A Practical Introduction to Bluetooth Low Energy security
without any special hardware

Slawomir Jasek

Trainer

SMARTLOCKPICKING.COM

HITB+CyberWeek UAE

Virtual Edition, 15-19 November 2020

Sławomir Jasek

<suavomeer> <yaseck>

#NFC #HCE #KNX
#BLE #embedded #SDR
#RFID #IoT

SMARTLOCKPICKING.COM

HITBSecConf

HITB+CyberWeek
Abu Dhabi, UAE: 12-17 October 2019

DEEPSEC

HACKING FOR B33R
BRUCON
WWW.BRUCON.ORG

HACK IN PARIS
Cyber Security Conference

GATTack.io
OUTSMART THE THINGS

hardwear.io
Hardware Security Conference and Training

Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

Bluetooth®
Classic

Focus on speed

Bluetooth™
4.0
Low Energy

Focus on preserving energy

Bluetooth 5?

- 2x speed (new modulation)
- 4x range (at lower speeds)
- Other extensions
- Not really yet rolled-out in devices (even claiming so)
- For us in short: **5.0 ≈ 4.0**





Parrot

Google play
twitter
YouTube
amazon
i-SMART
NETFLIX
skype
facebook
WhatsApp


VISA
1 2 3
4 5 6
7 8 9
0
sumup

MON 04-27
20576
20:08



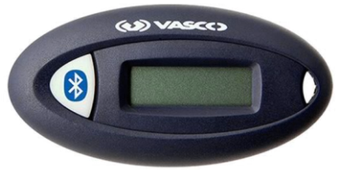




Jabra
Jabra



OXIMETER



VASCO

Security: vendor's claim

Military Grade Security

How Secure is [redacted]?

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

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

Meet the reliable *Bluetooth®* wireless technology enabled lock. With military-grade PKI encryption and inside-outside technology protecting the digital experience, backed by 70 years of door lock security,

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

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

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

But how secure are they, really?

ars TECHNICA BIZ & IT TECH SCIENCE PC

OPEN SESAME —

Top-selling handgun safe can remotely opened in seconds– needed

There's no online update mechanism for defective electronic safe

engadget

Reviews Gear Gaming Entertainment Products Tomorrow Audio Video Deals

—

Researcher finds huge security flaws in Bluetooth locks

You might want to rethink adding technology to your front door.

NEWS

Home | US Election | Coronavirus | Video | World | UK | Busine

Tech

Smart lock can be hacked 'in seconds'

75 Percent of Bluetooth Smart Locks Can Be Hacked

End users?

Security [edit]

Due to the inherent complexity of digital and wireless technologies, it can be difficult for the end user to confirm or refute the security claims of various product offerings on the market.[4]

https://en.wikipedia.org/wiki/Smart_lock#Security

Security professionals?

I don’t get this wireless sourcery

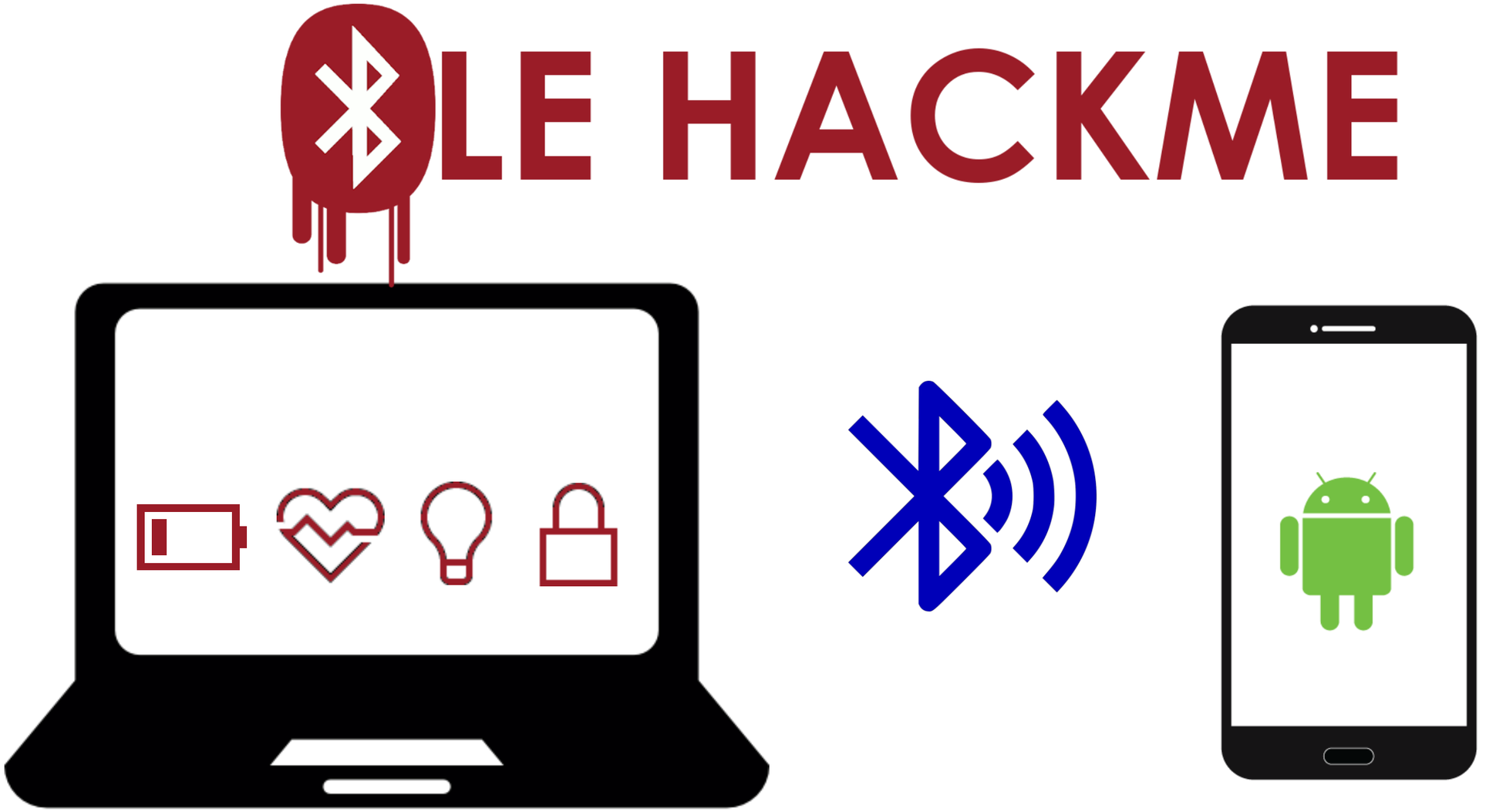
Must be some expensive hardware needed?

I don’t have time to train another set of complex tools!

Where do I get some vulnerable devices to practice?

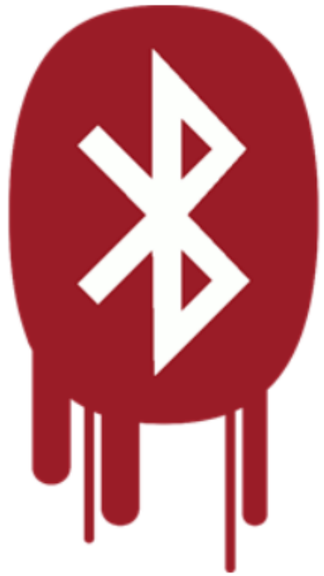
- Step-by-step **hands-on** introduction to BLE technology
- **Practical** challenges with increasing complexity level
- Devices simulated on **standard laptop**'s Bluetooth adapter, visible via radio just **like real** ones
- Standard Android **phone** as surprisingly effective "hacking tool"
- New skills easily **applicable to real devices.**
- Learn by having **fun**!

BLE HACKME



BLE HackMe: installation

https://www.microsoft.com/store/apps/9N7PNVS9J1B7



BLE HackMe

smartlockpicking.com • Education > Instructional tools

Bluetooth Low Energy HackMe - educational application which simulates various BLE devices to interact with.
In a series of tasks to solve you will get familiar with BLE advertisements, beacons, connections, take control over BLE smart bulb, reverse-engineer the communication protocol, brute force passwords, and hack real smart lock.

More

EVERYONE

Free

Get ...

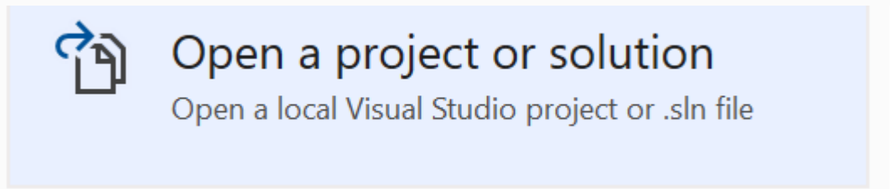
⚠ See System Requirements

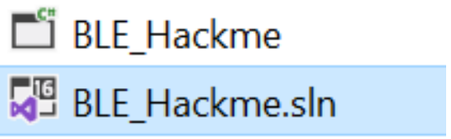
BLE HackMe – source code

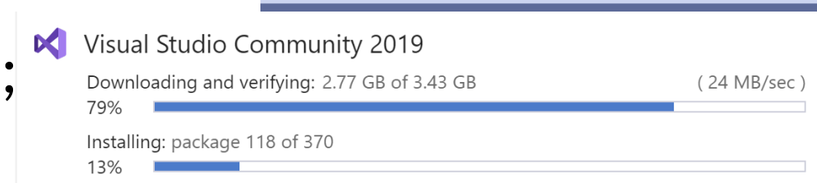
https://github.com/smartlockpicking/BLE_HackMe

Free, MIT license

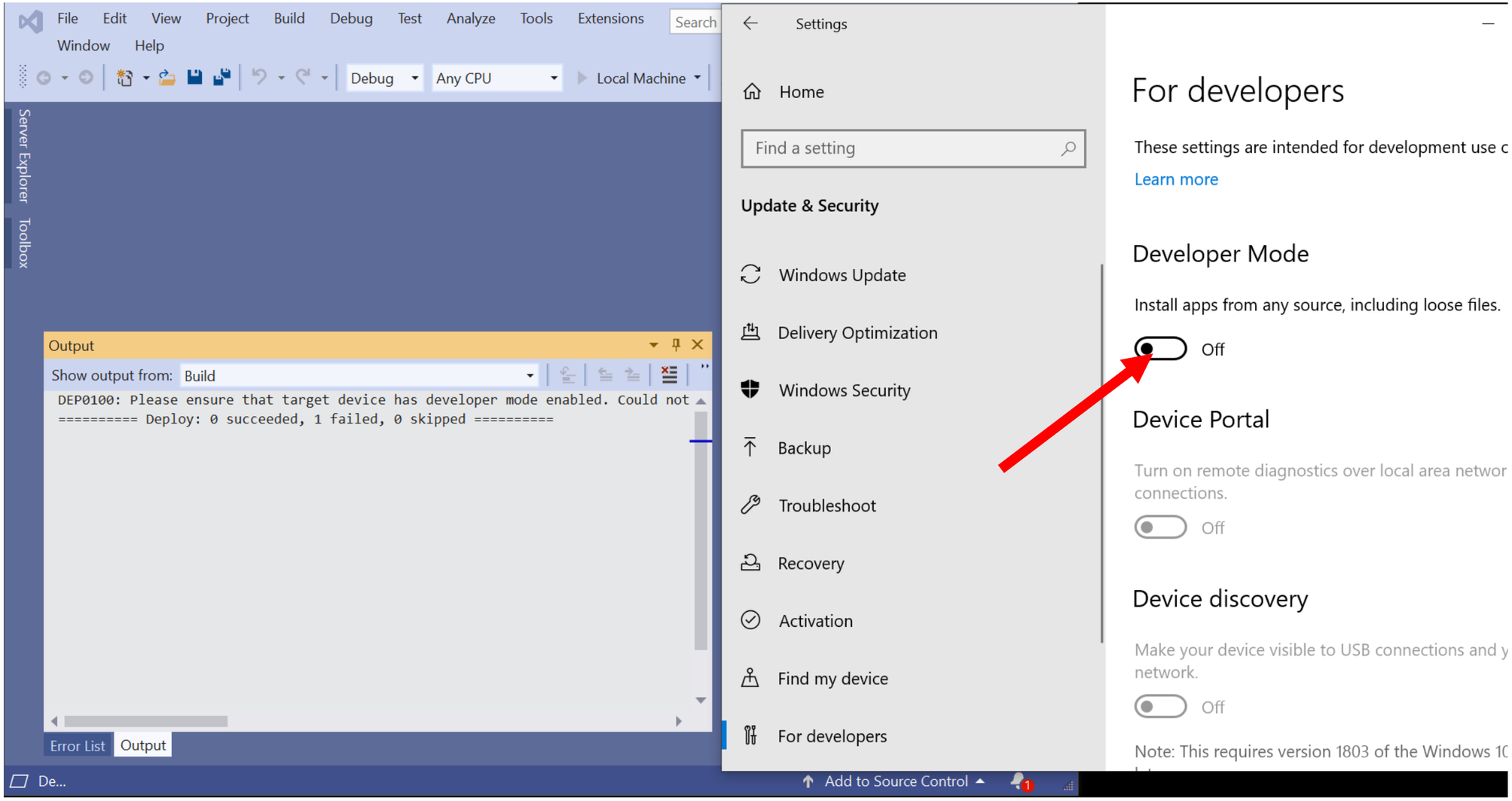
To compile: Visual Studio (free Community edition); UWP development

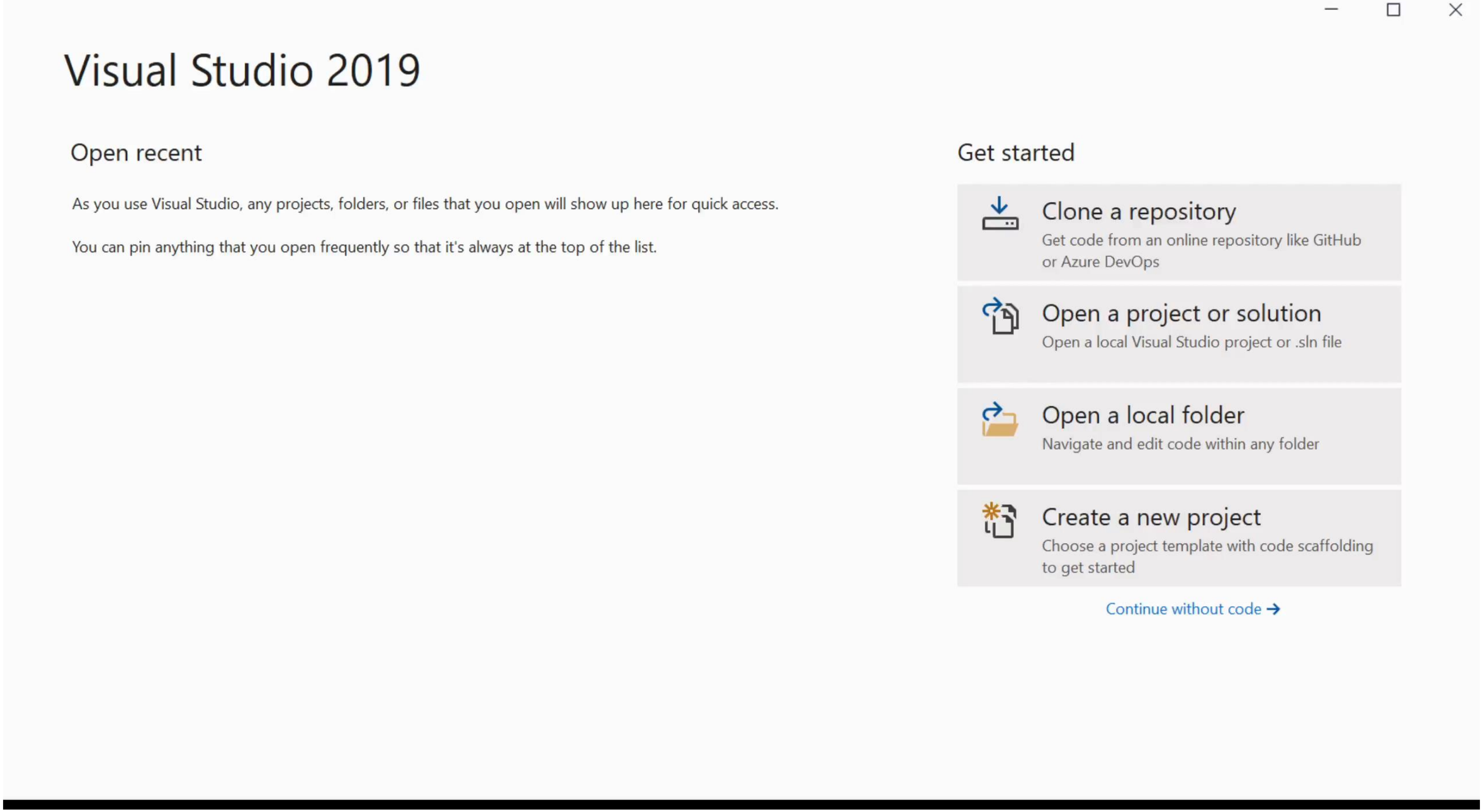






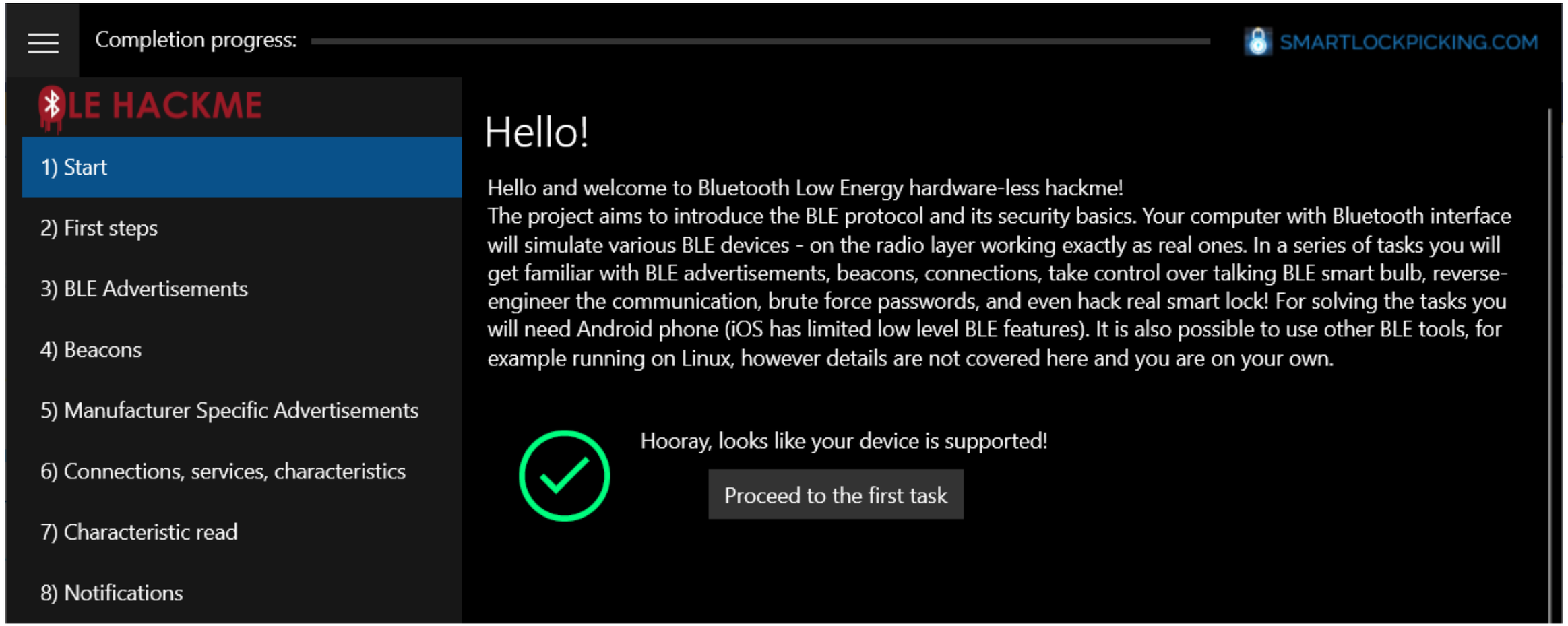
VS will ask to turn developer mode on



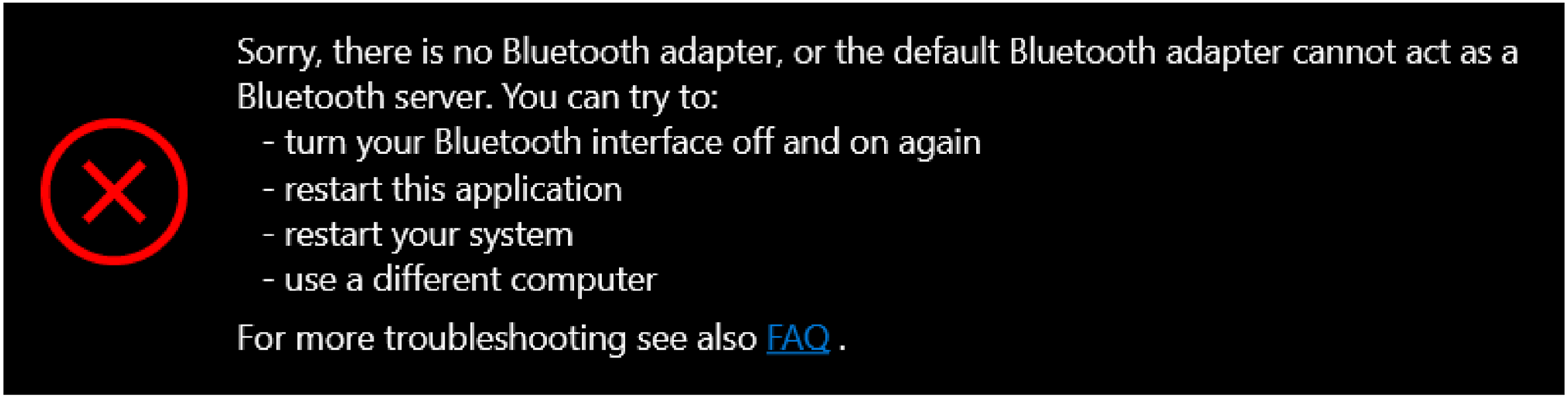

Visual Studio 2019
Open recent
As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.
You can pin anything that you open frequently so that it's always at the top of the list.
Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

BLE HackMe – compatibility check

Expected result:



Unsupported Bluetooth interface?



It will not work:

- in VM (with some exceptions)
- with external Bluetooth dongles (at least CSR8510 based)
- on some older (> 5 years old) laptops

If restarting does not help, try using a different computer…

Will it work in VM?

- Most cases: **unfortunately no**
  - Not for sharing Bluetooth with host
  - Not for typical CSR dongle USB passthrough
- Confirmed working with laptop's **internal Bluetooth adapter connected via USB**, not PCI (for example Lenovo Thinkpad X1 Carbon 7) - thanks Gerhard Klostermeier (@iiiikarus)!
- If you figure out workaround, let me know!

Windows „N” edition (uncommon)

- Windows „N” (rare edition) comes without media pack, required for a few tasks (text to speech functionality)
- If your system has no Windows Media Player available, please install „Microsoft Media Feature Pack”

Disclaimer

My first-ever C# code
Expect crashes, bugs, exceptions…
Features like:
- saving progress status
- changing color mode

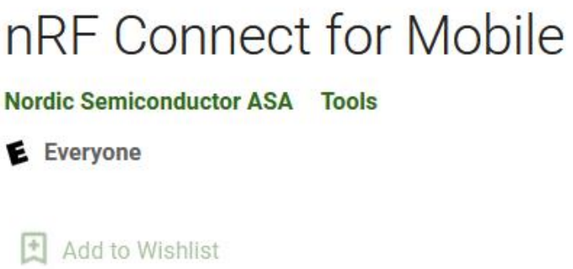
will come some next release ;)

https://pixabay.com/photos/cat-baby-kitten-sleep-hand-cat-2204590/

Feel free to file issues/PRs on Github:
https://github.com/smartlockpicking/BLE_HackMe/issues

Mobile app (our „hacking tool”)

„nRF Connect for Mobile”

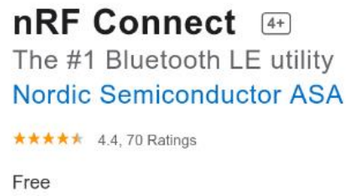


Android (recommended)

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

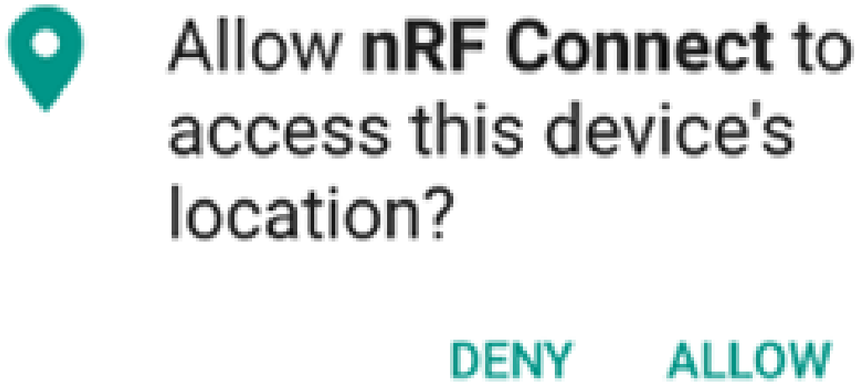
iOS – limited low-level BLE features,
you won’t be able to solve majority of tasks

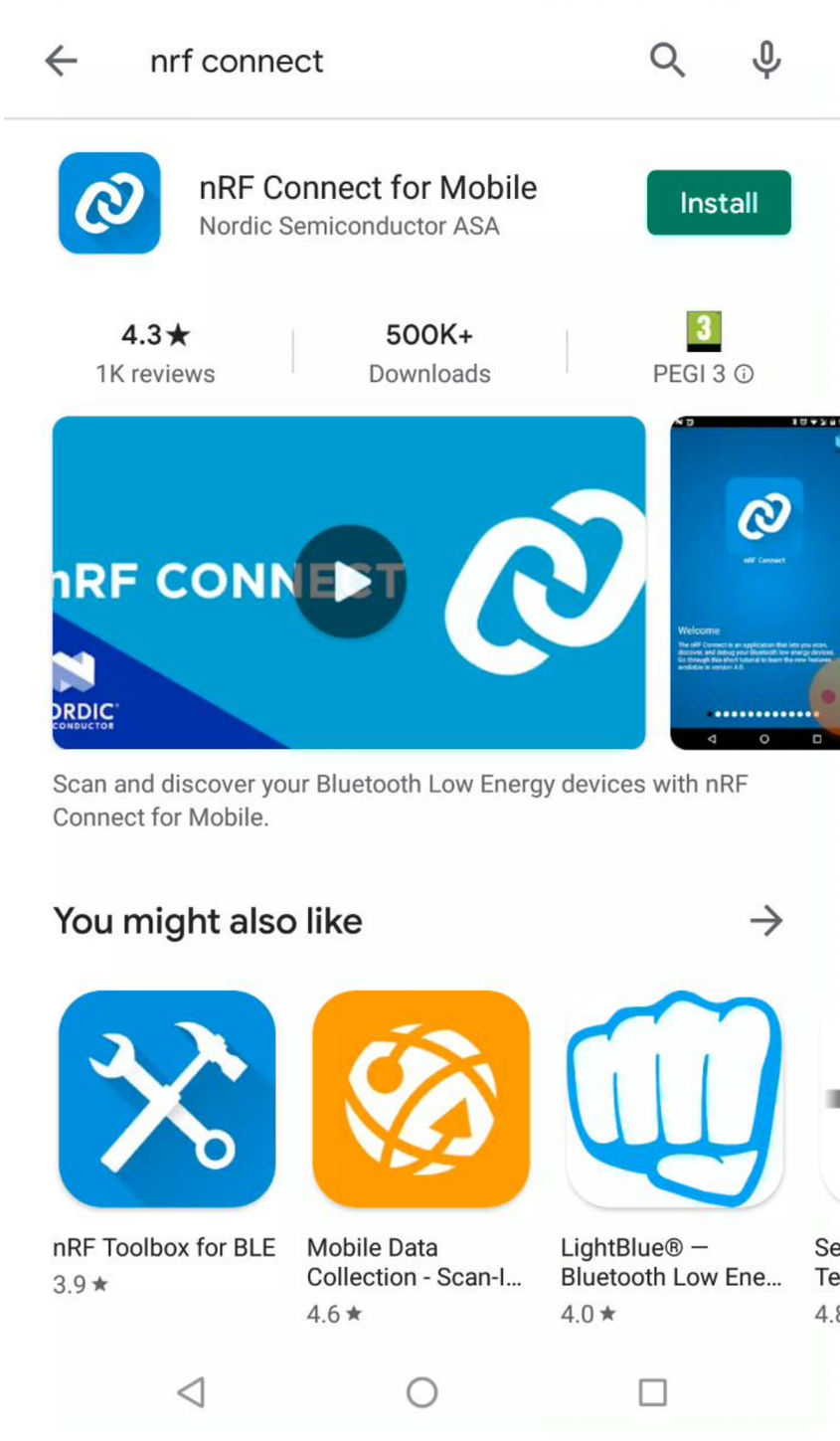
https://apps.apple.com/pl/app/nrf-connect/id1054362403



nRF Connect installation

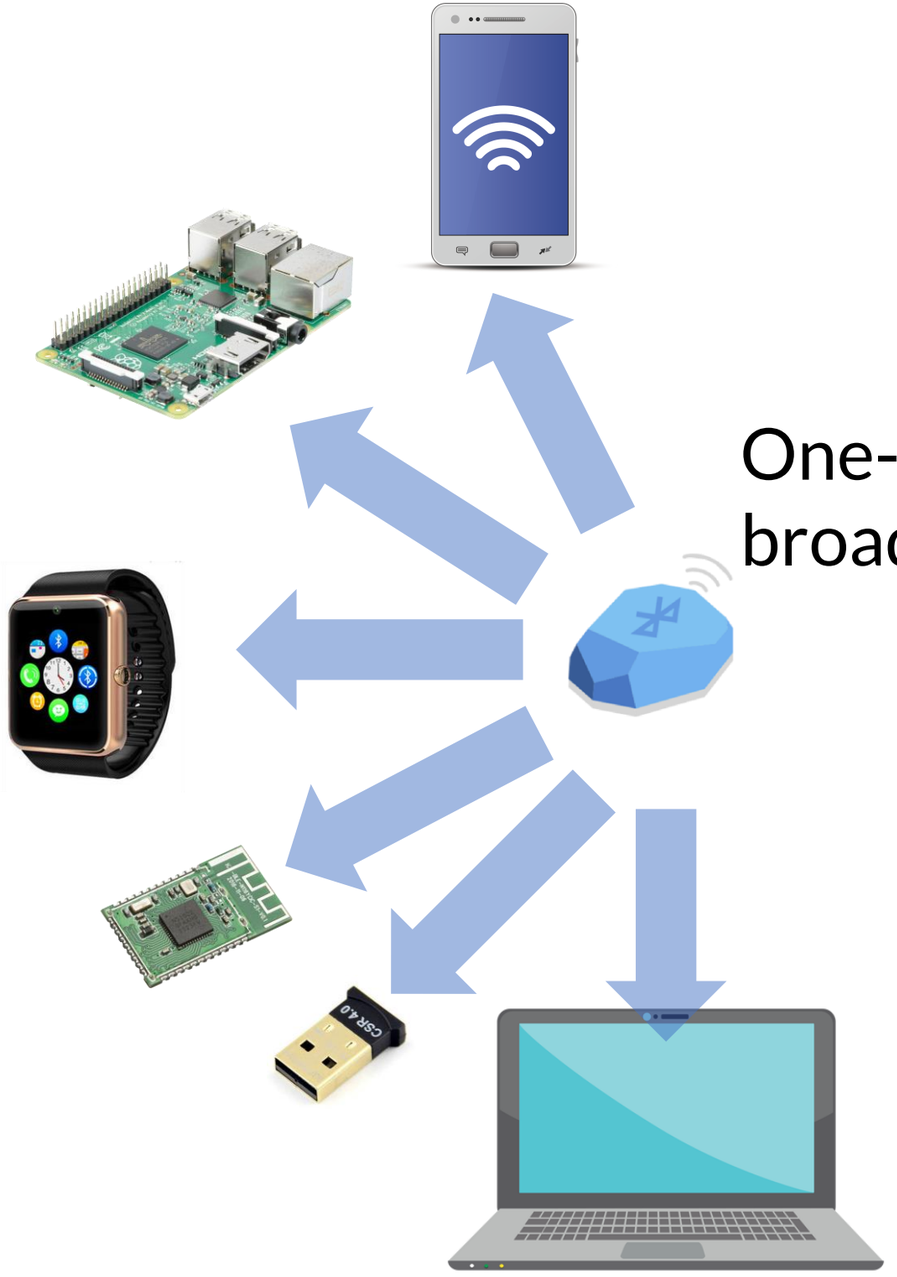
Android requires location permission from apps scanning Bluetooth





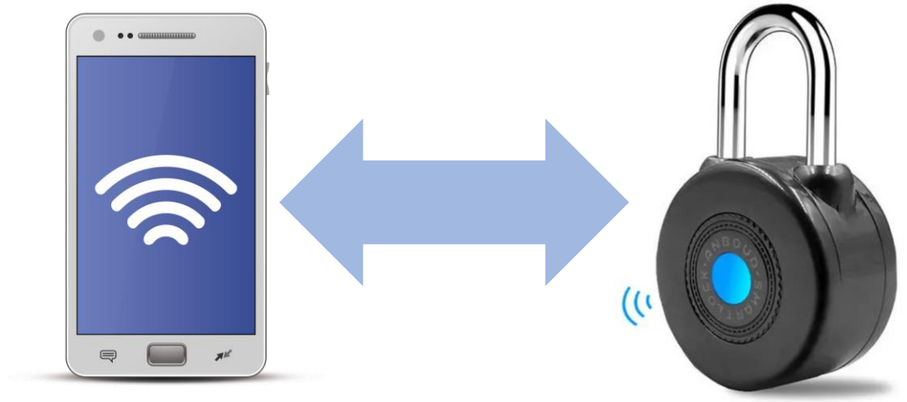
Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?



Bluetooth™ 4.0 Low Energy

One-way broadcast



Two-way communication (e.g. app controls smart lock)

Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

First steps

Theory introduction

You are undoubtely familiar with Bluetooth, and most likely use it every day - for example in wireless mouse, headset or car audio. Despite sharing common name, Bluetooth Low Energy is however a different technology. As the name implies - it aims to preserve energy, hence typical applications include rather occasional exchange of small data packets. Most common usage scenarios include:

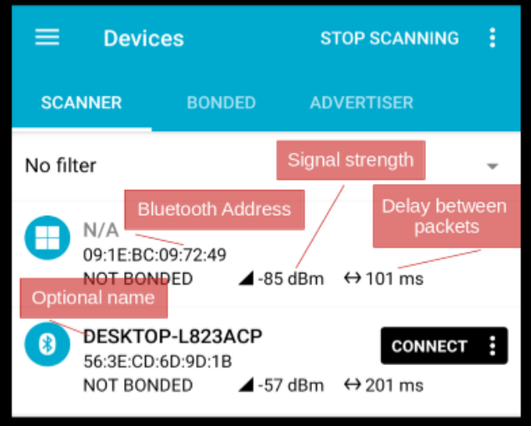
- a **Broadcaster** that transmits some one-way data ("**Advertisement**") to all nearby **Observers** (for example a "beacon" device broadcasting indoor location to nearby phones)
- BLE Client ("**Central**", for example mobile application) to Server ("**Peripheral**", for example smart lock) communication

We will start with the BLE broadcast advertisements.

Task

If everything went correctly, the HackMe application should now be broadcasting BLE packets. Let's see if it works! Probably the easiest way is to use your smartphone, and there are several free applications to do the job. The recommended one is nRF Connect, available for both Android and iOS, however iOS version lacks several important features required to solve some of the upcoming tasks.

For Bluetooth access, Android requires location permission from the application, so you will have to grant it during installation. Once started, the application will show nearby BLE devices. Beside optional device name, you will notice the device's adapter address, bonding (pairing) information, as well as signal strength (swiping to right will show its change in time) and frequency of the broadcasted packets (delay in ms). For connectable devices, there is also optional "CONNECT" button:

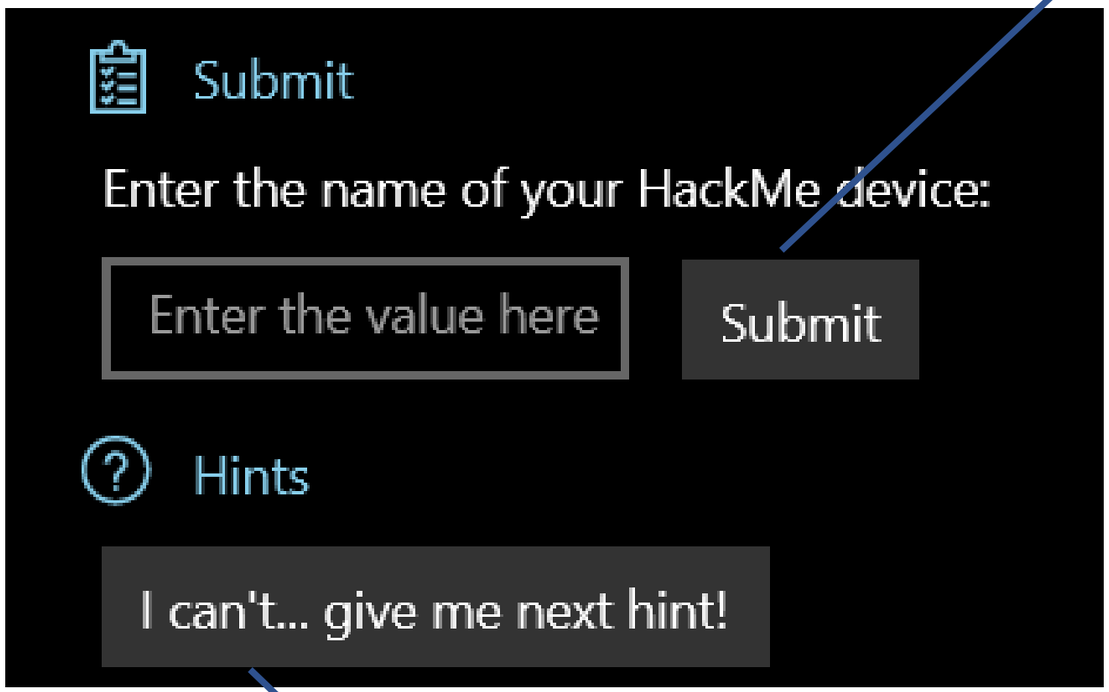


Short theory introduction

Task description

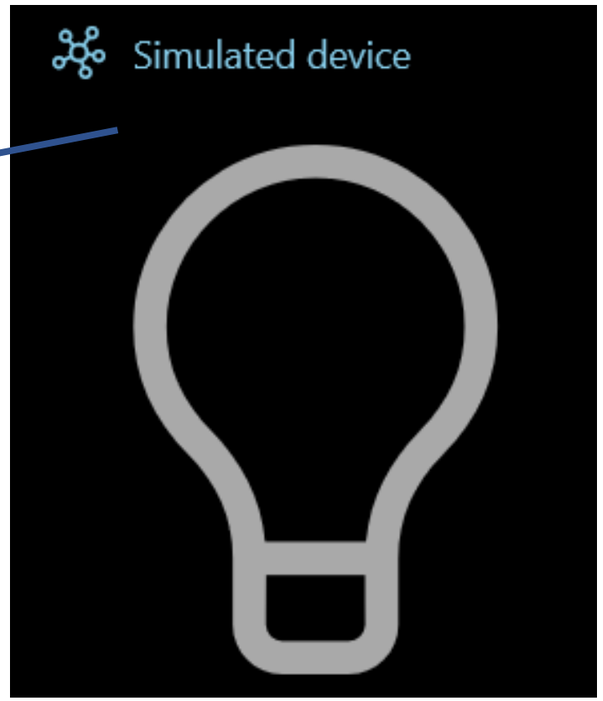
List of tasks

Submit solution



Submit task solution

Simulated device to hack (some next tasks)



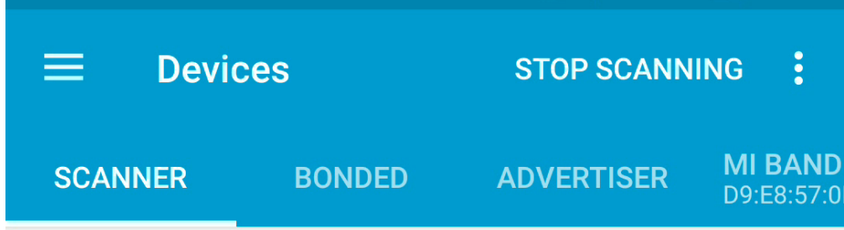
Hints (spoilers)

nRF Connect: scan

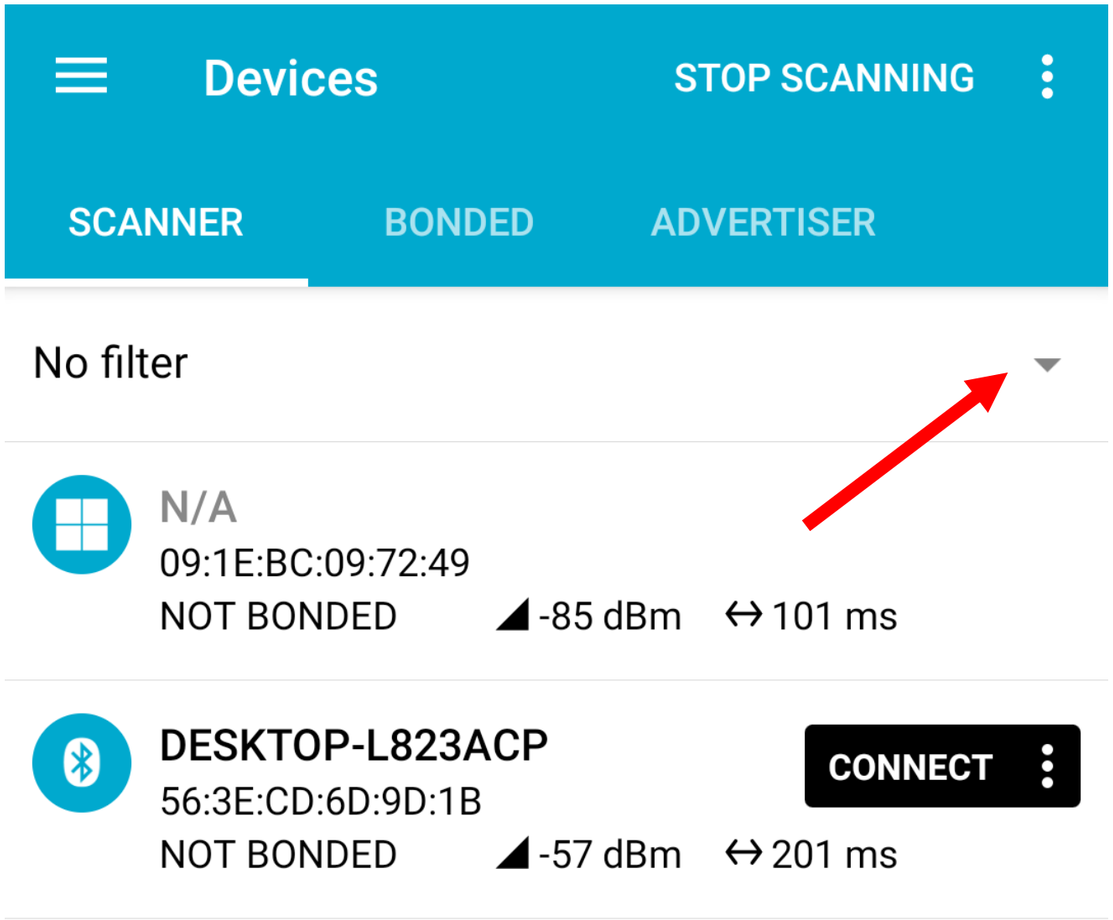
Scan starts automatically, stops after a while („SCAN” again if needed).

