Advertising data (**ADV_IND** + **SCAN_RSP**)
- Advertising interval

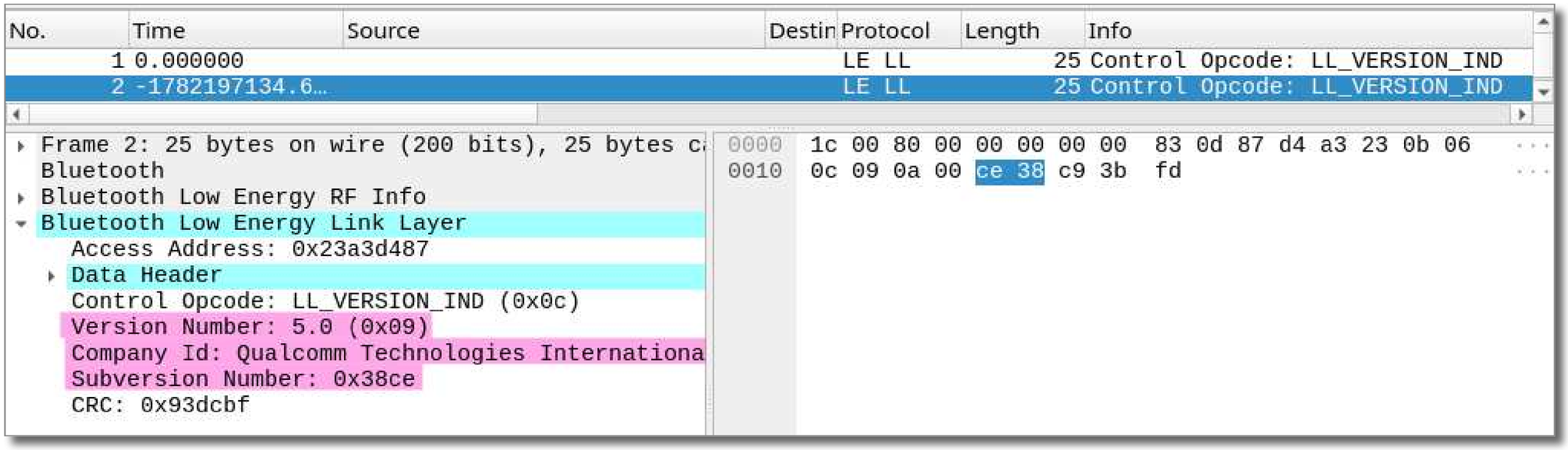
ACTIVE FINGERPRINTING

2.4.2.13 LL_VERSION_IND

The format of the CtrData field is shown in Figure 2.35.

CtrData		
Version (1 octet)	Company_Identifier (2 octets)	Subversion (2 octets)

BOSE NC-700 VERSION INFO



BLUE2THPRINTING!

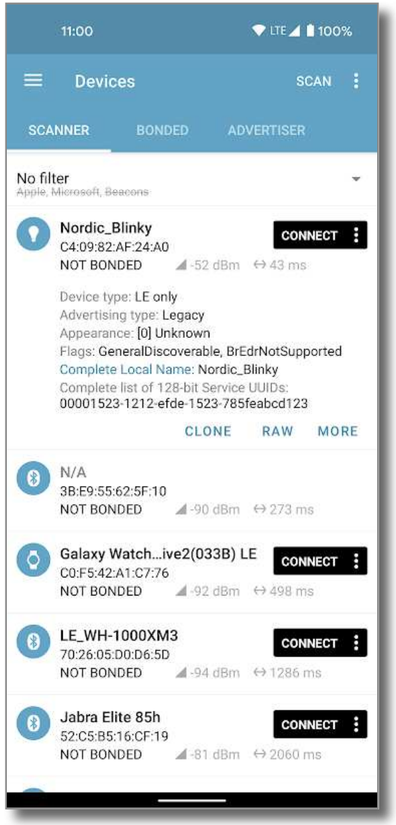
- **Xeno Kovah** (Dark Mentor LLC) maintains a collaborative fingerprint database of BLE devices (https://github.com/darkmentorllc/CLUES_Schema)

- **More than 500 UUIDs** identified and verified by real humans!

