



DECEMBER 9-10  
ARSENAL

# Bluetooth Low Energy hardware-less HackMe

Hands-on introduction to BLE security without any special hardware

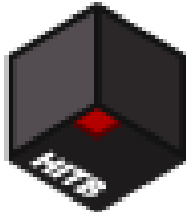
Slawomir.Jasek@smartlockpicking.com @slawekja



#BHEU @BLACKHATEVENTS



#BLE  HITB<sup>+</sup> CyberWeek #SDR #RFID

**Slawomir Jasek**  
**SMARTLOCKPICKING.COM** 

 HITB SecConf #NFC DEEPSEC #KNX

 #smartlock #embedded  GATTack.io #IoT

*OUTSMART THE THINGS*



#HCE



**hardwear.io**  
Hardware Security Conference and Training

# **Bluetooth**<sup>®</sup> Classic



# **Bluetooth**<sup>™</sup> 4.0 Low Energy





OPEN SESAME —

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

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

[Home](#) | [US Election](#) | [Coronavirus](#) | [Video](#) | [World](#) | [UK](#) | [Business](#)

[Tech](#)

## Smart lock can be hacked 'in seconds'

engadget

[Reviews](#) [Gear](#) [Gaming](#) [Entertainment](#) [Products](#) [Tomorrow](#) [Audio](#) [Video](#) [Deals](#)

# — Researcher finds huge security flaws in Bluetooth locks

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

[Smart Home](#)

## 75 Percent of Bluetooth Smart Locks Can Be Hacked

# BLE HACKME



Step-by-step **hands-on** introduction to BLE technology

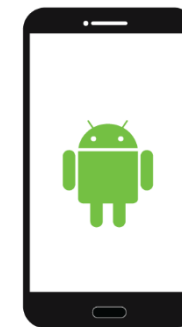
**Practical** challenges with increasing complexity level

Devices simulated on **standard laptop**'s Bluetooth adapter, visible via radio just **like real** ones

Android **phone** as surprisingly effective "hacking tool"

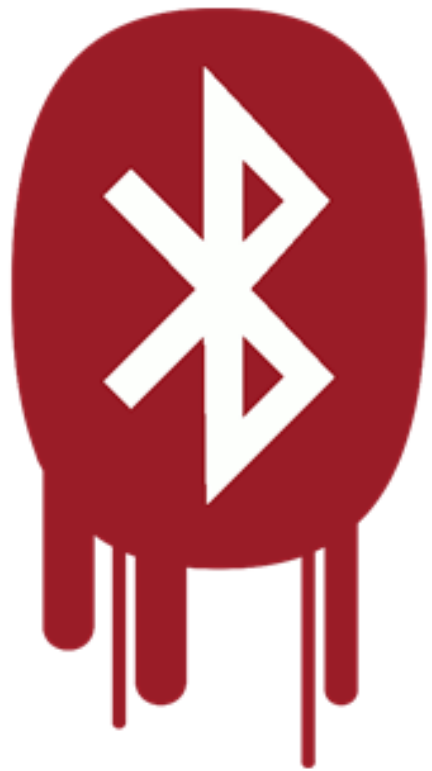
New skills easily **applicable to real devices**.

Learn by having **fun!**



# Install HackMe from Microsoft Store

<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

More



EVERYONE

Free

Get



⚠ See System Requirements

## Note

MS Store current discussion for compliance:



<http://www.pngall.com/?p=32983>

“encouraging illegal activity”...

If you cannot find it in the store, please check:  
[https://www.smartlockpicking.com/ble\\_hackme](https://www.smartlockpicking.com/ble_hackme)