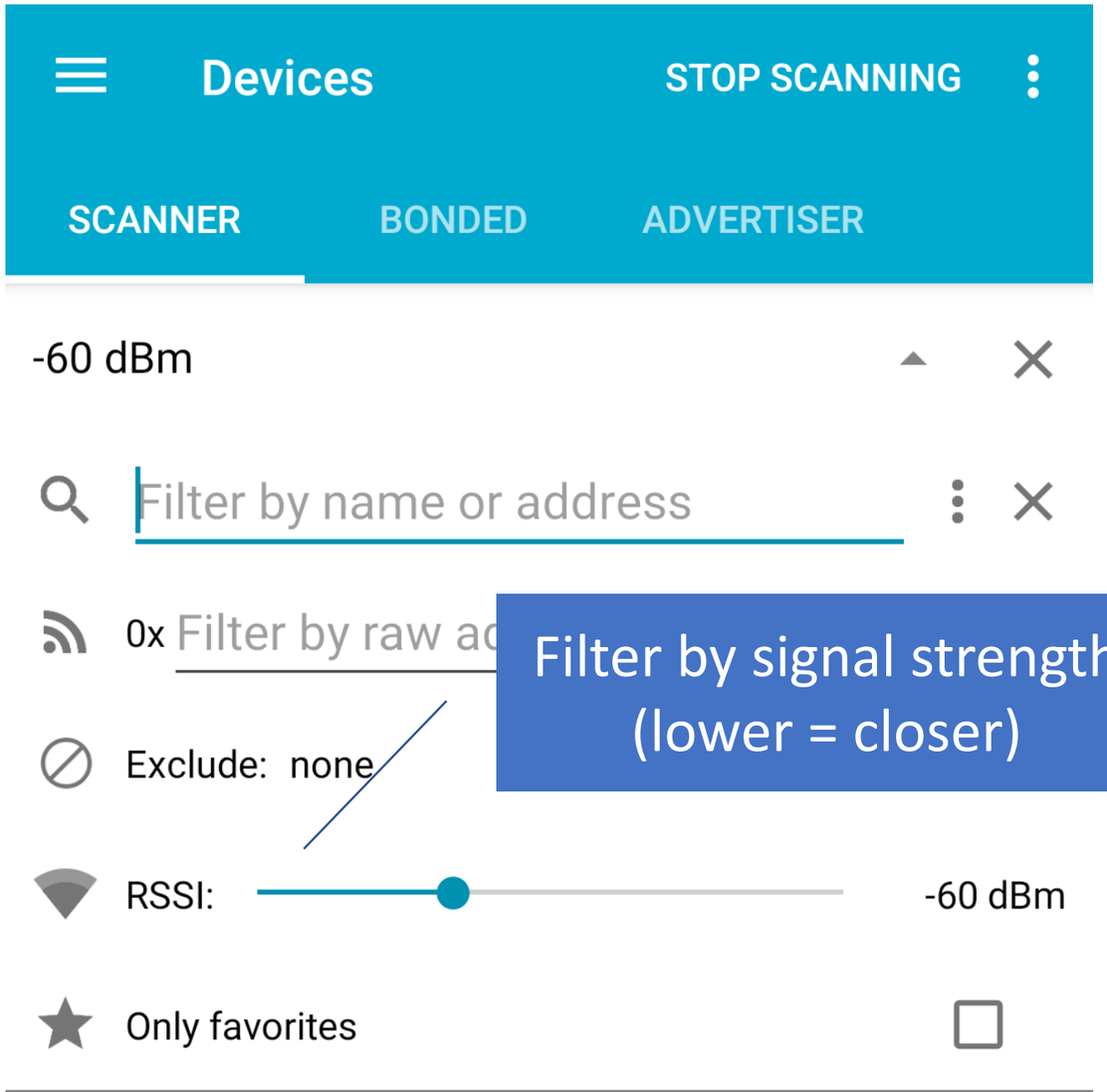
You may see lots of devices.

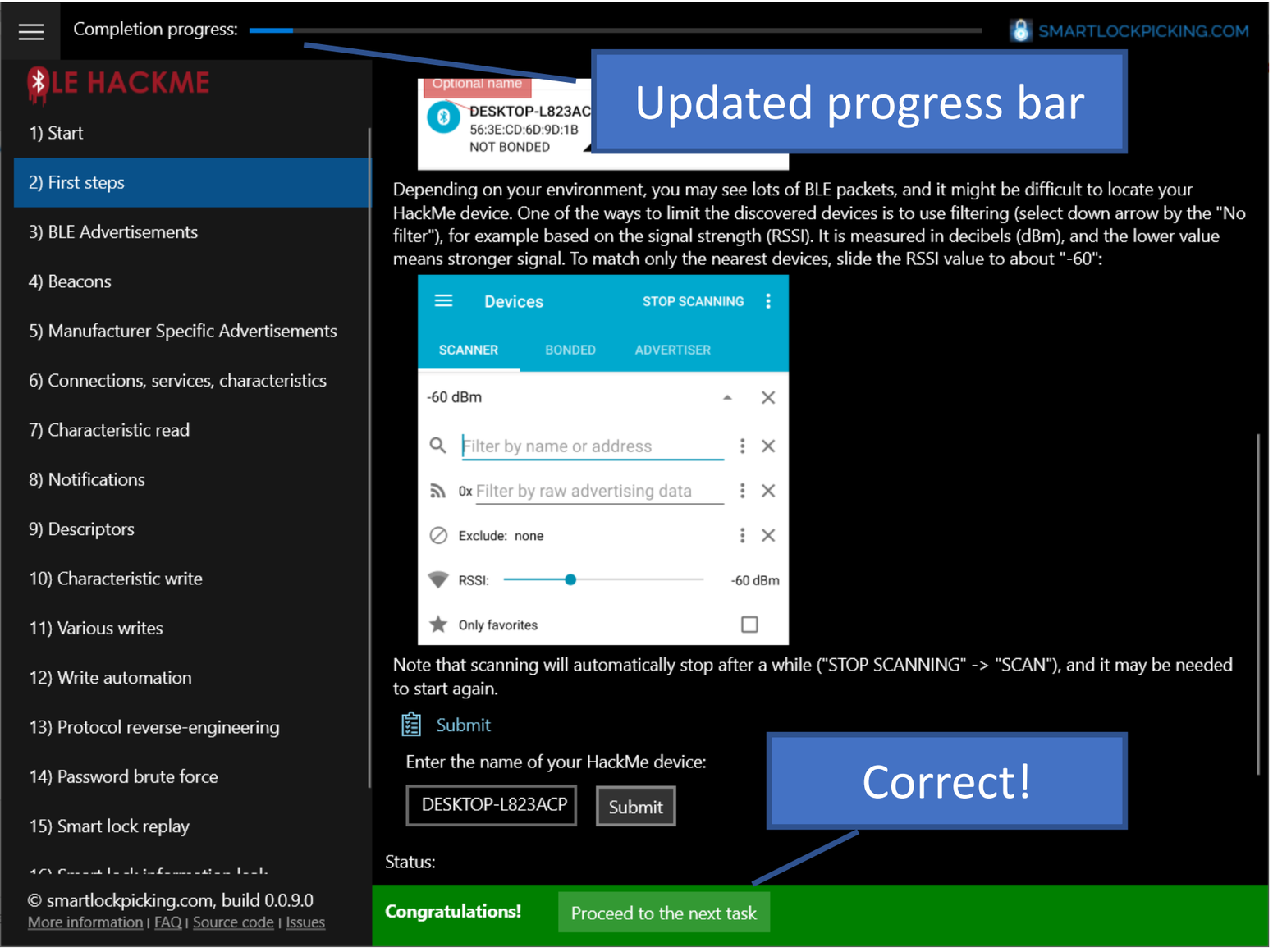
Where is our HackMe?

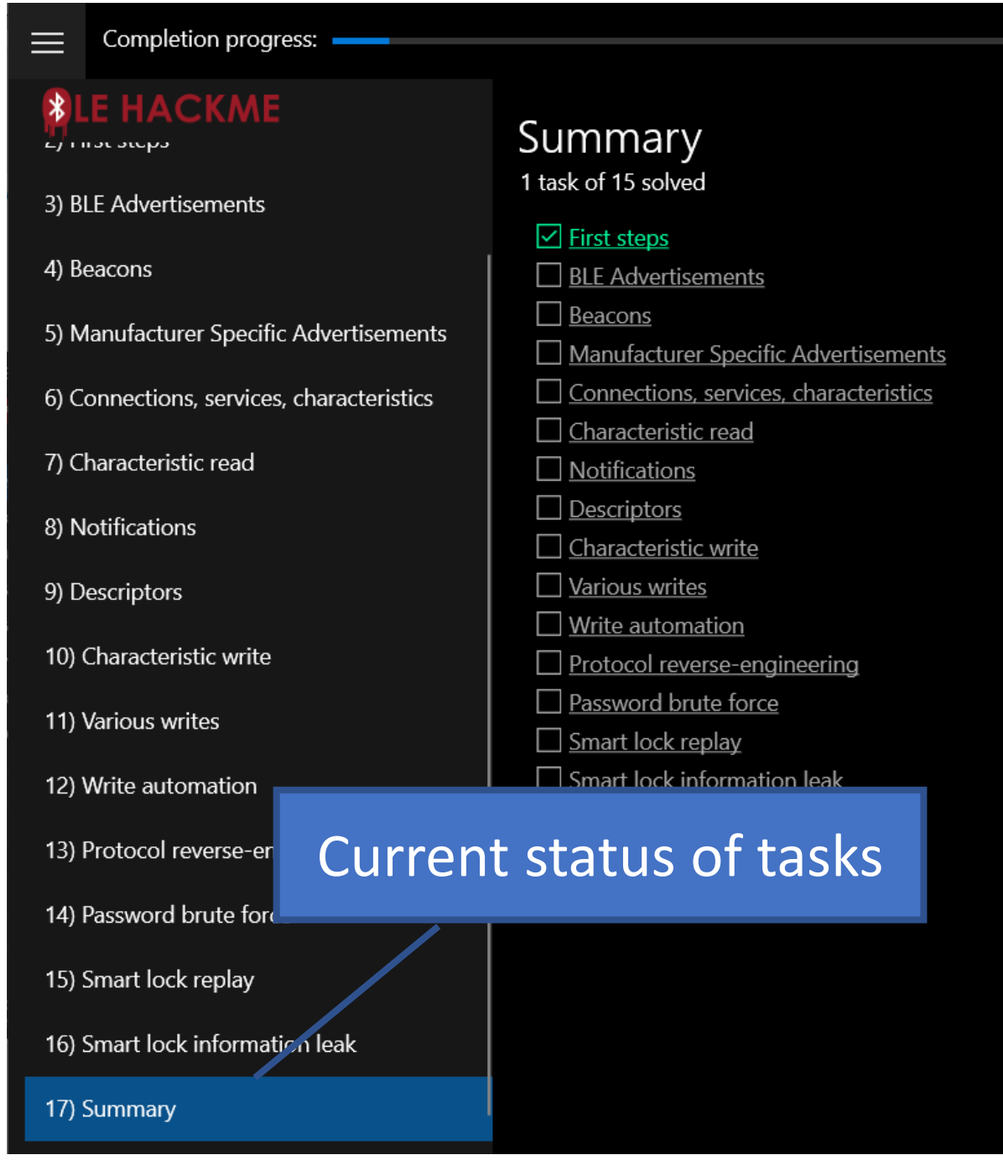


Too many devices? Filter to the rescue!






Completion progress:
SMARTLOCKPICKING.COM
BLE HACKME
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
Optional name
DESKTOP-L823AC
56:3E:CD:6D:9D:1B
NOT BONDED
Updated progress bar
Depending on your environment, you may see lots of BLE packets, and it might be difficult to locate your HackMe device. One of the ways to limit the discovered devices is to use filtering (select down arrow by the "No filter"), for example based on the signal strength (RSSI). It is measured in decibels (dBm), and the lower value means stronger signal. To match only the nearest devices, slide the RSSI value to about "-60":
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
-60 dBm
Filter by name or address
0x Filter by raw advertising data
Exclude: none
RSSI:
-60 dBm
Only favorites
Note that scanning will automatically stop after a while ("STOP SCANNING" -> "SCAN"), and it may be needed to start again.
Submit
Enter the name of your HackMe device:
DESKTOP-L823ACP
Submit
Correct!
Status:
Congratulations!
Proceed to the next task
© smartlockpicking.com, build 0.0.9.0
More information | FAQ | Source code | Issues


Completion progress:
BLE HACKME
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-er
14) Password brute for
15) Smart lock replay
16) Smart lock information leak
17) Summary
Summary
1 task of 15 solved
First steps
BLE Advertisements
Beacons
Manufacturer Specific Advertisements
Connections, services, characteristics
Characteristic read
Notifications
Descriptors
Characteristic write
Various writes
Write automation
Protocol reverse-engineering
Password brute force
Smart lock replay
Smart lock information leak
Current status of tasks

Device name?

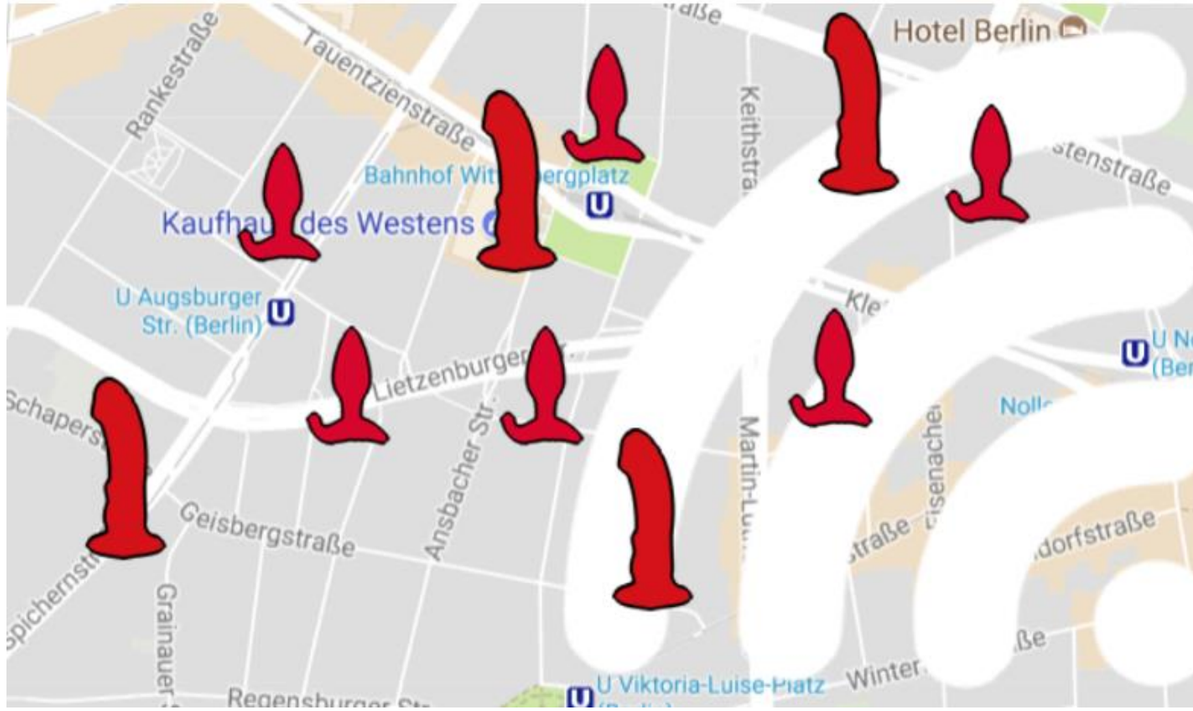
REALOV_VIBE
38:D2:69:E5:23:B1
NOT BONDED -91 dBm ↔ 302 ms

CONNECT

Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Complete Local Name: REALOV_VIBE
Incomplete List of 16-bit Service UUIDs: 0xFFF0
Complete Local Name: REALOV_VIBE
Slave Connection Interval Range: 7.50ms - 18.75ms
Tx Power Level: 0 dBm

Other interesting device names?

Vendor	Device Name	Ble Name
PicoBong	Blow hole	Blow hole
PicoBong	Blow hole	Picobong Male Toy
PicoBong	Life Guard	Life guard
PicoBong	Life Guard	Picobong Ring
PicoBong	Surfer	Surfer
PicoBong	Surfer	Picobong Butt Plug
PicoBong	Surfer	Egg driver
PicoBong	Surfer	Surfer_plug
PicoBong	Diver	Diver
PicoBong	Diver	Picobong Egg
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



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

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

Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

BLE Advertisements

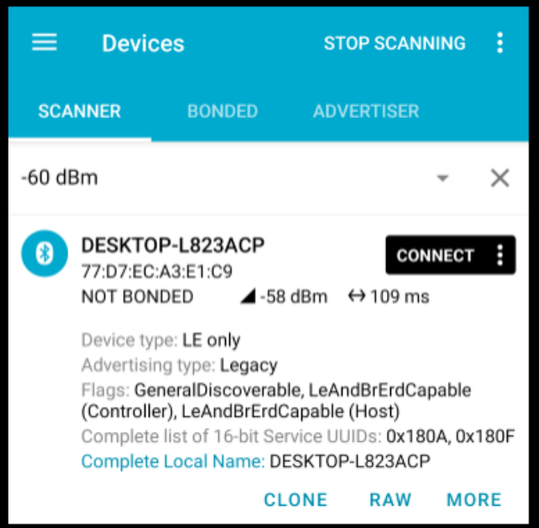
Theory introduction

Bluetooth Low Energy devices broadcast small packets called "advertisements". These packets are by design public (with a small exception of very uncommon "targeted advertisements"), and available for any receiver within Bluetooth range. The information broadcasted may contain for example some unique identifier, device name, state, sensor indication, or any other data the manufacturer would like to share. The data is formatted according to Bluetooth Generic Access Profile specification. Each transmitted field has associated GAP "Data Type", following its payload length and value. The most commonly used data types include:

- Flags - indicating device capabilities
- Device name
- List of Service UUIDs - device services (features) available after connecting (we'll get to it later)
- Manufacturer specific - proprietary, vendor data.

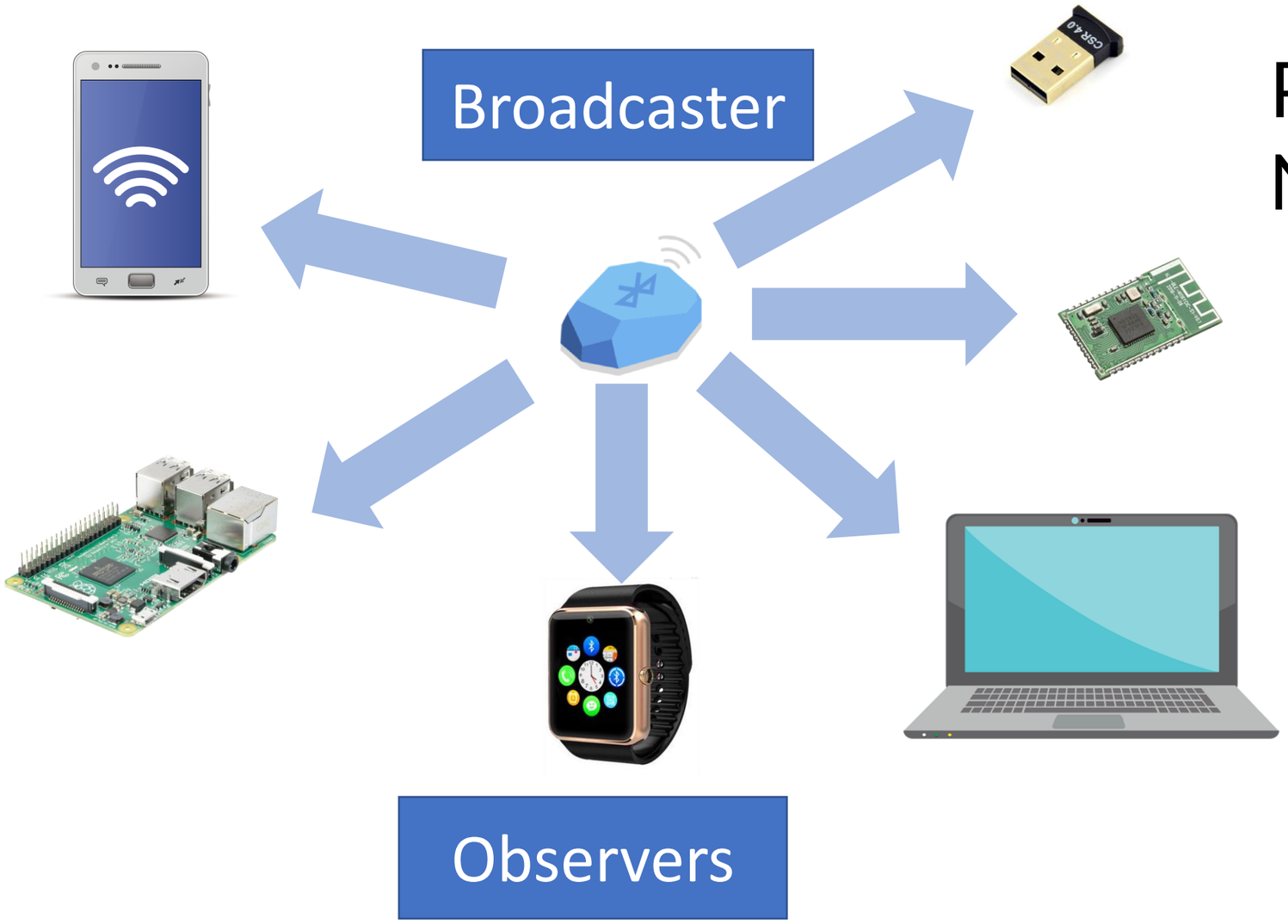
Task

In previous task you have identified your HackMe device, and noticed its name. Now tap on device name (not yet "CONNECT" button, we will get to it later), the application will expand some more information about the broadcasted packet:



The advertisement payload can change in time. Windows devices advertise their own BLE packets independently in the background, and you may notice it as a short "glitch" in the HackMe device advertisements. Just ignore this side effect:

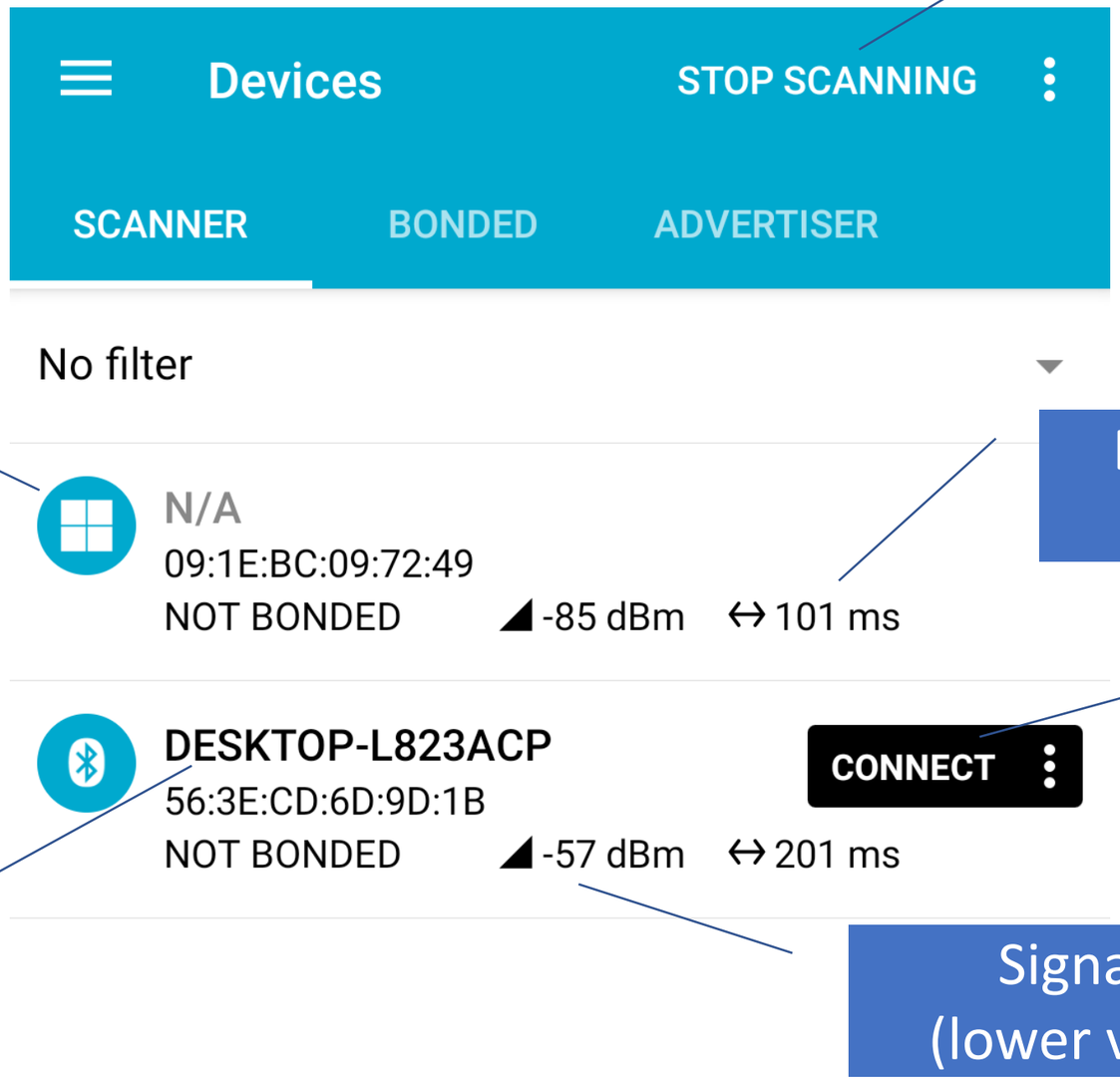
Advertisement: one-way broadcast



Public packets*
No pairing required

* except “targeted advertisements” (uncommon)

nRF Connect scan



Will stop after a while, may need to start again

Icon for device type

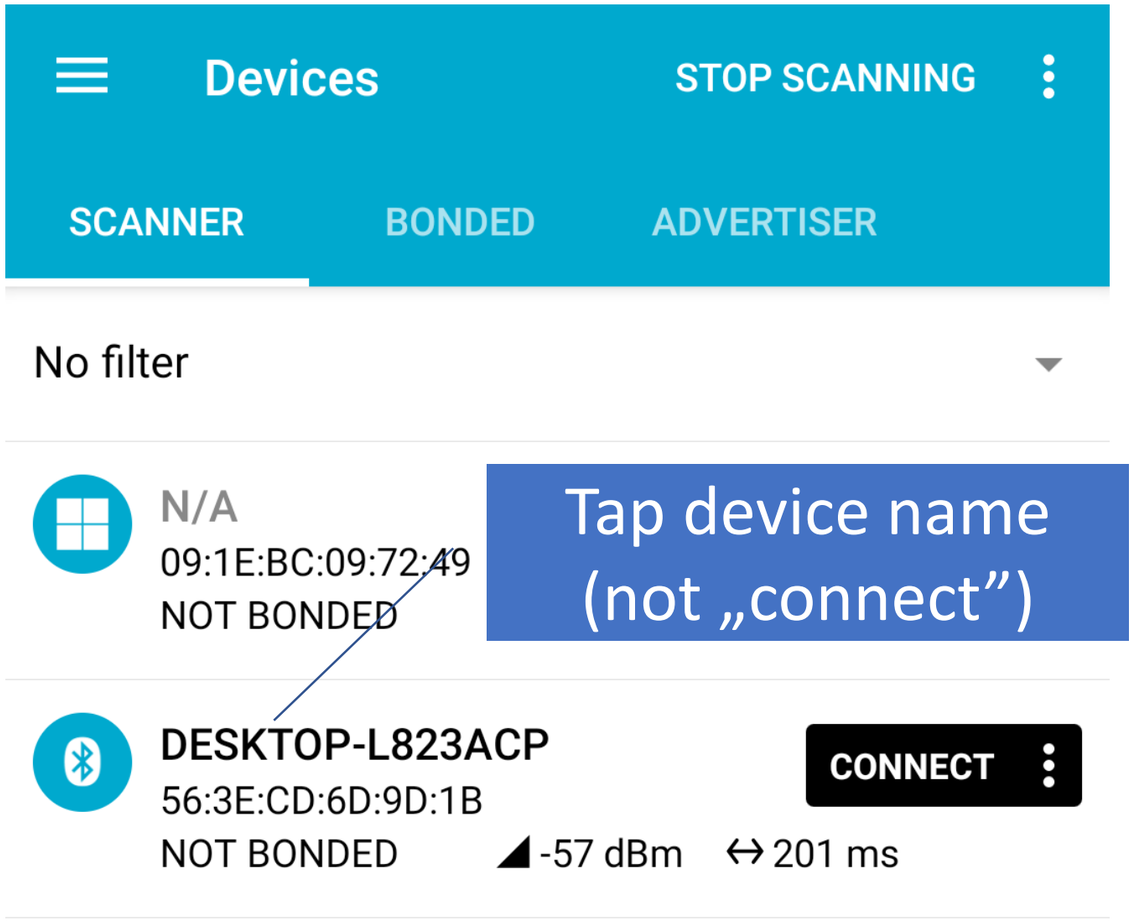
Delay between packets

Connectable device

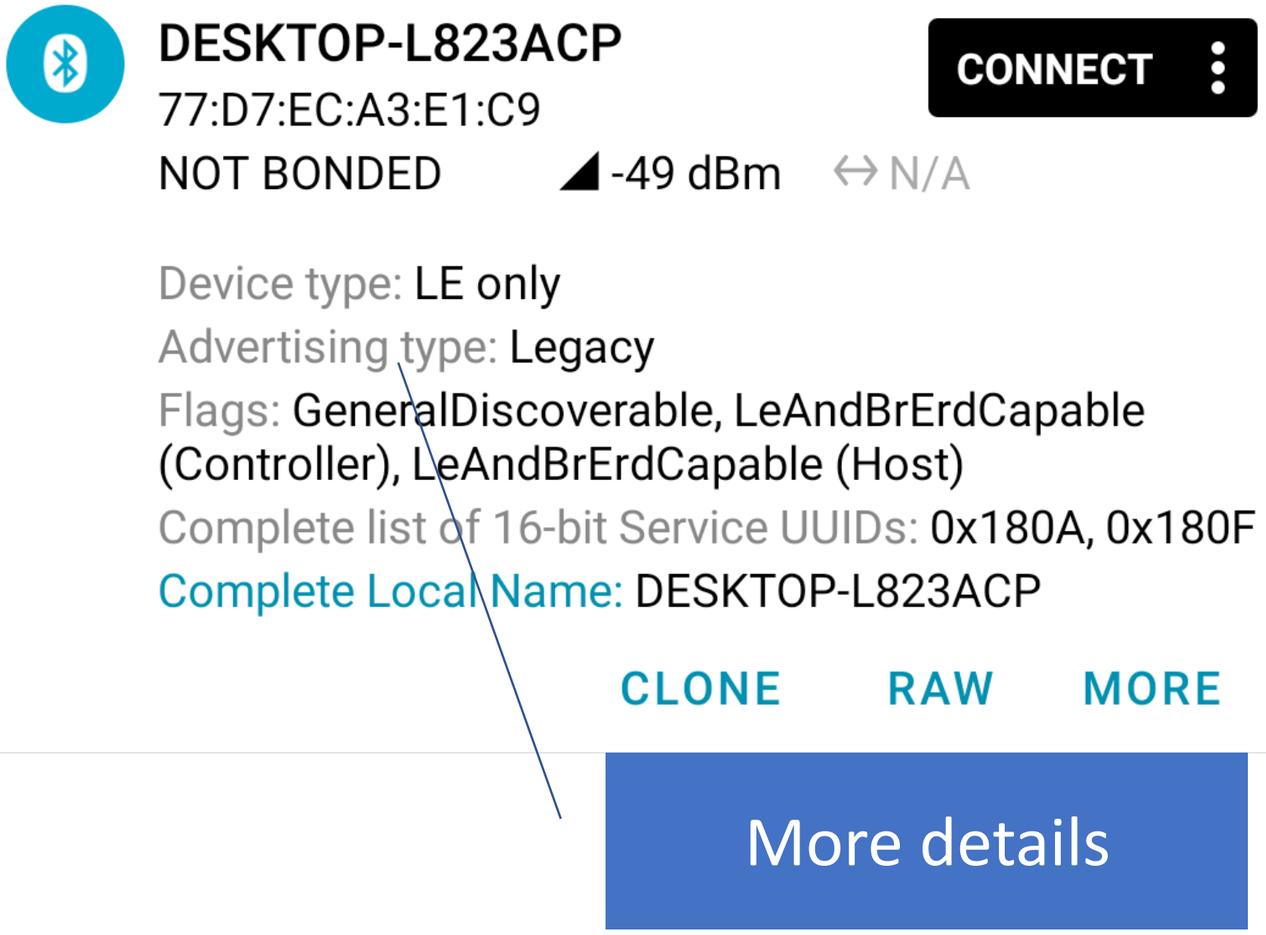
Optional device name

Signal strength (lower value=closer)

nRF Connect scan



Tap device name (not „connect”)



More details

BLE advertisements

DESKTOP-L823ACP
77:D7:EC:A3:E1:C9
NOT BONDED -49 dBm ↔ N/A

CONNECT

Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)
Complete list of 16-bit Service UUIDs: 0x180A, 0x180F
Complete Local Name: DESKTOP-L823ACP

CLONE RAW MORE

Raw hex bytes transmitted by device

Raw data:

0x02011A05030A180F1810094445534B544F
502D4C383233414350

Details:

LEN.	TYPE	VALUE
2	0x01	0x1A
5	0x03	0x0A180F18
16	0x09	0x4445534B544F502D4C383233414350

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

Decoded data

OK

Raw hex data: LEN, TYPE, VALUE

Raw data:

```
0x02011A05030A180F18100944445534B544F
502D4C383233414350
```

Details:

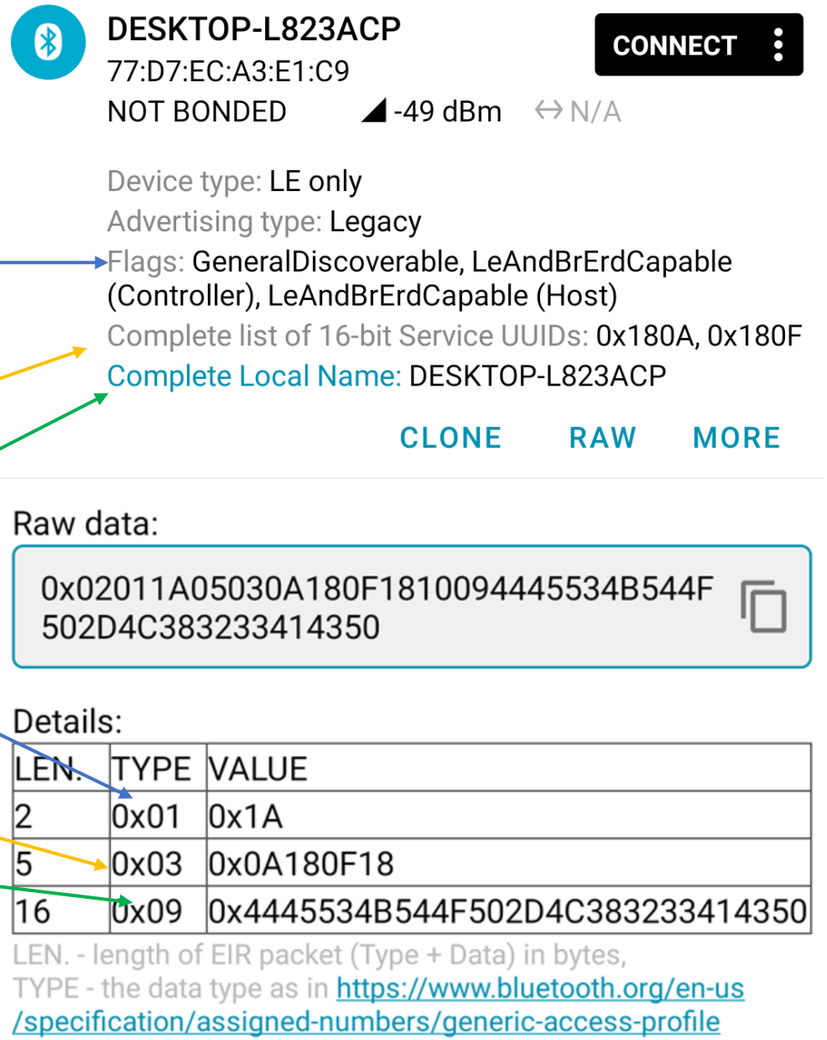
LEN.	TYPE	VALUE
2	0x01	0x1A
5	0x03	0x0A180F18
16	0x09	0x4445534B544F502D4C383233414350

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

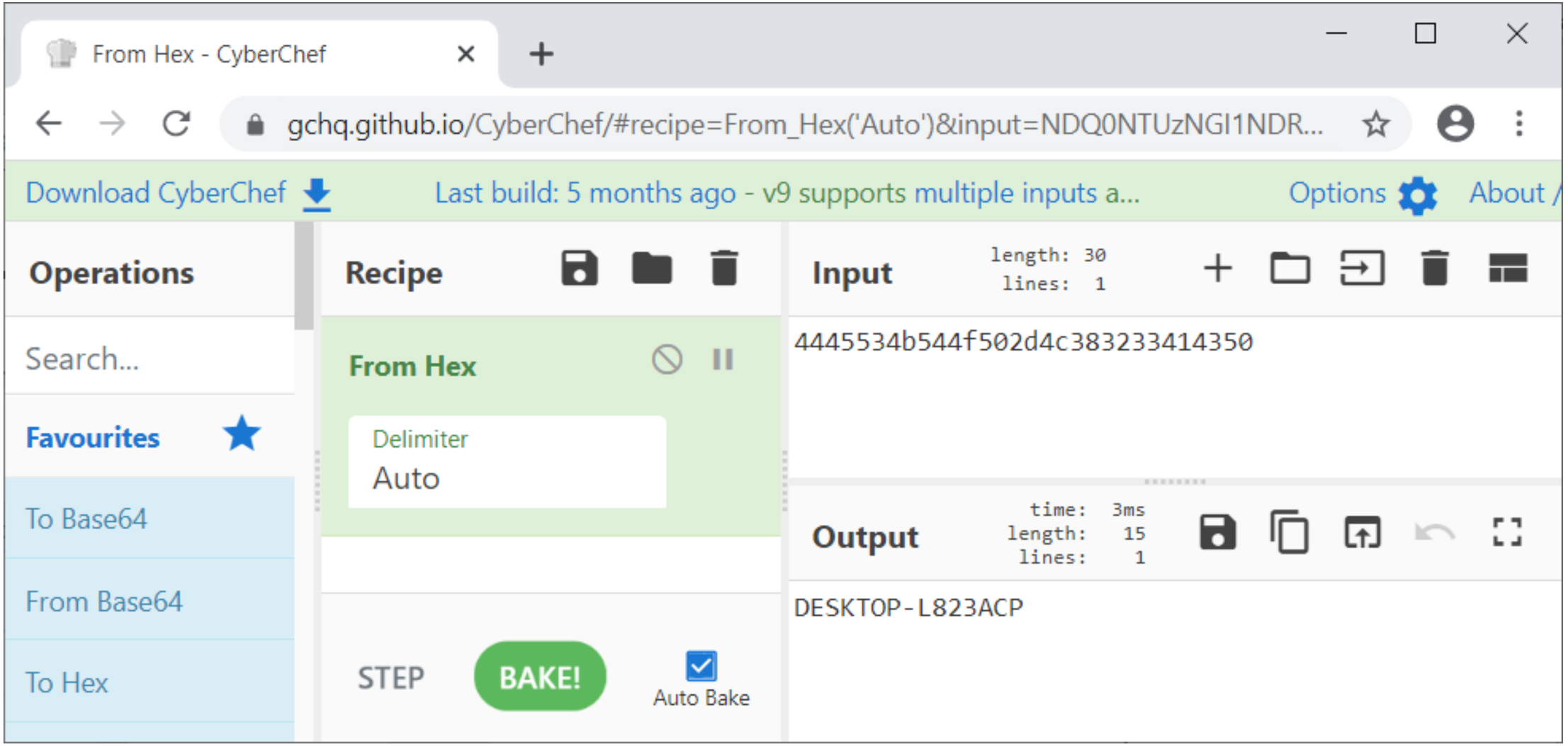
Generic Access Profile: types

Data Type Value	Data Type Name
0x01	«Flags»
0x02	«Incomplete List of 16-bit Service Class UUIDs»
0x03	«Complete List of 16-bit Service Class UUIDs»
0x09	«Complete Local Name»

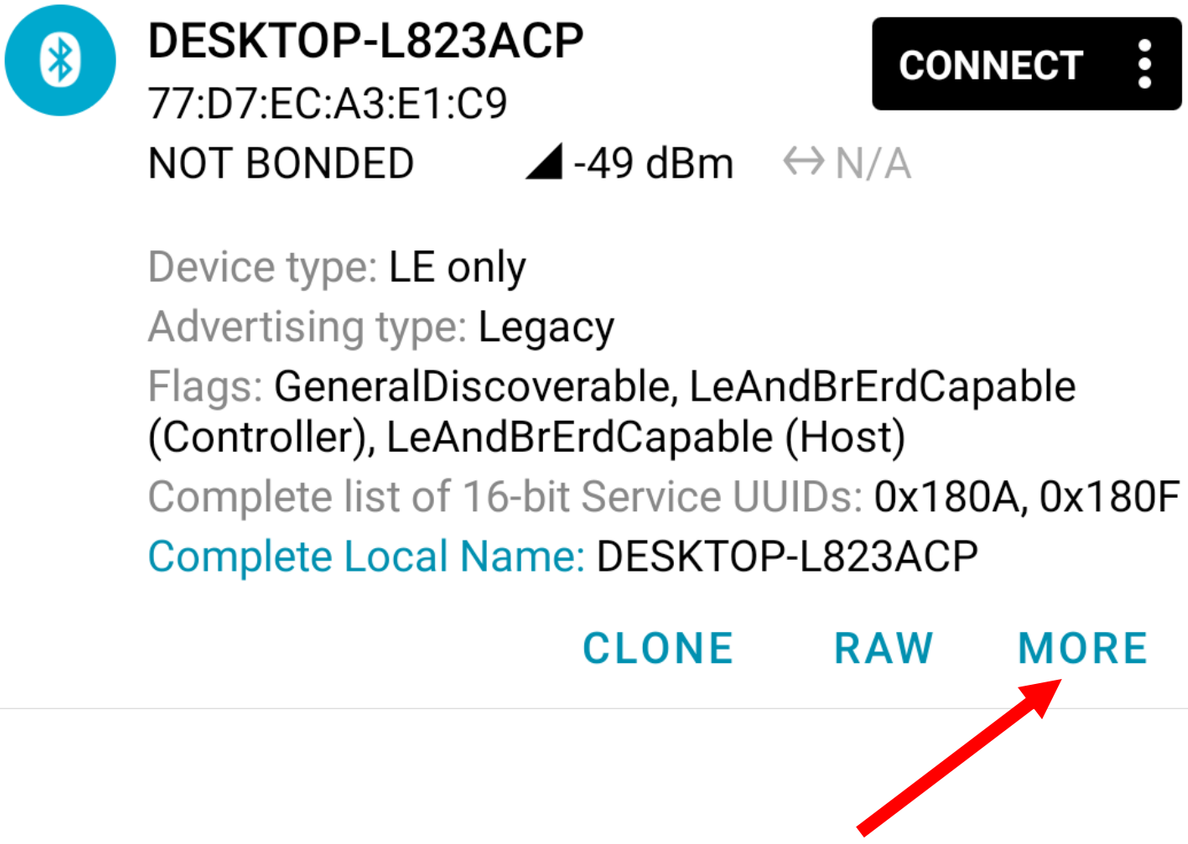
https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile/

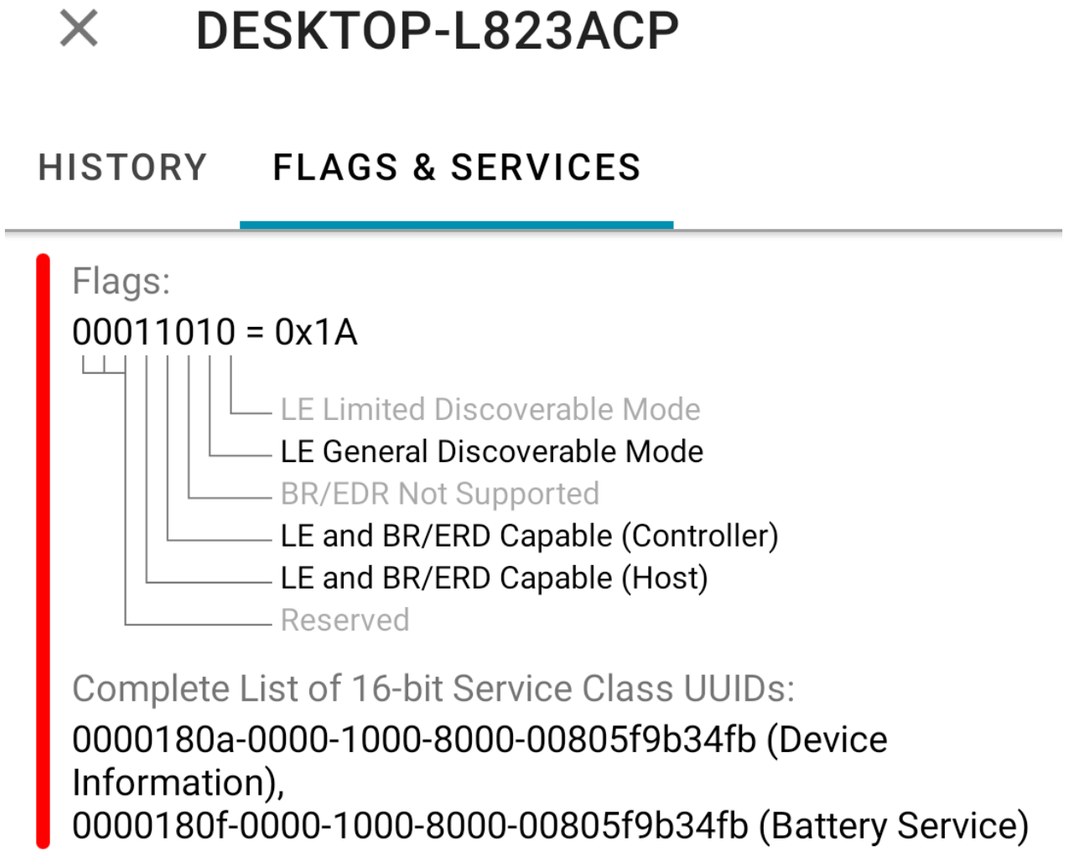


0x09 (complete local name) hex decoded



Flags (type 0x01) explained





Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

BLE iBeacons

Theory introduction

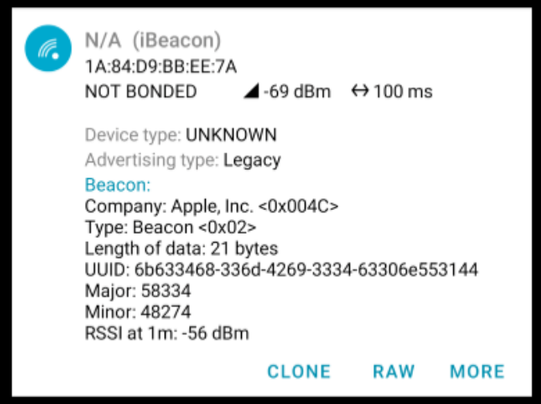
BLE advertisements are often used to broadcast some unique device identifiers, which can be used for example to identify specific device or pinpoint very precise indoor location of the receiving smartphone. One of the most commonly used formats is BLE iBeacon. The broadcasted packet contains:

- **UUID**, for example "00112233-4455-6677-8899-aabbccddeeff" - usually specific for vendor or installation
- Two numbers (0-65535): **Major** (usually common for group of devices) and **Minor** (for individual device).
- transmission signal strength, used to calculate the actual distance from device.