HACKING HACKERS

YOU WOULDN'T
EXPOSE YOUR PHONE

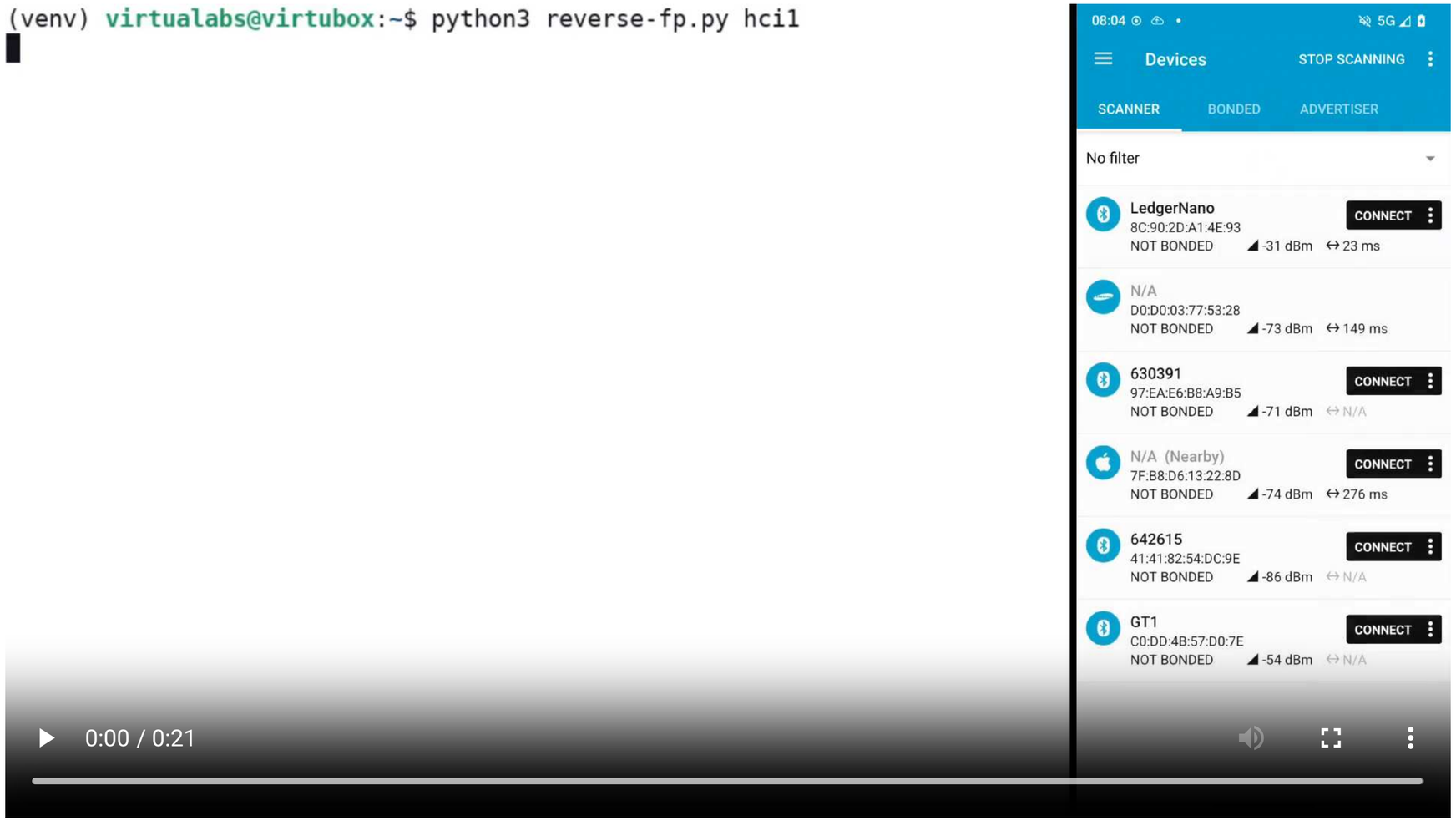
WHAT COULD POSSIBLY GO WRONG?



PERIPHERAL USING A GATT CLIENT

- **Totally legit**, defined in Bluetooth Specification 😏
- Yes, a **Peripheral can send requests** to a Central
- It's some kind of **reverse connection** 😋

REVERSE FINGERPRINTING



HACKERS WHINED DURING HARDWEAR.IO'S HWCTF

We saw the 'BleCallsHome' Bluetooth device but there is no characteristic marked as writeable... What are we supposed to do?

ALL YOUR BLUETOOTH LE DEVICES BELONG TO US?

- Finding known vulnerabilities **based on stack fingerprint**
- **Automated exploitation** & compromise
- **Steal** data, **hack** everything, cause **total chaos**!