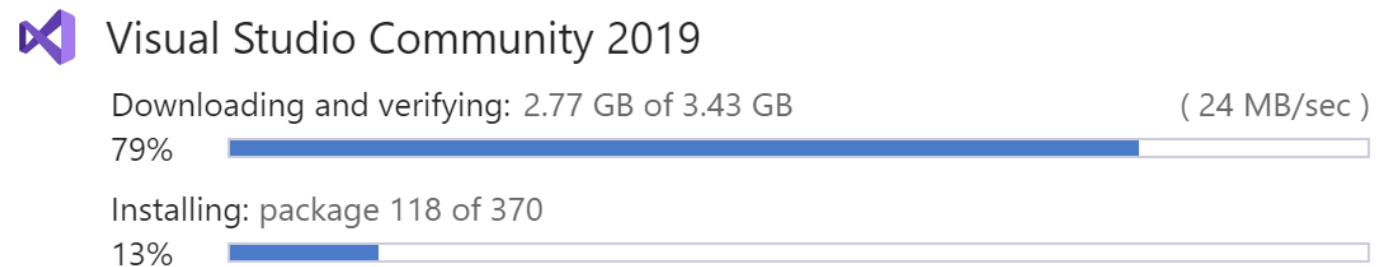


# Source code (MIT license)

[https://github.com/smartlockpicking/BLE\\_HackMe](https://github.com/smartlockpicking/BLE_HackMe)

Building:

Visual Studio (free Community edition)



UWP development



# Visual Studio: build&debug



Open a project or solution

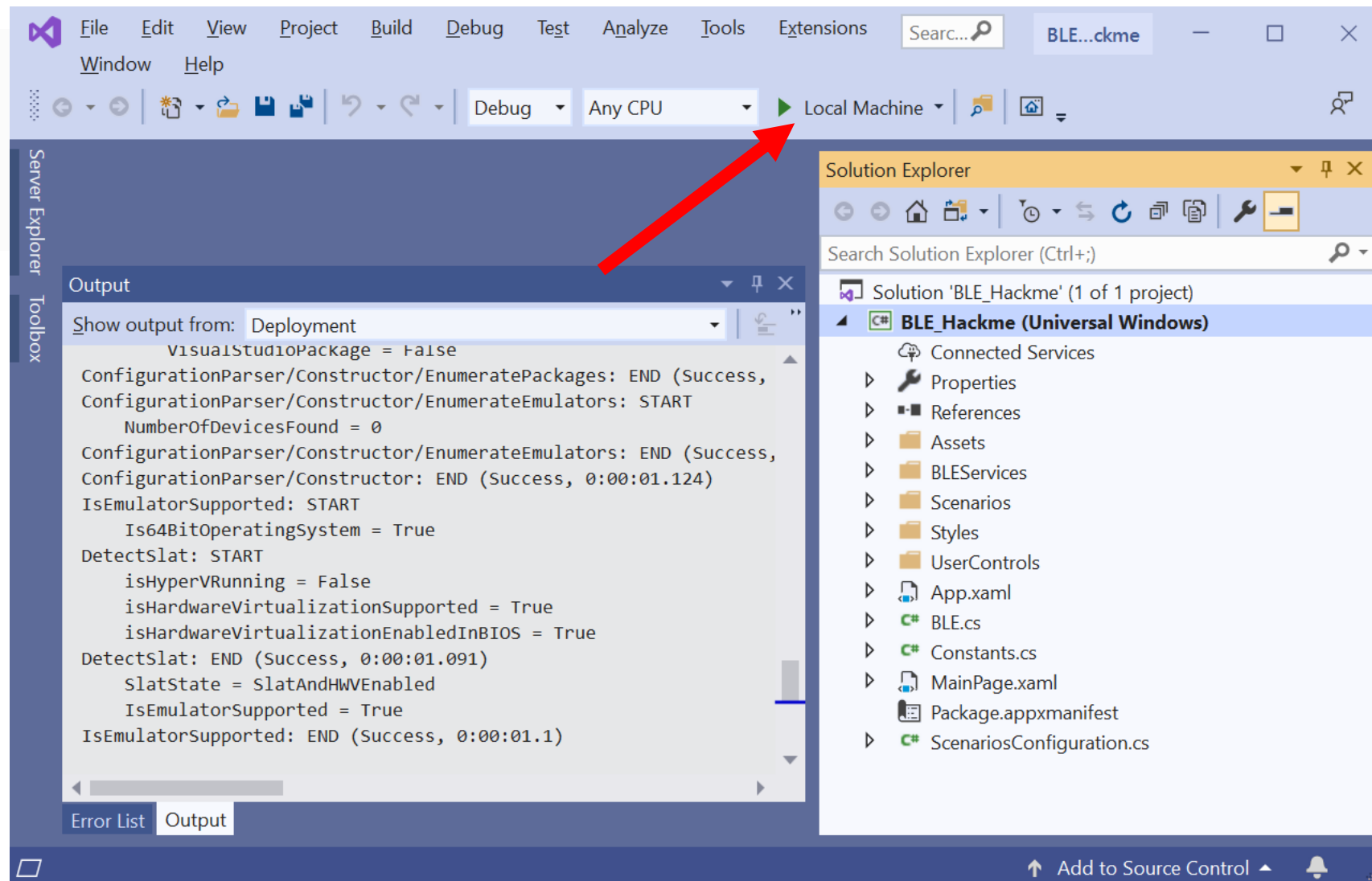
Open a local Visual Studio project or .sln file