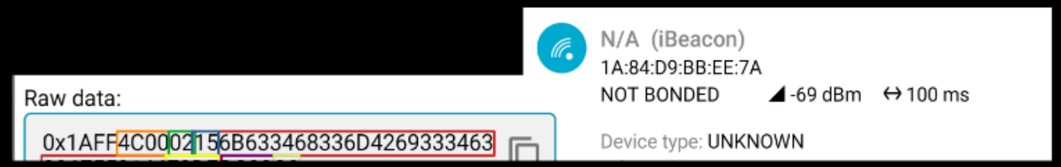
The beacon numbers are broadcasted as "manufacturer specific" (0xFF) data type in BLE advertisement packets.

Task

Your HackMe device advertisement has just changed. It does not broadcast its name any more (nRF Connect shows N/A), has different flags (no "CONNECT" button), and the Bluetooth address should also have changed. Your Windows still "glitches" with its own advertisement, the advertisement will switch just for a moment into "iBeacon", so it might be tricky to catch it:



Take a look at the "RAW" iBeacon packet (Note: if your raw packet starts with "0x1EFF0600" it means you caught this Microsoft packet, not iBeacon). You will quickly notice that there is no mystery - the data is simply embedded as raw hex into "0xFF" (Manufacturer Specific") field:



Might get tricky

Your windows advertises its own packets

N/A
5C:65:88:63:D5:91
NOT BONDED ◢ -48 dBm ↔ 105 ms

Device type: UNKNOWN
Advertising type: Legacy
Microsoft Advertising Beacon:
Scenario Type: Advertising Beacon <0x01>
Version: 0
Device Type: Windows 10 Desktop
Flags: 0x00 (version: 1)
Reserved: 0x02
Salt: 0x8AAF250E
Device Hash:
0xE82C1B012EF86FB6D1F7E8B39C10938F29BED9

CLONE RAW MORE

Changes into iBeacon only for a moment

N/A (iBeacon)
1A:84:D9:BB:EE:7A
NOT BONDED ◢ -69 dBm ↔ 100 ms

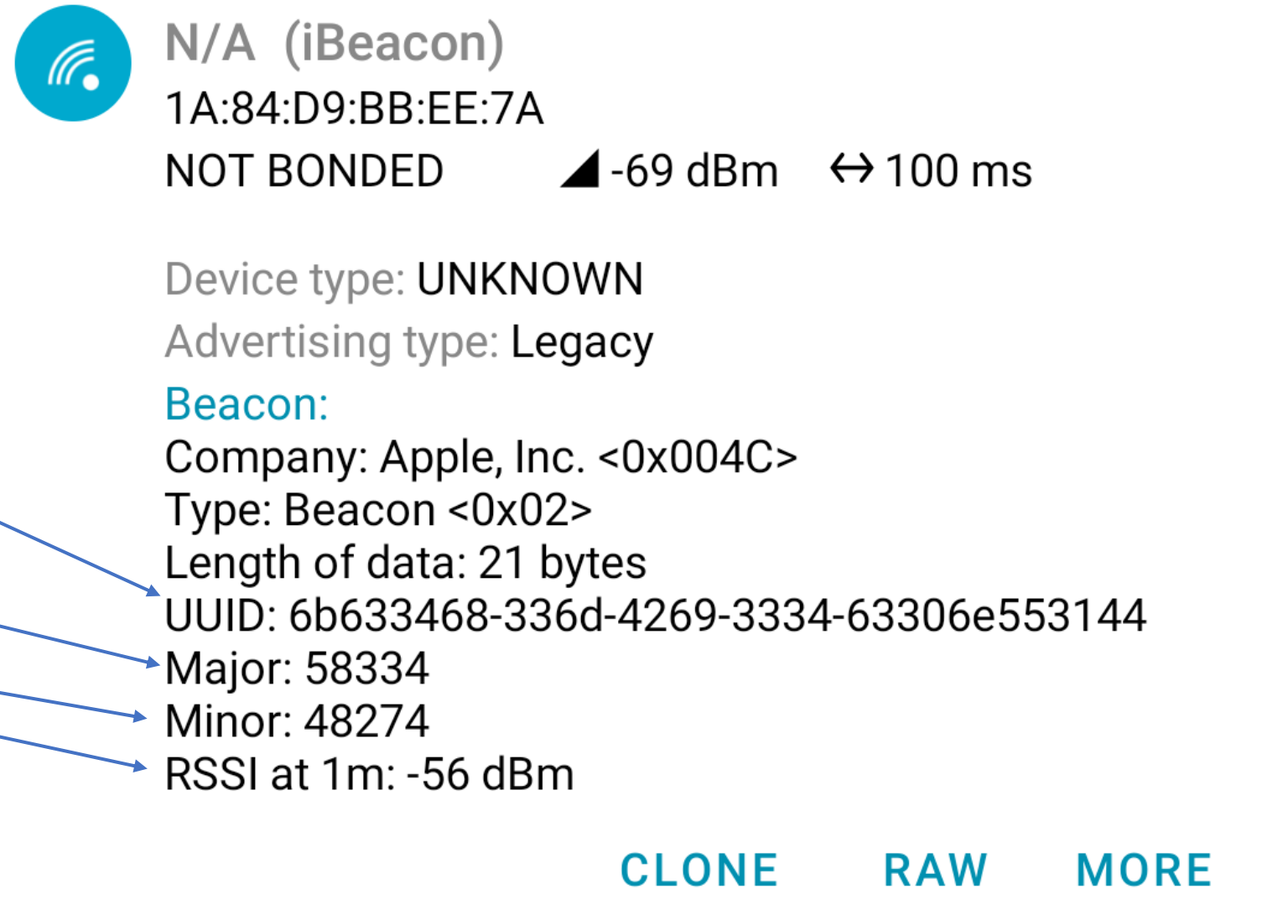
Device type: UNKNOWN
Advertising type: Legacy
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 6b633468-336d-4269-3334-63306e553144
Major: 58334
Minor: 48274
RSSI at 1m: -56 dBm

CLONE RAW MORE

iBeacon

Transmits

- UUID
- Two numbers:
  - Major
  - Minor
- Signal strength (RSSI)



iBeacon raw advertisement: 0xFF

Data Type Value	Data Type Name
0xFF	«Manufacturer Specific Data»

Raw data:

0x1AFF4C000215 6B633468336D4269333463 306E553144E3DEBC92C8

Details:

LEN.	TYPE	VALUE
26	0xFF	0x4C000215 6B633468336D4269333463 306E553144E3DEBC92C8

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

N/A (iBeacon)
1A:84:D9:BB:EE:7A
NOT BONDED -69 dBm ↔ 100 ms

Device type: UNKNOWN
Advertising type: Legacy
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 6b633468-336d-4269-3334-63306e553144
Major: 58334
Minor: 48274
RSSI at 1m: -56 dBm

CLONE RAW MORE

Beacons in practice?

- Indoor location, track users
- Display context information in a shop or gallery
- Automatic check-in at places (get free food!)
- Key-finder
- Integration with home automation
- Trigger other location based actions

https://www.aislelabs.com/reports/beacon-guide/

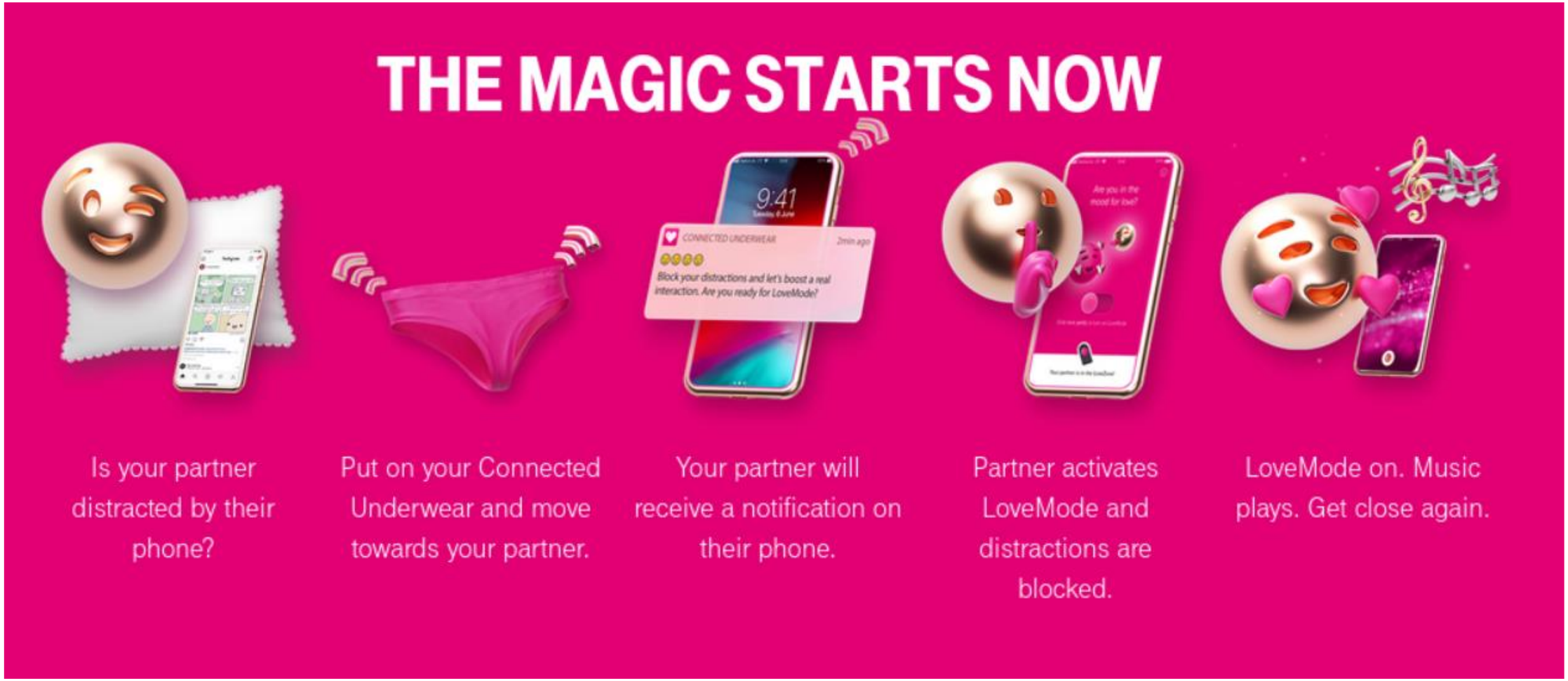
Connected underwear

https://www.lovemagenta.com/connected_underwear

Connected underwear: v2

https://www.lovemagenta.com/connected_underwear

How does it work?



https://www.lovemagenta.com/connected_underwear

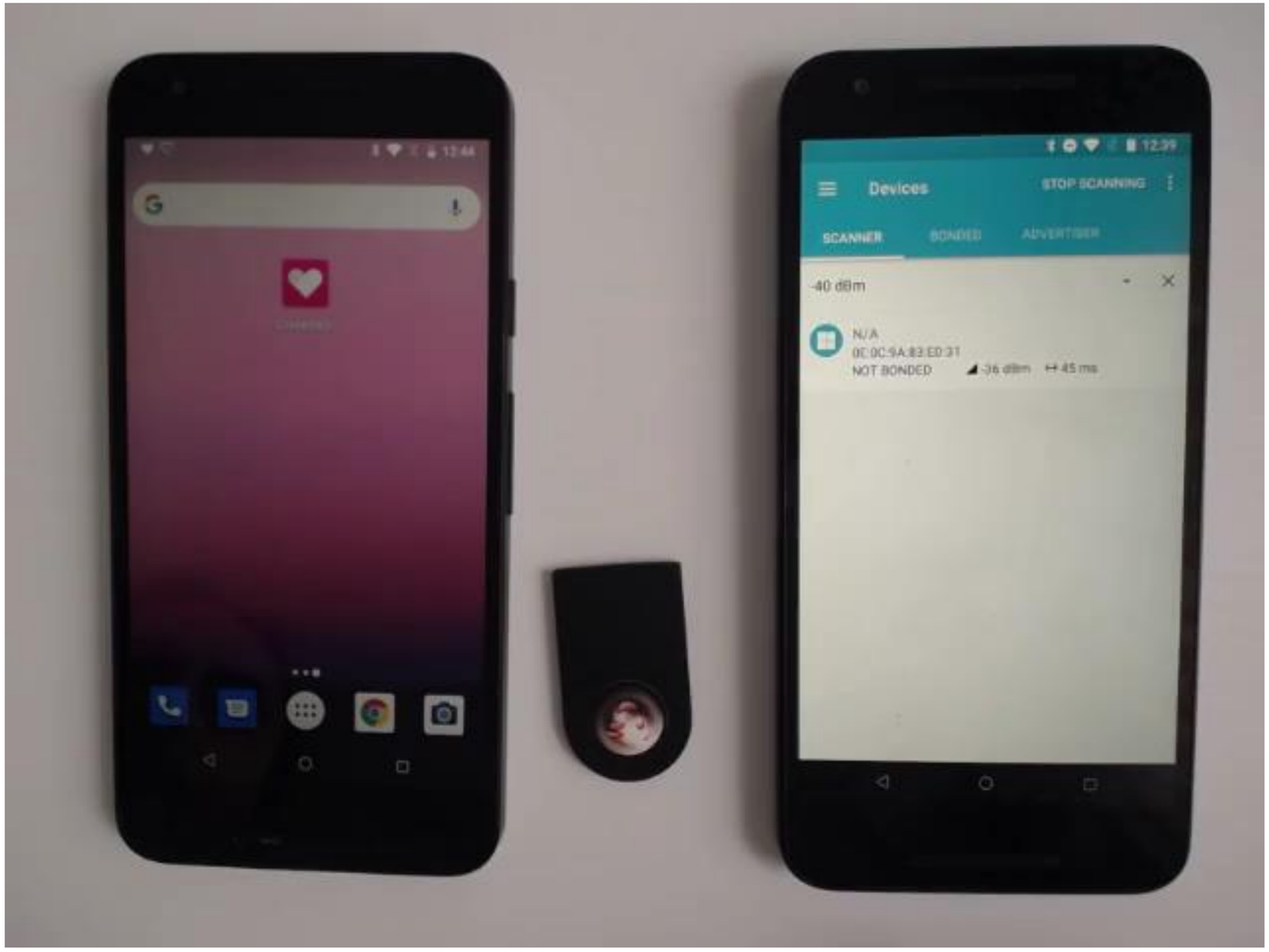
The BLE chip: iBeacon advertisement

N/A (iBeacon)
EA:1A:42:8F:B5:AB
NOT BONDED -36 dBm ↔ 305 ms

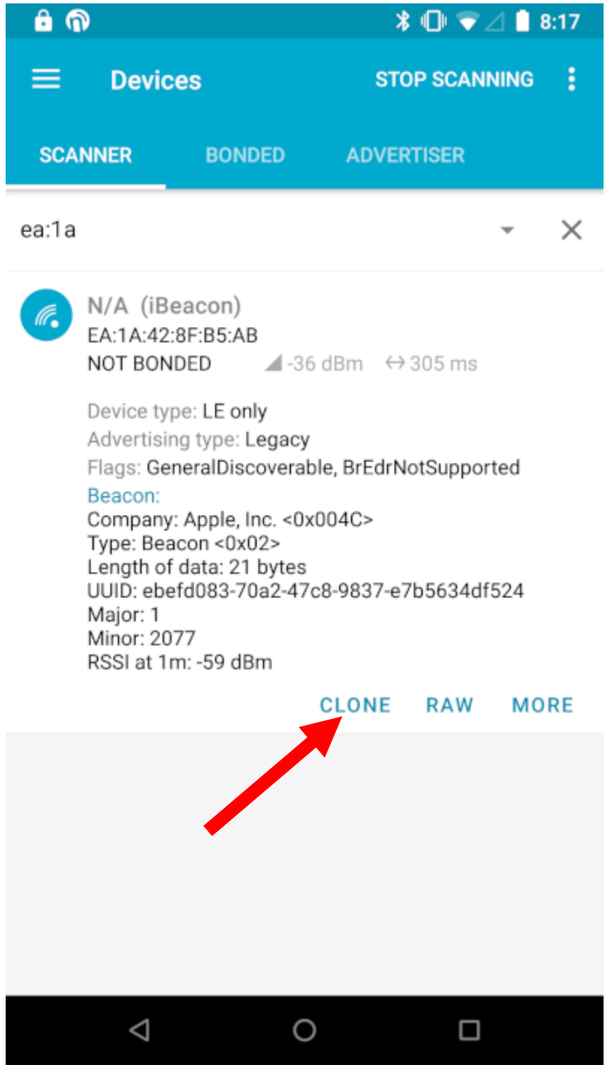
Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, BrEdrNotSupported
Beacon:
Company: Apple, Inc. <0x004C>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: ebefd083-70a2-47c8-9837-e7b5634df524
Major: 1
Minor: 2077
RSSI at 1m: -59 dBm

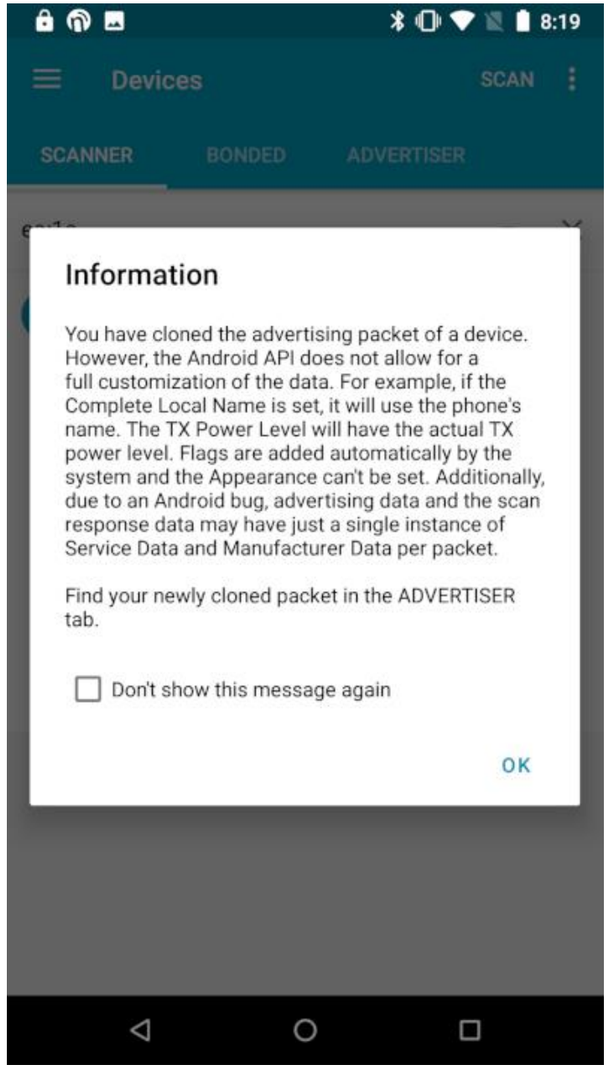
CLONE RAW MORE

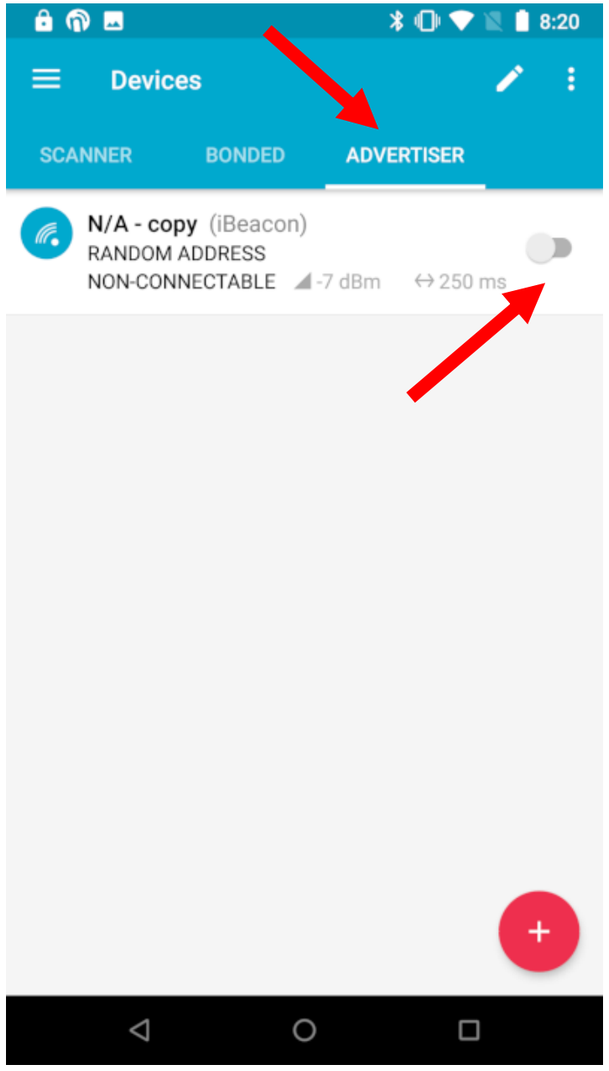
Smart pants: iBeacon clone demo

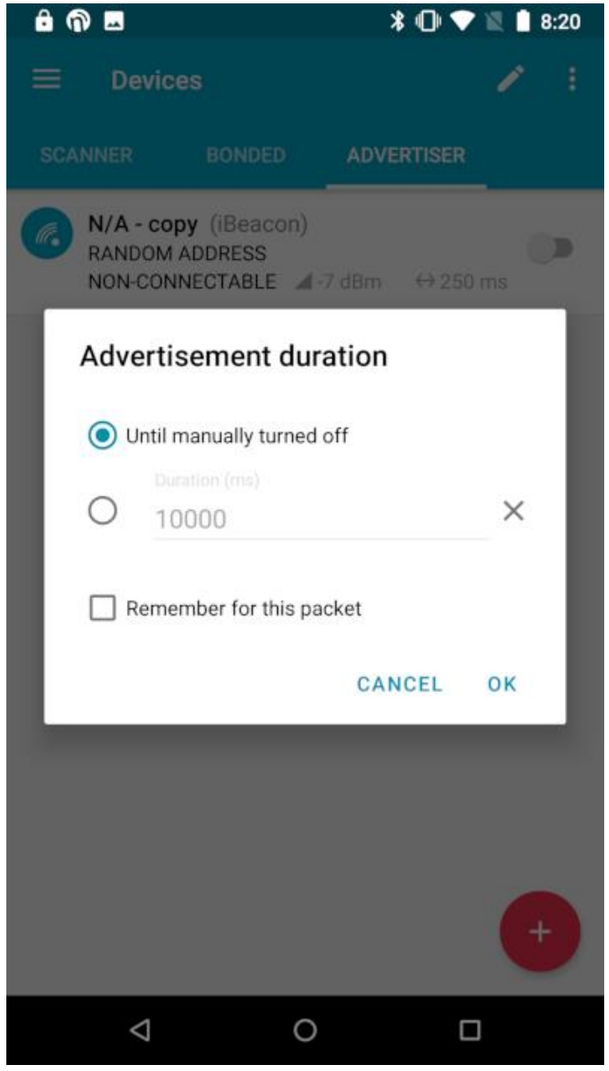


Cloning iBeacon advertisement in nRF





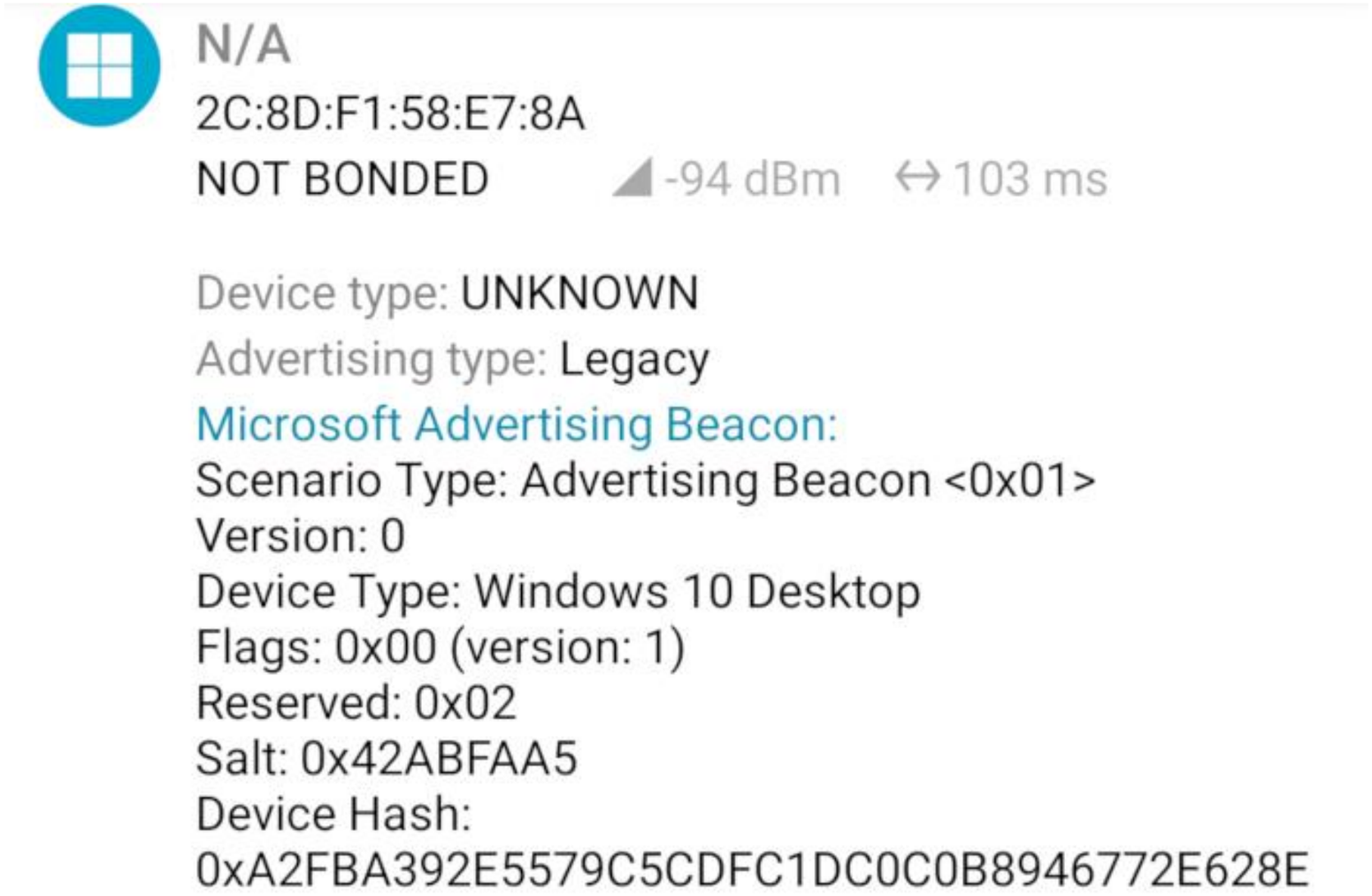




Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

Windows 10 advertisements



Windows®

Microsoft BLE “beacon” spec

Beacon Data (24 bytes): The beacon data section is further broken down. Note that the Scenario and Subtype Specific Data section requirements will differ based on the Scenario and Subtype.

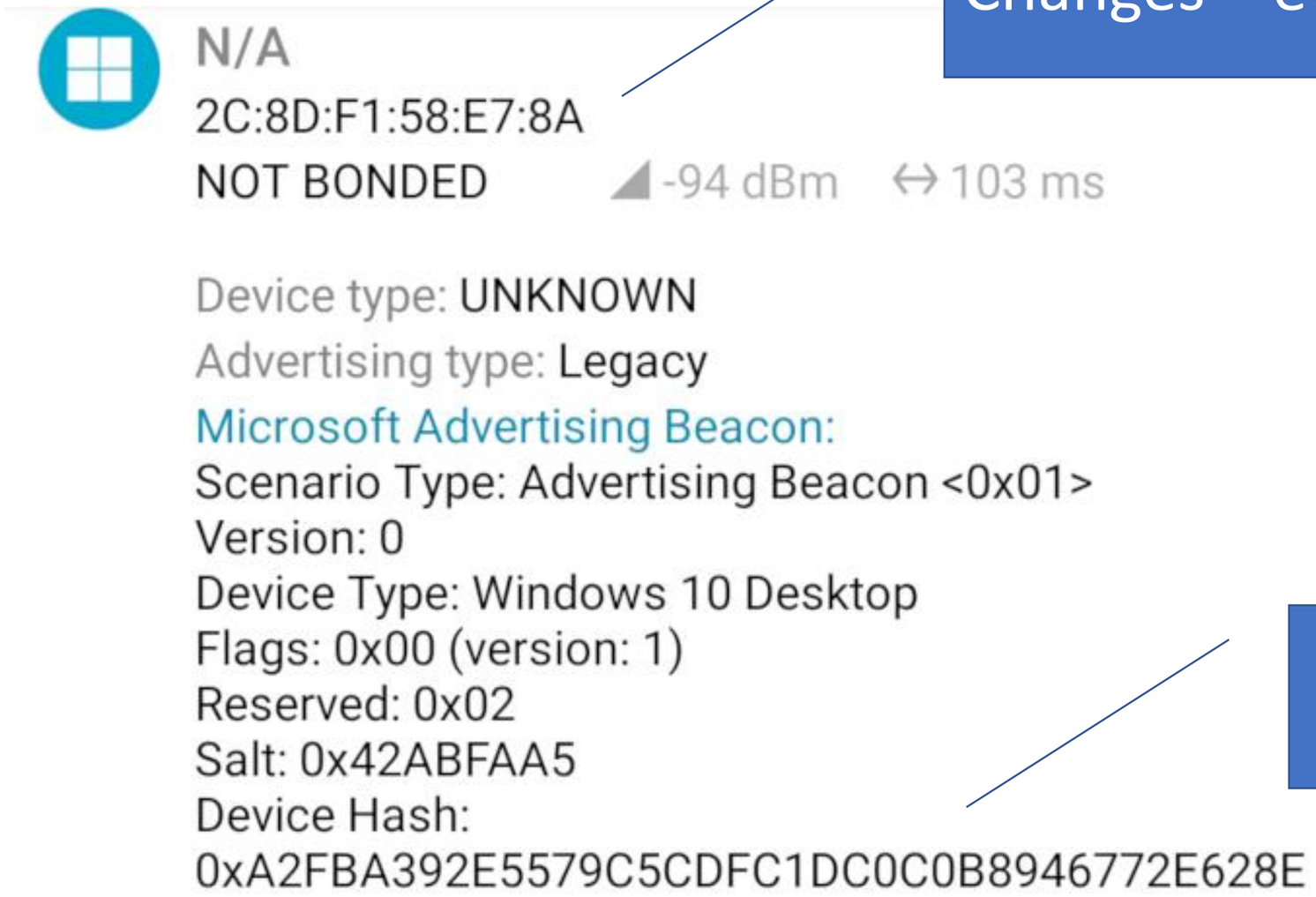
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Scenario Type								Version and Device Type								Version and Flags								Reserved							
Salt																															
Device Hash (24 bytes)																															

Version and Device Type (1 byte): The high two bits are set to 00 for the version number; the lower6 bits are set to Device Type values as in section 2.2.2.2.2:

Value	Meaning
1	Xbox One
6	Apple iPhone
7	Apple iPad
8	Android device
9	Windows 10 Desktop
11	Windows 10 Phone
12	Linus device
13	Windows IoT
14	Surface Hub

https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cdp/77b446d0-8cea-4821-ad21-fabdf4d9a569

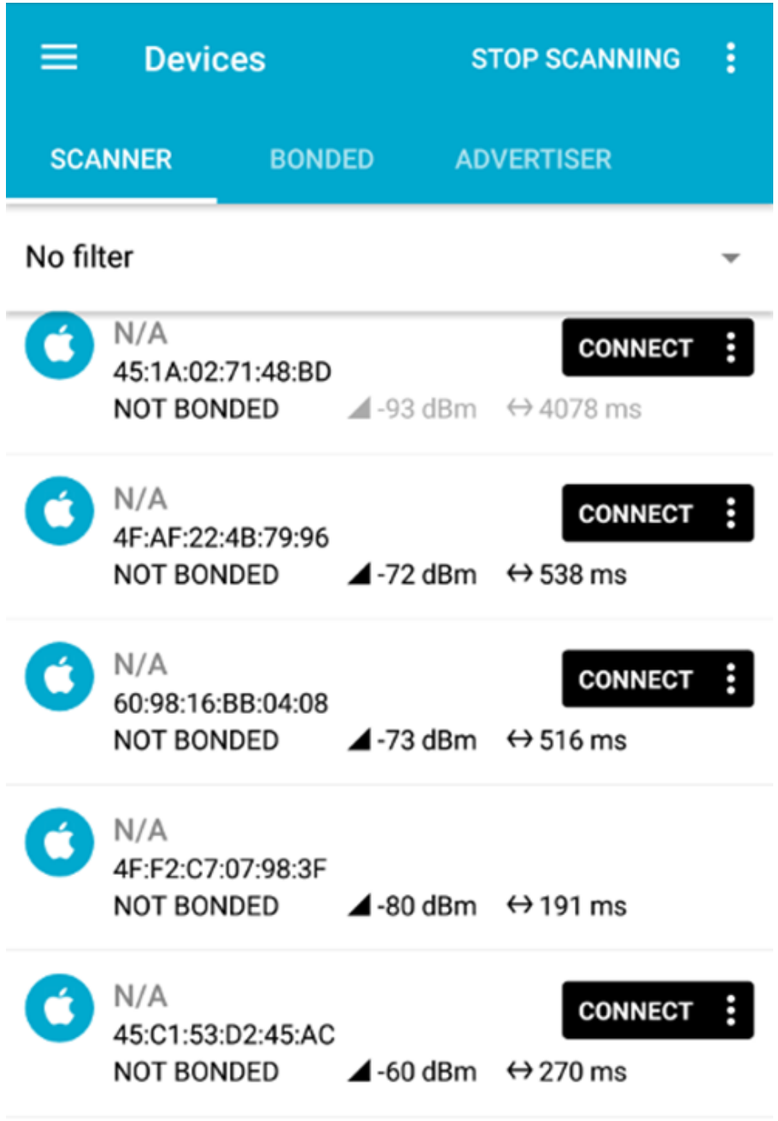
Tracking?