(Well, it does not exist … yet)

ABUSING BLUETOOTH LE STACKS

ATTRIBUTES DATABASE (GATT)

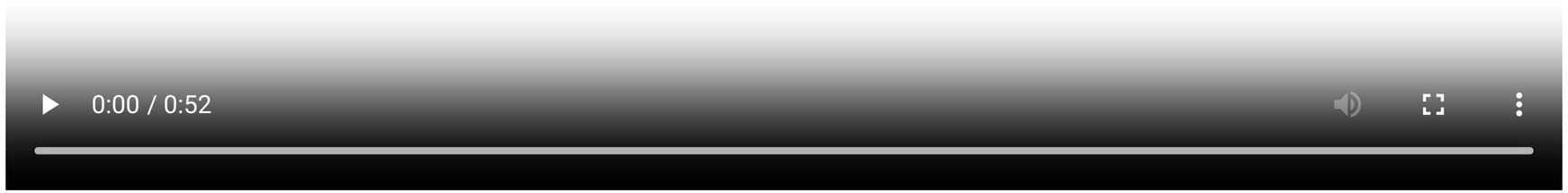
Handle	Type UUID	Value (Hex.)	Properties
1	0x2800	00 18	Read
2	0x2803	02 03 00 00 2A	Read
3	0x2A00	00 00 00 00 00 00	Read/Write

ATTRIBUTE PROPERTIES

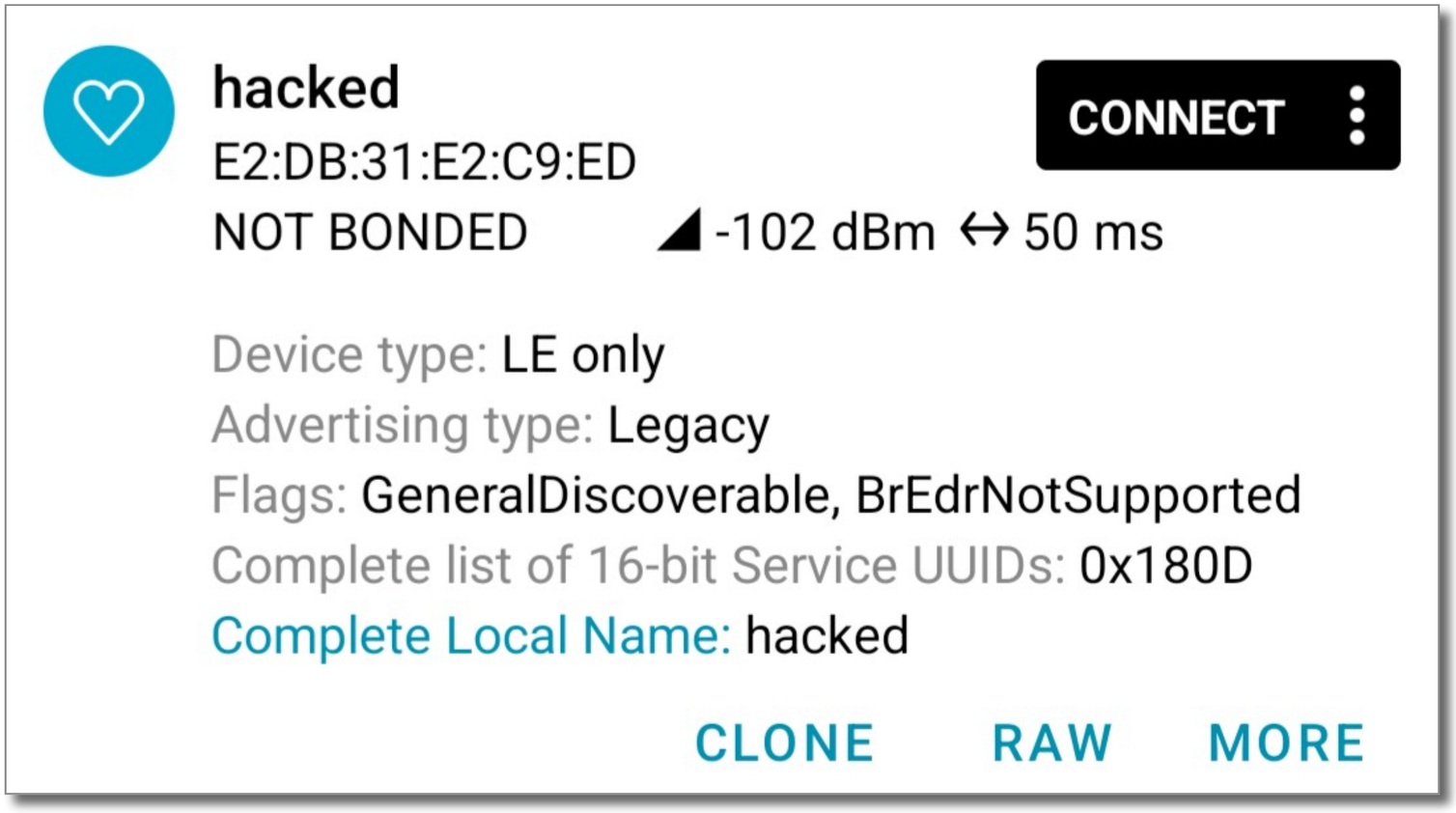
- **ATT-level**: defines access rights for the attribute value
- **GATT-level**: defines access rights for a characteristic's value