BLE\_Hackme



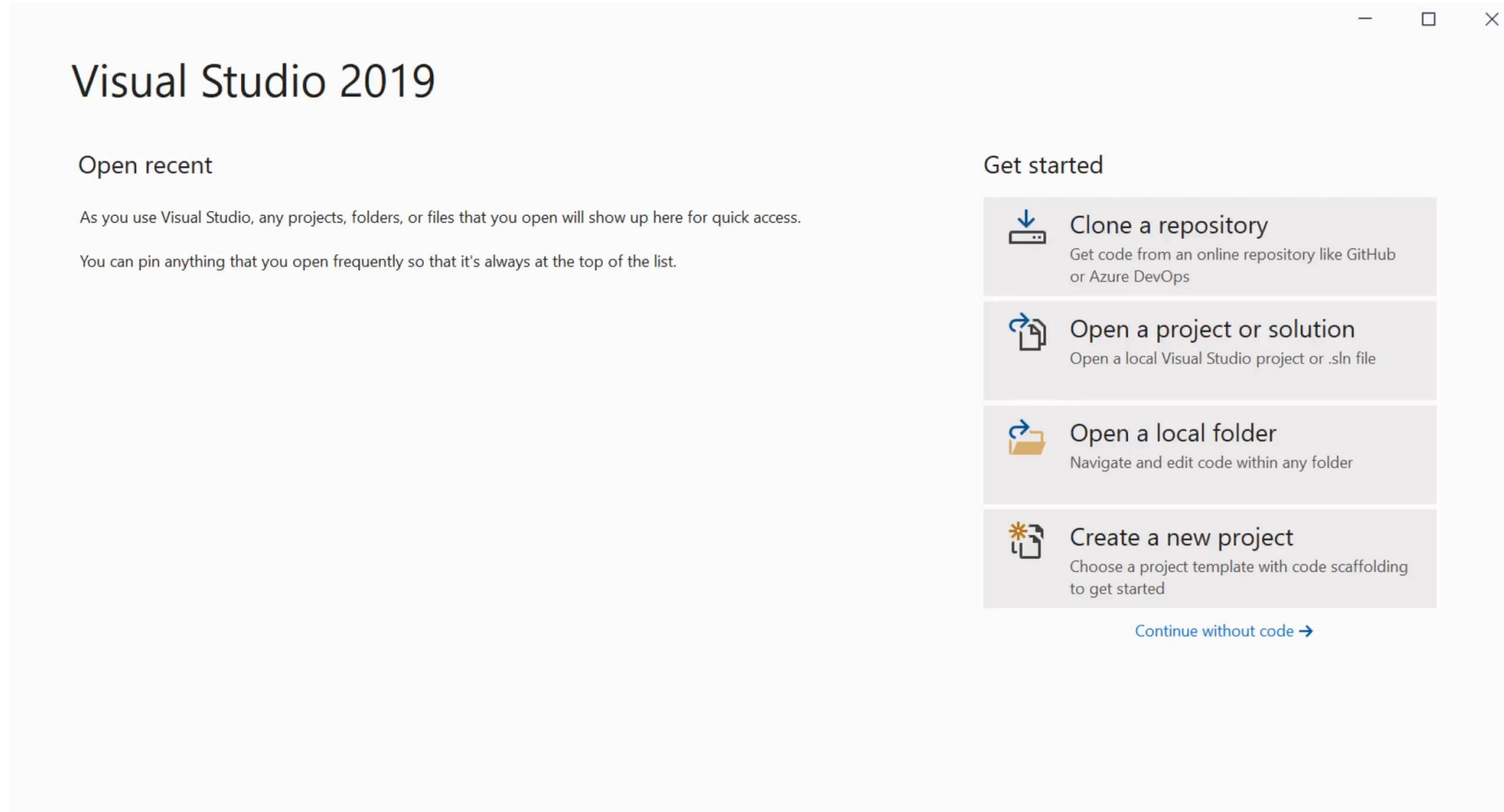
BLE\_Hackme.sln



# VS will ask for developer mode

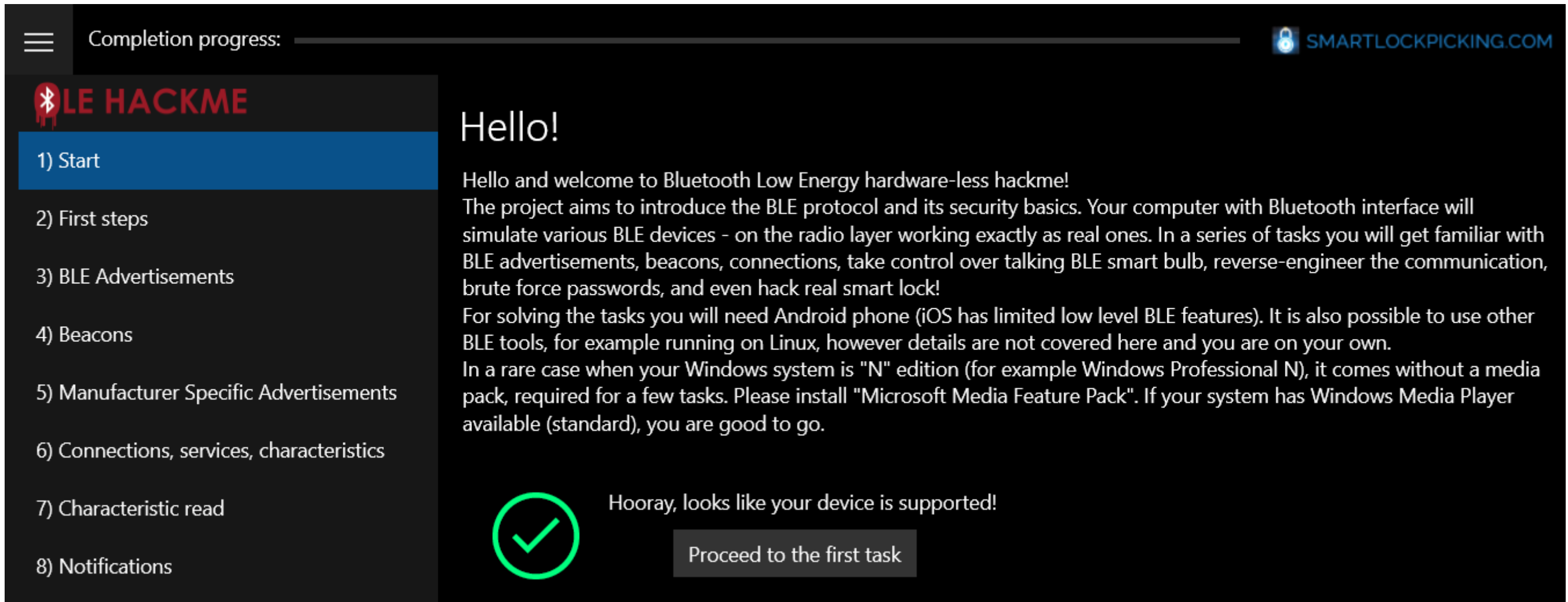
The image shows a composite screenshot. On the left is the Visual Studio interface with the Output window displaying the error: "DEP0100: Please ensure that target device has developer mode enabled. Could not ===== Deploy: 0 succeeded, 1 failed, 0 skipped =====". On the right is the Windows Settings app, specifically the 'For developers' section. The 'Developer Mode' toggle switch is currently turned off, and a red arrow points to it. Below it, the 'Device Portal' and 'Device discovery' toggles are also shown as off.