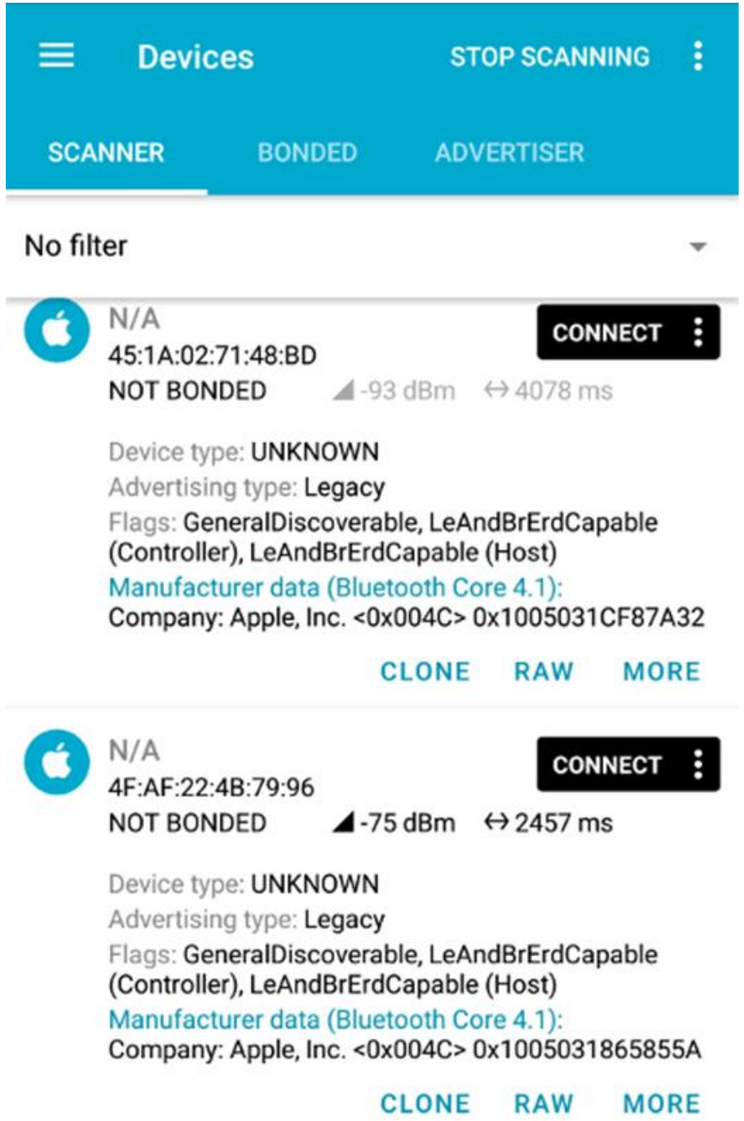


Changes ~ every 15 min

Changes ~ every 1 h

iPhone BLE advertisements





Raw data:

0x02011A07FF4C0010020B00

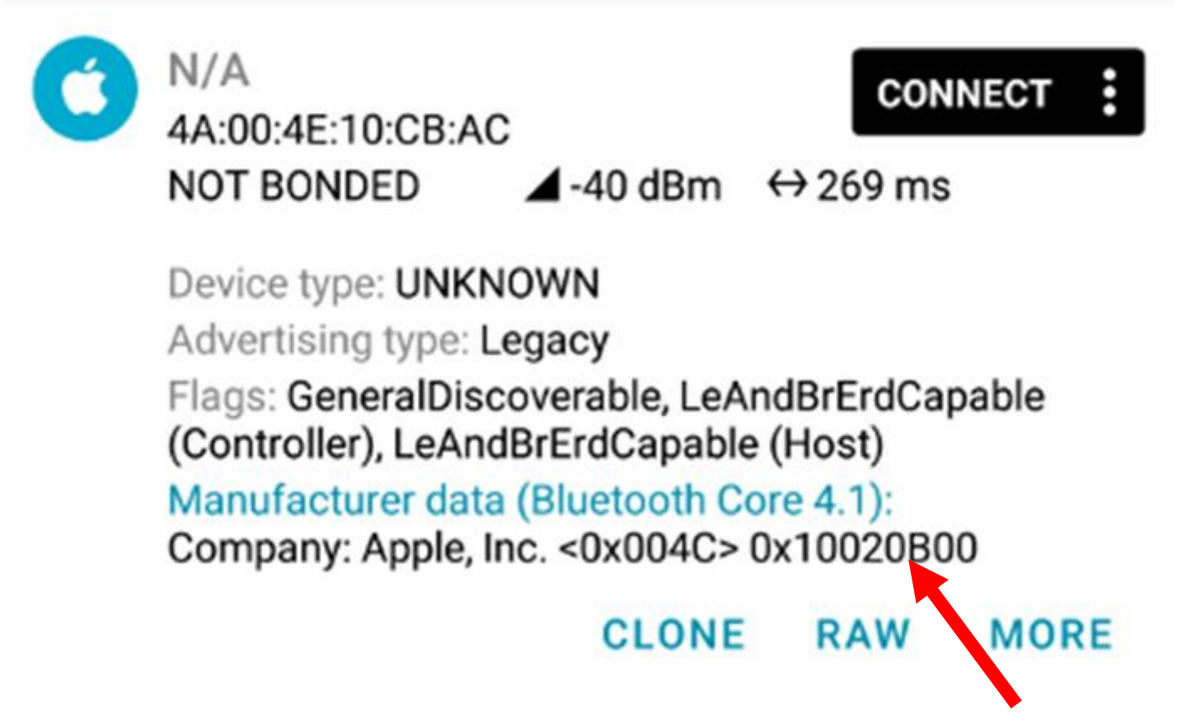
Details:

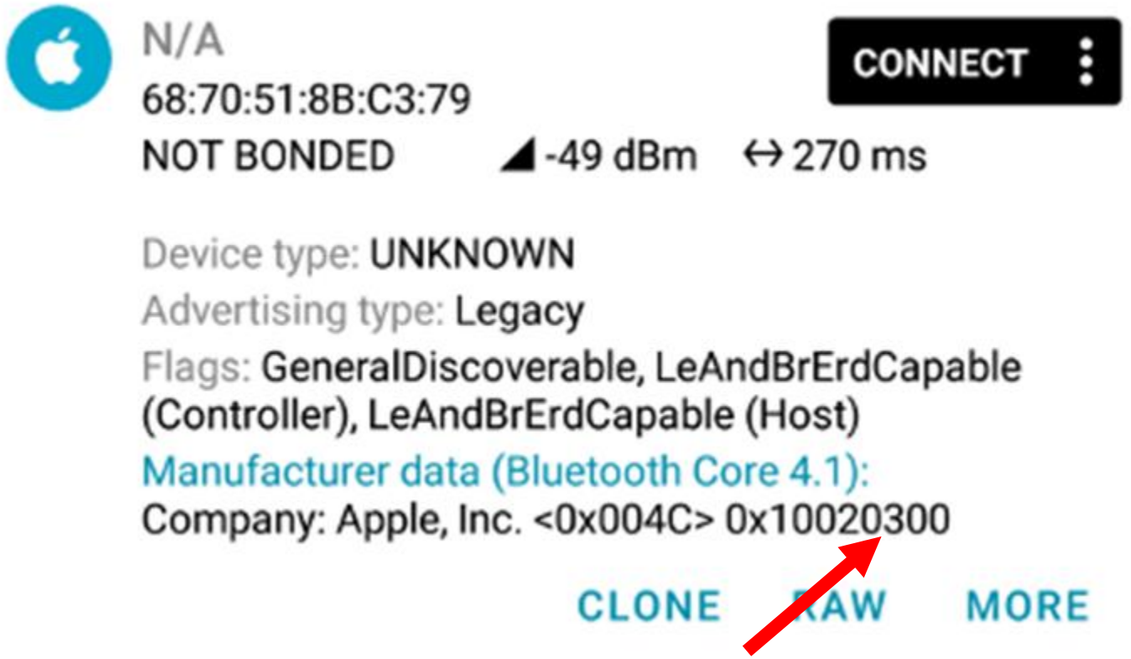
LEN.	TYPE	VALUE
2	0x01	0x1A
7	0xFF	0x4C0010020B00

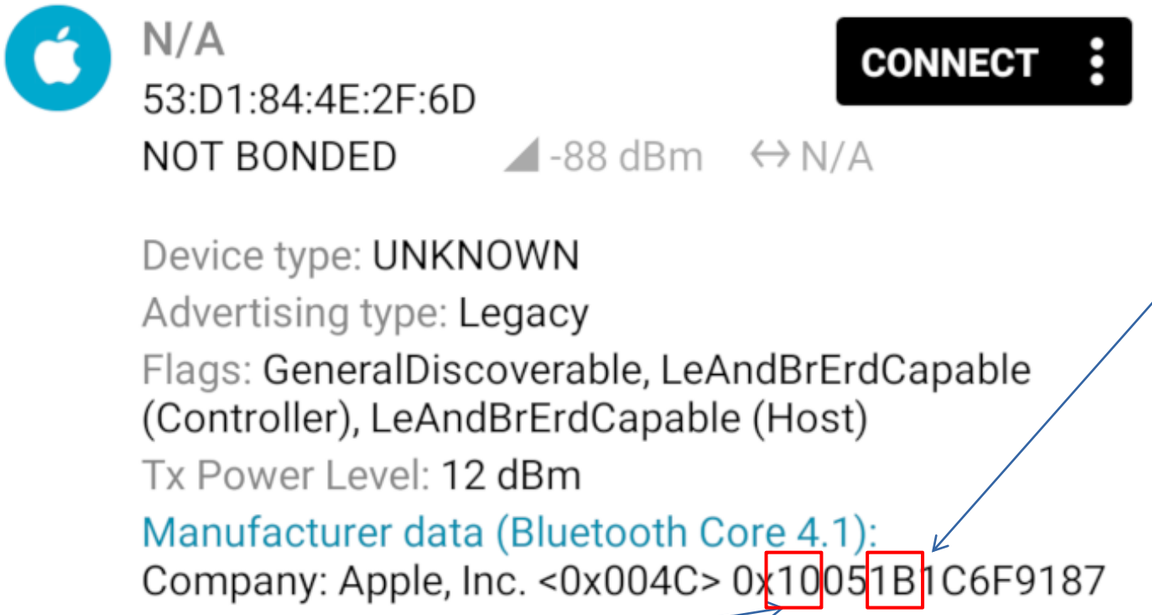
LEN. - length of EIR packet (Type + Data) in bytes,
TYPE - the data type as in https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile

OK

Switching screen on/off?

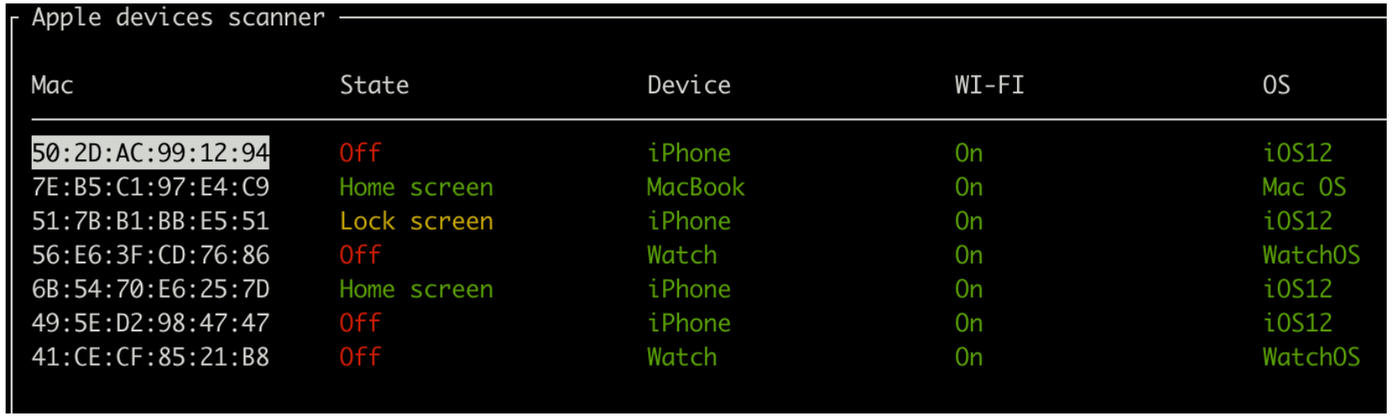






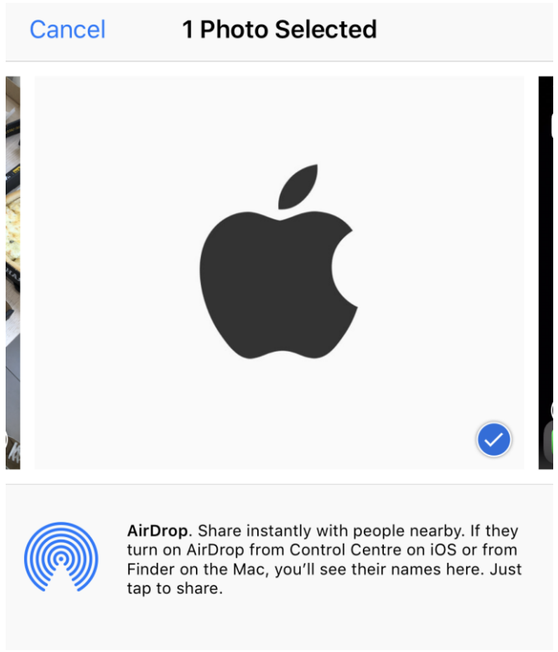

```
0x0b - Home screen
0x1c - Home screen
0x1b - Home screen
0x11 - Home screen
0x03 - Off
0x18 - Off
0x09 - Off
0x13 - Off
0x0a - Off
0x1a - Off
0x01 - Off
0x07 - Lock screen
0x17 - Lock screen
0x0e - Calling
0x5b - Home screen
0x5a - Off
```

```
0x05 - Airdrop
0x07 - Airpods
0x10 - Nearby
0x0b - Watch Connection
0x0c - Handoff
0x0d - Wi-Fi Settings
0x0e - Hotspot
0x0f - Wi-Fi Join Network
```

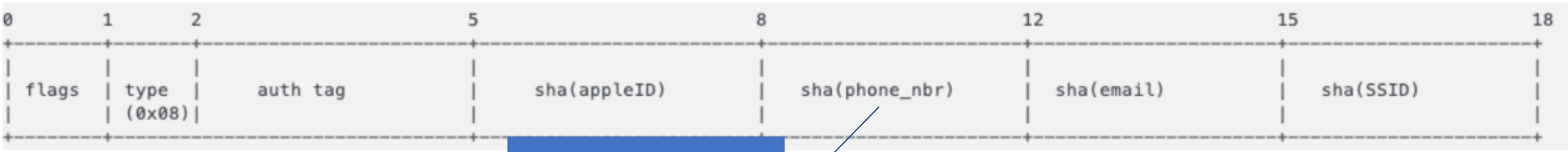


https://github.com/hexway/apple_bleee

Airdrop or wifi sharing: longer data



Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, [Device specific]
Manufacturer data (Bluetooth Core 4.1):
Company: Apple, Inc. <0x004C>
0x0F11C0086258BE3125C9D087555A77E359
AD1B10020B0C



First few bytes of SHA(phone numer)

https://github.com/hexway/apple_bleee

First bytes of SHA(phone number)

Limited format phone numbers:

+ 1 - 213 - xxx-xxxx

+ (country) (area) (number)

Create „rainbow tables” of SHA(all phone numbers)

Look up the advertised first bytes SHA(target number)

-> target’s phone number

Possible collisions easy to discard

More info: https://hexway.io/research/apple-bleee/

More Apple Continuity, Wireshark dissector:
https://github.com/furiousMAC/continuity

Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

G+A „exposure notification”

Google

COVID-19 Exposure Notifications: Technology To Help Public Health Institutions Fight Pandemic

Google and Apple have jointly developed a COVID-19 exposure alert system to help government agencies and the global community fight the pandemic through a technique known as 'contact tracing'.

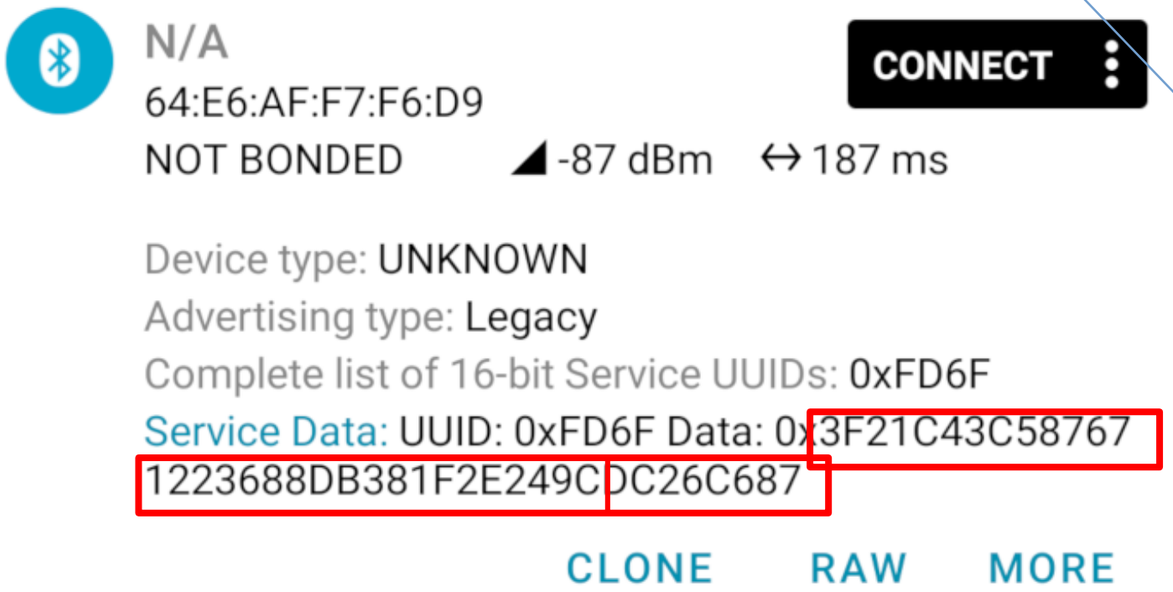
https://www.google.com/covid19/exposurenotifications/

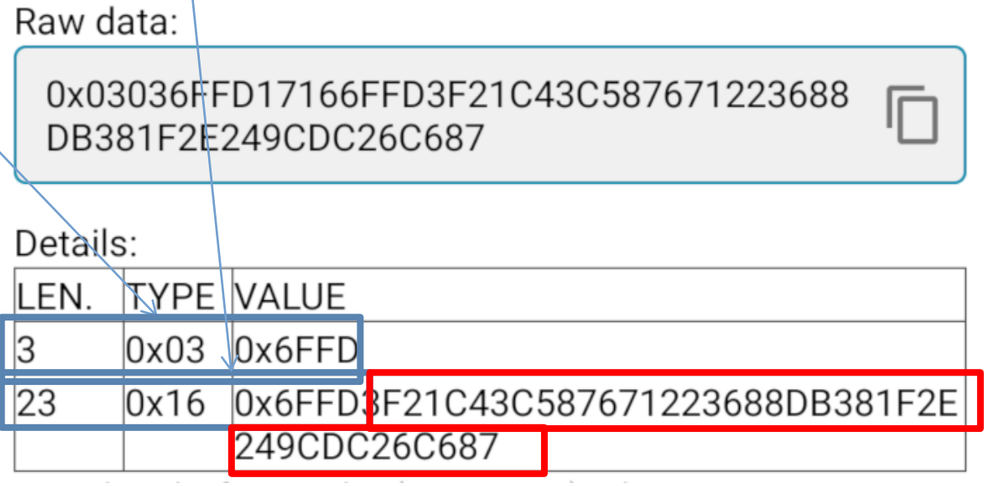
BLE packets

Flags			Complete 16-bit Service UUID			Service Data - 16 bit UUID				
Length	Type	Flags	Length	Type	Service UUID	Length	Type	Service Data		
0x02	0x01 (Flag)	0x1A	0x03	0x03 (Complete 16-bit Service UUID)	0xFD6F (Exposure Notification Service)	0x17	0x16 (Service Data - 16 bit UUID)	0xFD6F (Exposure Notification Service)	**16 bytes** **Rolling Proximity Identifier**	**4 bytes** **Associated Encrypted Metadata**

?

https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf





“Encrypted metadata”

b. A 4 byte Associated Encrypted Metadata that contains the following (LSB first):

   i. Byte 0 — Versioning.

      - Bits 7:6 — Major version (01).
      - Bits 5:4 — Minor version (00).
      - Bits 3:0 — Reserved for future use.

   ii. Byte 1 — Transmit power level.

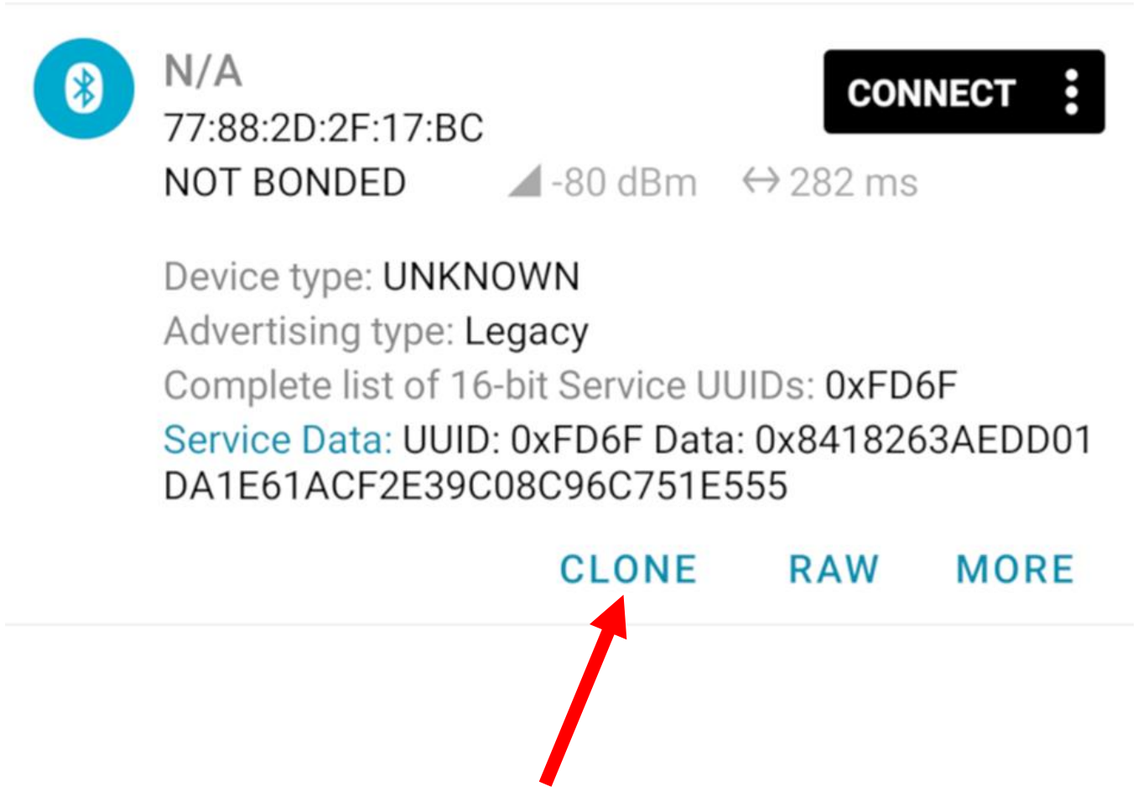
      - This is the measured radiated transmit power of Bluetooth Advertisement packets, and is used to improve distance approximation. The range of this field shall be -127 to +127 dBm.

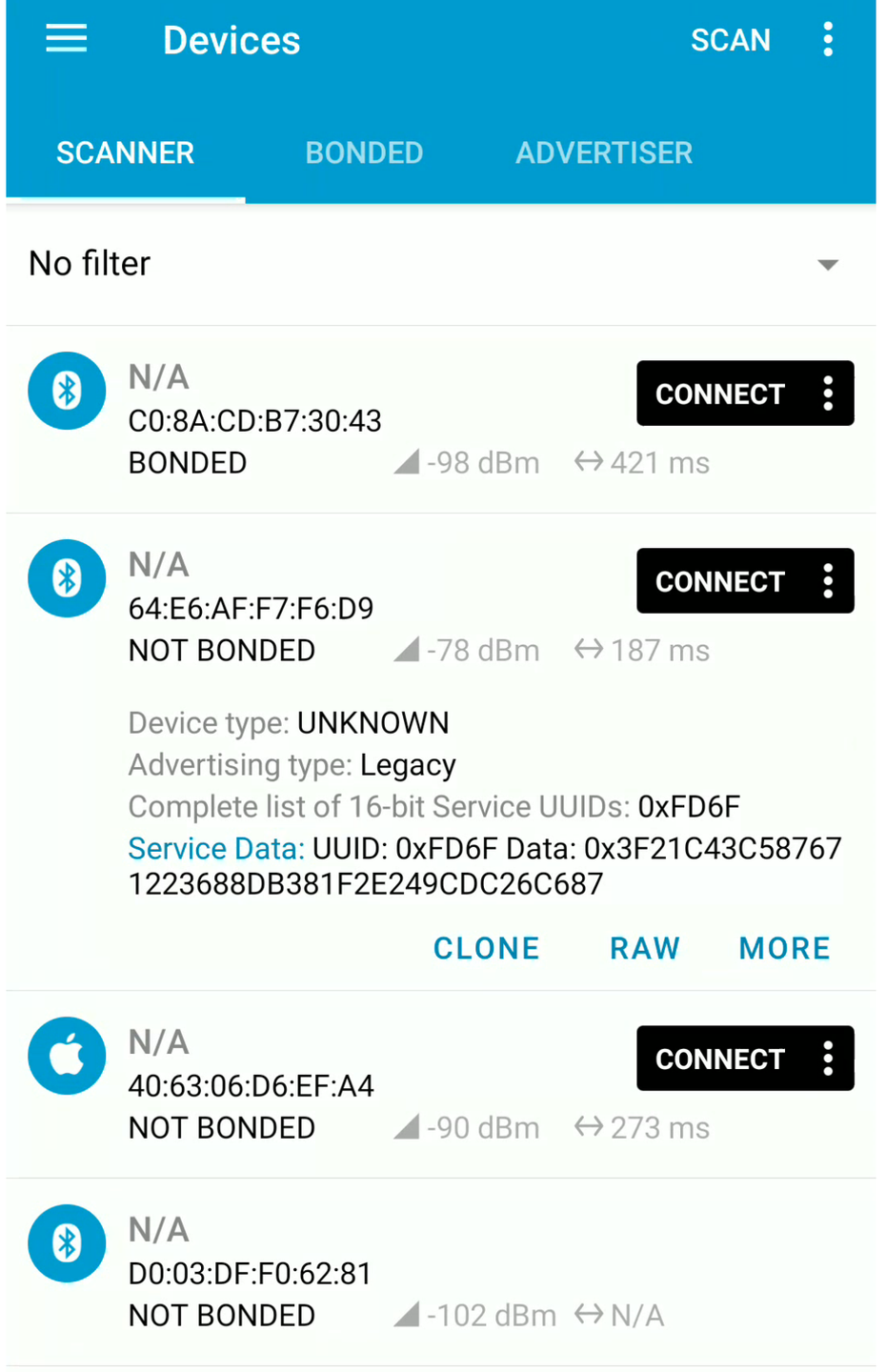
   iii. Byte 2 — Reserved for future use.

   iv. Byte 3 — Reserved for future use.

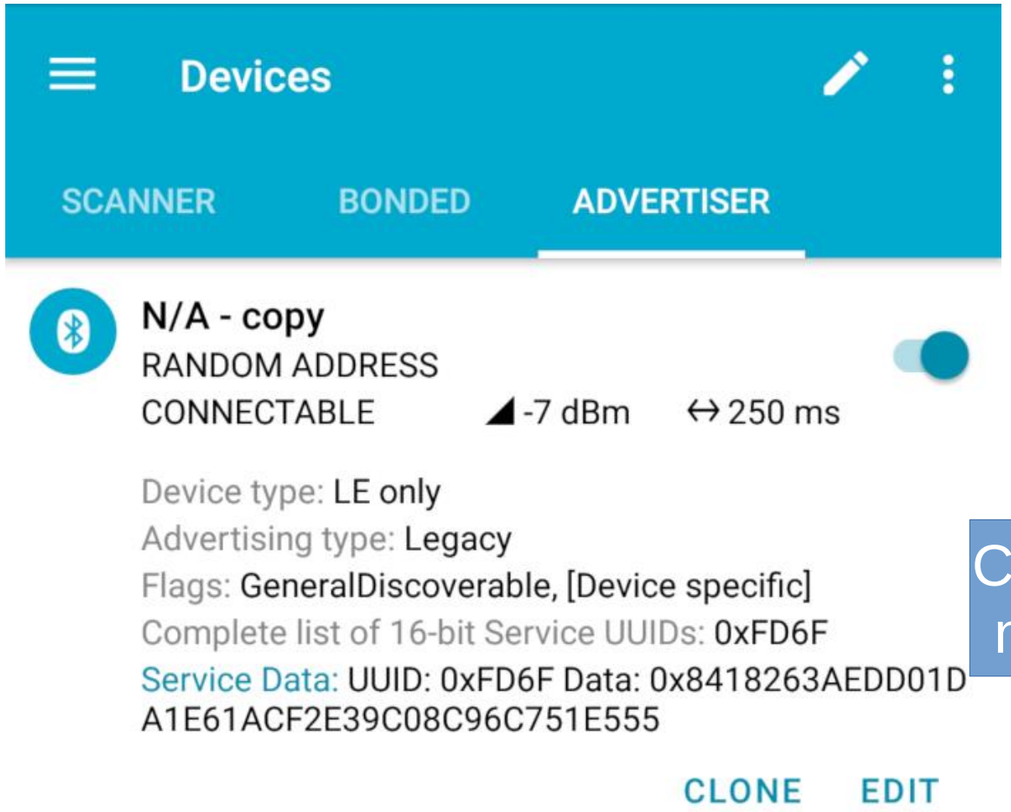
https://blog.google/documents/70/Exposure_Notification_-_Bluetooth_Specification_v1.2.2.pdf

nRF Connect – clone

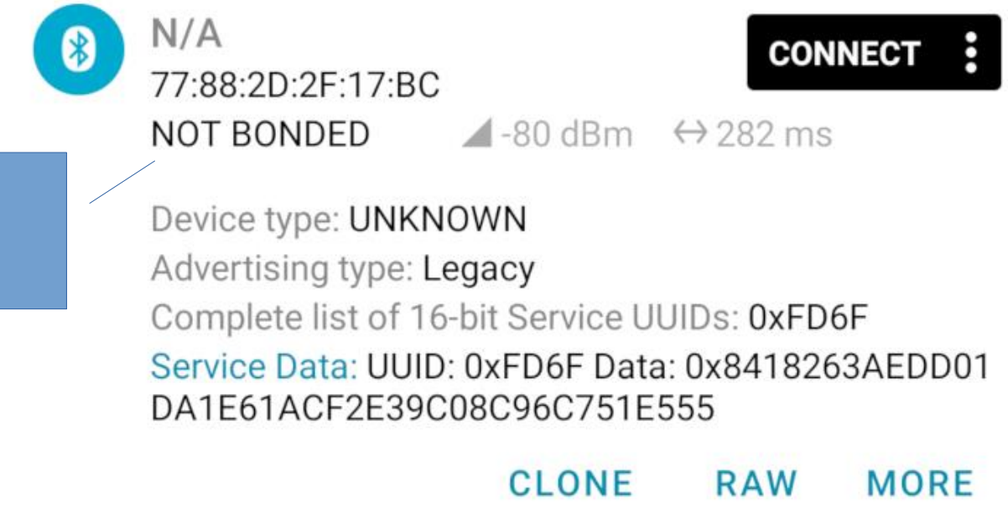




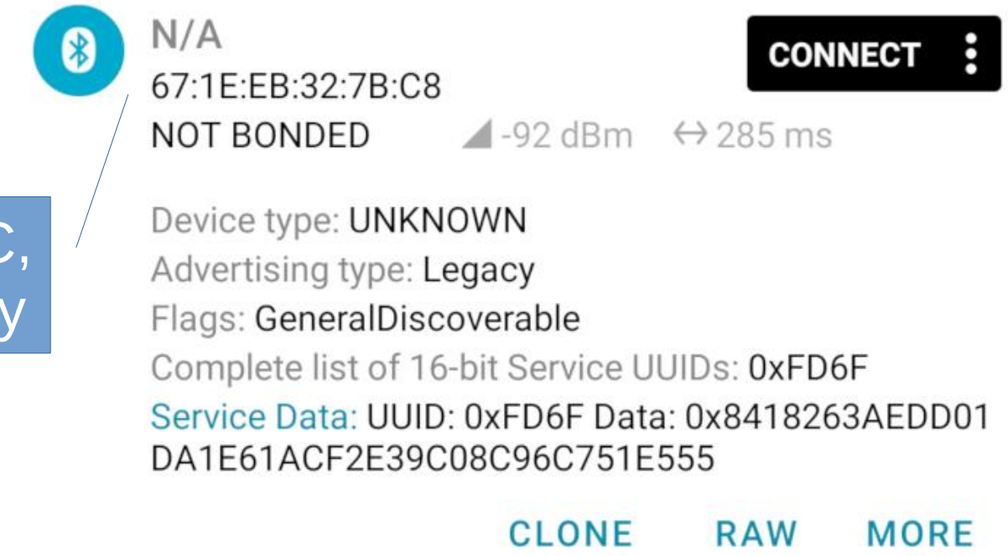
Cloned advertisement



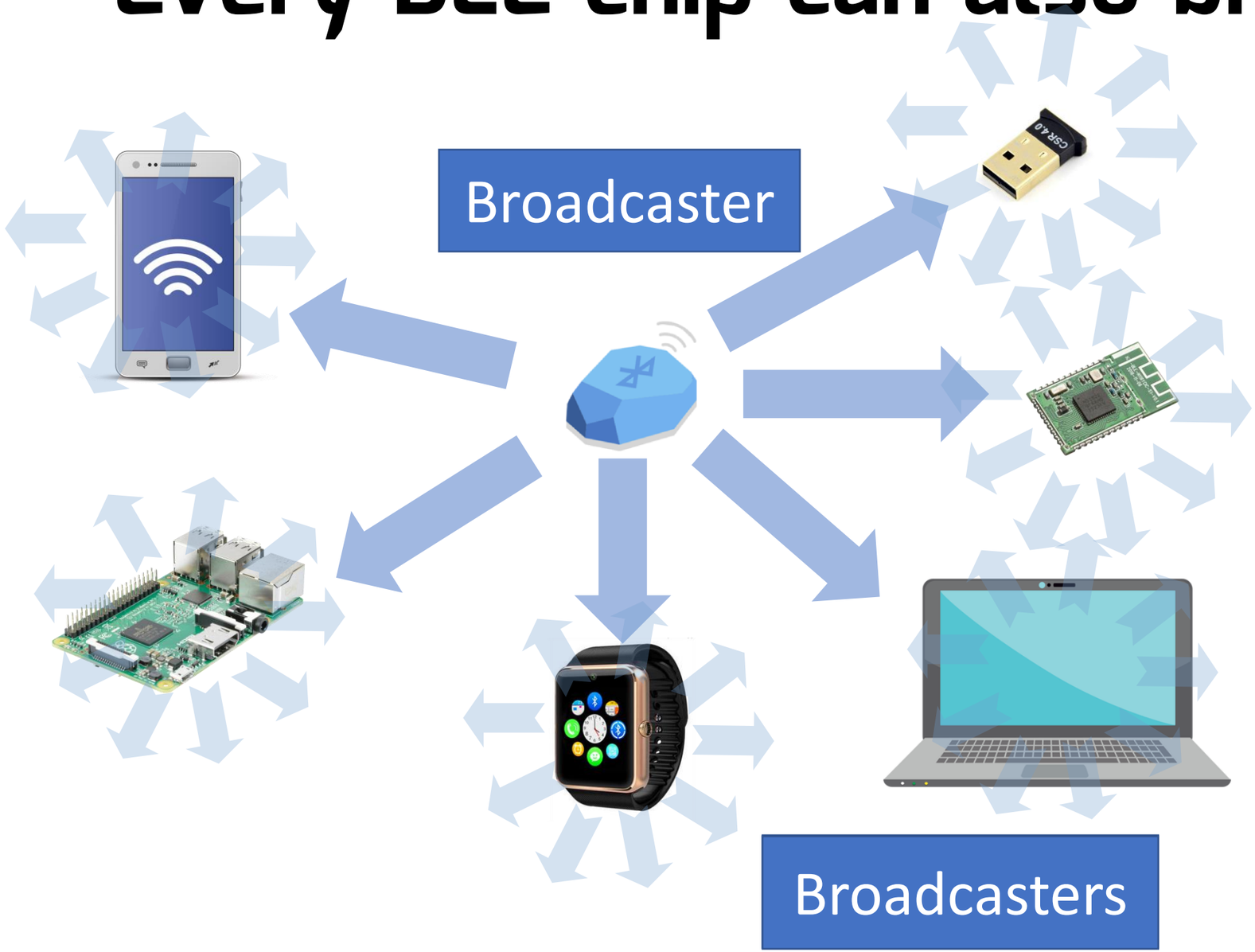
Original



Clone: different MAC, not checked anyway



Every BLE chip can also broadcast



Advertise from console (BlueZ)

- `# hcitool cmd 0x08 0x0008 1F 02 01 1A 03 03 6F FD 17 16 6F FD 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 01 02 00 00`

For the LE Controller Commands, the OGF code is defined as 0x08.

7.8.7 LE Set Advertising Data Command

Command	OCF	Command parameters	Return Parameters
HCI_LE_Set_Advertising_Data	0x0008	Advertising_Data_Length, Advertising_Data	Status

Specification of the Bluetooth System v4.2, page 1282

Advertise from console (BlueZ)

- `# hcitool cmd 0x08 0x0008 1F 02 01 1A 03 03 6F FD 17 16 6F FD 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 01 02 00`

Flags			Complete 16-bit Service UUID			Service Data - 16 bit UUID				
Length	Type	Flags	Length	Type	Service UUID	Length	Type	Service Data		
0x02	0x01 (Flag)	0x1A	0x03	0x03 (Complete 16-bit Service UUID)	0xFD6F (Exposure Notification Service)	0x17	0x16 (Service Data - 16 bit UUID)	0xFD6F (Exposure Notification Service)	**16 bytes** **Rolling Proximity Identifier**	**4 bytes** **Associated Encrypted Metadata**

7.8.7

Com			
HCI_LE_Set_Advertising_Data	0x0008	Advertising_Data_Length, Advertising_Data	Status

Specification of the Bluetooth System v4.2, page 1282

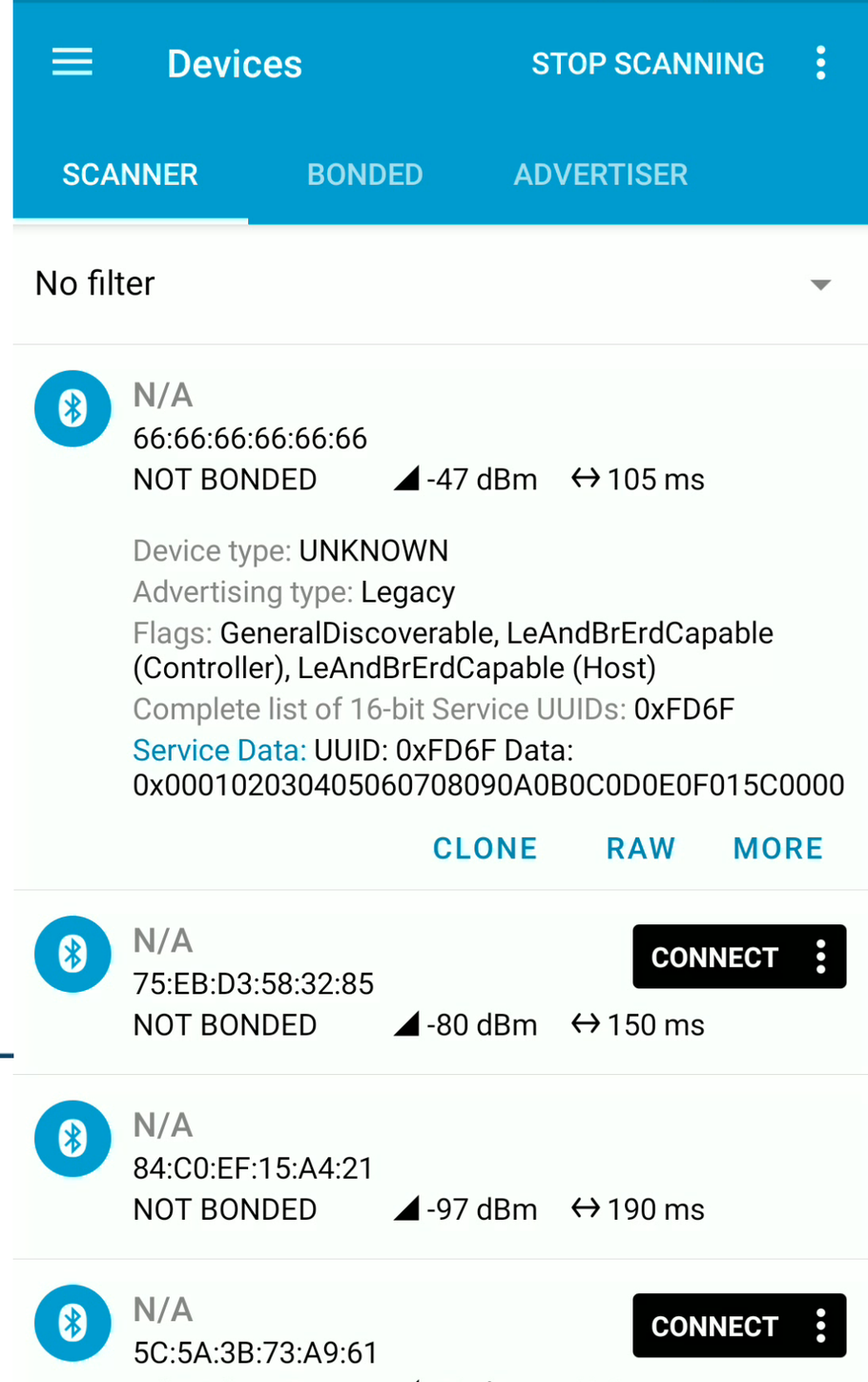
Simple script simulating on Linux

```
# set advertising parameters (100ms)
hcitool -i $HCI cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00 00 00 00 00 00 07 00
COUNT=0;
# send 255 various IDs
while [ $COUNT -lt 255 ]; do
    HEX=`printf '%02X' $COUNT`
    ID="$HEX 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F"
    echo "====== Advertising: $ID ========= "
    #set advertising payload
    hcitool -i $HCI cmd 0x08 0x0008 1F 02 01 1A 03 03 6F FD 17 16 6F FD $ID
$META
    #start advertising
    hcitool -i $HCI cmd 0x08 0x000a 01
    COUNT=$((COUNT+1))
    sleep 1
done
```

https://github.com/smartlockpicking/covid-sim/

Script in action

```
root@kali:~# ./covid
```



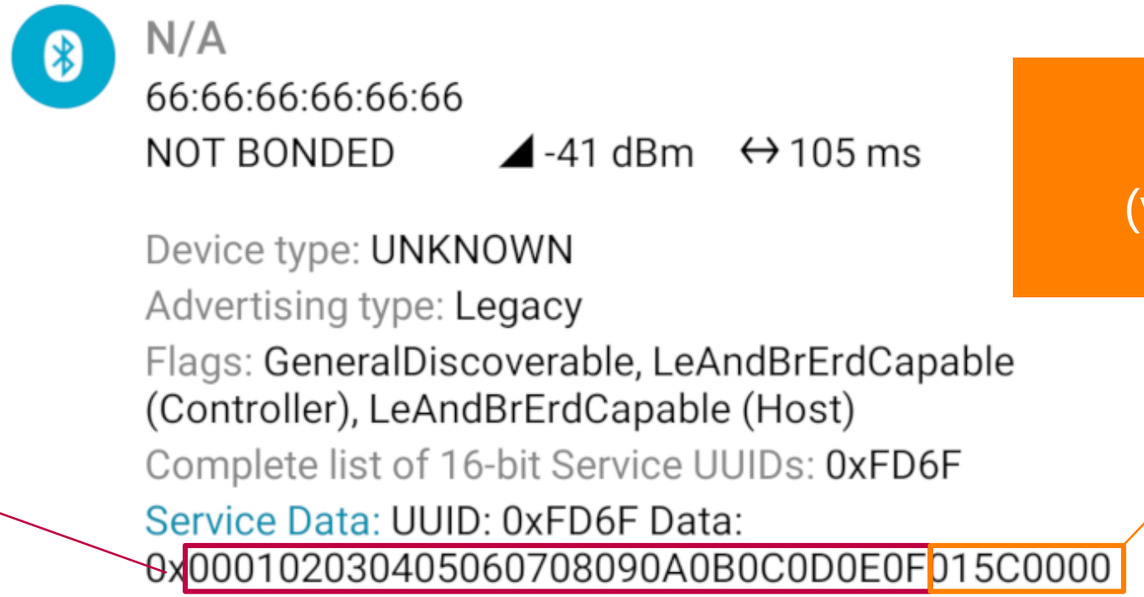
LevelDB database on the phone (GMS)

```
bullhead:/data/data/com.google.android.gms/app_contact-tracing-contact-record-db # ls
000191.ldb 000193.log CURRENT LOCK LOG LOG.old MANIFEST-000192
```

Sample entry in the LevelDB

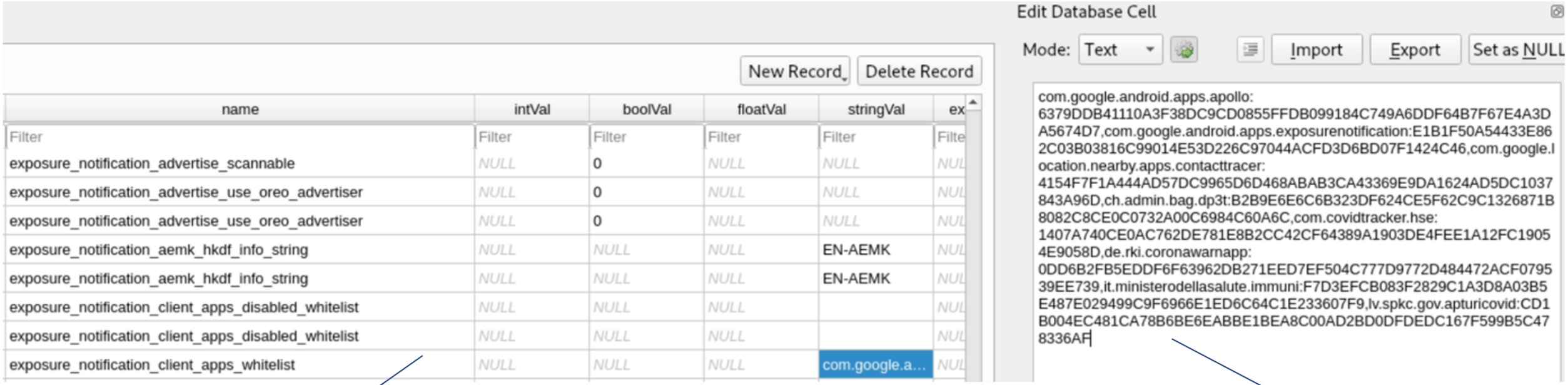
```
Key:b'47fd000102030405060708090a0b0c0d0e0f'
Value:b'
0a1708caa5a2f70518d2ffffffffffffff012204015c0000 0a1708caa5a2f70518d4ffffffffffffff012204015c0000
0a1708caa5a2f70518ceffffffffffffff012204015c0000 0a1708f8a5a2f70518d0ffffffffffffff012204015c0000
0a1708caa5a2f70518d0ffffffffffffff012204015c0000 0a1708f8a5a2f70518d2ffffffffffffff012204015c0000
0a1708caa5a2f70518ceffffffffffffff012204015c0000 0a1708f9a5a2f70518d2ffffffffffffff012204015c0000
0a1708caa5a2f70518cfffffffffffffff012204015c0000 0a1708f9a5a2f70518d4ffffffffffffff012204015c0000
0a1708caa5a2f70518d0ffffffffffffff012204015c0000 0a1708f9a5a2f70518d5ffffffffffffff012204015c0000
0a1708caa5a2f70518d0ffffffffffffff012204015c0000 0a1708f9a5a2f70518d3ffffffffffffff012204015c0000
0a1708caa5a2f70518d2ffffffffffffff012204015c0000 0a1708f9a5a2f70518d1ffffffffffffff012204015c0000
```

16-byte identifier



„Encrypted metadata”
(version, signal strength)

Want to develop tracing app yourself?