CHECKING PERMISSIONS IS HARD

```
(venv) virtualabs@virtubox:~$
```



WIRELESS DEFACING

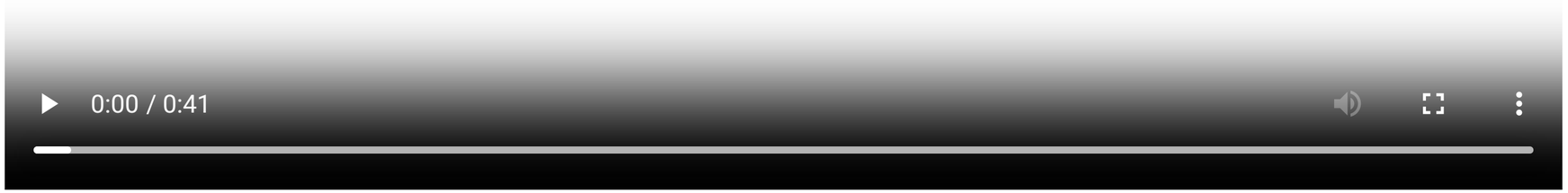


DID I TELL YOU CHECKING PERMISSIONS WAS HARD?

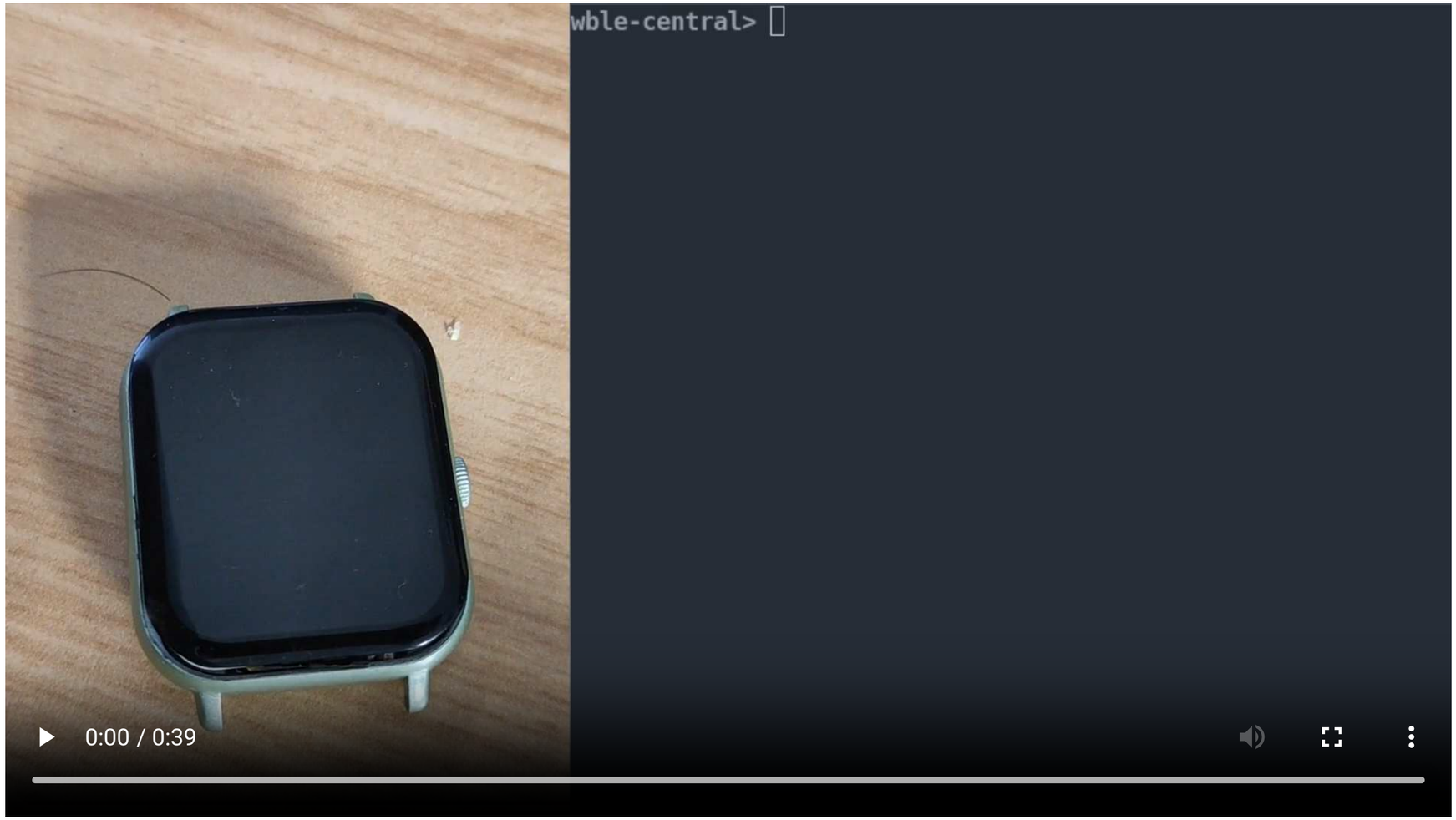
```
wble-central>
```

LIKE, REALLY HARD... 😖

```
wble-central>
```



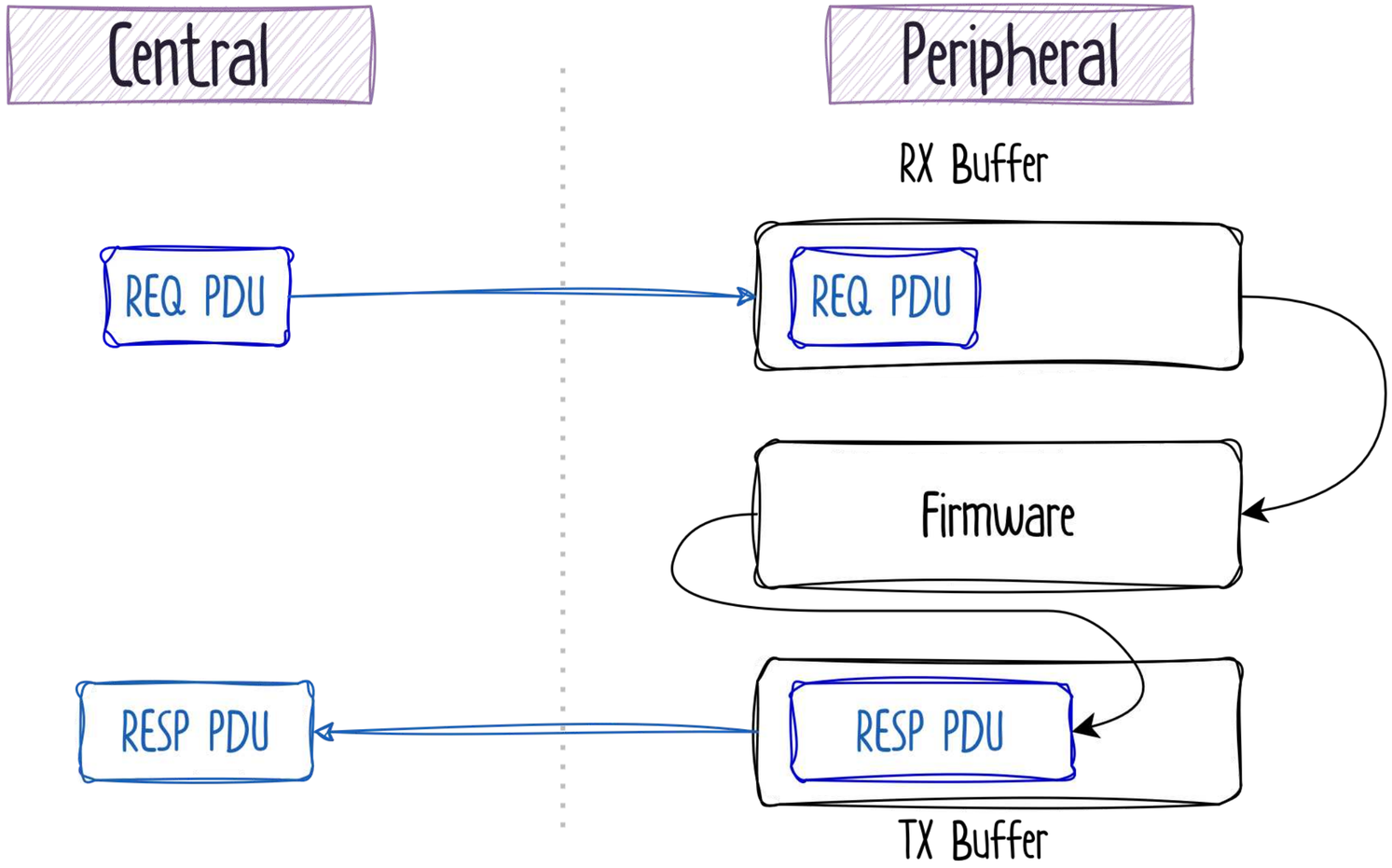