# Visual studio: build&debug demo



<https://youtu.be/F9GejjagKOY>

# Initial compatibility check



The screenshot shows a web application interface for a project titled "BLE HACKME". At the top, there is a "Completion progress:" indicator and a logo for "SMARTLOCKPICKING.COM". The main content area is divided into a left sidebar and a main text area. The sidebar contains a list of steps: 1) Start, 2) First steps, 3) BLE Advertisements, 4) Beacons, 5) Manufacturer Specific Advertisements, 6) Connections, services, characteristics, 7) Characteristic read, and 8) Notifications. The main text area displays a "Hello!" message, a welcome message, and a confirmation message with 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.  
In a rare case when your Windows system is "N" edition (for example Windows Professional N), it comes without a media pack, required for a few tasks. Please install "Microsoft Media Feature Pack". If your system has Windows Media Player available (standard), you are good to go.

 Hooray, looks like your device is supported!

[Proceed to the first task](#)

# Compatibility check fail...



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](#).

FAQ: [https://github.com/smartlockpicking/BLE\\_HackMe/wiki/FAQ](https://github.com/smartlockpicking/BLE_HackMe/wiki/FAQ)

# Sorry, it will not work...

- On some older (> 5 years old) laptops (Bluetooth 4 required)
- With most external Bluetooth dongles (CSR8510)
  - Confirmed working with Realtek 8761B-based ones
  - List will be updated in FAQ
- In VM – unless direct USB pass-through possible to internal Bluetooth adapter (e.g. Lenovo Thinkpad X1 Carbon 7)
  - or compatible dongle (see above)

# Disclaimer

My first ever C# code.

Expect bugs, crashes,  
exceptions...

Some basic functionality (like  
saving progress state) missing in  
the initial release.



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



# Our “hacking tool”: nRF Connect

Android (recommended)