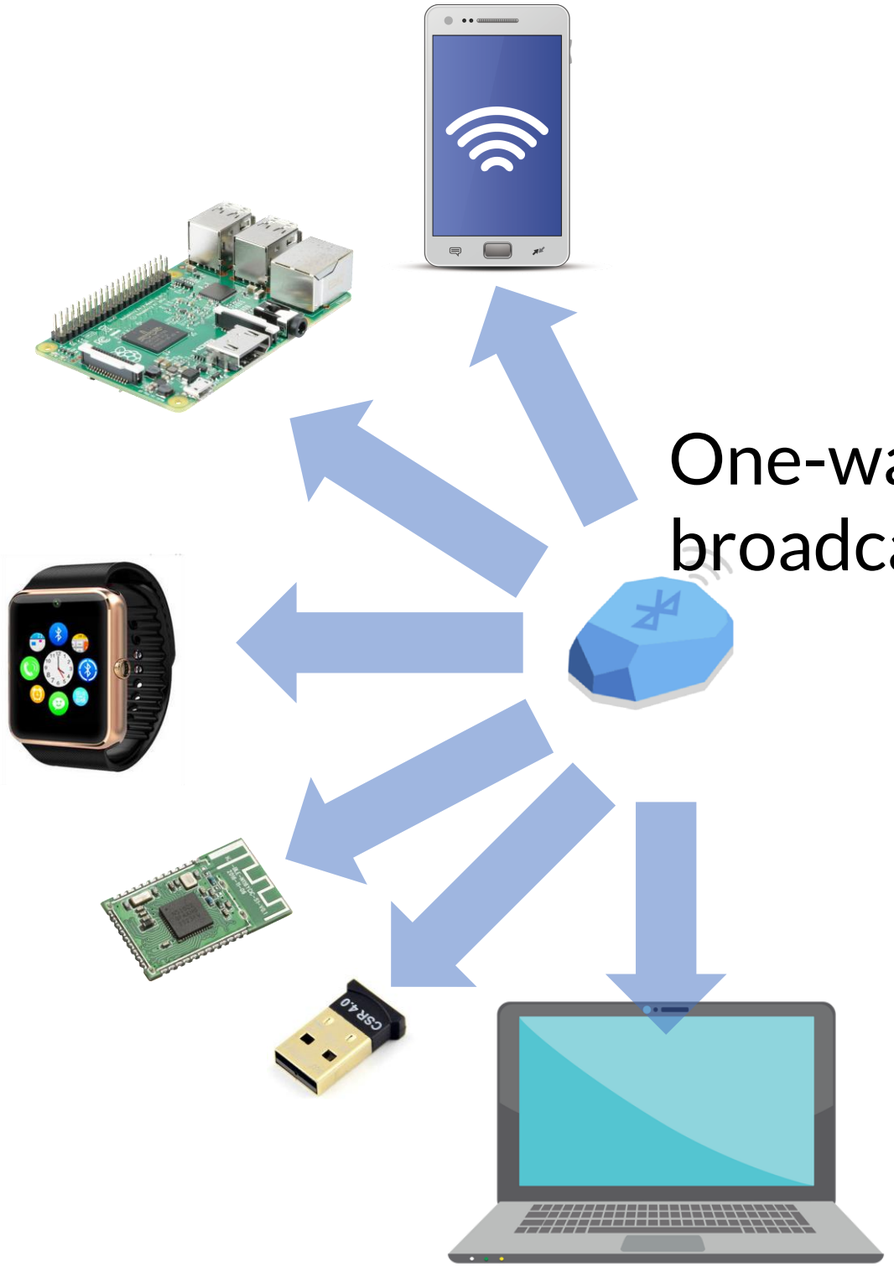


GMS phenotype db

You can add here your app’s key and get access to the API

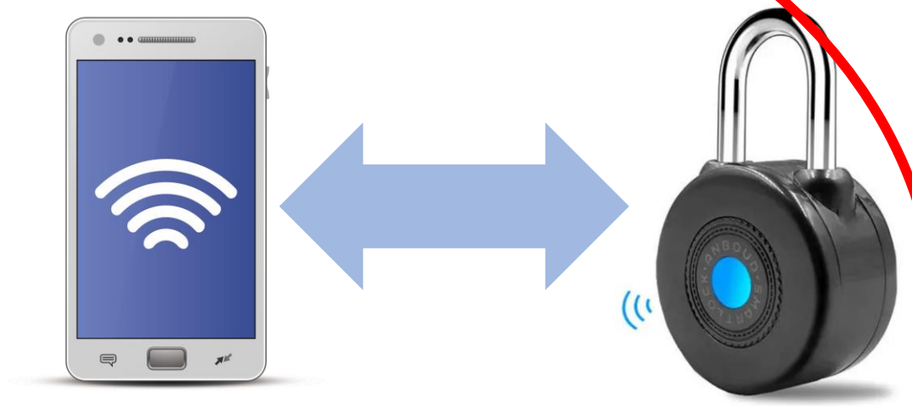
Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

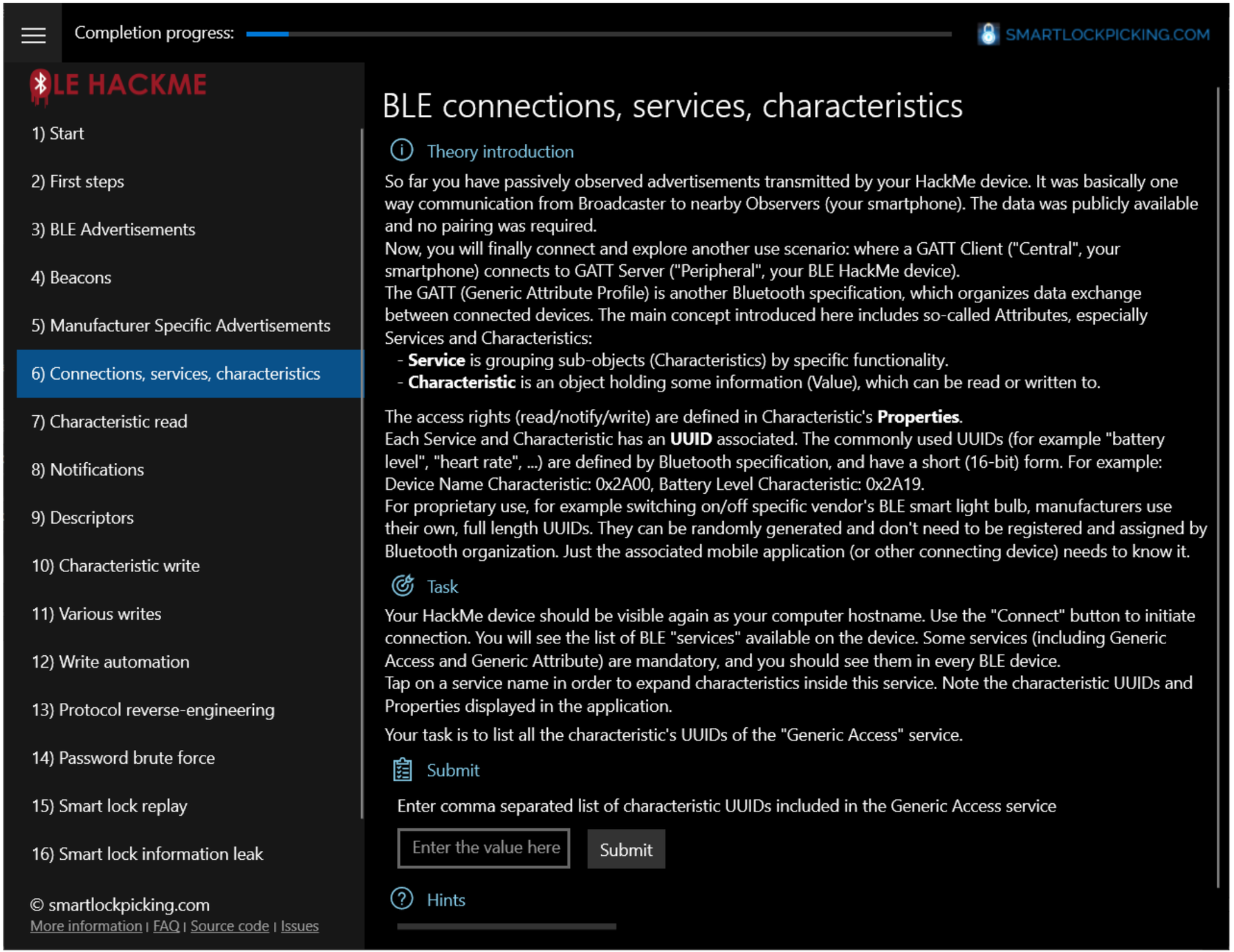


Bluetooth™ 4.0 Low Energy

One-way public broadcast



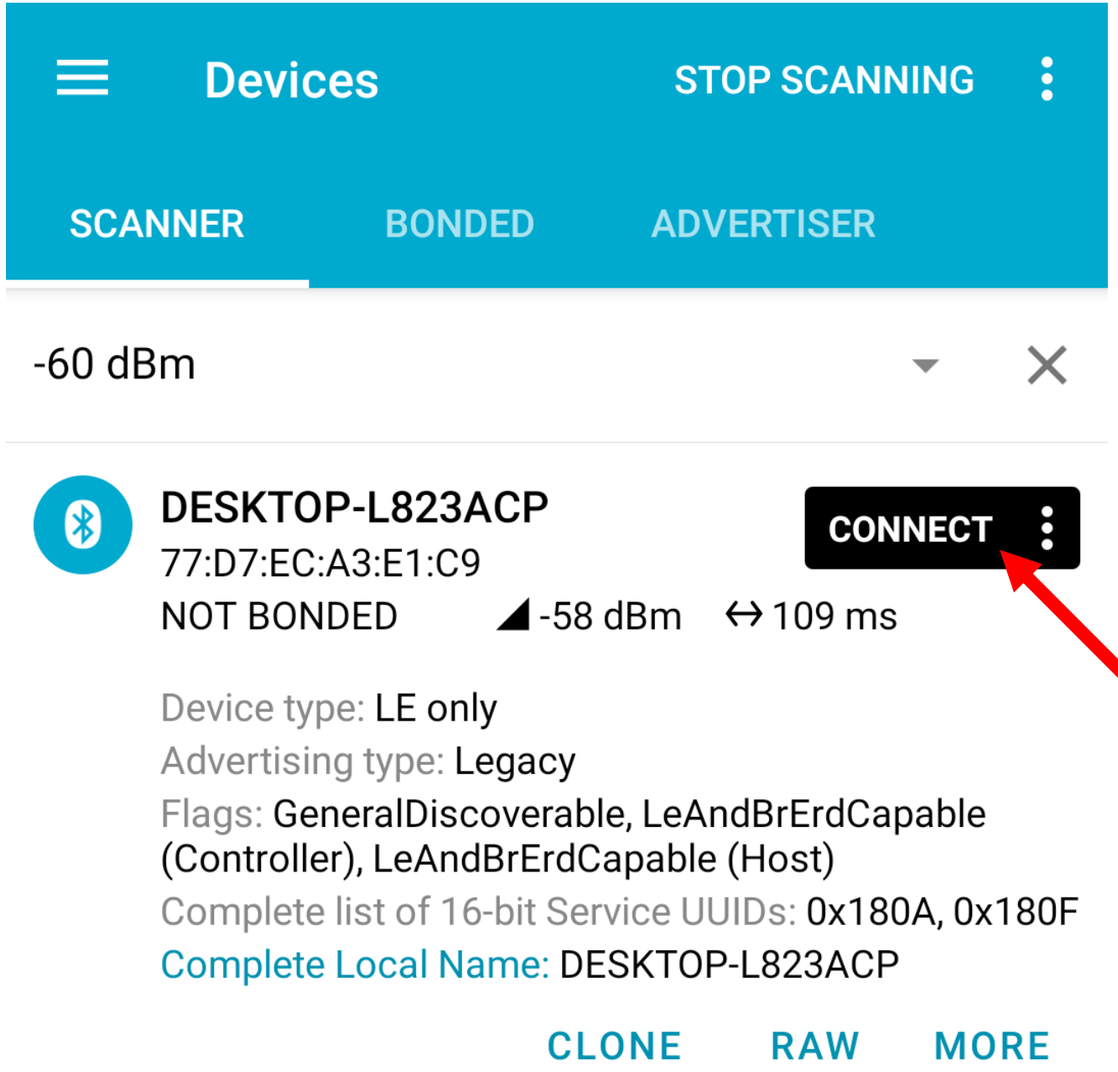
Two-way communication (e.g. app controls smart lock)

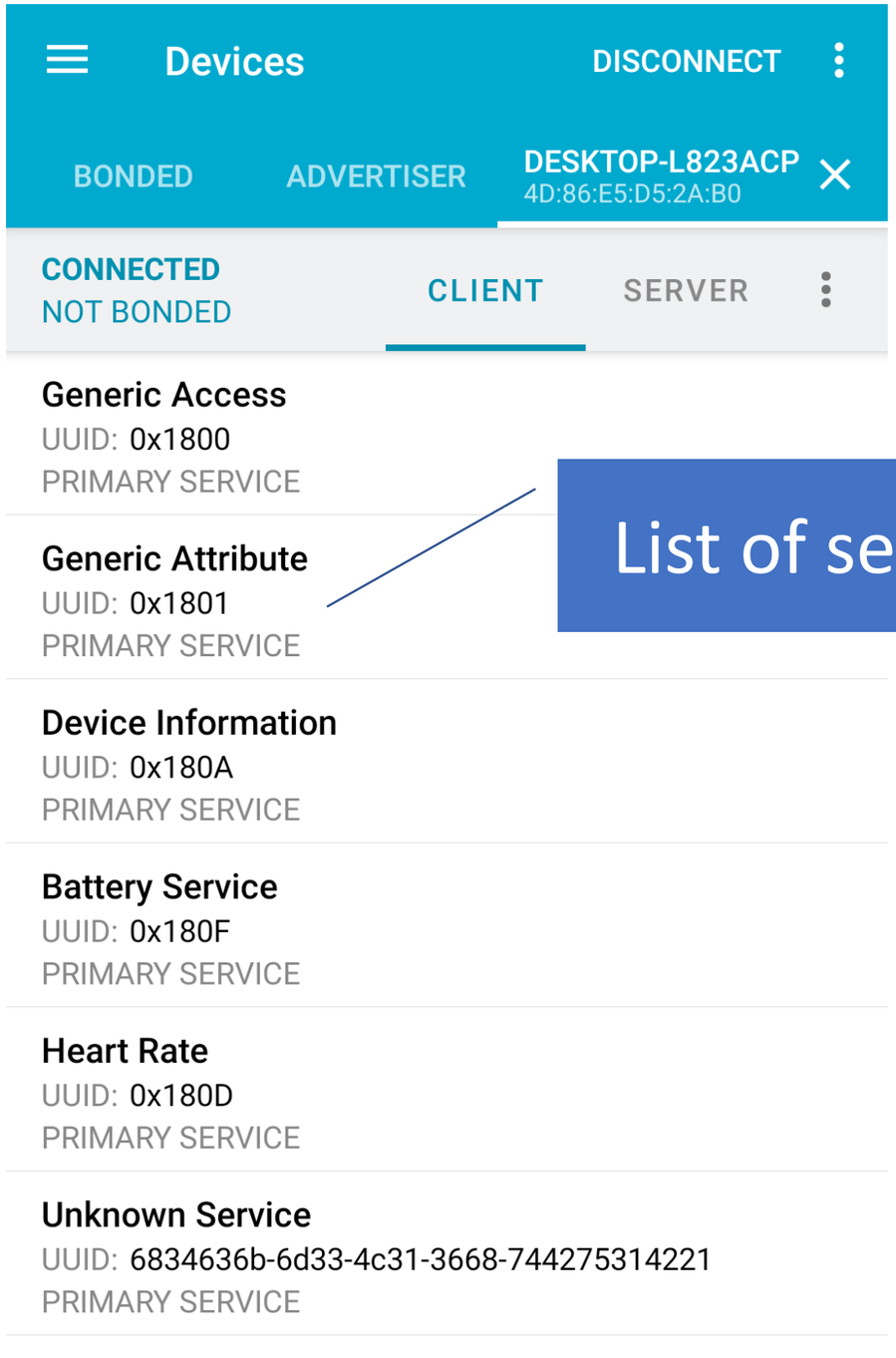

Completion progress:
SMARTLOCKPICKING.COM
BLE HACKME
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak
© smartlockpicking.com
More information | FAQ | Source code | Issues
BLE connections, services, characteristics
Theory introduction
So far you have passively observed advertisements transmitted by your HackMe device. It was basically one way communication from Broadcaster to nearby Observers (your smartphone). The data was publicly available and no pairing was required.
Now, you will finally connect and explore another use scenario: where a GATT Client ("Central", your smartphone) connects to GATT Server ("Peripheral", your BLE HackMe device).
The GATT (Generic Attribute Profile) is another Bluetooth specification, which organizes data exchange between connected devices. The main concept introduced here includes so-called Attributes, especially Services and Characteristics:
- Service is grouping sub-objects (Characteristics) by specific functionality.
- Characteristic is an object holding some information (Value), which can be read or written to.
The access rights (read/notify/write) are defined in Characteristic's Properties.
Each Service and Characteristic has an UUID associated. The commonly used UUIDs (for example "battery level", "heart rate", ...) are defined by Bluetooth specification, and have a short (16-bit) form. For example: Device Name Characteristic: 0x2A00, Battery Level Characteristic: 0x2A19.
For proprietary use, for example switching on/off specific vendor's BLE smart light bulb, manufacturers use their own, full length UUIDs. They can be randomly generated and don't need to be registered and assigned by Bluetooth organization. Just the associated mobile application (or other connecting device) needs to know it.
Task
Your HackMe device should be visible again as your computer hostname. Use the "Connect" button to initiate connection. You will see the list of BLE "services" available on the device. Some services (including Generic Access and Generic Attribute) are mandatory, and you should see them in every BLE device.
Tap on a service name in order to expand characteristics inside this service. Note the characteristic UUIDs and Properties displayed in the application.
Your task is to list all the characteristic's UUIDs of the "Generic Access" service.
Submit
Enter comma separated list of characteristic UUIDs included in the Generic Access service
Enter the value here
Submit
Hints

GATT: services and characteristics

- **G**eneric **ATT**ribute Profile.
- Attributes are: Services, Characteristics, Descriptors.
- Identified by **UUID** – short (registered), long – proprietary.
- A **Service** is grouping sub-objects (Characteristics).
- A **Characteristic** holds a single **Value**.
- For example: Battery Level Service has Battery Level Characteristic which holds Battery Level Value.
- You will feel it much better in practice!

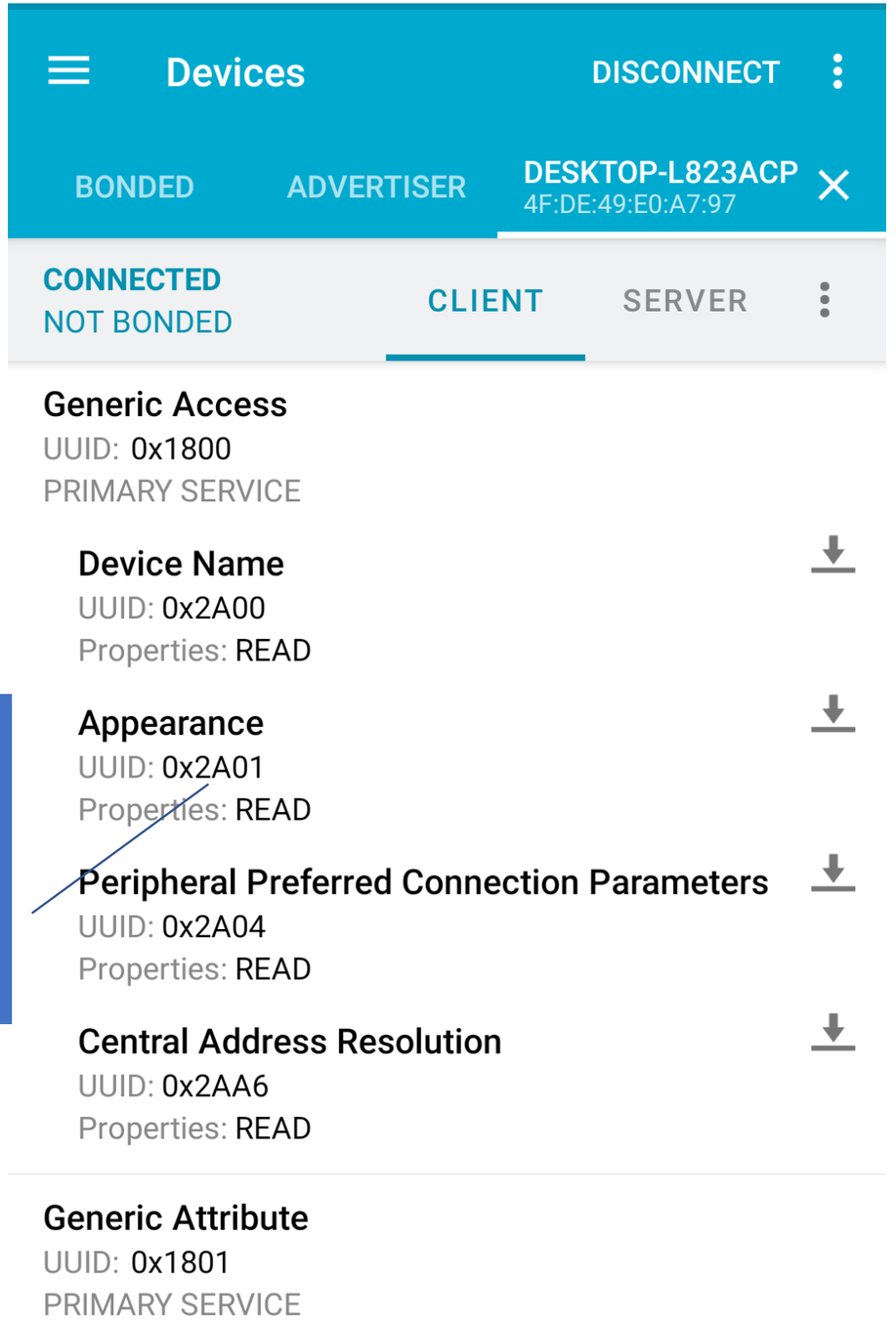
nRF Connect: CONNECT

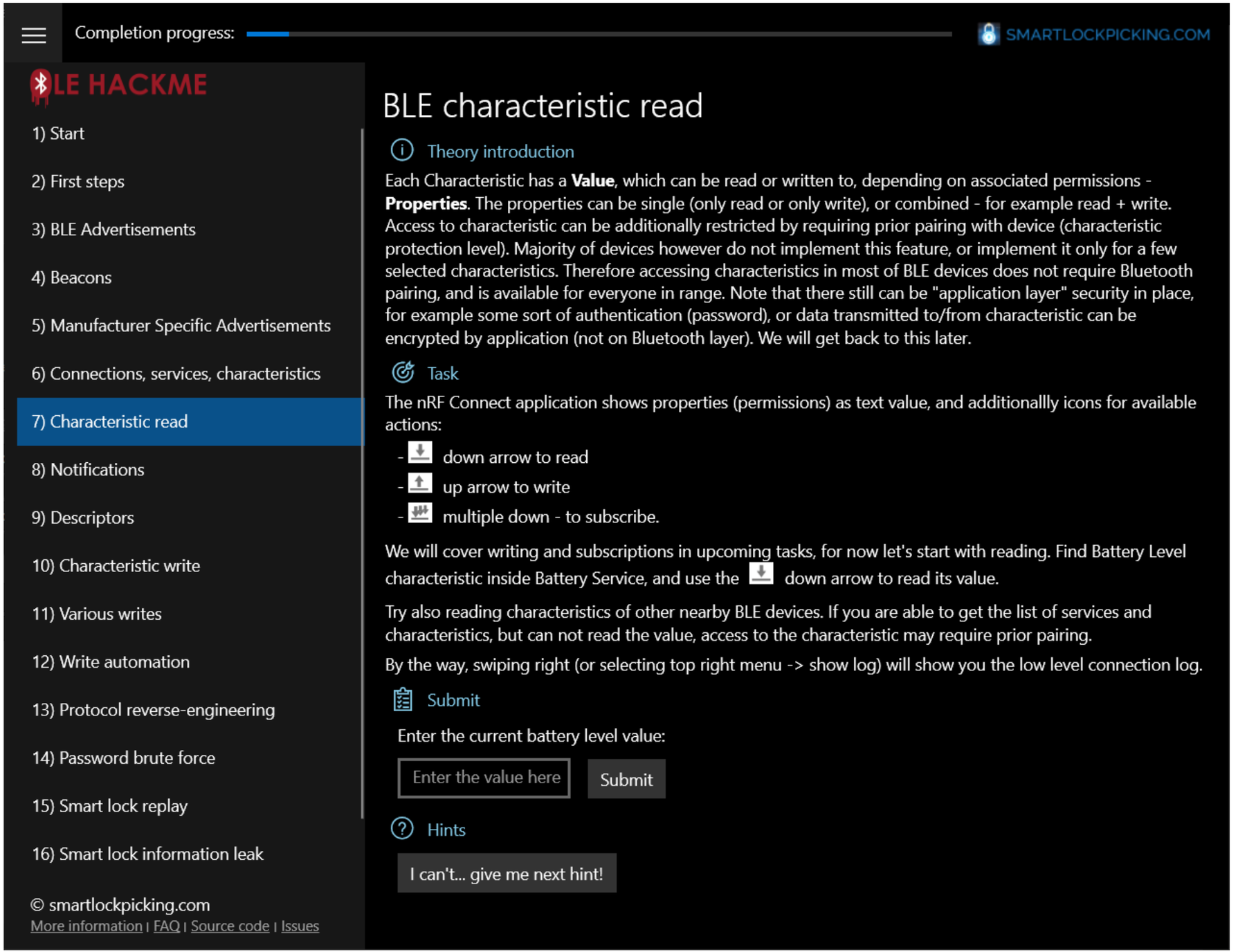




List of services

List of characteristics in the service




Completion progress:
SMARTLOCKPICKING.COM
BLE HACKME
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak
© smartlockpicking.com
More information | FAQ | Source code | Issues
BLE characteristic read
Theory introduction
Each Characteristic has a Value, which can be read or written to, depending on associated permissions - Properties. The properties can be single (only read or only write), or combined - for example read + write. Access to characteristic can be additionally restricted by requiring prior pairing with device (characteristic protection level). Majority of devices however do not implement this feature, or implement it only for a few selected characteristics. Therefore accessing characteristics in most of BLE devices does not require Bluetooth pairing, and is available for everyone in range. Note that there still can be "application layer" security in place, for example some sort of authentication (password), or data transmitted to/from characteristic can be encrypted by application (not on Bluetooth layer). We will get back to this later.
Task
The nRF Connect application shows properties (permissions) as text value, and additionallly icons for available actions:
- down arrow to read
- up arrow to write
- multiple down - to subscribe.
We will cover writing and subscriptions in upcoming tasks, for now let's start with reading. Find Battery Level characteristic inside Battery Service, and use the down arrow to read its value.
Try also reading characteristics of other nearby BLE devices. If you are able to get the list of services and characteristics, but can not read the value, access to the characteristic may require prior pairing.
By the way, swiping right (or selecting top right menu -> show log) will show you the low level connection log.
Submit
Enter the current battery level value:
Enter the value here
Submit
Hints
I can't... give me next hint!

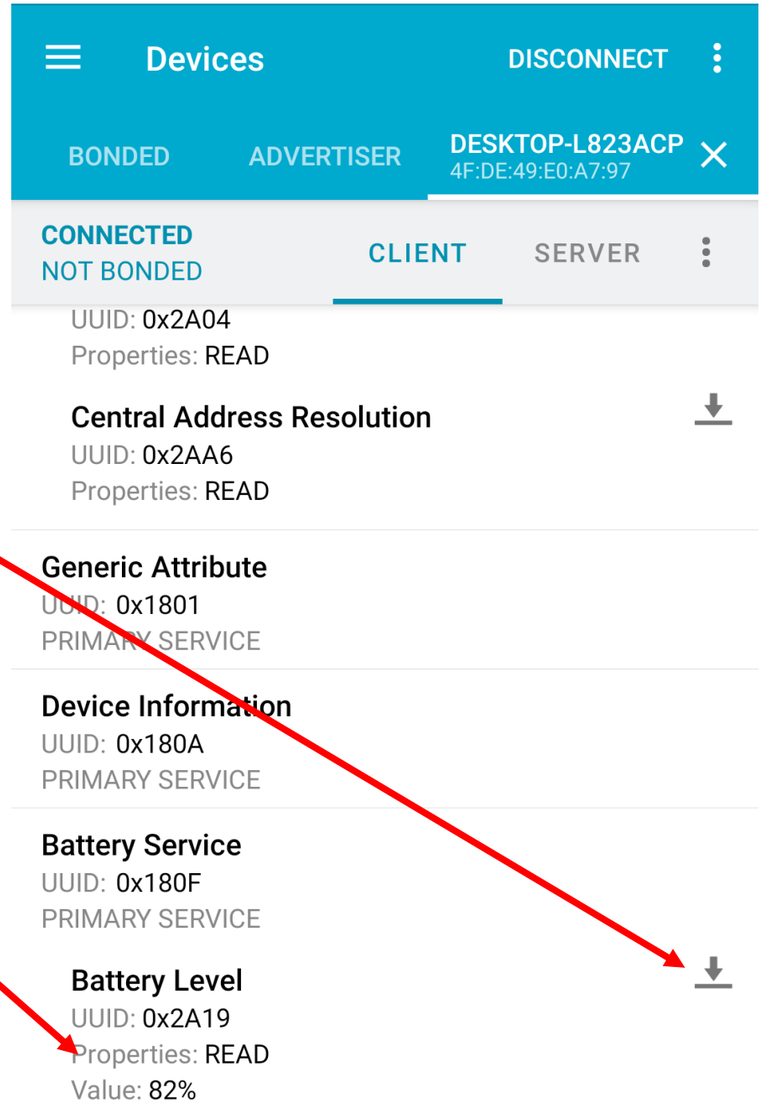
Characteristic properties

Read

Notify

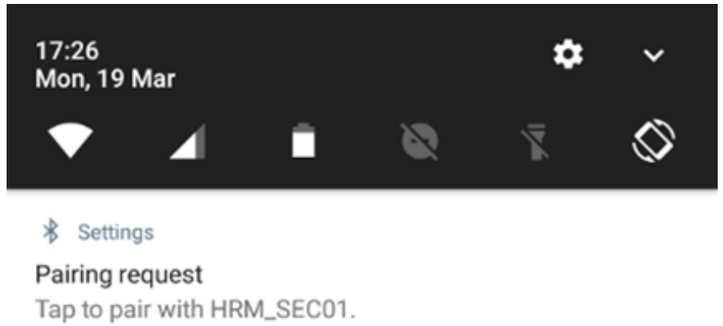
Write

Can be combined

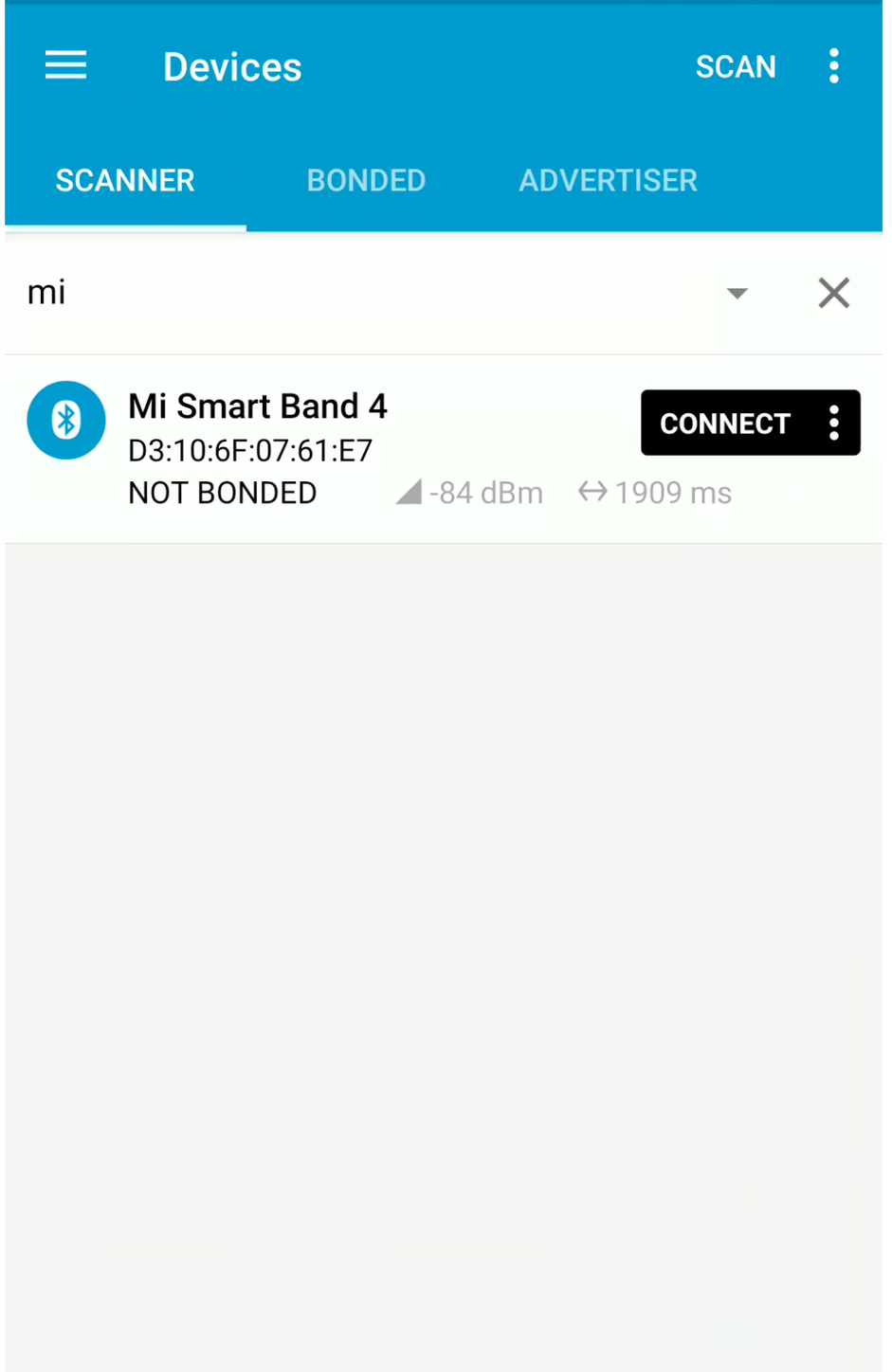


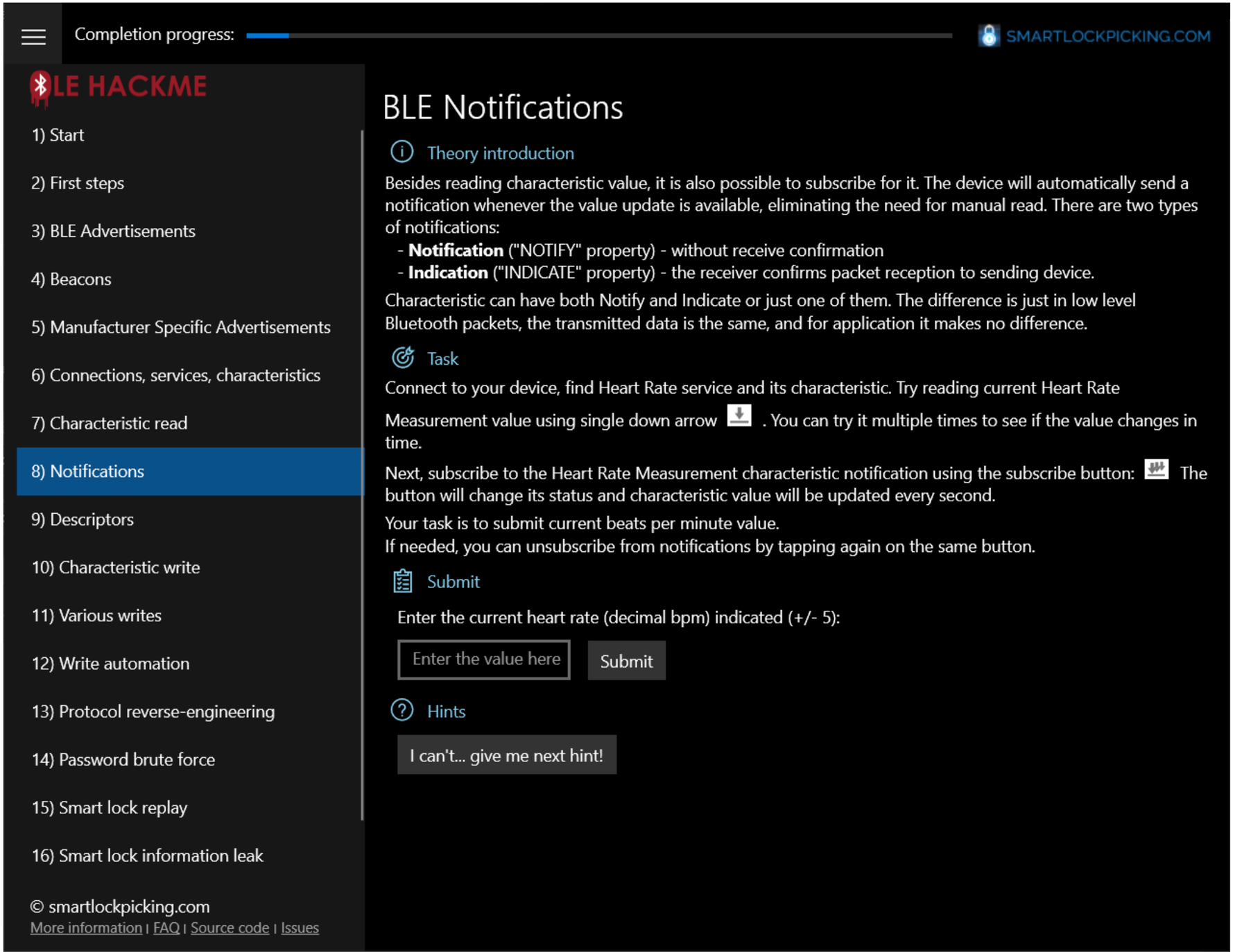
Wait, what about pairing?

- We just read data from the device, but no pairing was required?!
- Pairing (/bonding) in BLE is possible, but optional.
- Majority of devices do not implement it!
- … or secure just selected characteristics.
- In such case you will notice:
  - stall/disconnect (device requires switch to pairing mode)
  - pairing request (device allows for pairing with anyone)



Popular sport band




Completion progress:
SMARTLOCKPICKING.COM
BLE HACKME
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak
© smartlockpicking.com
More information | FAQ | Source code | Issues
BLE Notifications
Theory introduction
Besides reading characteristic value, it is also possible to subscribe for it. The device will automatically send a notification whenever the value update is available, eliminating the need for manual read. There are two types of notifications:
- Notification ("NOTIFY" property) - without receive confirmation
- Indication ("INDICATE" property) - the receiver confirms packet reception to sending device.
Characteristic can have both Notify and Indicate or just one of them. The difference is just in low level Bluetooth packets, the transmitted data is the same, and for application it makes no difference.
Task
Connect to your device, find Heart Rate service and its characteristic. Try reading current Heart Rate
Measurement value using single down arrow . You can try it multiple times to see if the value changes in time.
Next, subscribe to the Heart Rate Measurement characteristic notification using the subscribe button: The button will change its status and characteristic value will be updated every second.
Your task is to submit current beats per minute value.
If needed, you can unsubscribe from notifications by tapping again on the same button.
Submit
Enter the current heart rate (decimal bpm) indicated (+/- 5):
Enter the value here
Submit
Hints
I can't... give me next hint!