BAD HANDLING OF SOME PDUS



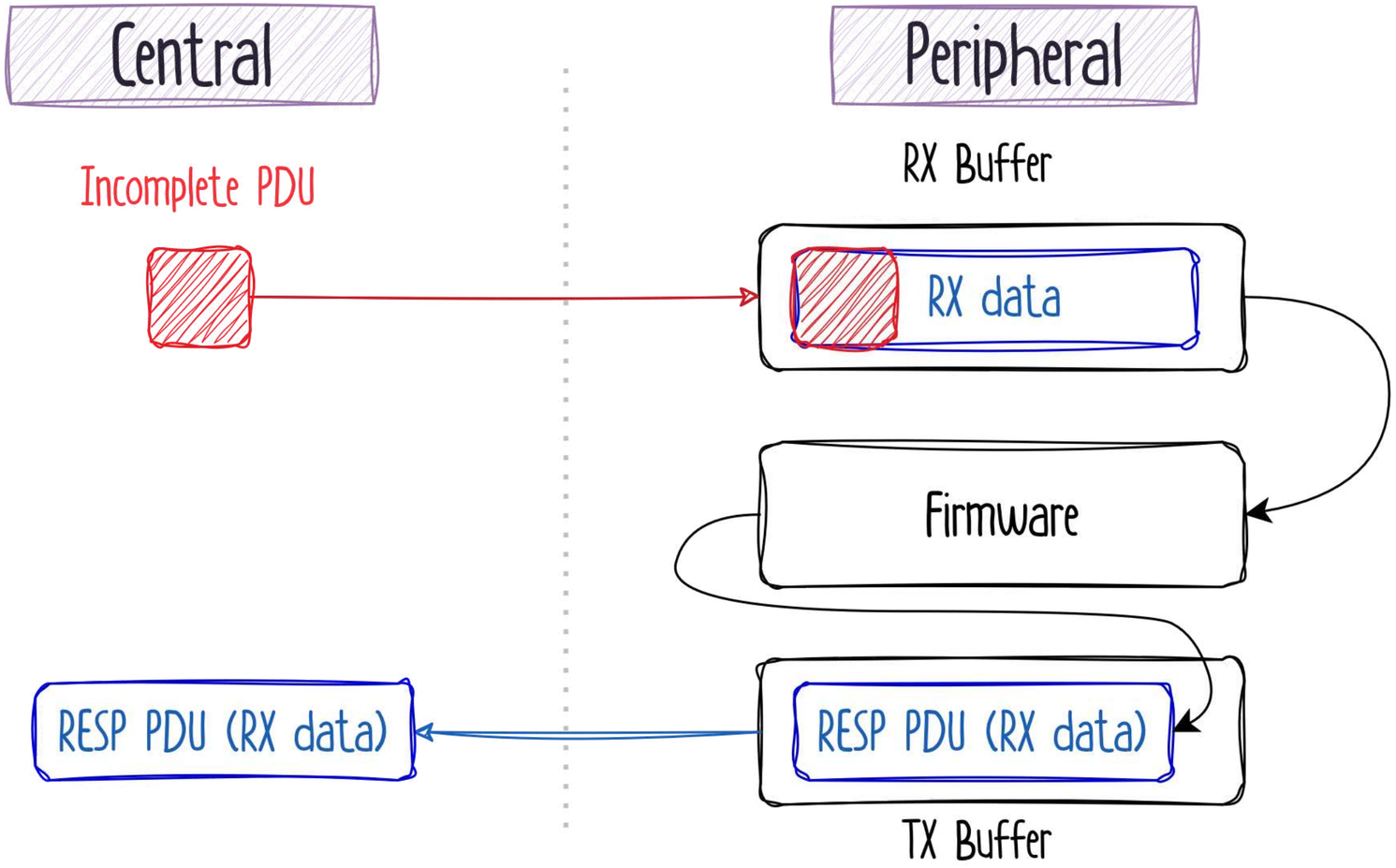
STEALING DATA FROM SYSTEM-ON-CHIPS

OR HOW I THOUGHT I FOUND SOMETHING HUGE...