nRF Connect for Mobile

Nordic Semiconductor ASA Tools

Everyone

Add to Wishlist

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

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



**nRF Connect** 4+

The #1 Bluetooth LE utility  
Nordic Semiconductor ASA

★★★★ 4.4, 70 Ratings

Free

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

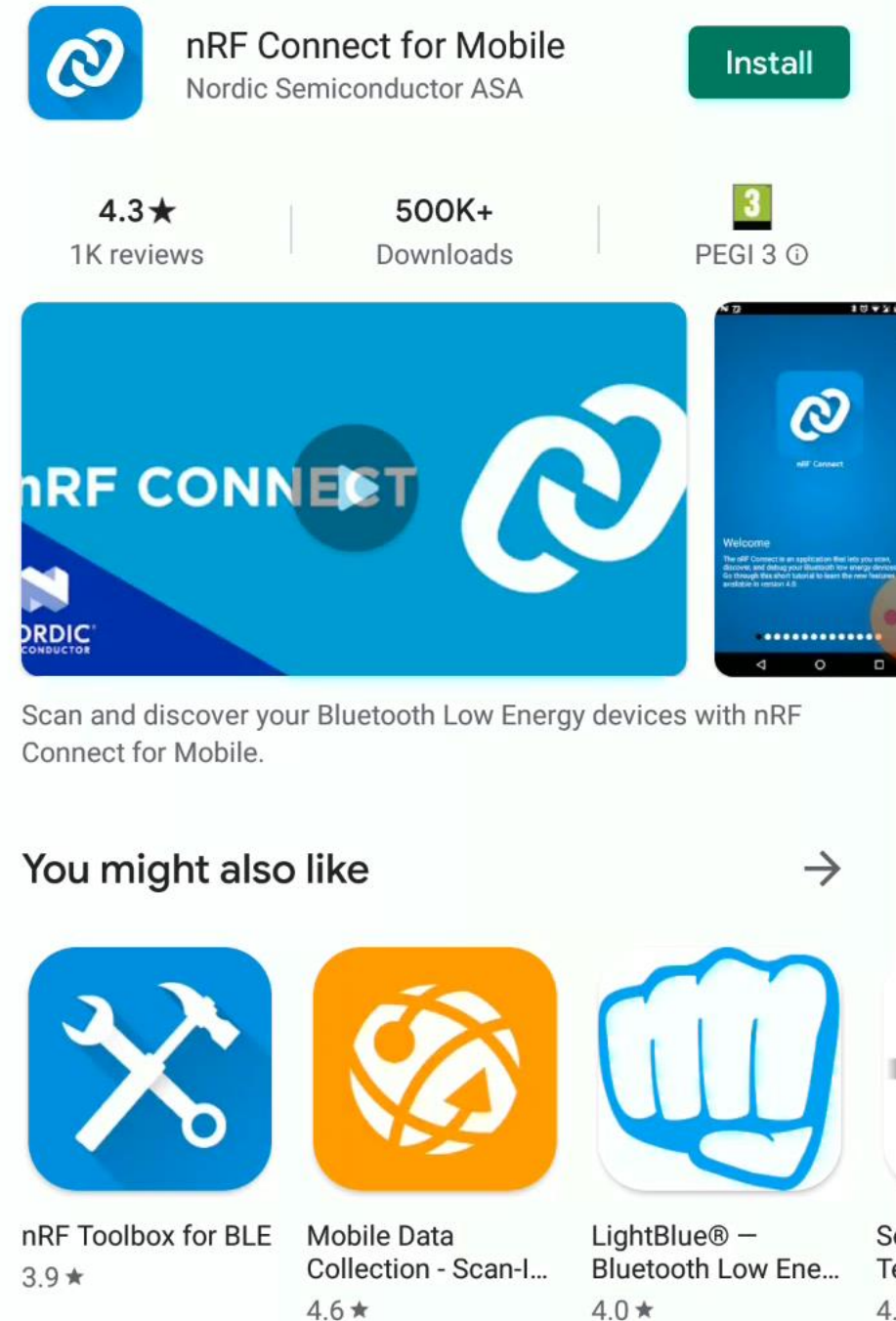
# nRF Connect: permissions

Android requires location permission from apps scanning Bluetooth



Allow nRF Connect to access this device's location?

DENY ALLOW



The screenshot shows the Google Play Store listing for 'nRF Connect for Mobile' by Nordic Semiconductor ASA. The app has a 4.3 star rating from 1K reviews and over 500K downloads. It is rated PEGI 3. The main image shows the app's logo and a play button. Below the image is a description: 'Scan and discover your Bluetooth Low Energy devices with nRF Connect for Mobile.' Underneath, there is a 'You might also like' section with three recommended apps: 'nRF Toolbox for BLE' (3.9 stars), 'Mobile Data Collection - Scan-I...' (4.6 stars), and 'LightBlue® - Bluetooth Low Ene...' (4.0 stars).

- ☰ 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 0.0.9.0  
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

List of tasks

## First steps

### i 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:

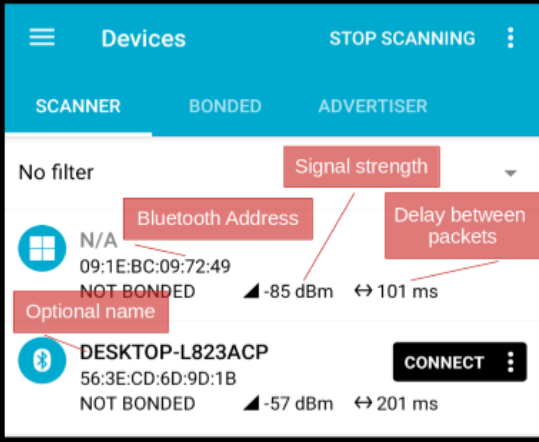
- a **Broadcaster** that transmits some one-way data ("**Advertisement**") to all nearby **Observers** (for example a "beacon" device broadcasting indoor location to nearby phones)
- BLE Client ("**Central**", for example mobile application) to Server ("**Peripheral**", for example smart lock) communication

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.