Subscribe to notifications

Tap to subscribe for value change

Heart Rate
UUID: 0x180D
PRIMARY SERVICE

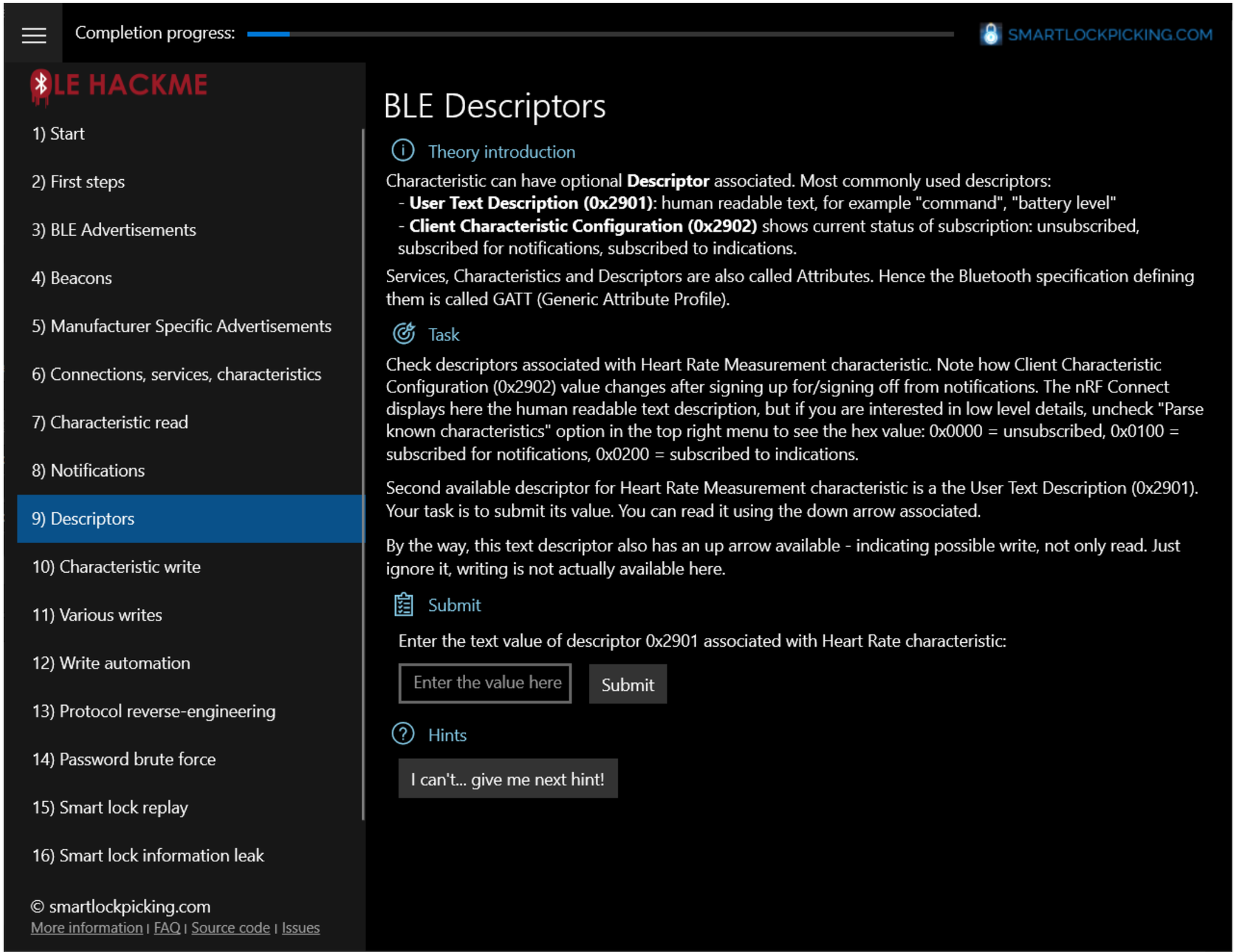
Heart Rate Measurement
UUID: 0x2A37
Properties: NOTIFY, READ
Descriptors:
Characteristic User Description
UUID: 0x2901
Value: Beats per minute 8690
Client Characteristic Configuration
UUID: 0x2902

Value updates automatically

Heart Rate
UUID: 0x180D
PRIMARY SERVICE

Heart Rate Measurement
UUID: 0x2A37
Properties: NOTIFY, READ
Value: Heart Rate Measurement: 123 bpm, Contact is Detected
Descriptors:
Characteristic User Description
UUID: 0x2901
Client Characteristic Configuration
UUID: 0x2902
Value: Notifications enabled

Notifications enabled


Completion progress:
SMARTLOCKPICKING.COM
BLE HACKME
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak
© smartlockpicking.com
More information | FAQ | Source code | Issues
BLE Descriptors
Theory introduction
Characteristic can have optional Descriptor associated. Most commonly used descriptors:
- User Text Description (0x2901): human readable text, for example "command", "battery level"
- Client Characteristic Configuration (0x2902) shows current status of subscription: unsubscribed, subscribed for notifications, subscribed to indications.
Services, Characteristics and Descriptors are also called Attributes. Hence the Bluetooth specification defining them is called GATT (Generic Attribute Profile).
Task
Check descriptors associated with Heart Rate Measurement characteristic. Note how Client Characteristic Configuration (0x2902) value changes after signing up for/signing off from notifications. The nRF Connect displays here the human readable text description, but if you are interested in low level details, uncheck "Parse known characteristics" option in the top right menu to see the hex value: 0x0000 = unsubscribed, 0x0100 = subscribed for notifications, 0x0200 = subscribed to indications.
Second available descriptor for Heart Rate Measurement characteristic is a the User Text Description (0x2901). Your task is to submit its value. You can read it using the down arrow associated.
By the way, this text descriptor also has an up arrow available - indicating possible write, not only read. Just ignore it, writing is not actually available here.
Submit
Enter the text value of descriptor 0x2901 associated with Heart Rate characteristic:
Enter the value here
Submit
Hints
I can't... give me next hint!

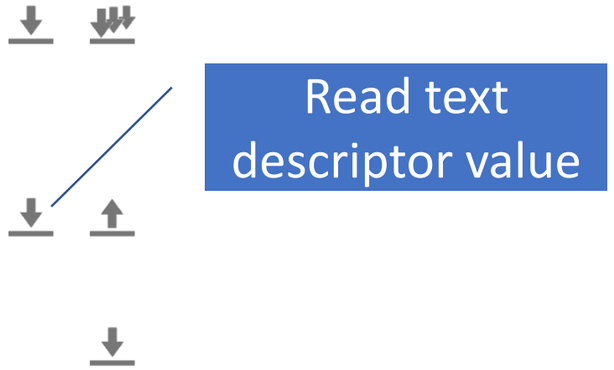
Characteristic's Descriptors

Most commonly used:

- **User Text Descriptor 0x2901** – optional human readable text
- **Client Characteristic Configuration Descriptor 0x2902** – current status of subscription to notifications

Heart Rate
UUID: 0x180D
PRIMARY SERVICE

Heart Rate Measurement
UUID: 0x2A37
Properties: NOTIFY, READ
Descriptors:
Characteristic User Description
UUID: 0x2901
Value: Beats per minute 8690
Client Characteristic Configuration
UUID: 0x2902



Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

Characteristic Write

Theory introduction

So far you have learned how to receive data from device by reading and subcribing for characteristic value. Characteristic may also have "write" property, which allows for submitting a value to device.

Task

Using your new skills in regards to services, characteristics and descriptors, connect to your HackMe device, find proprietary service responsible for light bulb control, and a characteristic inside it that allows to switch the light on and off. Read the current state of the switch, and try to turn it on.

In nRF Connect use the up arrow by the characteristic to write a value to it. Once succeeded, you can turn the light off again if you like.

Simulated device

Hints

I can't... give me next hint!

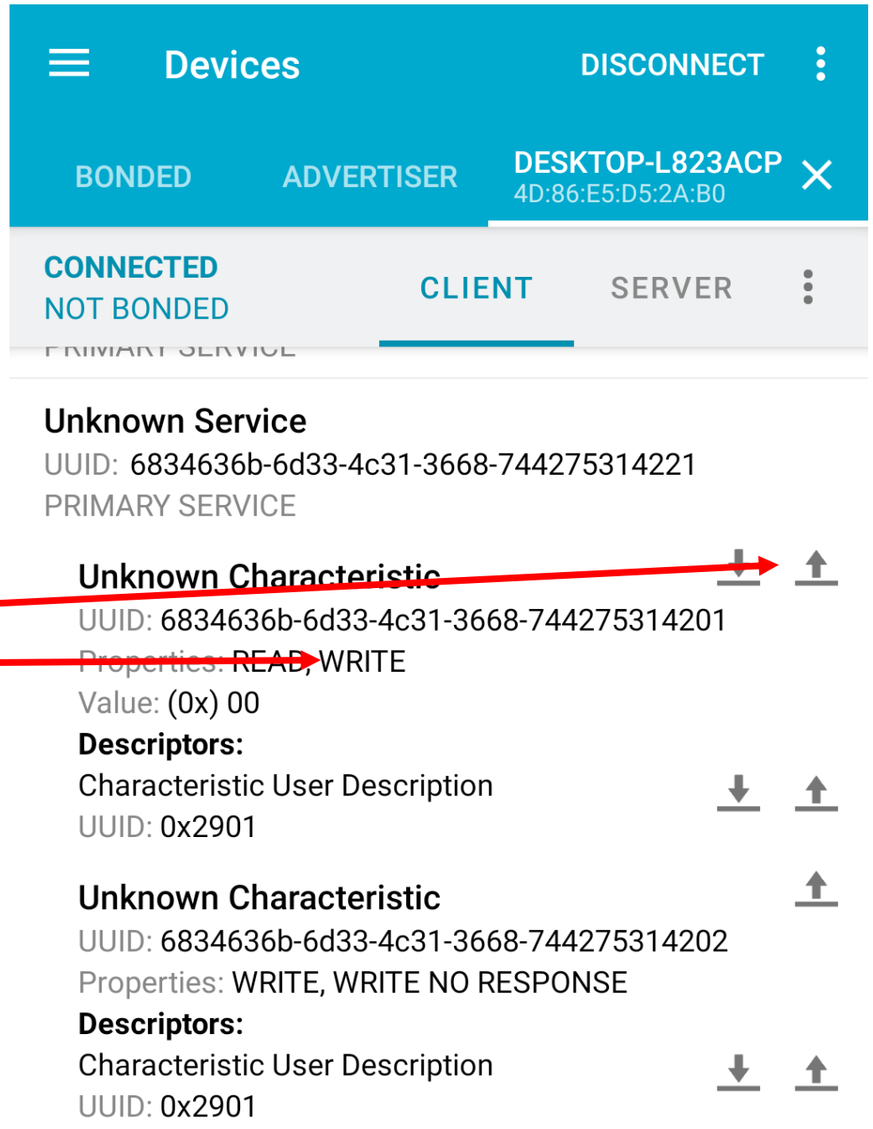
Write

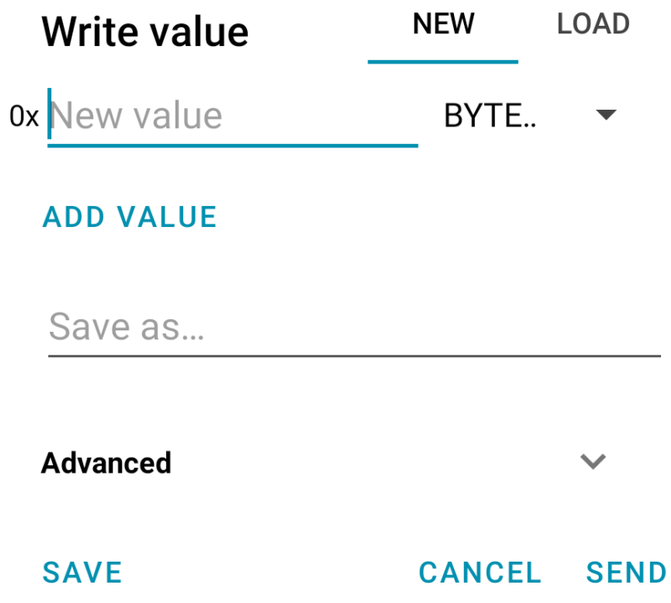
Read 

Notify 

Write 

Can be combined





Completion progress: SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

Characteristic Write

Theory introduction

So far you have learned how to receive data from device by reading and subcribing to characteristic value. Characteristics may also have a "write" property, which allows for submitting a value to device.

Task

Using your new skills in regards to services, characteristics and descriptors, connect to your HackMe device, find proprietary service responsible for light bulb control, and a characteristic inside it that allows to switch the light on and off. Read the current state of the switch, and try to turn it on.

In nRF Connect use the up arrow by the characteristic to write a value to it. Once succeeded, you can turn the light off again if you like.

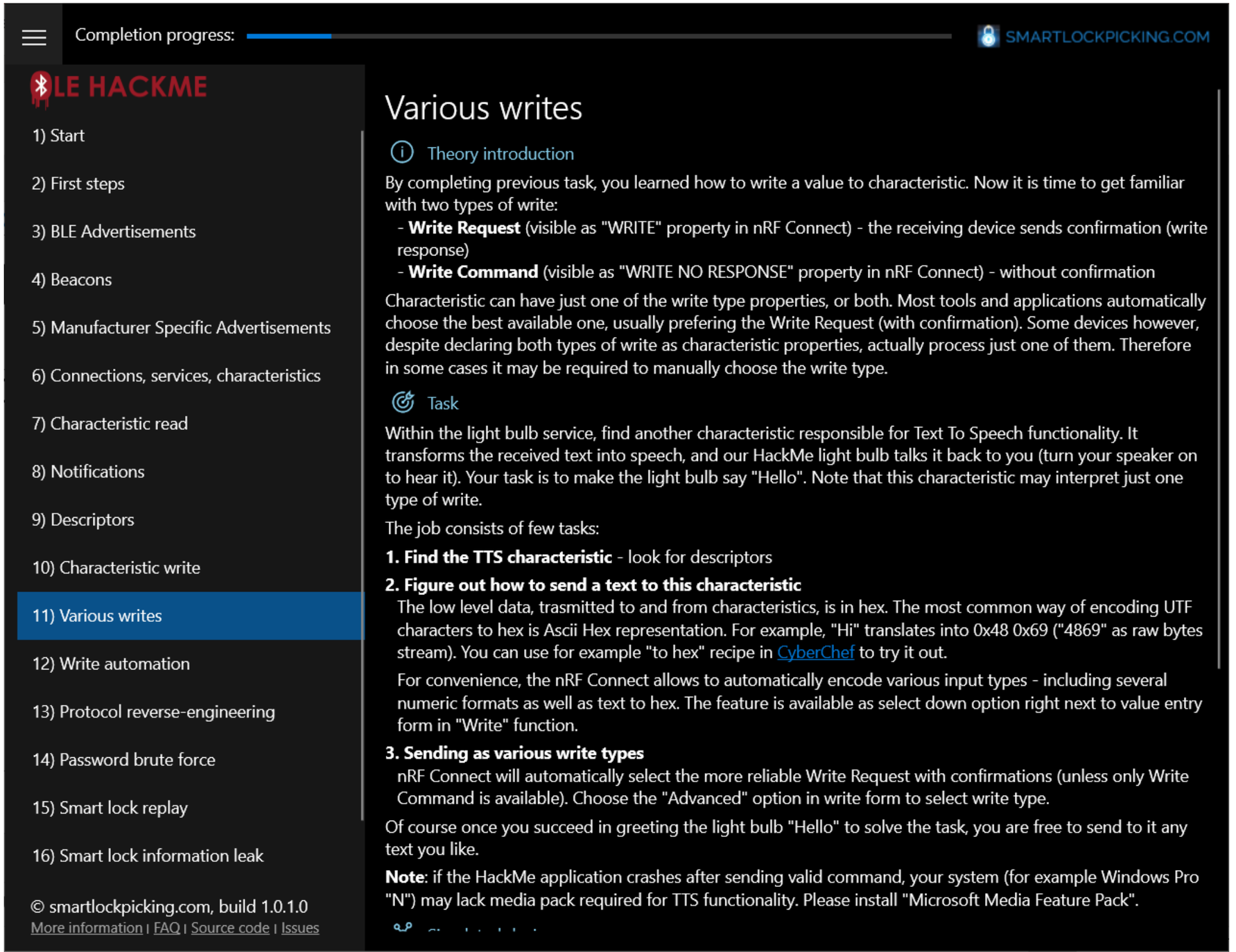
Simulated device

Hints

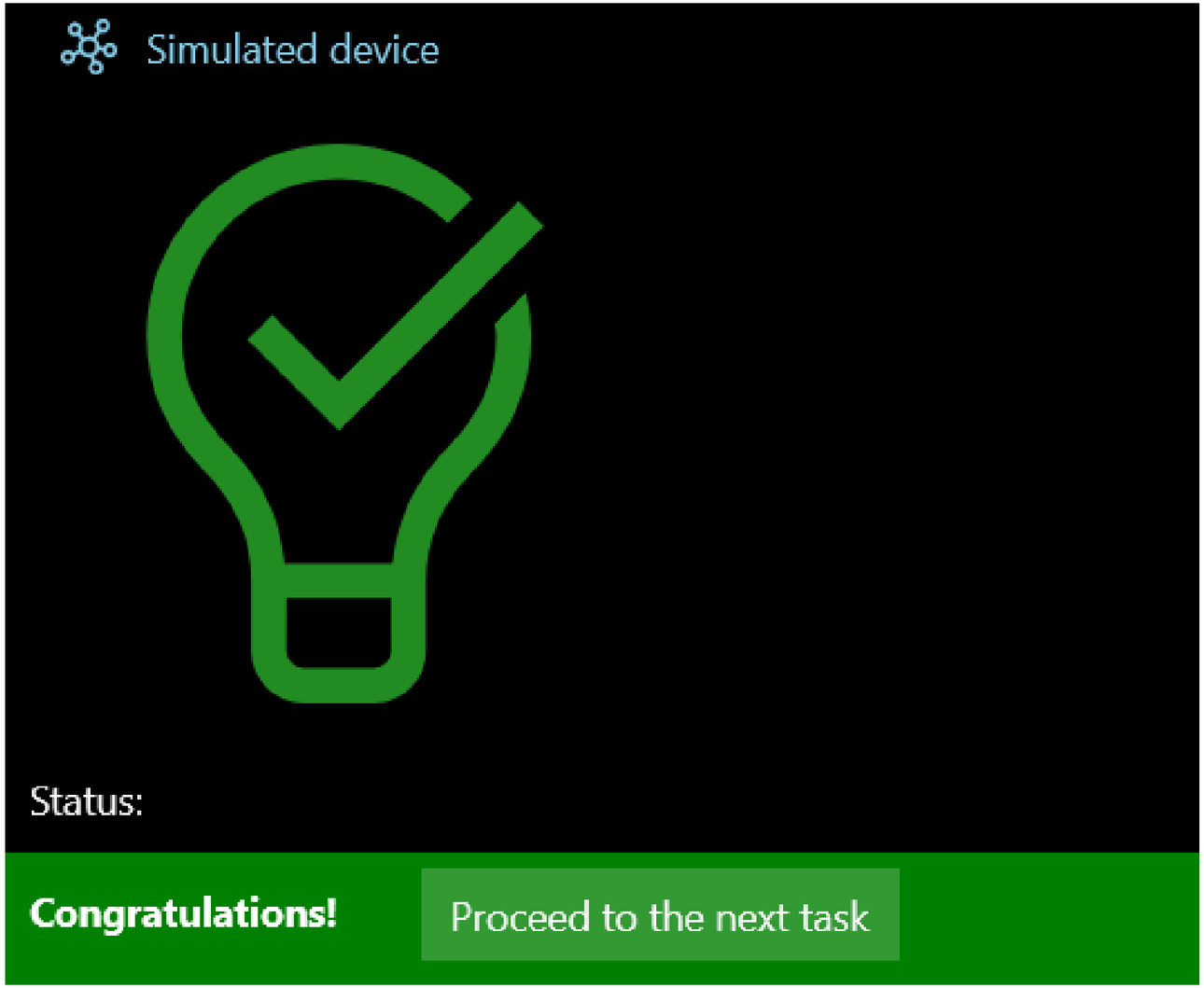
I can't... give me next hint!

Status:

Congratulations! Proceed to the next task


Completion progress:
SMARTLOCKPICKING.COM
BLE HACKME
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak
© smartlockpicking.com, build 1.0.1.0
More information | FAQ | Source code | Issues
Various writes
Theory introduction
By completing previous task, you learned how to write a value to characteristic. Now it is time to get familiar with two types of write:
- Write Request (visible as "WRITE" property in nRF Connect) - the receiving device sends confirmation (write response)
- Write Command (visible as "WRITE NO RESPONSE" property in nRF Connect) - without confirmation
Characteristic can have just one of the write type properties, or both. Most tools and applications automatically choose the best available one, usually prefering the Write Request (with confirmation). Some devices however, despite declaring both types of write as characteristic properties, actually process just one of them. Therefore in some cases it may be required to manually choose the write type.
Task
Within the light bulb service, find another characteristic responsible for Text To Speech functionality. It transforms the received text into speech, and our HackMe light bulb talks it back to you (turn your speaker on to hear it). Your task is to make the light bulb say "Hello". Note that this characteristic may interpret just one type of write.
The job consists of few tasks:
1. Find the TTS characteristic - look for descriptors
2. Figure out how to send a text to this characteristic
The low level data, trasmitted to and from characteristics, is in hex. The most common way of encoding UTF characters to hex is Ascii Hex representation. For example, "Hi" translates into 0x48 0x69 ("4869" as raw bytes stream). You can use for example "to hex" recipe in CyberChef to try it out.
For convenience, the nRF Connect allows to automatically encode various input types - including several numeric formats as well as text to hex. The feature is available as select down option right next to value entry form in "Write" function.
3. Sending as various write types
nRF Connect will automatically select the more reliable Write Request with confirmations (unless only Write Command is available). Choose the "Advanced" option in write form to select write type.
Of course once you succeed in greeting the light bulb "Hello" to solve the task, you are free to send to it any text you like.
Note: if the HackMe application crashes after sending valid command, your system (for example Windows Pro "N") may lack media pack required for TTS functionality. Please install "Microsoft Media Feature Pack".

Talking BLE smart light bulb!



Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

Task

The light bulb has yet another characteristic, which allows to change its color and brightness level. You will surely find it in the light bulb service. The valid data format to send via write is however unknown. Fortunately, there was a mobile application possible to decompile. The decompiled source code snippet responsible for sending valid request follows:

```
public static final byte ARGB_FRAME_PREFIX = (byte) -86;
public static final byte FRAME_SUFFIX = (byte) -1;

public bool a(int i) {
  byte alpha = (byte) Color.alpha(i);
  byte red = (byte) Color.red(i);
  byte green = (byte) Color.green(i);
  byte blue = (byte) Color.blue(i);
  byte[] bArr = new byte[]{ARGB_FRAME_PREFIX, alpha, red, green, blue,
  FRAME_SUFFIX};
  return this.c.e.b(bArr);
}
```

Your task is to analyse the decompiled source code, and based on it create a valid request to light bulb RGB characteristic - setting it to half-dim pure red.

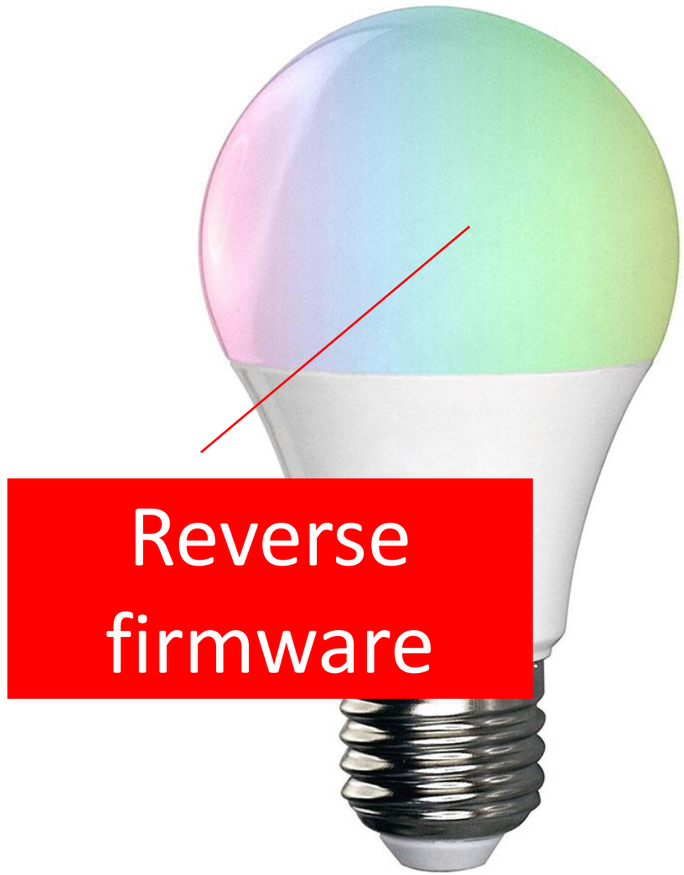
Of course you can then set any color and brightness level you like. Maybe even record a macro to change the colors?

Simulated device

Hints

I can't... give me next hint!

Data format?



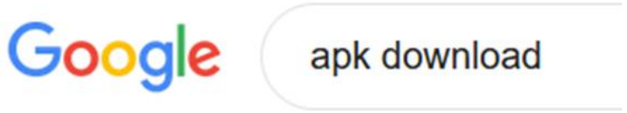
Reverse firmware

Intercept

Reverse mobile app

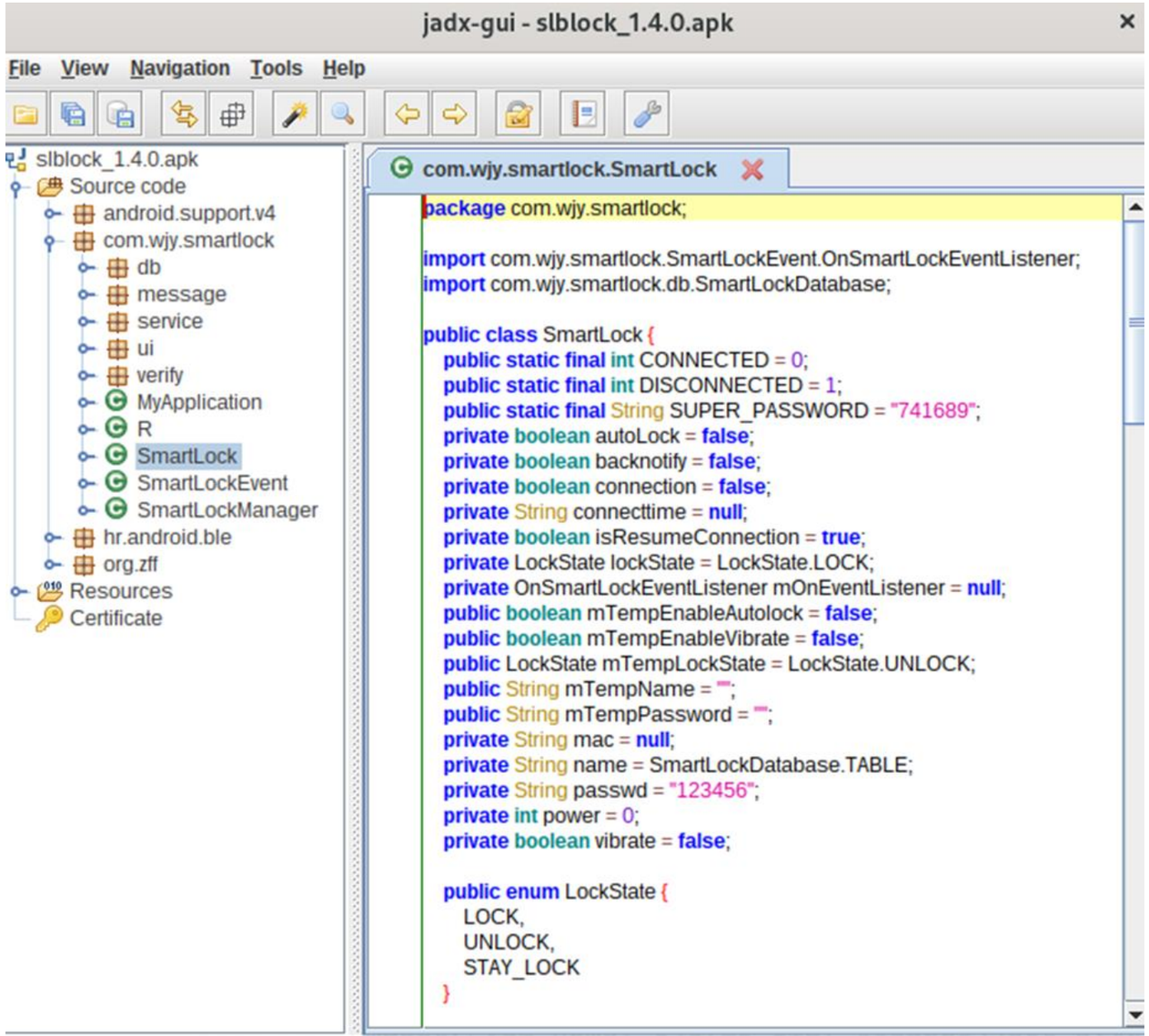
Mobile app reversing?

- Grab the “apk” binary



- Decompile
  - JADX
    https://github.com/skylot/jadx
  - BytecodeViewer
    https://github.com/Konloch/bytecode-viewer

  - Many others…



Completion progress: SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

Task

The light bulb has yet another characteristic, which allows to change its color and brightness level. You will surely find it in the light bulb service. The valid data format to send via write is however unknown.
Fortunately, there was a mobile application possible to decompile. The decompiled source code snippet responsible for sending valid request follows:

```
public static final byte ARGB_FRAME_PREFIX = (byte) -86;
public static final byte FRAME_SUFFIX = (byte) -1;

public bool a(int i) {
   byte alpha = (byte) Color.alpha(i);
   byte red = (byte) Color.red(i);
   byte green = (byte) Color.green(i);
   byte blue = (byte) Color.blue(i);
   byte[] bArr = new byte[]{ARGB_FRAME_PREFIX, alpha, red, green, blue,
   FRAME_SUFFIX};
   return this.c.e.b(bArr);
}
```

Your task is to analyse the decompiled source code, and based on it create a valid request to light bulb RGB characteristic - setting it to half-dim pure red.

Of course you can then set any color and brightness level you like. Maybe even record a macro to change the colors?

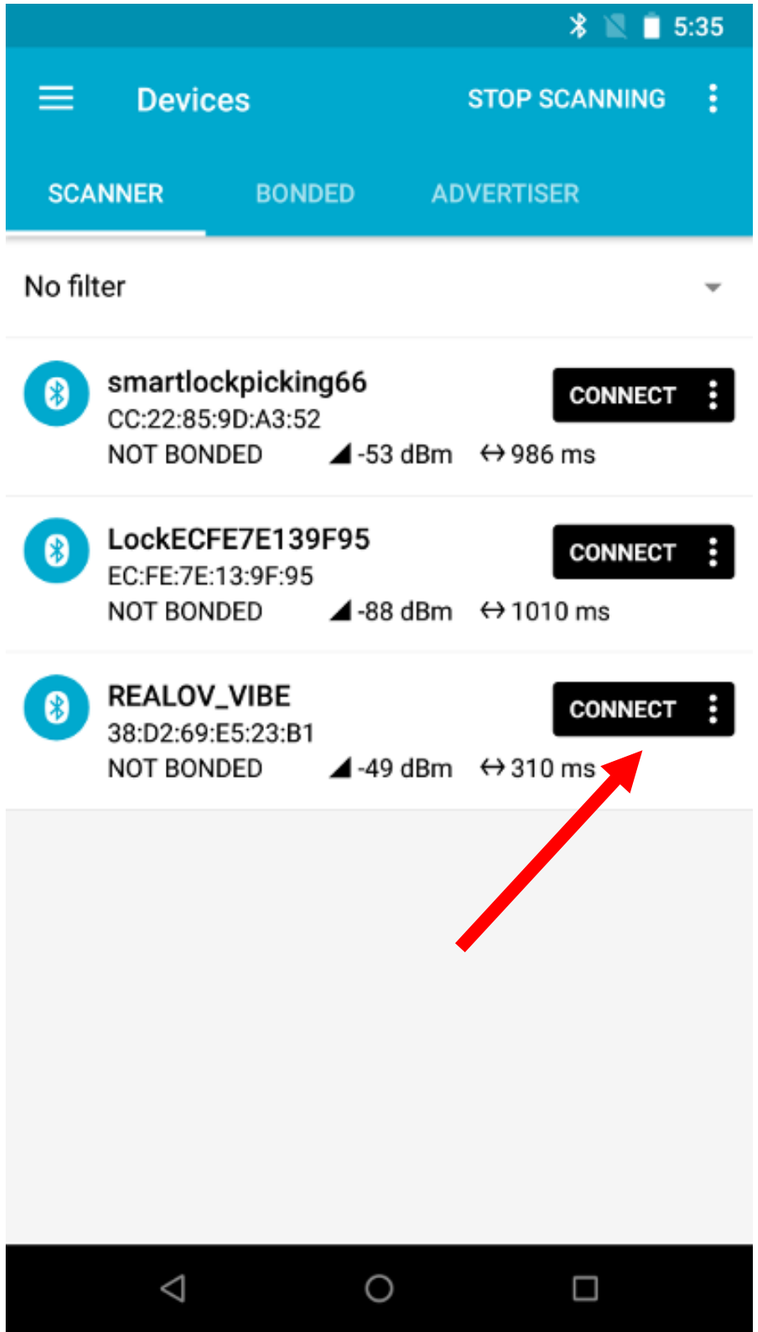
Simulated device

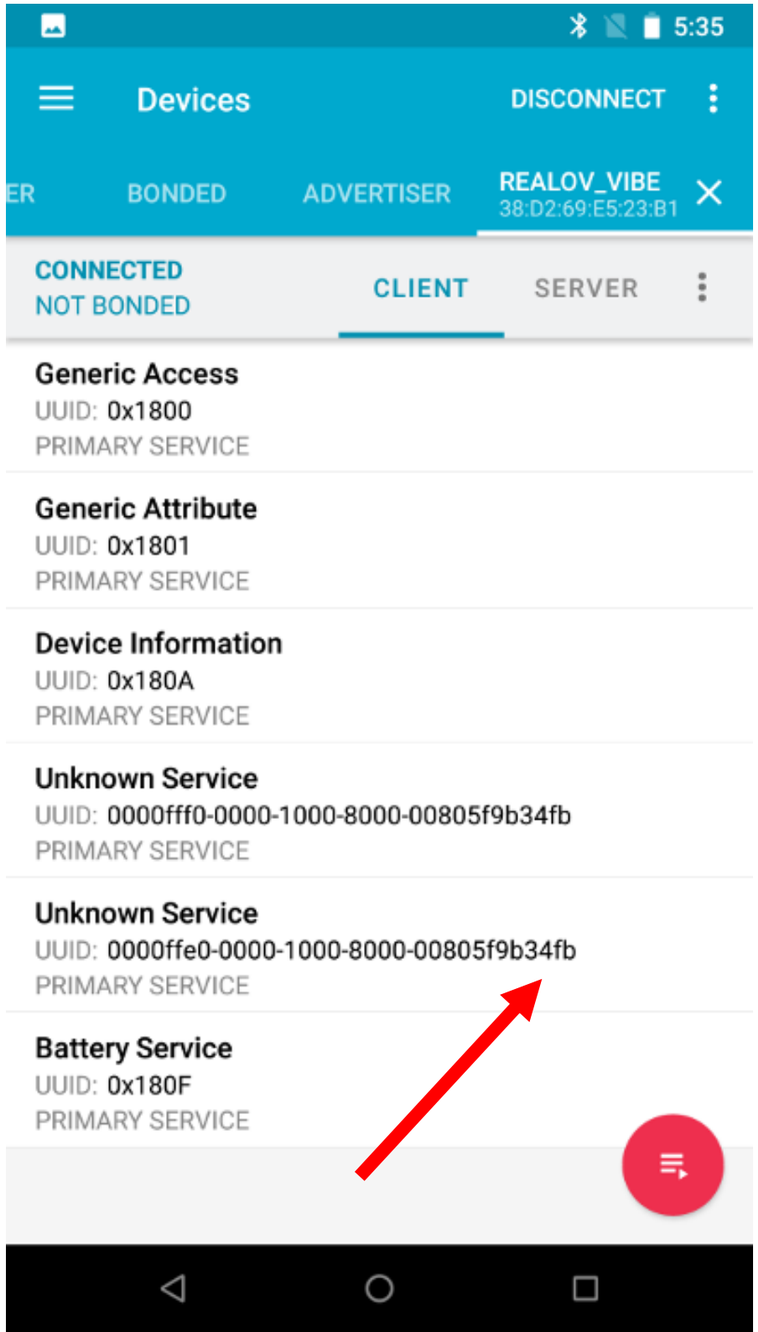
Status:

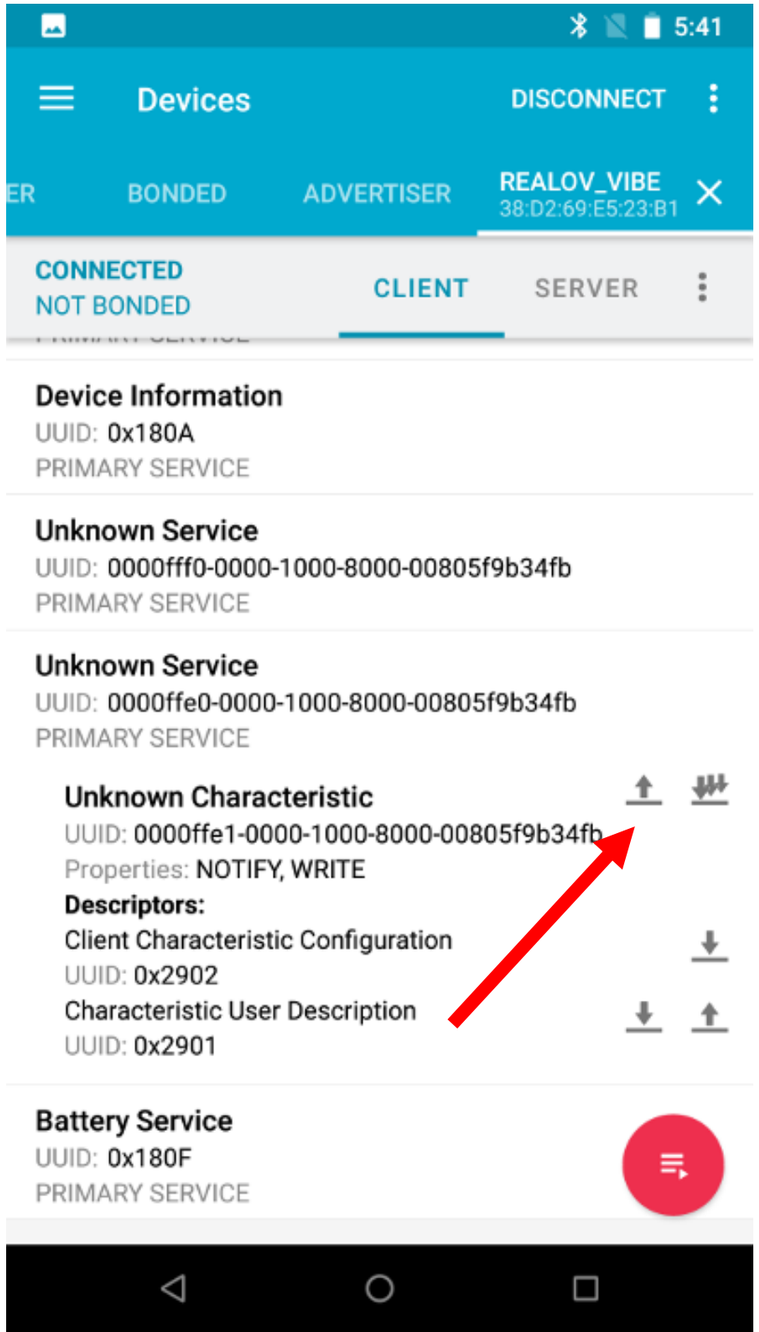
Congratulations! Proceed to the next task

The moment you've been waiting for:

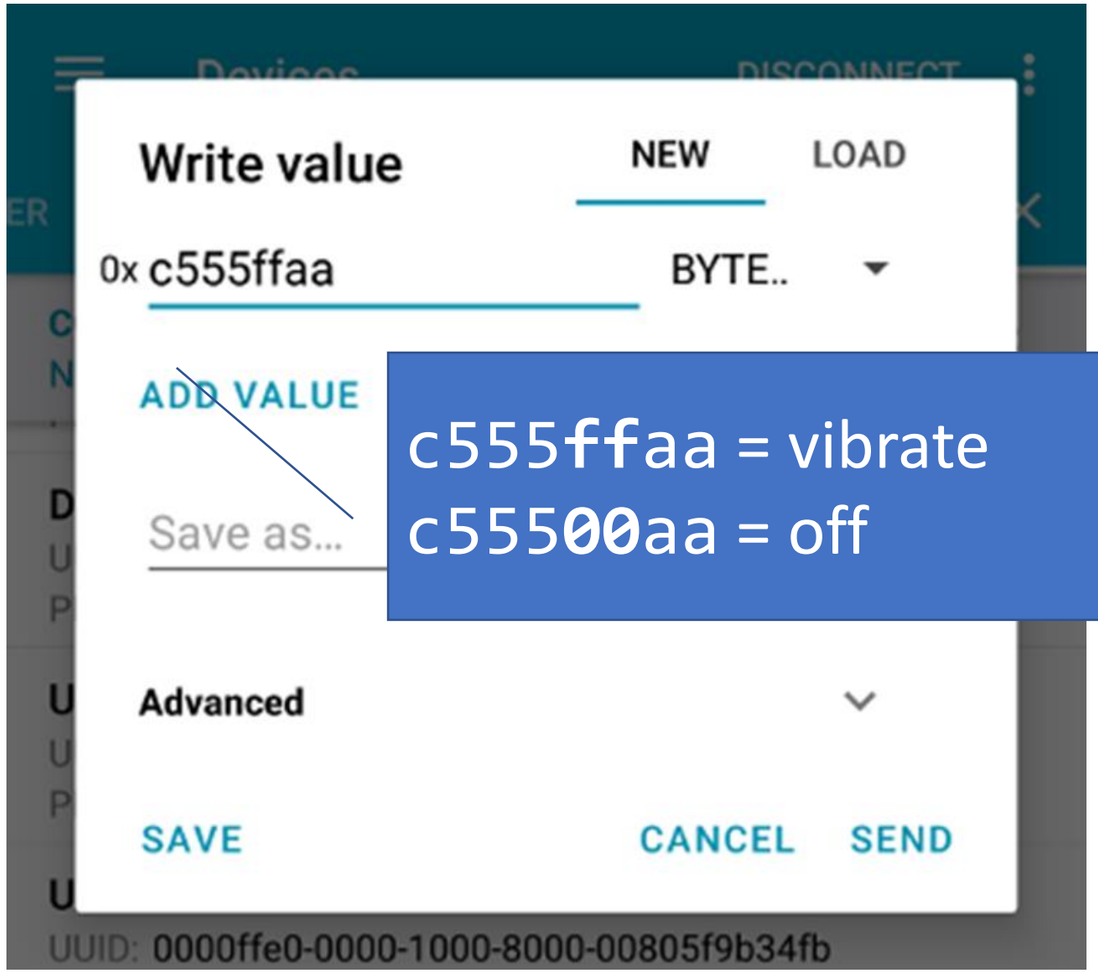
Dildo demo!