BLUETOOTH PDU RX PROCESSING



MESSING WITH HEADER TO LEAK RESIDUAL DATA

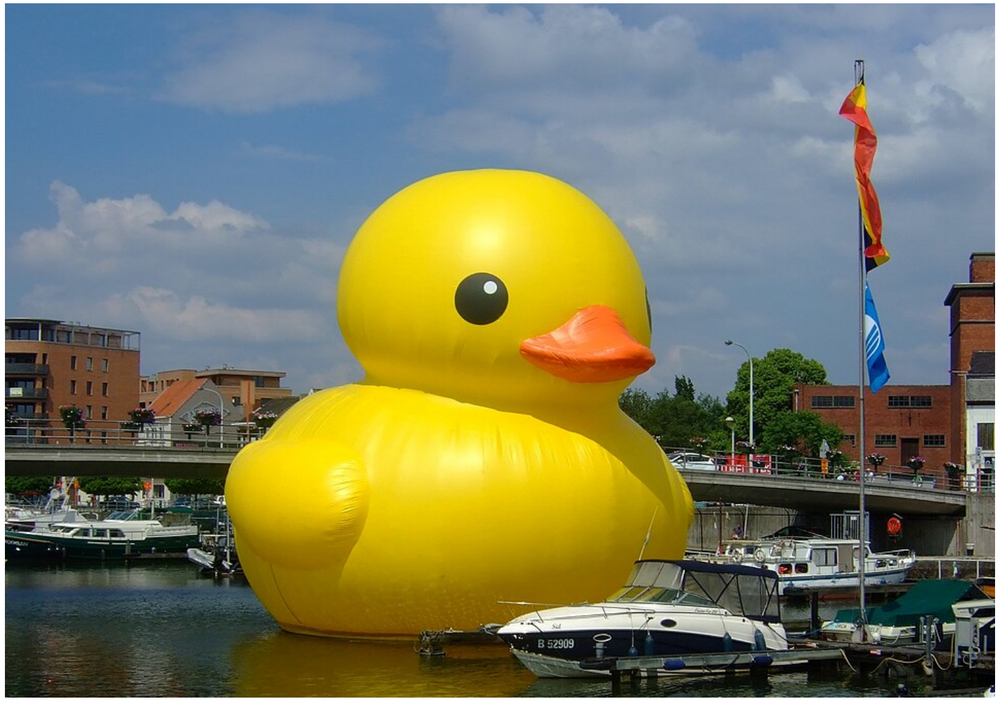


THE PLAN

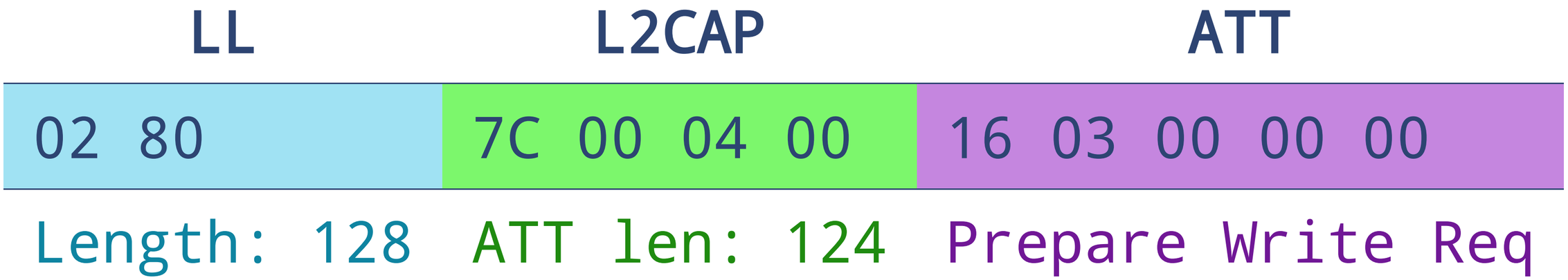
- Send a **PrepareWrite Request** with bad length field
- Wait for device to answer with a **PrepareWrite Response**
- Extract **residual data** from response and profit!

EXPLOITING THIS FLAW

- Wrote a proof-of-concept and tested against multiple devices
- I was able to retrieve at most 48 bytes of data, but no more…
- Looked good enough to include this in a talk 👌



I LEAKED MY OWN DATA 🤦



System-on-chips use **LL's length field**

to figure out the payload size...

I'M A FRAUD.

TIME TO THINK

- Radio cares about **LL length**, but not **L2CAP length**
- Maybe there is **another way to leak a SoC's RX buffer**?
- Lucky me, I have a lot of BLE devices for my research

AND I FOUND THIS

	LL	L2CAP	ATT
TX:	02 05	01 00 04 00	0A
RX:	02 06	02 00 04 00	0B 64

Device answers to a **ATT_READ_REQ that has no handle!**

OMG, I HAVE A NEW PLAN!

- We can **leak** data by **abusing weak ATT length checks** 🥳
- **ATT** provides **all the primitives** we need!