5:35
Devices
STOP SCANNING
SCANNER
BONDED
ADVERTISER
No filter
smartlockpicking66
CC:22:85:9D:A3:52
NOT BONDED
-53 dBm
986 ms
CONNECT
LockECFE7E139F95
EC:FE:7E:13:9F:95
NOT BONDED
-88 dBm
1010 ms
CONNECT
REALOV_VIBE
38:D2:69:E5:23:B1
NOT BONDED
-49 dBm
310 ms
CONNECT


5:35
Devices
DISCONNECT
BONDED
ADVERTISER
REALOV_VIBE
38:D2:69:E5:23:B1
CONNECTED
NOT BONDED
CLIENT
SERVER
Generic Access
UUID: 0x1800
PRIMARY SERVICE
Generic Attribute
UUID: 0x1801
PRIMARY SERVICE
Device Information
UUID: 0x180A
PRIMARY SERVICE
Unknown Service
UUID: 0000fff0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Service
UUID: 0000ffe0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Battery Service
UUID: 0x180F
PRIMARY SERVICE


5:41
Devices
DISCONNECT
BONDED
ADVERTISER
REALOV_VIBE
38:D2:69:E5:23:B1
CONNECTED
NOT BONDED
CLIENT
SERVER
Device Information
UUID: 0x180A
PRIMARY SERVICE
Unknown Service
UUID: 0000fff0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Service
UUID: 0000ffe0-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE
Unknown Characteristic
UUID: 0000ffe1-0000-1000-8000-00805f9b34fb
Properties: NOTIFY, WRITE
Descriptors:
Client Characteristic Configuration
UUID: 0x2902
Characteristic User Description
UUID: 0x2901
Battery Service
UUID: 0x180F
PRIMARY SERVICE

Write request



c555**ff**aa = vibrate
c555**00**aa = off

Or just download the app and connect ;)

Realov

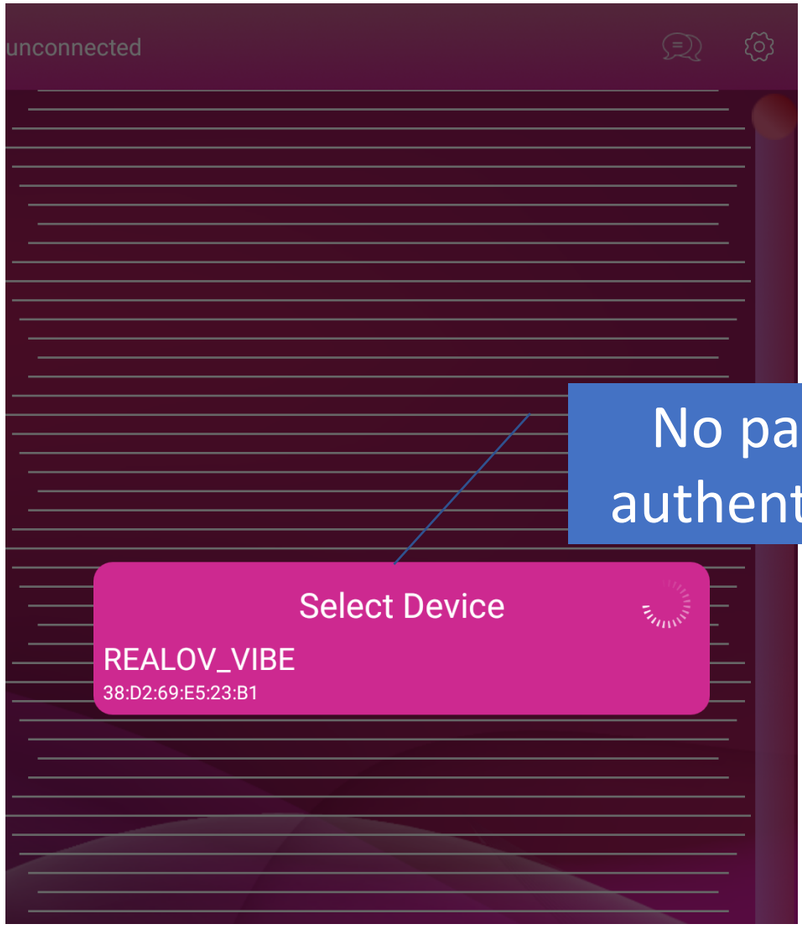
PauloCostacx66 Entertainment

★★★☆☆ 103

PEGI 3

Add to Wishlist

Install



No pairing/ authentication

Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

Write automation

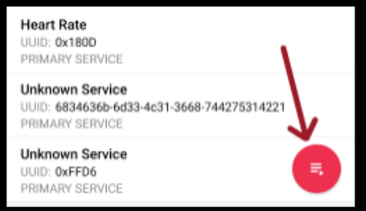
Theory introduction

Typical communication with BLE device consists of a series of writes / reads / notifications. In order to trigger specific functionality, it may be necessary to send multiple writes in sequence, sometimes to various characteristics, and often within short, limited time.

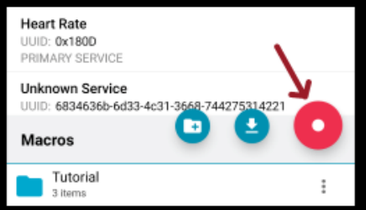
Task

You already know how to turn the light bulb on and off. Your current task is to blink it twice per second for a few seconds.
Don't worry - the job does not require you to master extreme fast clicking in the application. Instead, let's introduce a very handy feature of nRF Connect: **Macros**.

The functionality is available after connecting to device and selecting the small red circle in bottom right corner:



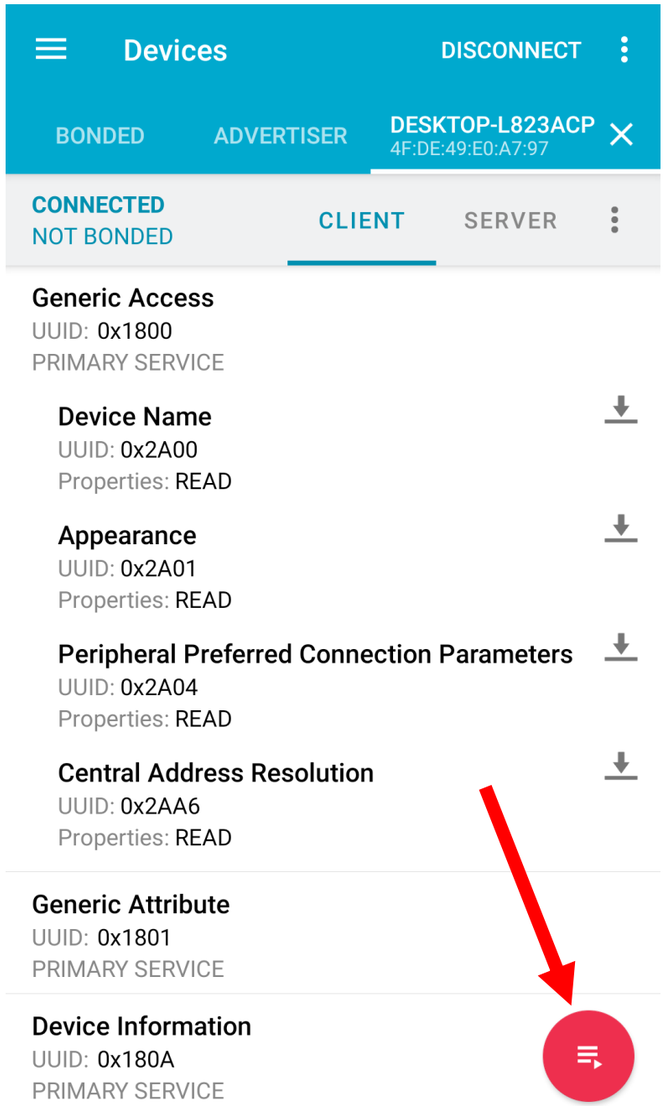
Tap the red circle to start recording:

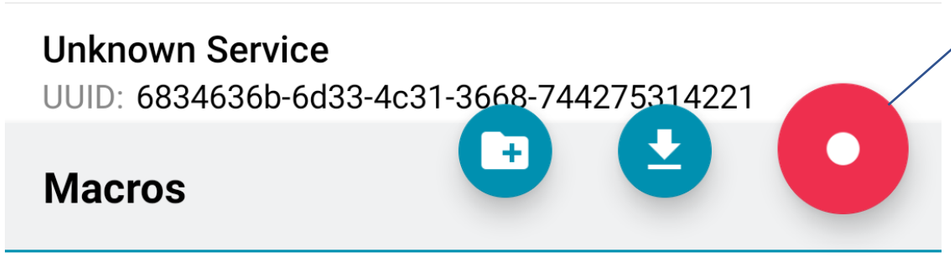


Now send as usual any write you would like to record - for example turn the light on. Next, we can introduce a delay before sending another write. Tap the "hourglass" icon:

Use + and - to set desired delay. 200-300ms will allow to blink twice per second.

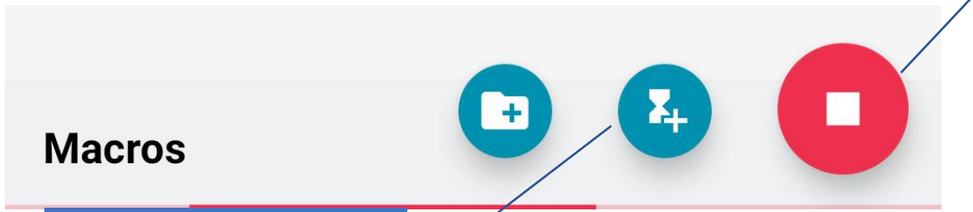
nRF Connect macros



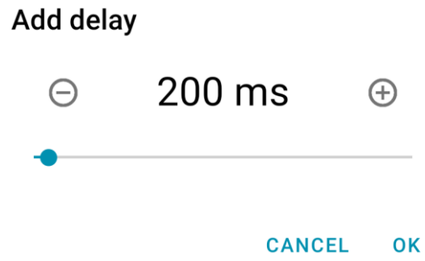


Start recording

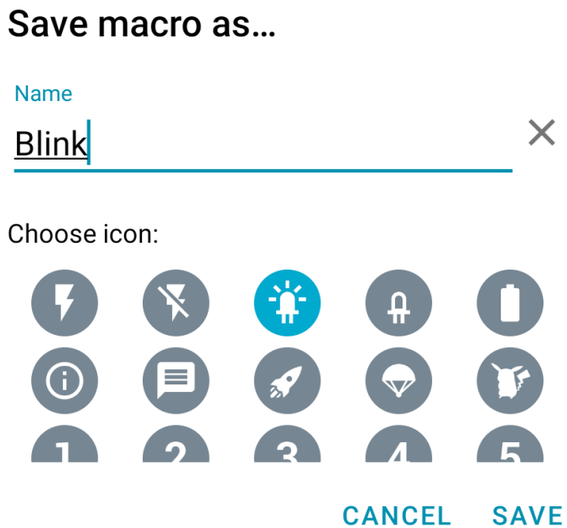
Now send something to device like previously



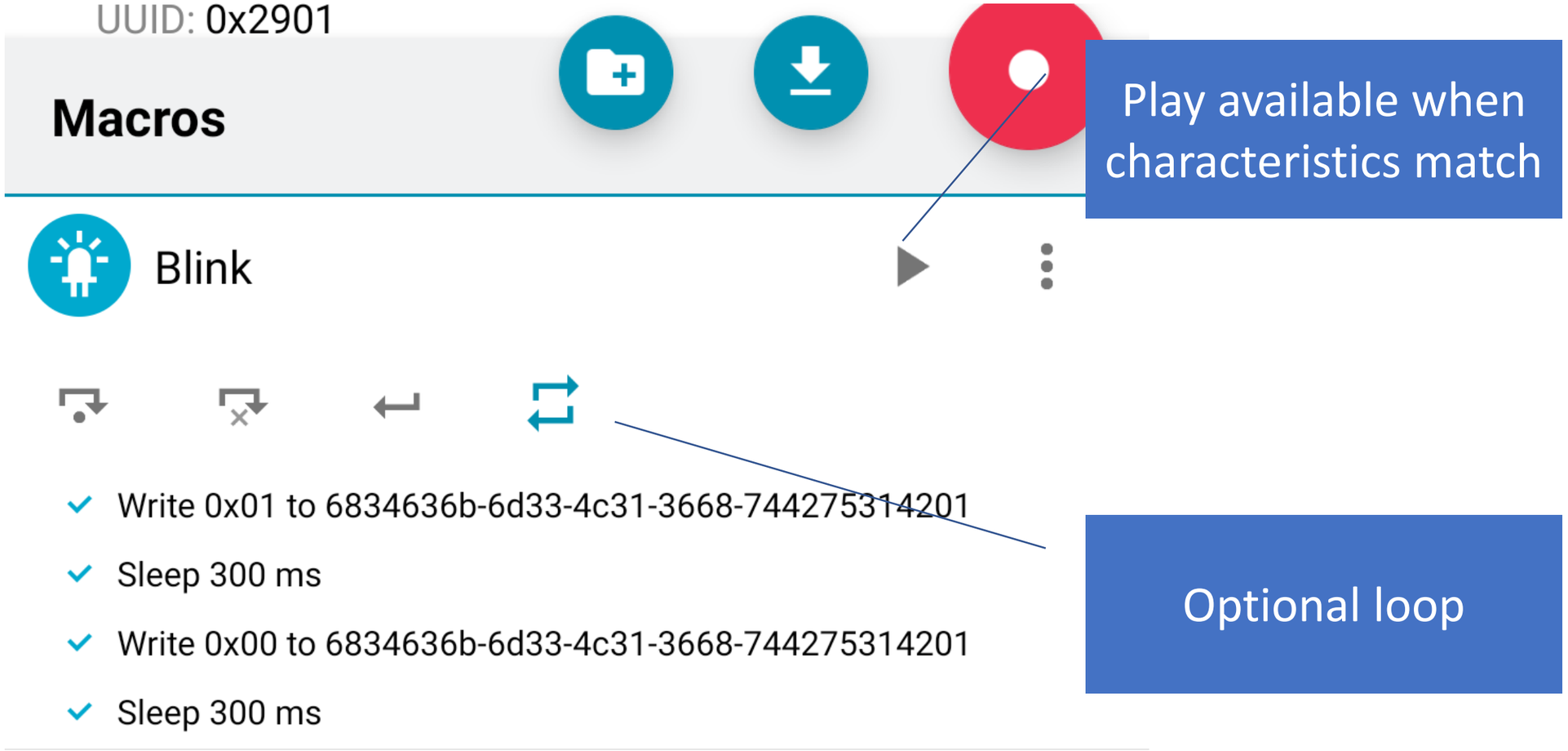
optional delay between requests



Stop recording



Play macro


UUID: 0x2901
Macros
Blink
Write 0x01 to 6834636b-6d33-4c31-3668-744275314201
Sleep 300 ms
Write 0x00 to 6834636b-6d33-4c31-3668-744275314201
Sleep 300 ms
Play available when characteristics match
Optional loop

Completion progress:

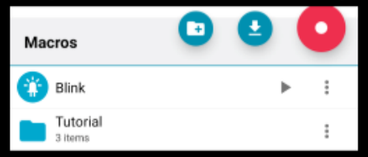
SMARTLOCKPICKING.COM

BLE HACKME

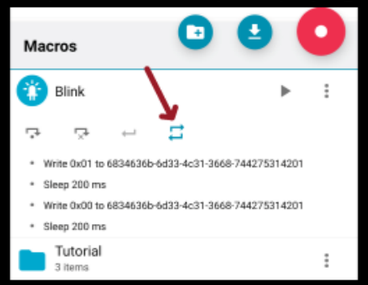
1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

Now, every time you are connected to a device with matching characteristics, the macro will light up blue, and the play button will be enabled. Try if it works:



The only missing part is repeating it multiple times. Tap on the macro name to see its content and options. Select the "repeat" icon, and press play again:



Simulated device

Status:

Congratulations! Proceed to the next task

Completion progress:

SMARTLOCKPICKING.COM

BLE HACKME

Password brute force

Theory introduction

Lots of simple BLE devices work just like you saw in the previous tasks. There is no security, anyone can connect to such device, and in order to control it, just valid data format to send is needed. Slightly more complex devices implement some sort of authentication, for example user password. Only the user who entered valid password in mobile application is authorized to operate it. In many cases the password is then sent by the application in plain, unencrypted form via BLE characteristic write. Devices often do not enforce changing default password (and many users leave this "12345678"), not to mention password complexity. Also, most devices do not have any password brute force prevention mechanisms in place.

Task

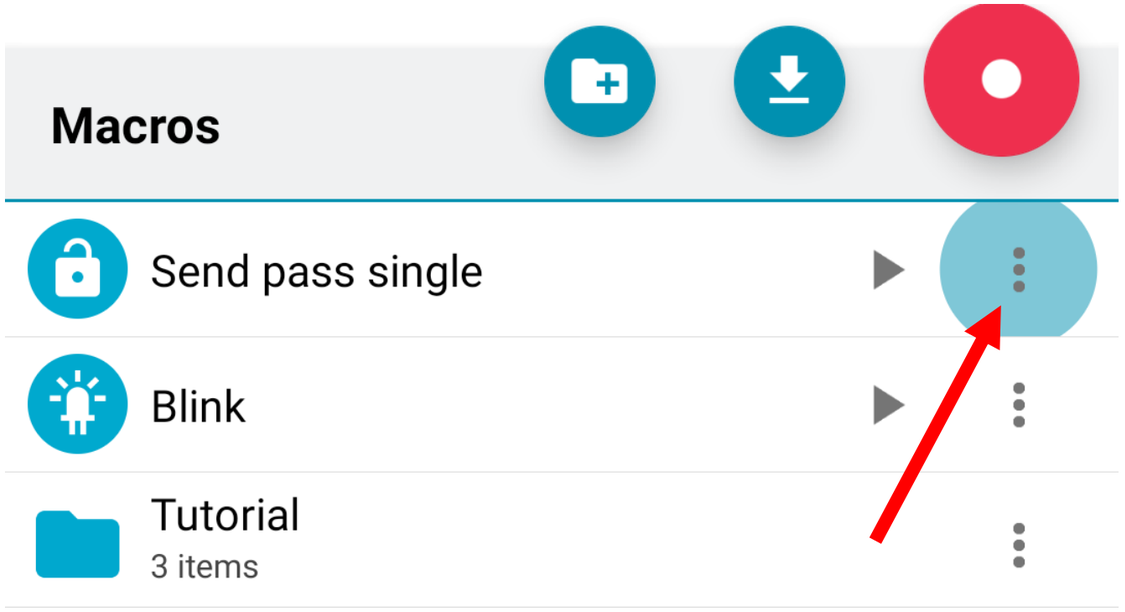
The same light bulb RGB characteristic that you have exploited in previous task, has even more features. By sending another command to it, you can enable light bulb "special effects" mode. This special mode is however password protected. The password is just 3 digits (0-9).
Here is the relevant decompiled source code fragment:

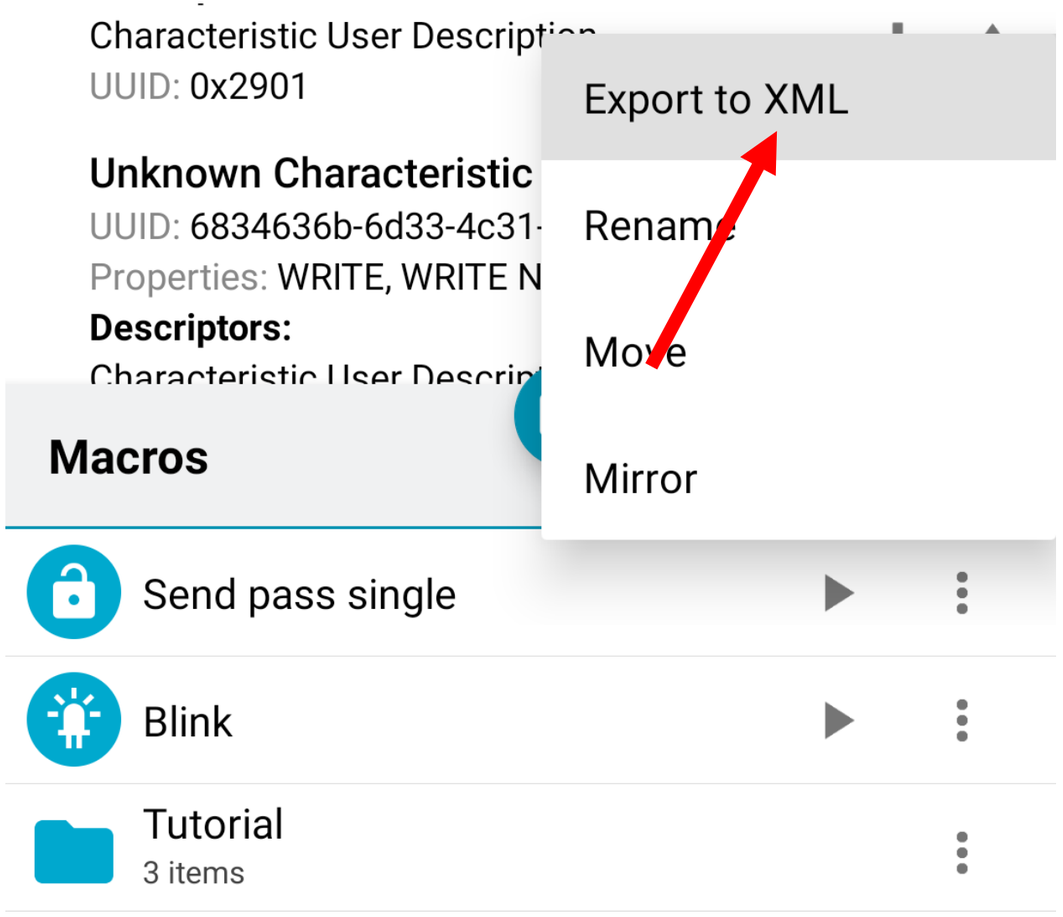
```
public static final byte FRAME_SUFFIX = (byte) -1;
public static final byte FX_FRAME_PREFIX = (byte) -66;
public static final byte FX_ON = (byte) 1;
public static final byte FX_OFF = (byte) 0;

public bool f(bool b) {
  byte a;
  if (b == true) {
    a = FX_ON;
  }
  else
  {
    a = FX_OFF;
  }
  byte[] bArr = new byte[]{FX_FRAME_PREFIX, this.pass[0], this.pass[1],
  this.pass[2], a, FRAME_SUFFIX};
  return this.c.e.b(bArr);
}
```

Your job is to:

1. **Figure out proper command format** - analyse the decompiled code just like in previous task. The HackMe application will let you know in the status if the format of received command is valid but password wrong.
2. **Brute force the password.** Trying each combination by manual writes is possible, but very time

Macros can be exported and edited





nRF Connect macro file (XML)

Ensure that specific characteristics available

```
<macro name="Blink" icon="LED_ON">
  <assert-service description="Ensure 6834636b-6d33-4c31-3668-744275314221 service"
uuid="6834636b-6d33-4c31-3668-744275314221">
        <assert-characteristic description="Ensure 6834636b-6d33-4c31-3668-744275314201
characteristic" uuid="6834636b-6d33-4c31-3668-744275314201">
                <property name="WRITE" requirement="MANDATORY"/>
        </assert-characteristic>
  </assert-service>
  <write description="Write 0x01 to 6834636b-6d33-4c31-3668-744275314201" characteristic-
uuid="6834636b-6d33-4c31-3668-744275314201" service-uuid="6834636b-6d33-4c31-3668-
744275314221" value="01" type="WRITE_REQUEST"/>
  <sleep description="Sleep 200 ms" timeout="200"/>
</macro>
```

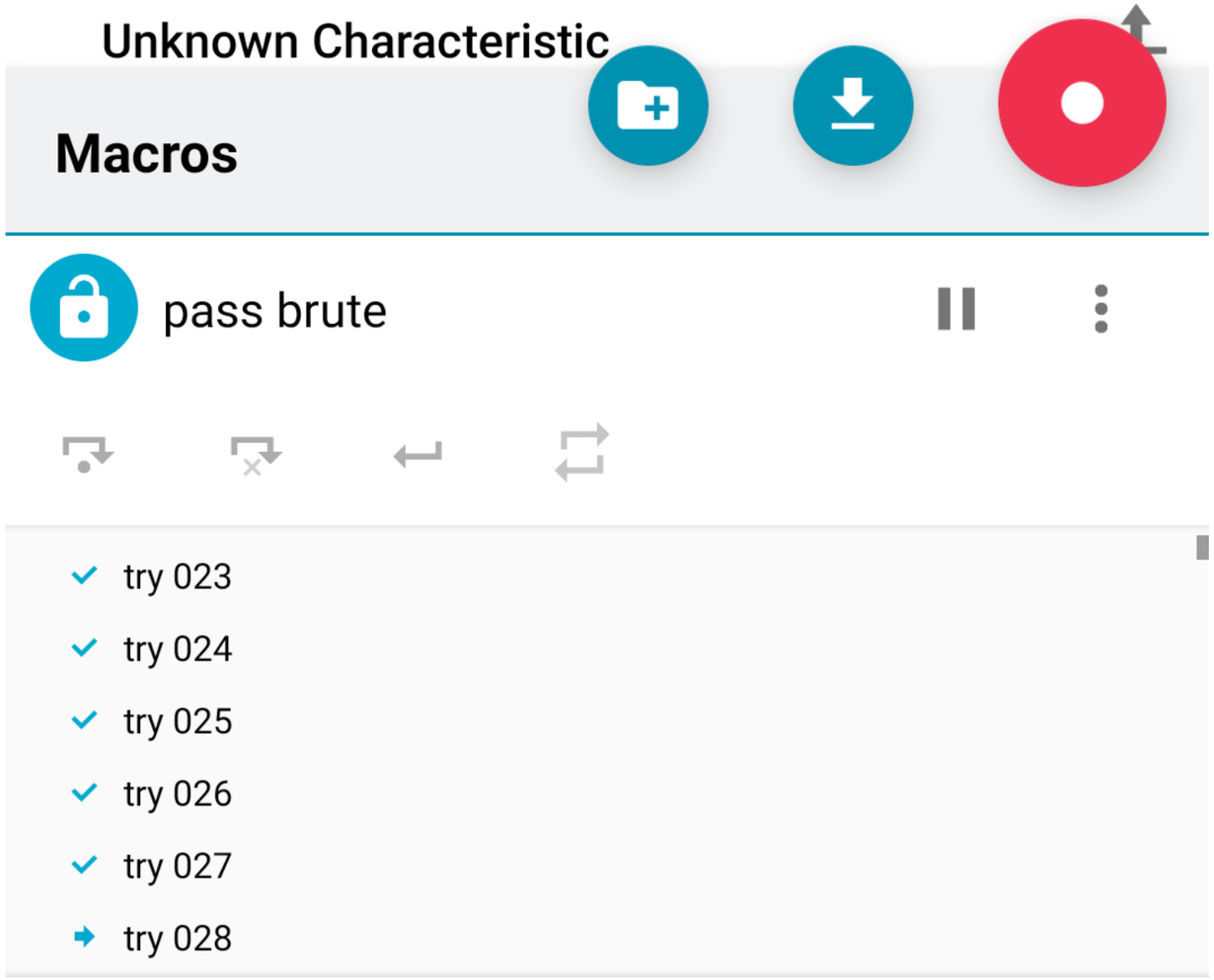
Write request to specified characteristic

Macros documentation:

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

Demo: brute force password

Takes about 100 sec to try all 1000 combinations (10/s)



Completion progress:

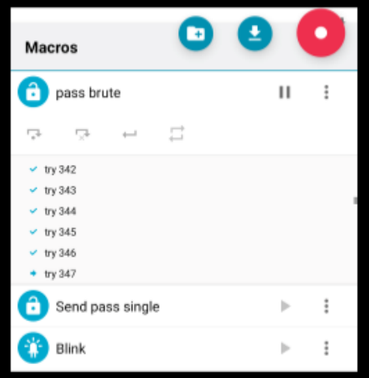
SMARTLOCKPICKING.COM

BLE HACKME

1) Start
2) First steps
3) BLE Advertisements
4) Beacons
5) Manufacturer Specific Advertisements
6) Connections, services, characteristics
7) Characteristic read
8) Notifications
9) Descriptors
10) Characteristic write
11) Various writes
12) Write automation
13) Protocol reverse-engineering
14) Password brute force
15) Smart lock replay
16) Smart lock information leak

More information | FAQ | Source code | Issues

It is worth changing the "write description" in XML file (from the default generated by application) - for example to "try 003", "try 004" etc. This way you will see the progres live during brute force:



The password is randomly generated, and it will be different after you restart HackMe application.

Simulated device

Valid PIN: 6 5 0

Hints

Status:

Invalid password, access denied!

Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

Vaultek Bluetooth gun safe

Password setting instruction manual

Master Code Programming

First time users should change the default code as soon as possible to prevent unauthorized access to your safe. You will also use this master code to pair the Vaultek™ app on your smartphone, so it should be kept confidential.

TIP: Code Requirements

- Your code can be a minimum of 4 and a maximum of 8 keypad entries.
- Two keys cannot be pressed simultaneously.
- Programming a new code will overwrite your previous code.

1 Enter default code 1-2-3-4-5 on the keypad to open your safe.

https://vaulteksafe.com/wp-content/uploads/2018/05/Vaultek-VT20-User-Manual-REV031817-007-OL-2.pdf

Rough assumptions

Number of digits in code	4	6	8
Number of combinations	5^4=625	5^6 = 15625	5^8 = 390625
Time to crack (assuming 10 tries/s)*	62.5 seconds	4.3 hours	4.5 days

- No soft-locking, brute preventions
- Dictionary attack would speed up the chances significantly

* I did not measure in practice how many tries/s this device is capable of

BTW, you don't need the code ;)

ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE

OPEN SESAME —

Top-selling handgun safe can be remotely opened in seconds—no PIN needed

There's no online update mechanism for defective electronic safe.

DAN GOODIN - 12/9/2017, 11:20 PM

https://arstechnica.com/information-technology/2017/12/top-selling-handgun-safe-can-be-remotely-opened-in-seconds-no-pin-needed/

https://www.twosixlabs.com/bluesteal-popping-gatt-safes/

Remotely Cracking Bluetooth Enabled Gun Safes

In this blog post, we will detail BlueSteal, or the ability to exploit multiple security failures in the Vaultek VT20i. These vulnerabilities highlight the need to include security audits early in the product manufacturing process. These vulnerabilities include CVE-2017-17435 and CVE-2017-17436. The VT20i is a very popular product designed for the safe storage of firearms and is one of Amazon's top sellers in several categories. We appreciate the form and fit of the safe as it is one of the more well constructed safes we have interacted with. Along with this post we detail a redacted proof of concept which can unlock Vaultek VT20i Gun Safes that we own through transmission of specially formatted Bluetooth messages.

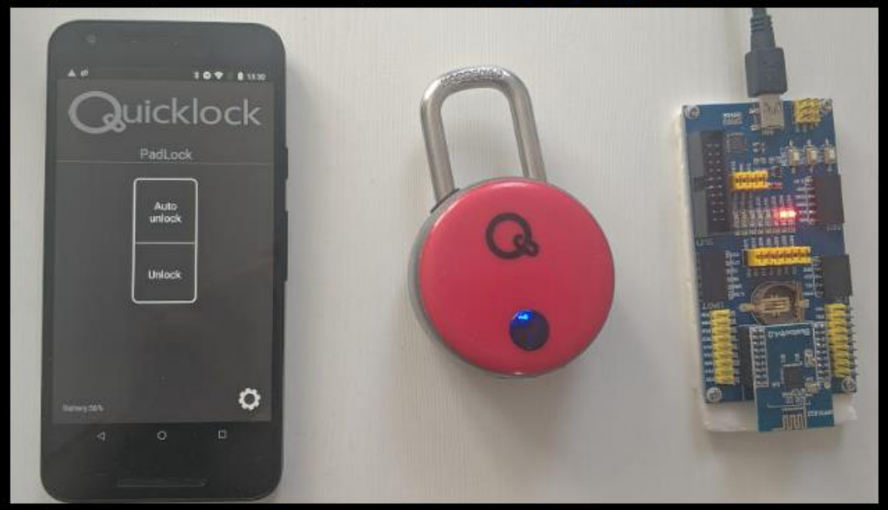
Completion progress:

SMARTLOCKPICKING.COM

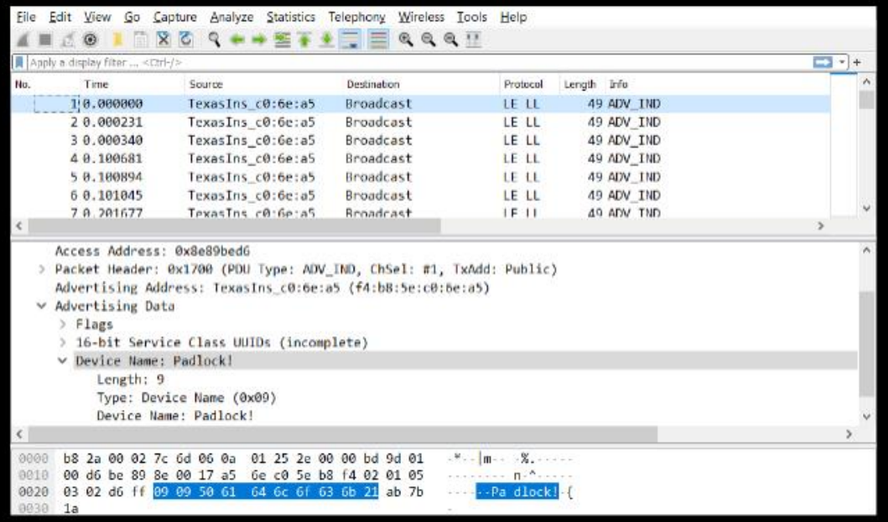
BLE HACKME

Task

Your task is to replay communication of Quicklock Bluetooth Smart padlock and its mobile application, intercepted using nRF Sniffer running on a $15 nRF51 development kit:



Download pcap file with intercepted packets and open it in Wireshark. You will see lots of packets, starting with the lock device advertisements. Note by the way how device name advertisement is visible in Wireshark:



Real smart lock, real vulnerabilities!

https://www.thequicklock.com/product-padlock.php

Defcon 24: Anthony Rose, Ben Ramsey

>>> Picking Bluetooth Low Energy Locks from a Quarter Mile Away

Anthony Rose & Ben Ramsey

MERCULITE SECURITY

>>> Plain Text Passwords

* Are they even trying?
* Found on 4 separate locks
  - Quicklock Doorlock
  - Quicklock Padlock
  - iBluLock Padlock
  - Plantraco Phantomlock

```
▶ Frame 278: 49 bytes on wire (392 bits)
▶ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low
▶ Bluetooth Low Energy Link Layer
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Request (0x12)
    Handle: 0x002d
    Value: 001234567812345678
```

00 12345678 12345678
Opcode Current Password New Password

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

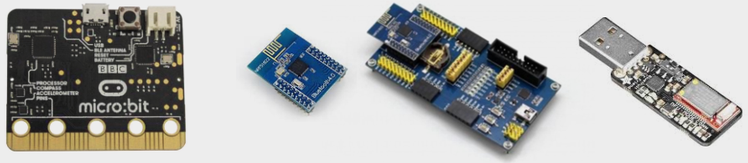
Sniffing #1 – do I actually need a sniffer?

Application running on your phone: use Android btsnoop

- Saves pcap (readable in Wireshark)
- Some phones (e.g. Nexus 5X 8.1) have adb TCP service for live integration

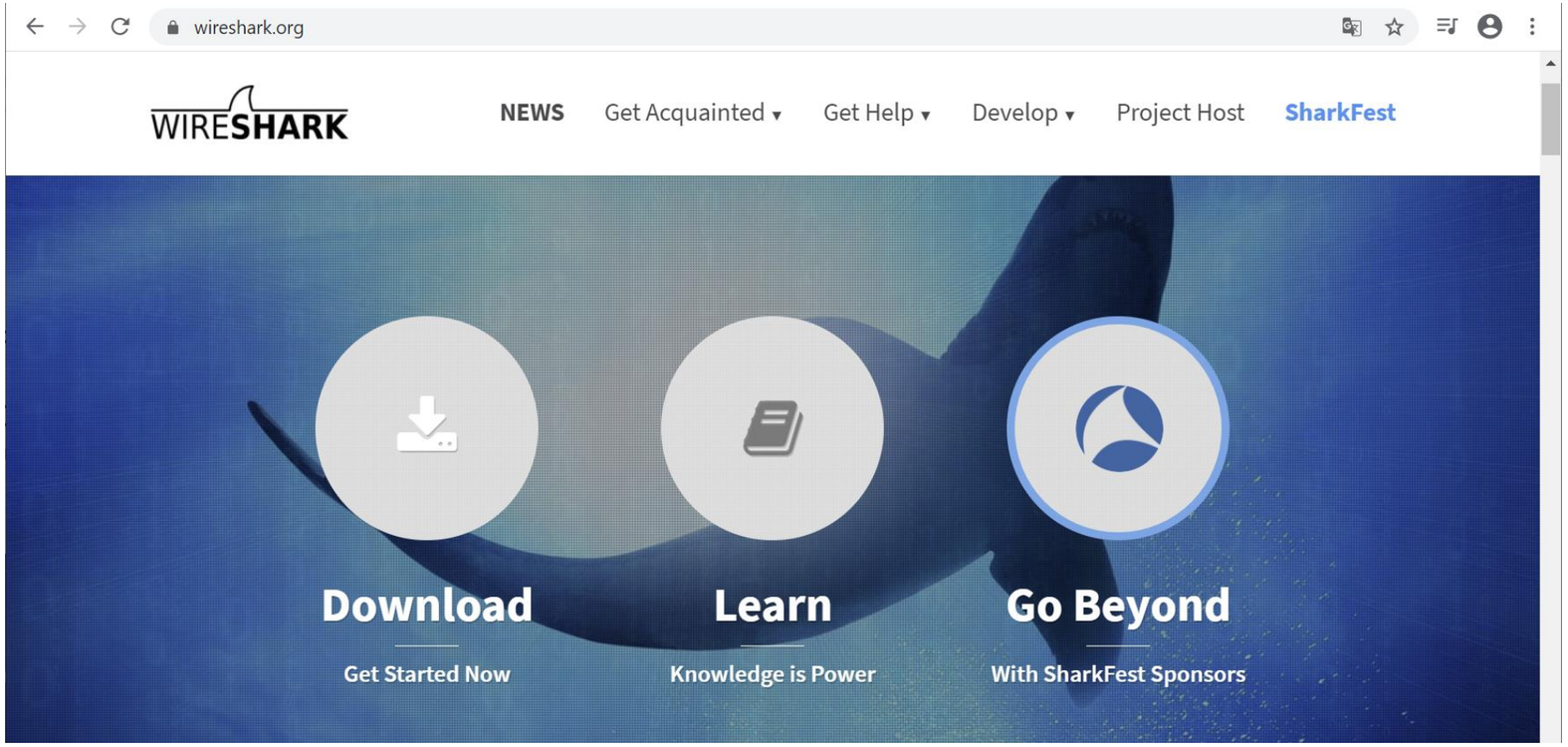
usbmon1
usbmon2
Android Bluetooth Btsnoop Net Nexus_5X 00fd60f3e2c5
Android Logcat Crash Nexus_5X 00fd60f3e2c54c2b: andr
Android Logcat Events Nexus_5X 00fd60f3e2c54c2b: and

Open BLE sniffers

Ubertooth	Open hardware/firmware. First open Bluetooth sniffer. https://www.greatscottgadgets.com/ubertoothone/	Open hardware 	120$
nRF Sniffer	Closed (but free) firmware. Nice integration with Wireshark (toolbar). https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Sniffer-for-Bluetooth-LE	Nordic Semiconductor nRF51/52 	$5- $50
BtleJack	Open firmware. Can also jam and hijack connections. https://github.com/virtualabs/btlejack	nRF51 (including BBC: microbit) 	$5- $25
SniffLE	Open firmware. BLE 5; improved reliability. https://github.com/nccgroup/Sniffle	Texas Instructions CC1352/CC26x2 	$40

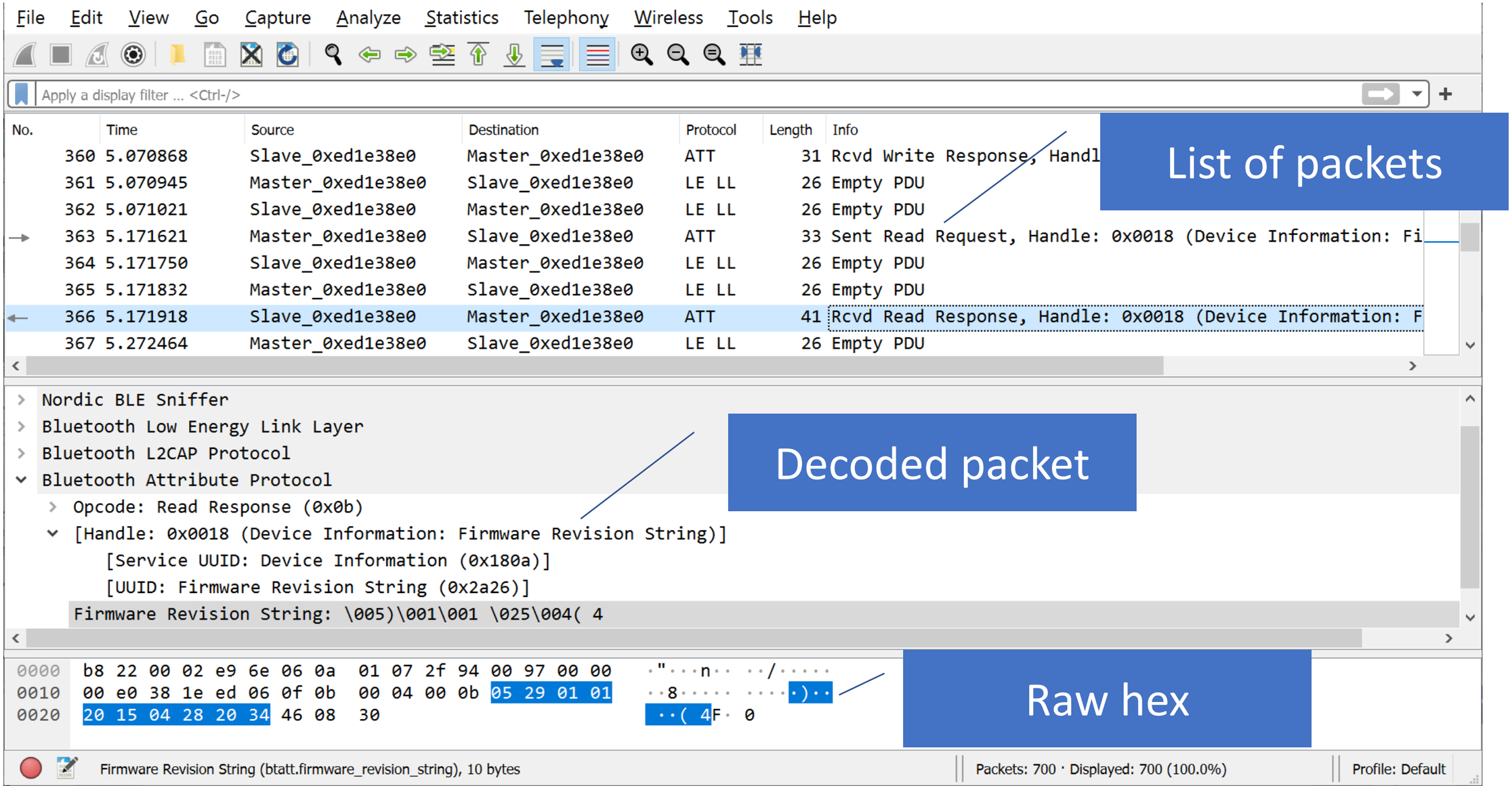
Wireshark

- The free, open “industry standard” for network analysing
- All open BLE sniffers can dump to its pcap format