LEAKING THE FIRST 2 BYTES

- We send an **ATT_READ_REQ** with no handle
- If the **next 2 bytes match a handle**, attribute's value is returned
- **If not**, an ATT_ERROR_RSP containing **the handle value is returned**!

LEAKING 2 MORE BYTES

3.4.3.1 ATT_FIND_INFORMATION_REQ

The ATT_FIND_INFORMATION_REQ PDU is used to obtain the mapping of attribute handles with their associated types. This allows a client to discover the list of attributes and their types on a server.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x04 = ATT_FIND_INFORMA-TION_REQ
Starting Handle	2	First requested handle number
Ending Handle	2	Last requested handle number

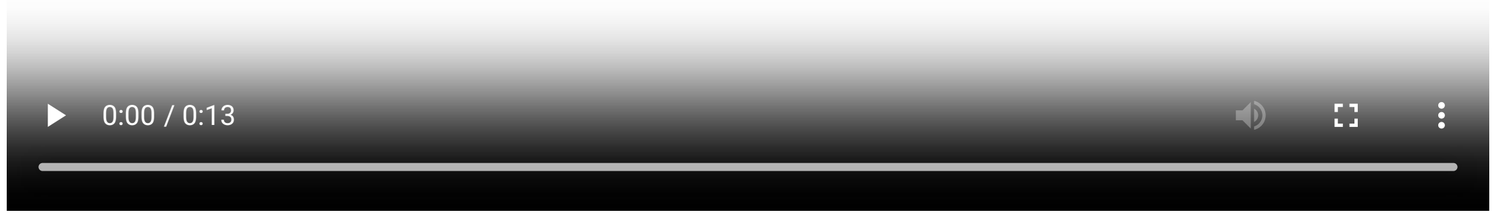
Table 3.7: Format of ATT_FIND_INFORMATION_REQ PDU

THAT'S A 4-BYTE MEMORY LEAK!

- Exploits an **invalid ATT length check**
- Combines two ATT requests to **leak 4 bytes of residual data**

PROOF OF CONCEPT

```
(venv) virtualabs@virtubox:~$
```



IMPACT

- Device can leak at most 4 bytes of data (not really impressive)
- Only a few stacks are vulnerable (not the most popular ones)
- Impact is indeed limited, but stealing bytes from a remote device's RX buffer is a thing

Vendor	Subversion	Vulnerable?
Bluetrum	0x0003	✅
JieLi	-	✅
Realtek	0x8773	❌
Qualcomm	0x38ce	❌
Cypress	0x2310	❌

CONCLUSION

BLE IS NOT EASY

- We often only have a look at the application layer, but the stack could be pretty interesting too

- Some BLE stacks **fail to correctly parse PDUs**
  (we're in 2026, guys...)

- BLE **Peripheral role can have a GATT client** and use it!

HOW TO AVOID BEING HACKED?

- **Don't connect to random devices** using your smartphone!
- Don't use devices based on **some Bluetrum and JieLi stacks** (for now)
- **Keep** firmwares and OSes **up-to-date** (as usual)

FAILURE IS ALWAYS AN OPTION

- I felt really dumb when I realized my mistake
- It made me dig into the ATT protocol to find a similar issue
- Sorry guys, the awesome vulnerability you were waiting for is in another talk 😔

THANK YOU, Q&A TIME!

REFERENCES

- BLERP: BLE Re-Pairing Attacks and Defenses, Tommaso Sacchetti, Daniele Antonioli, NDSS Symposium, 2026
- Blue2thprinting (blue-[tooth)-printing]: Answering the Question of 'WTF am I Even Looking at?!', Xeno Kovah, ShmooCon 2024
- Open Wounds: The last 5 years have left Bluetooth to bleed, Xeno Kovah, Hack.lu 2023
- InjectaBLE: Injection de Traffic Malveillant dans une Connexion Bluetooth Low Energy, Romain Cayre, Florent Galtier, SSTIC 2021
- BLESA: Spoofing Attaacks against Reconnections in Bluetooth Low Energy, Jianliang Wu et al., WOOT 2020
- KNOB Attack, Daniele Antonioli, Nils Ole Tippenhauer, Kasper B. Rasmussen, Usenix 2019
- Defeating Bluetooth Low Energy 5 PRNG For Fun and Jamming, Damien Cauquil, DEF CON 27, 2019
- You'd Better Secure Your BLE Devices, Damien Cauquil, DEF CON 26, 2018
- Bluetooth: With Low Energy comes Low Security, Mike Ryan, WOOT 2013
- Bluetooth Core Specification version 6.3, Bluetooth SIG, 2026