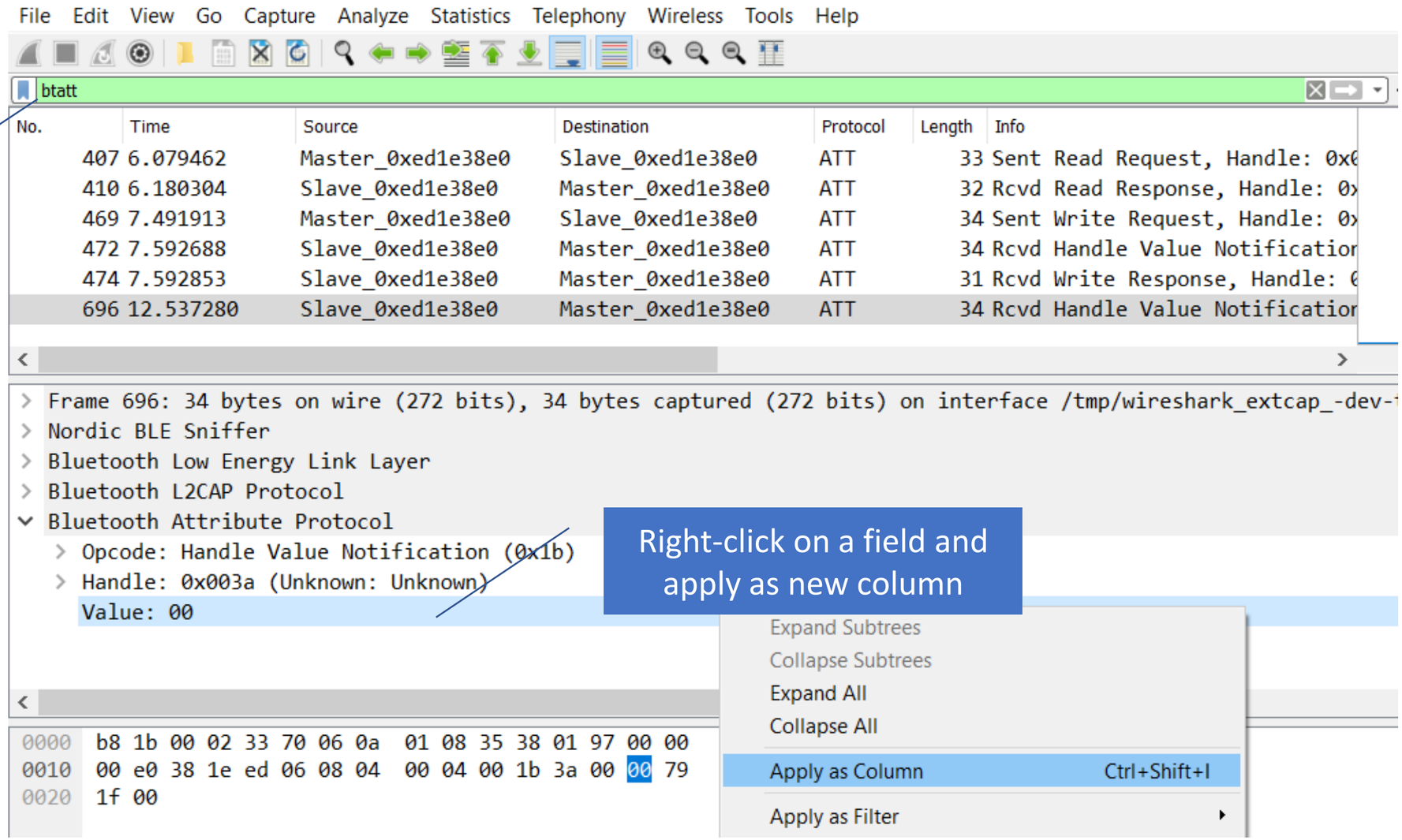
https://www.wireshark.org

Wireshark

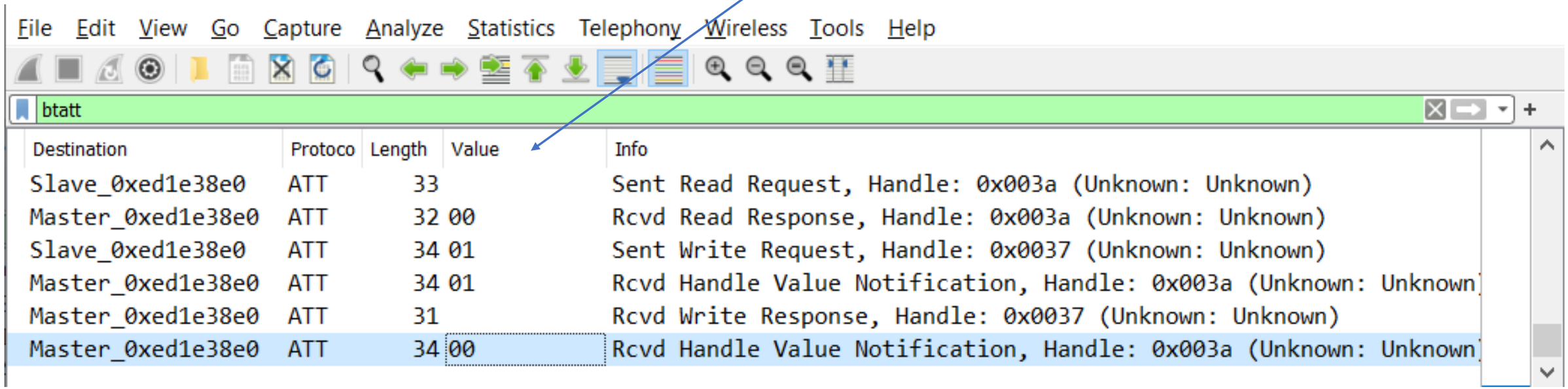

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
360 5.070868 Slave_0xed1e38e0 Master_0xed1e38e0 ATT 31 Rcvd Write Response, Handl
361 5.070945 Master_0xed1e38e0 Slave_0xed1e38e0 LE LL 26 Empty PDU
362 5.071021 Slave_0xed1e38e0 Master_0xed1e38e0 LE LL 26 Empty PDU
363 5.171621 Master_0xed1e38e0 Slave_0xed1e38e0 ATT 33 Sent Read Request, Handle: 0x0018 (Device Information: Fi
364 5.171750 Slave_0xed1e38e0 Master_0xed1e38e0 LE LL 26 Empty PDU
365 5.171832 Master_0xed1e38e0 Slave_0xed1e38e0 LE LL 26 Empty PDU
366 5.171918 Slave_0xed1e38e0 Master_0xed1e38e0 ATT 41 Rcvd Read Response, Handle: 0x0018 (Device Information: F
367 5.272464 Master_0xed1e38e0 Slave_0xed1e38e0 LE LL 26 Empty PDU
List of packets
Nordic BLE Sniffer
Bluetooth Low Energy Link Layer
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Read Response (0x0b)
[Handle: 0x0018 (Device Information: Firmware Revision String)]
[Service UUID: Device Information (0x180a)]
[UUID: Firmware Revision String (0x2a26)]
Firmware Revision String: \005)\001\001 \025\004(4
Decoded packet
0000 b8 22 00 02 e9 6e 06 0a 01 07 2f 94 00 97 00 00
0010 00 e0 38 1e ed 06 0f 0b 00 04 00 0b 05 29 01 01
0020 20 15 04 28 20 34 46 08 30
Raw hex
Firmware Revision String (btatt.firmware_revision_string), 10 bytes
Packets: 700 · Displayed: 700 (100.0%)
Profile: Default

Wireshark tricks

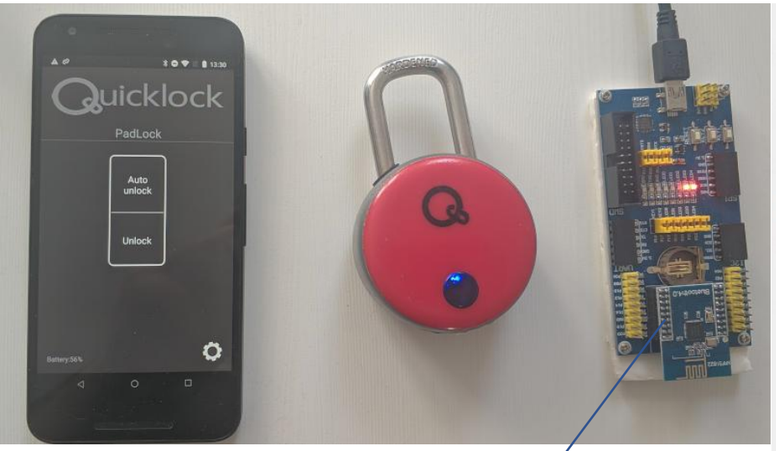
"btatt" filter: display only ATT (read/write/notify)



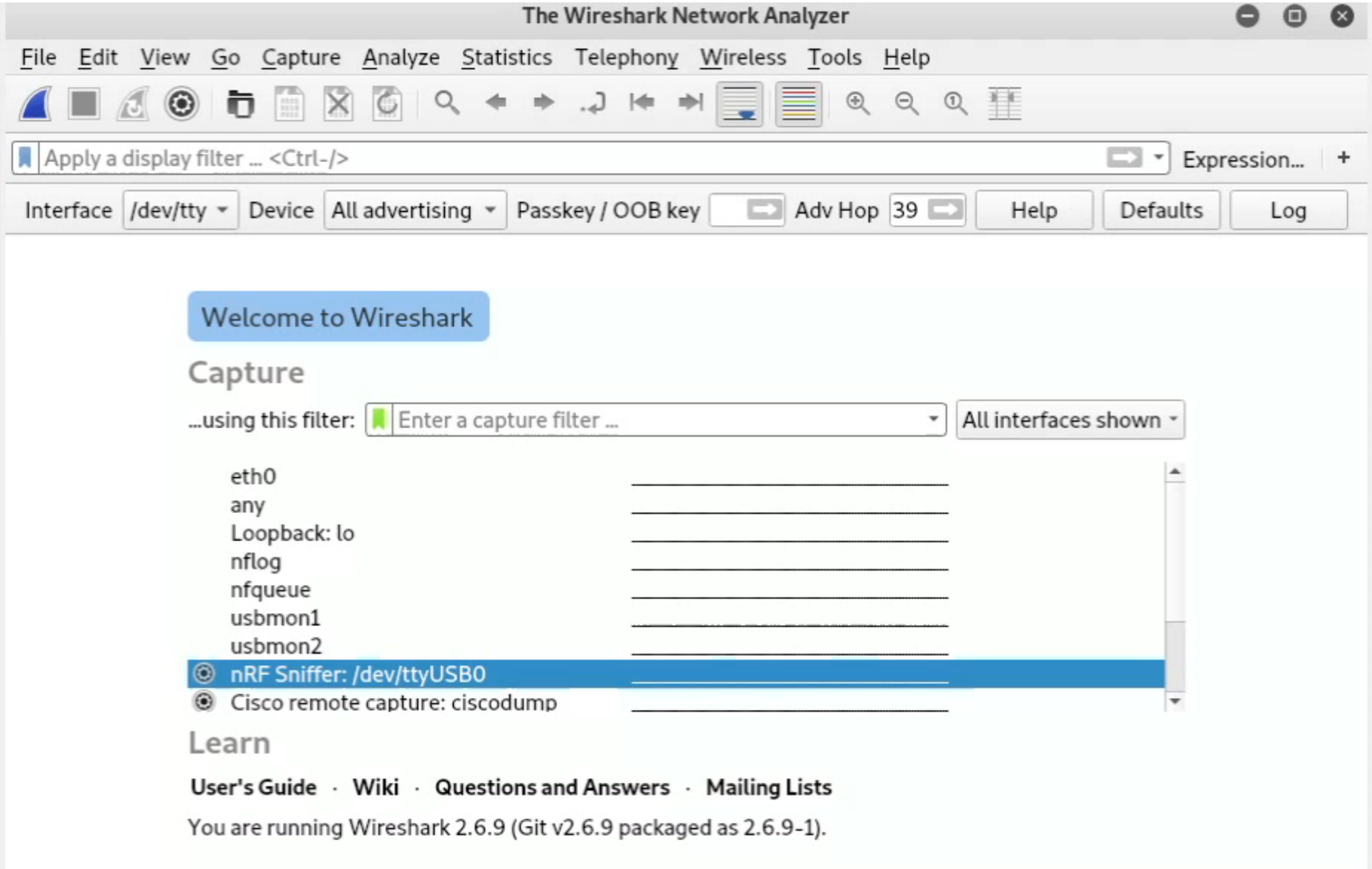
Wireshark tricks: new “value” column



How did I sniff it?



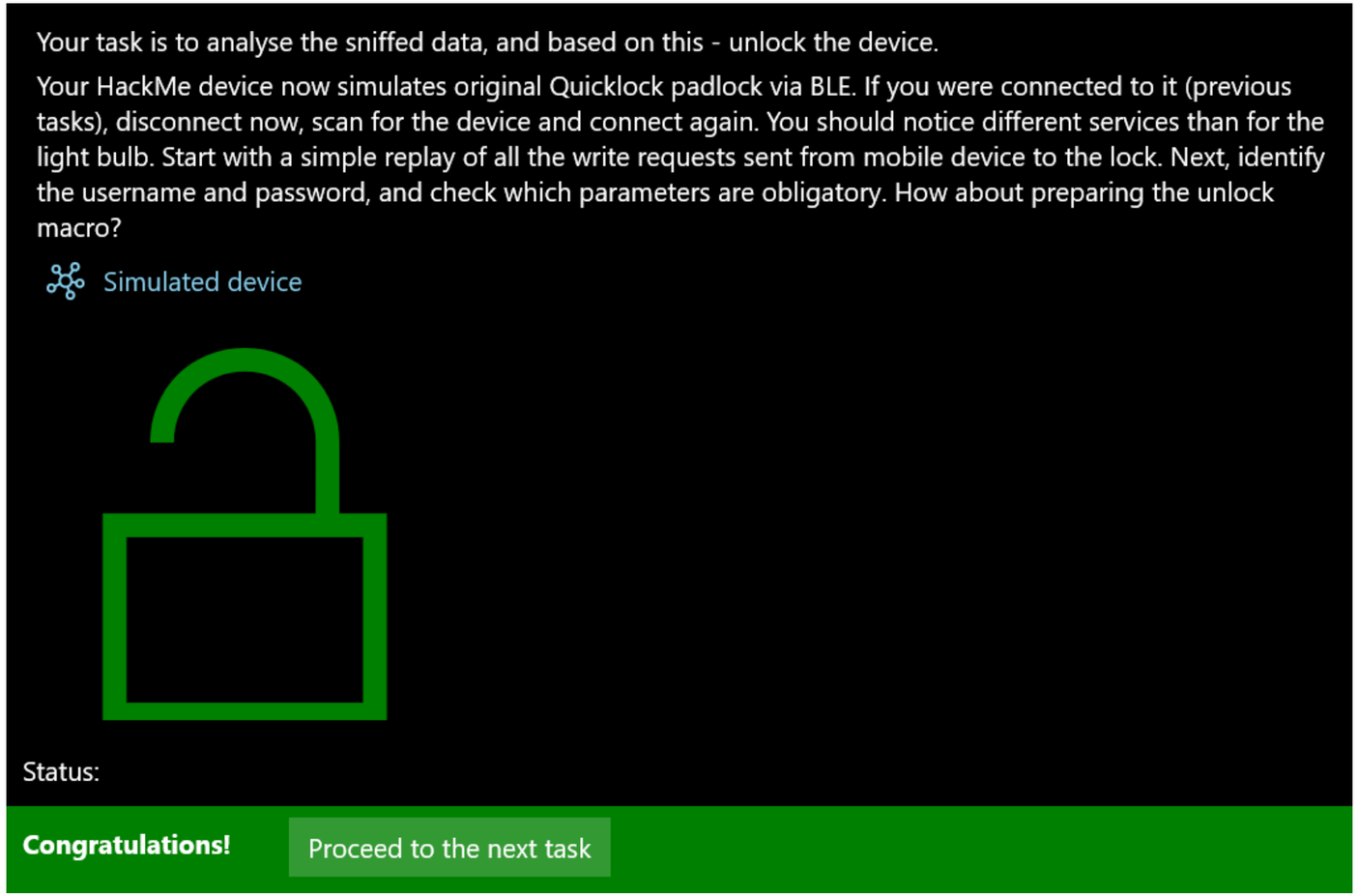
$15 nRF51822 flashed with nRF Sniffer firmware



HackMe: replay

quicklock unlock replay

- send username
- send password
- send unlock



Vendor's response

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.

(...)

12345678

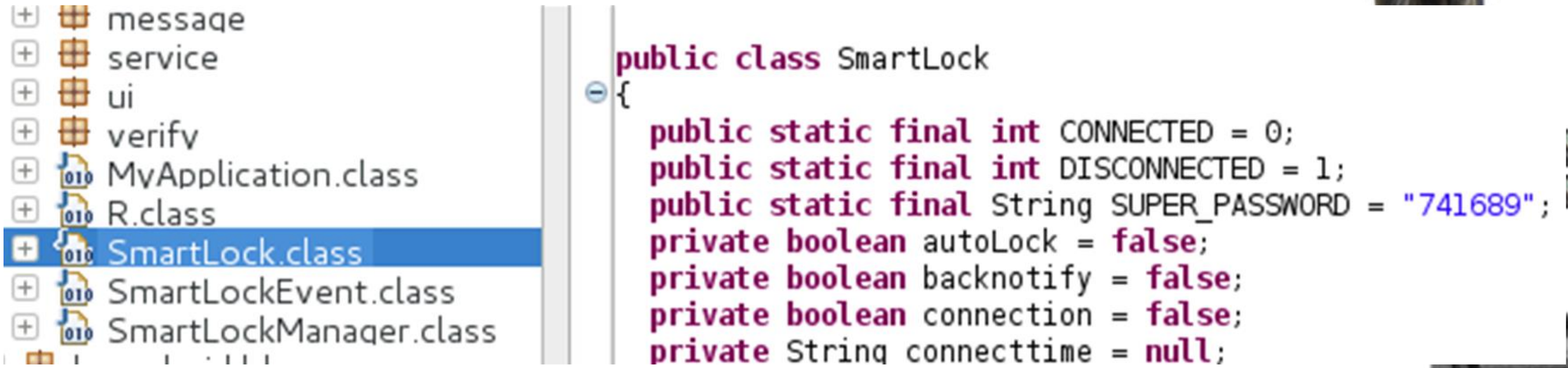
Many users of the products never update the default password and when they call for tech support our first option is to have them try the default Bluetooth password– which often works. !!!

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

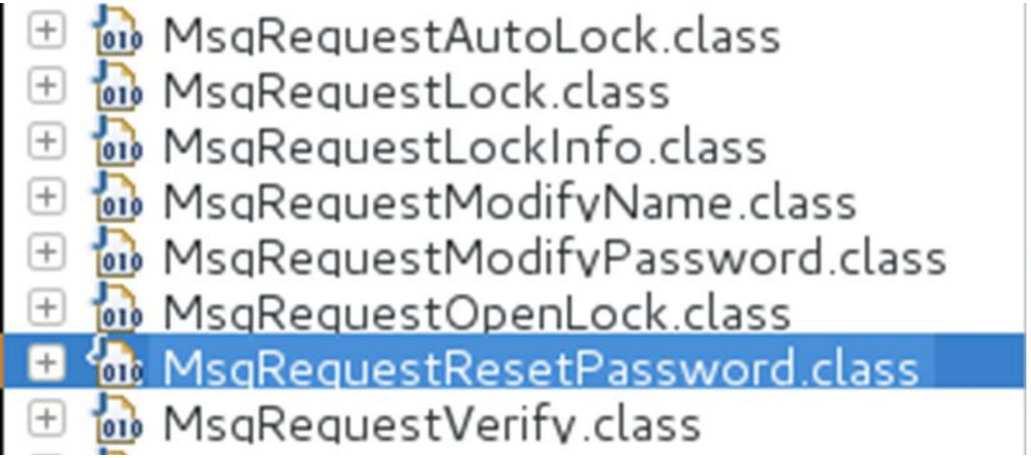
BTW, it also features RFID card

https://www.youtube.com/watch?v=gutxTyyxbcg
https://twitter.com/slawekja/status/1103675187860512768

Other smart locks?




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




```
public static final int MSG_CMD = 8;
public static final int MSG_LENGTH = 8;
public static final int MSG_STX = 161;
```

Unlock without knowing the password

a13734313638390578 9a3b246c6c17164f0121

a20500f0c77f162e8bd2 80

a13734313638390 9bcaafbae83b3babc02b81 7a0

a20900

a137343136383908

a131323334353601

Initial communication based on lock's MAC (just replay)

SUPER_PASSWORD: 741689

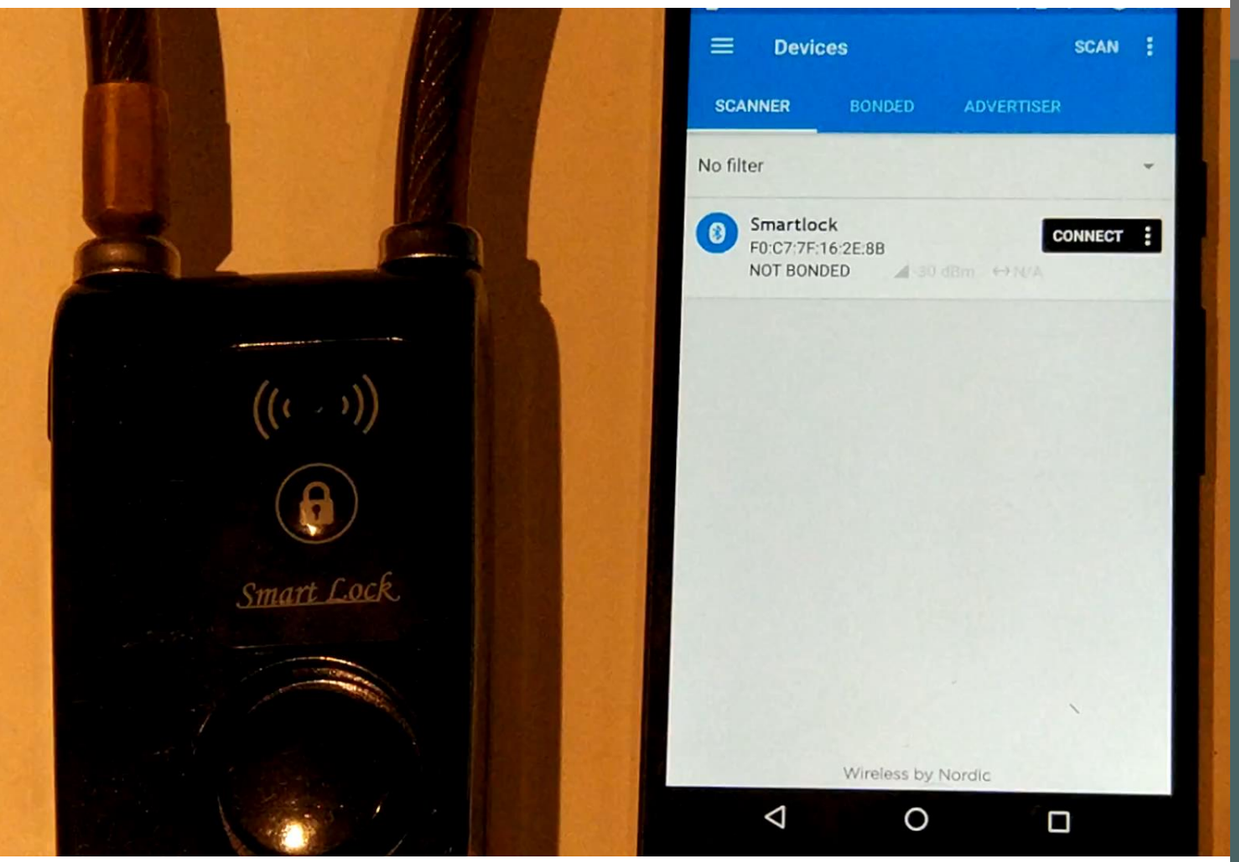
Reset password command

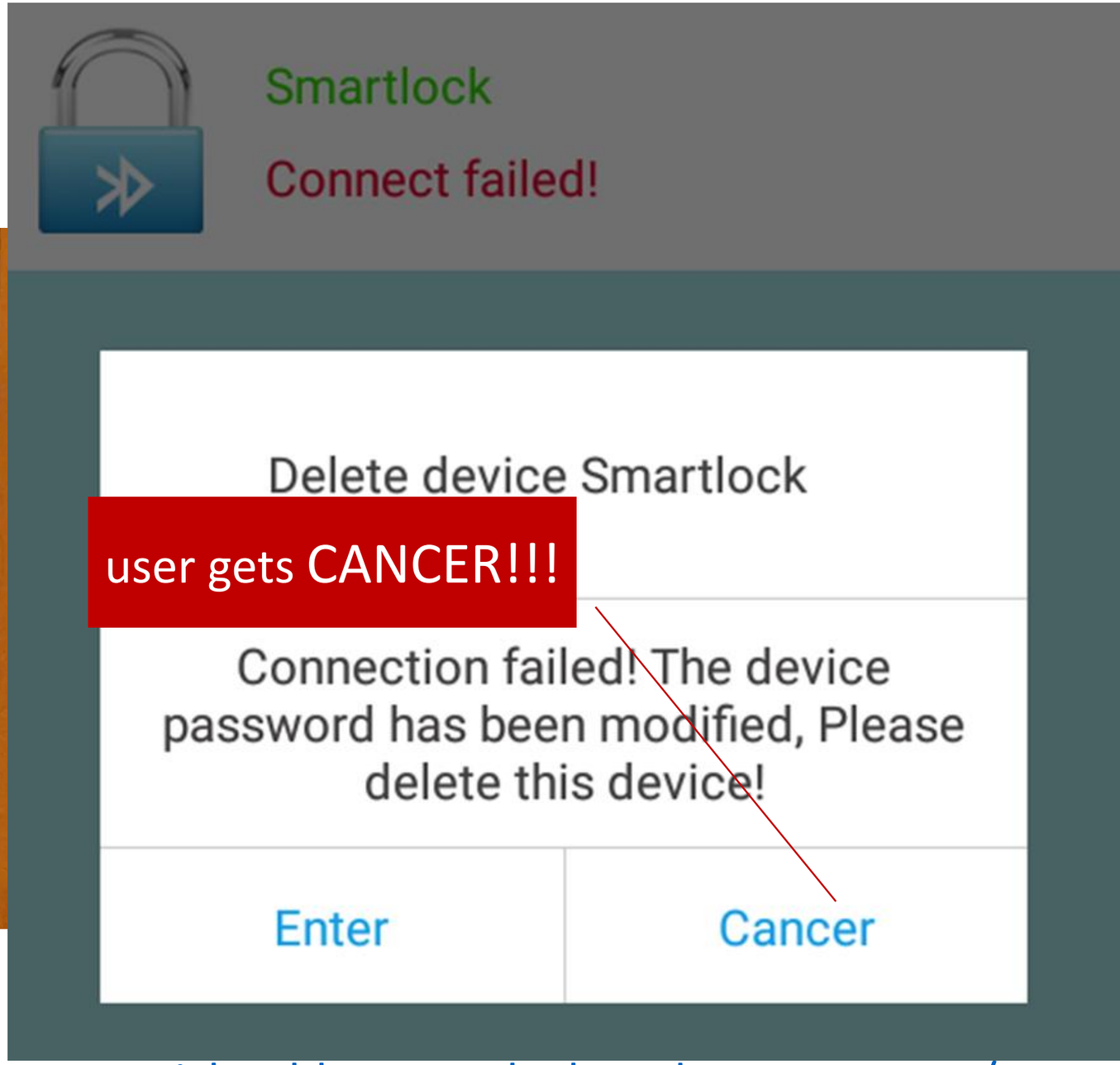
Unlock command

Default password: 123456

https://smartlockpicking.com/tutorial/how-to-pick-a-ble-smart-lock-and-cause-cancer/

Deadly hack!





https://smartlockpicking.com/tutorial/how-to-pick-a-ble-smart-lock-and-cause-cancer/

Tapplock: pass=MD5(MAC)

```
public static String keyAndSerialNo(String str, String str2) {
    if (str == null) {
        return NULL_ONE;
    }
    str = AndroidTool.md5(str.toUpperCase()).toUpperCase();
    int i = -1;
    int hashCode = str2.hashCode();
    if (hashCode != -2081830932) {
        if (hashCode != -96187322) {
            if (hashCode == -96182228 && str2.equals(KEY_TWO)) {
                i = 1;
            }
        } else if (str2.equals(KEY_ONE)) {
            i = 0;
        }
    } else if (str2.equals(SERIAL_NO)) {
        i = 2;
    }
    switch (i) {
        case 0:
            str = str.substring(0, 8);
            break;
        case 1:
            str = str.substring(8, 16);
            break;
        case 2:
            str = str.substring(16, 24);
            break;
        default:
            str = NULL_ONE;
            break;
    }
    return str;
}
```

Calculate MD5 from uppercase MAC

KEY_ONE = first 8 characters of the MD5

SERIAL_NO = characters 16-24 of the MD5

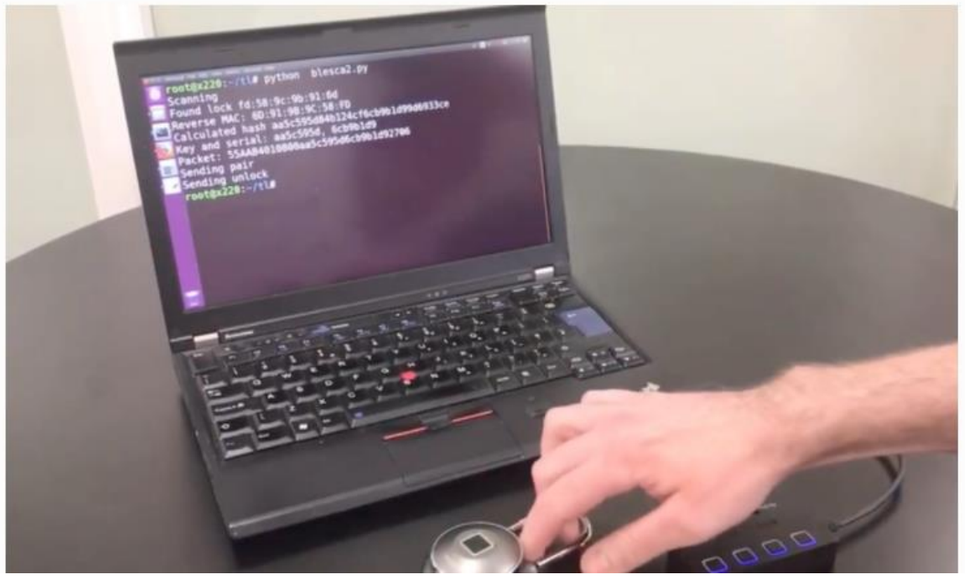
Forbes

Jun 13, 2018, 05:25am EDT

Tapplock: This $100 'Smart Lock' Can Be Hacked Open In 2 Seconds

Thomas Brewster Forbes Staff
Cybersecurity
Associate editor at Forbes, covering cybercrime, privacy, security and surveillance.

This article is more than 2 years old.



https://www.forbes.com/sites/thomasbrewster/2018/06/13/tapplock-smart-lock-hacked-in-2-seconds

Details: https://www.pentestpartners.com/security-blog/totally-pwning-the-tapplock-smart-lock/

So I bought one on E-bay…

… and the seller turned out to be another security researcher ;)

unic0de000 155 points · 10 days ago

5. Their business model is entirely based on selling padlocks to security researchers who want to publish critiques of the lock

Reply Share Report Save Give gold

msuozzo 27 points · 10 days ago

It's genius when you think about it.

Reply Share Report Save Give gold

https://www.reddit.com/r/netsec/comments/8qsmkq/unlocking_a_smart_padlock_using_md5_and_thats_it/

Tapplock: early firmware

@cybergibbons Tapplock auth: MD5(MAC)

55AAB4010800BEDEF425020FEB279405

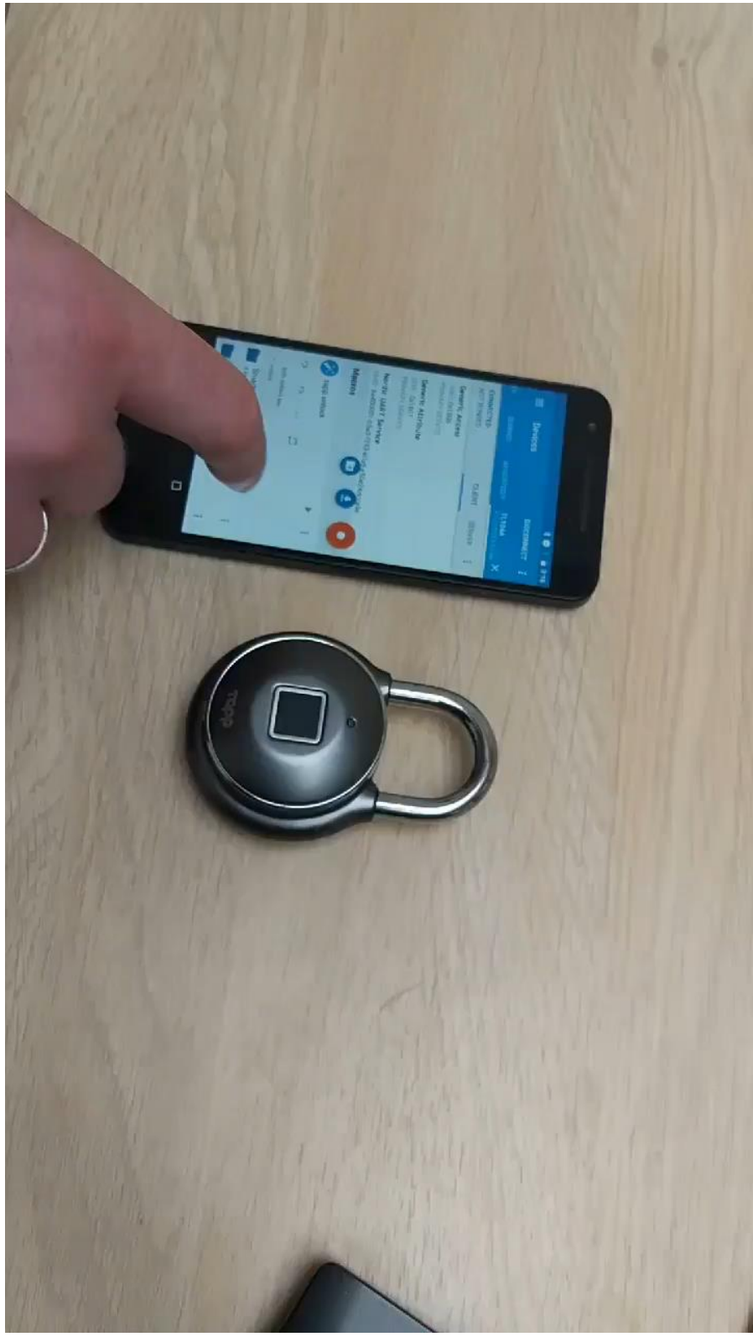
Static header | KEY1 (MD5 of MAC) | SERIAL_NO (MD5 of MAC) | CRC

@LucaBongiorni Tapplock auth:

55AAB4010800010203040000000C601

KEY1 (static) | SERIAL_NO (static)

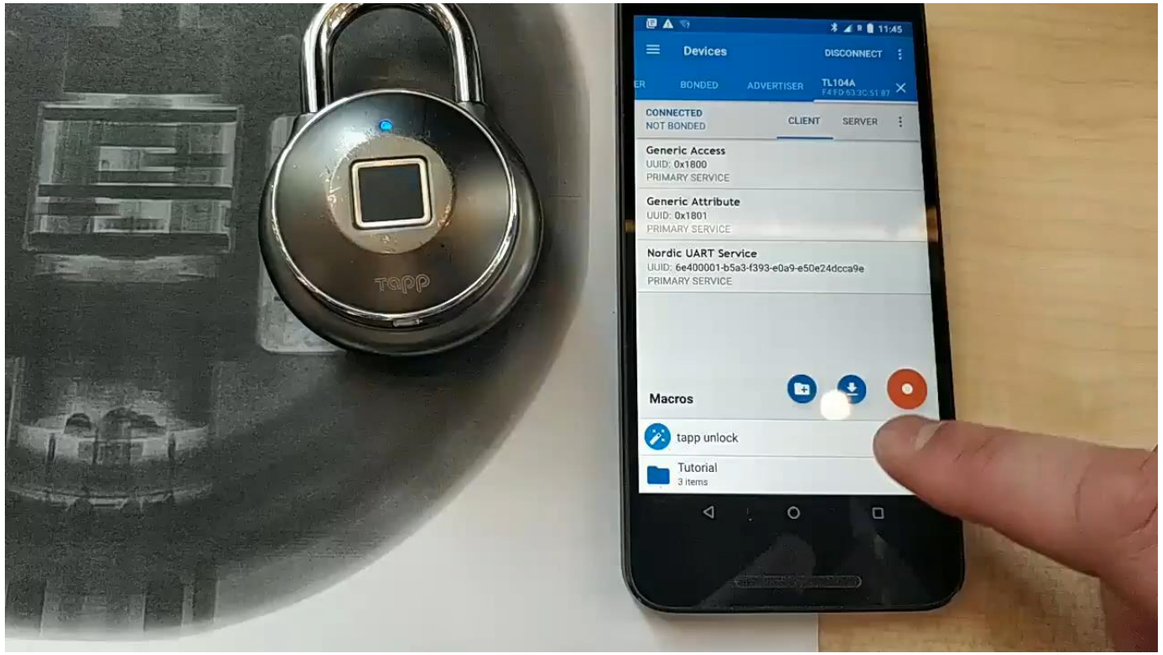
Tapplock nRF Connect macro



https://twitter.com/LucaBongiorni/status/1012671111845294081

https://twitter.com/slawekja/status/1012687779887763456

Works for more of them!



https://twitter.com/slawekja/status/1040177919153397760

Win a BLE sniffer!!!

- First 3 people to solve all HackMe tasks will get

  Adafruit LE Sniffer (nRF)!

- Send:
  - screenshot of “Summary” with all tasks solved*
  - macro scripts for tasks 14 and 15

  hitbchallenge@smartlockpicking.com

* If the app crashes and you can’t solve it, send detailed description

Agenda

1. Introduction to BLE, HackMe lab setup
2. BLE advertisements
   - Packet format, beacons, other advertisements
   - Windows, iOS devices BLE broadcast
   - COVID-19 contact tracing
3. BLE connections
   - GATT services and characteristics
   - Hacking simple devices using just a phone
   - Hacking smart locks
4. What next?

What next?

- Test your new skills on real devices!
- A lot can be done with simple tools and $3 dongle
  - gatttool BlueZ command-line
  - BLESuite (python) https://github.com/nccgroup/BLESuite
  - Bettercap BLE https://github.com/bettercap/bettercap

- BLE relay/MITM
  - Mirage https://homepages.laas.fr/rcayre/mirage-documentation/
  - BtleJuice https://github.com/DigitalSecurity/btlejuice
  - Gattacker https://github.com/securing/gattacker/

Bluetooth link-layer encryption?

Crack PIN pairing: CrackLE

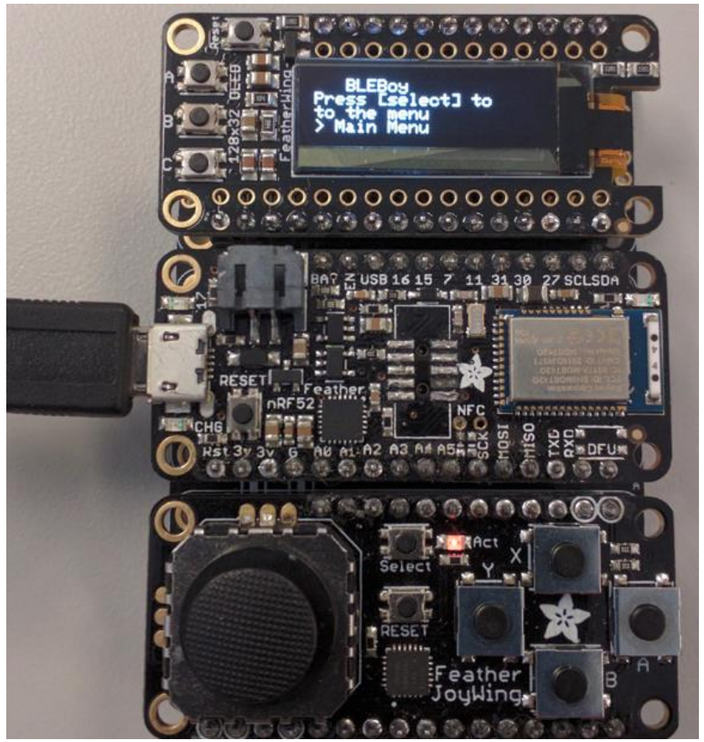
https://github.com/mikeryan/crackle

Test various pairing methods

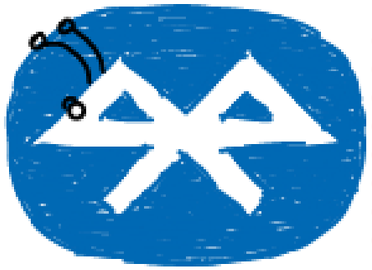
https://github.com/nccgroup/BLEBoy

Nice intro to complex BLE security

https://duo.com/decipher/understanding-bluetooth-security



Bluetooth related vulnerabilities?



CVE-2020-6616 (Broadcom) PRNG
CVE-2020-0022 (Android) RCE BlueFrag
CVE-2019-18614 (Broadcom) RCE
CVE-2019-15063 (Broadcom) reboot iOS and Android
CVE-2019-13916 (Broadcom) RCE
CVE-2019-11516 (Broadcom) RCE
CVE-2019-6994 (Broadcom) crash
CVE-2018-19860 (Broadcom) RCE

BLE Spoofing
Attack BLESA

BLEEDINGTOOTH

CVE-2020-0022 an Android 8.0-9.0
Bluetooth Zero-Click RCE – BlueFrag

Critical Bluetooth BIAS Attack
Let Hackers Access Billions of Devices

KNOB Attack
Key Negotiation of Bluetooth Attack: Breaking Bluetooth Security

What else?

- BLE CTF running on ESP32 by Ryan Holeman @hackgnar
  https://github.com/hackgnar/ble_ctf
- My old “hackmelock” (linux/rpi + android mobile app):
  https://smartlockpicking.com/hackmelock/
- Check www.smartlockpicking.com for updates - new tutorials, trainings, hacking smart locks, new HackMe features…

Next online trainings (BLE, NFC/RFID) probably Feb’21:

HITBSECTRAIN https://sectrain.hitb.org

Thank You

Sławomir Jasek, slawomir.jasek@smartlockpicking.com

See you at HITB's Discord channel for questions & answers!

HITB+CyberWeek UAE

Virtual Edition, 15-19 November 2020