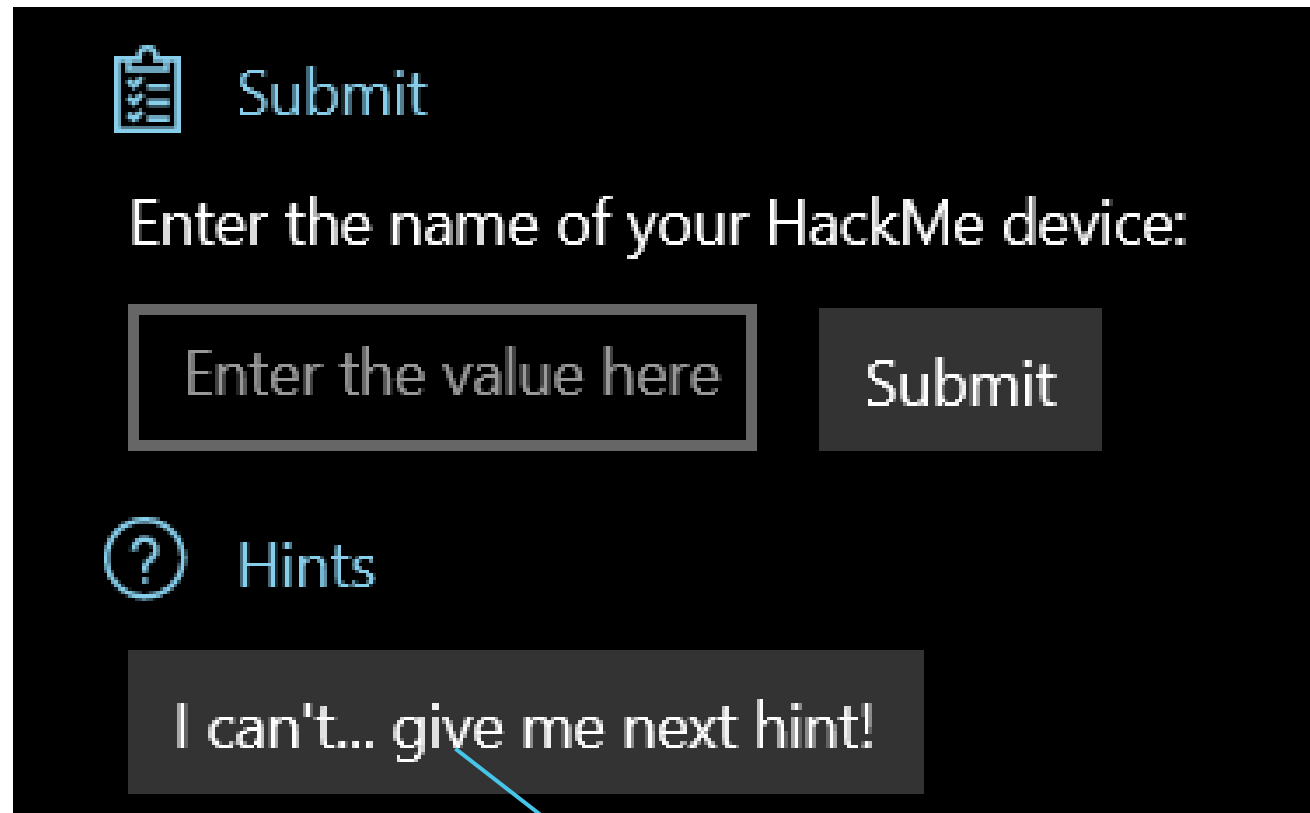
For Bluetooth access, Android [requires](#) location permission from the application, so you will have to grant it during installation. Once started, the application will show nearby BLE devices. Beside optional device name, you will notice the device's adapter address, bonding (pairing) information, 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:



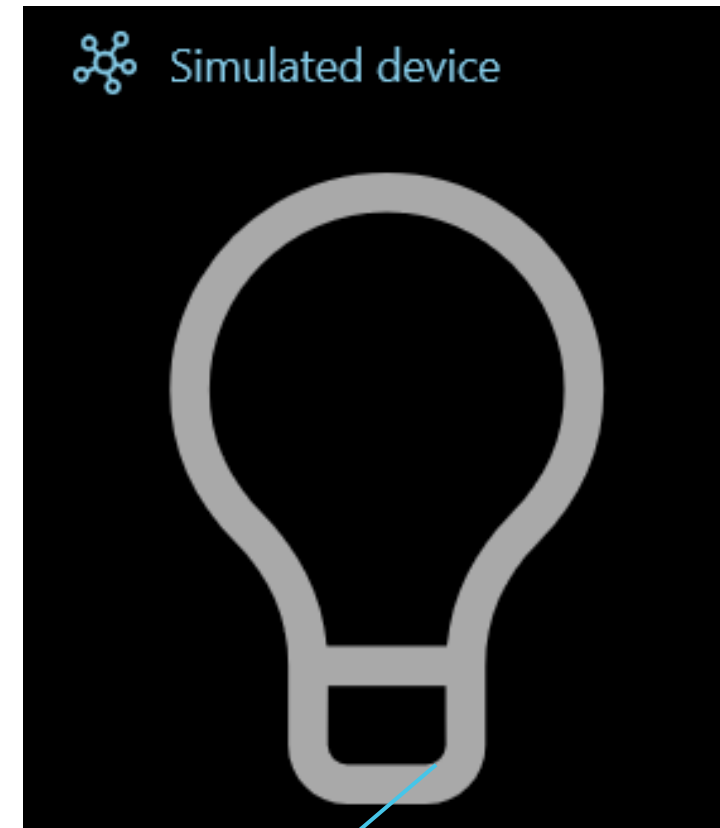
Theory introduction

Task description

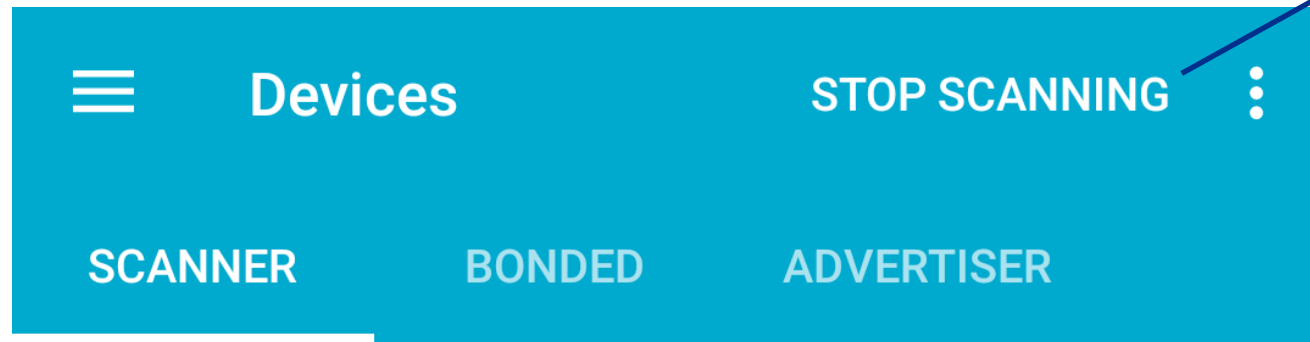
# Submit solution



Spoilers ;)






# nRF Connect: Scan



Will stop after a while, may need to start again




Icon for device type

No filter

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

Delay between packets

Optional device name

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

CONNECT 

Connectable device

Signal strength (lower value=closer)

# Too many devices? Filter!

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

No filter

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

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

CONNECT

Devices STOP SCANNING

SCANNER BONDED ADVERTISER

-60 dBm

Filter by

0x Filter b

Exclude: none

RSSI: -60 dBm

Only favorites

**Filter by signal strength  
(lower = closer)**

Completion progress:  SMARTLOCKPICKING.COM

## BLE HACKME

- Start
- 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

Optional name

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

**CONNECT** ⋮

Depending on your environment, you may see lots of BLE packets, and it might be difficult to locate your HackMe device. One of the ways to limit the discovered devices is to use filtering (select down arrow by the "No filter"), for example based on the signal strength (RSSI). It is measured in decibels (dBm), and the lower value means stronger signal. To match only the nearest devices, slide the RSSI value to about "-60":

**Devices** STOP SCANNING ⋮

SCANNER	BONDED	ADVERTISER
-60 dBm		
🔍 Filter by name or address		⋮ X
📶 0x Filter by raw advertising data		⋮ X
🚫 Exclude: none		⋮ X
📶 RSSI: <span style="display: inline-block; width: 100px; border-bottom: 1px solid #ccc; position: relative;"> <span style="position: absolute; left: 0; top: -10px;">-</span> <span style="position: absolute; right: 0; top: -10px;">-60 dBm</span> </span>		
★ Only favorites		<input type="checkbox"/>

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:

Status:

**Congratulations!**
Proceed to the next task

submit

Completion progress:

## BLE HACKME

- Start
- 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
- Summary

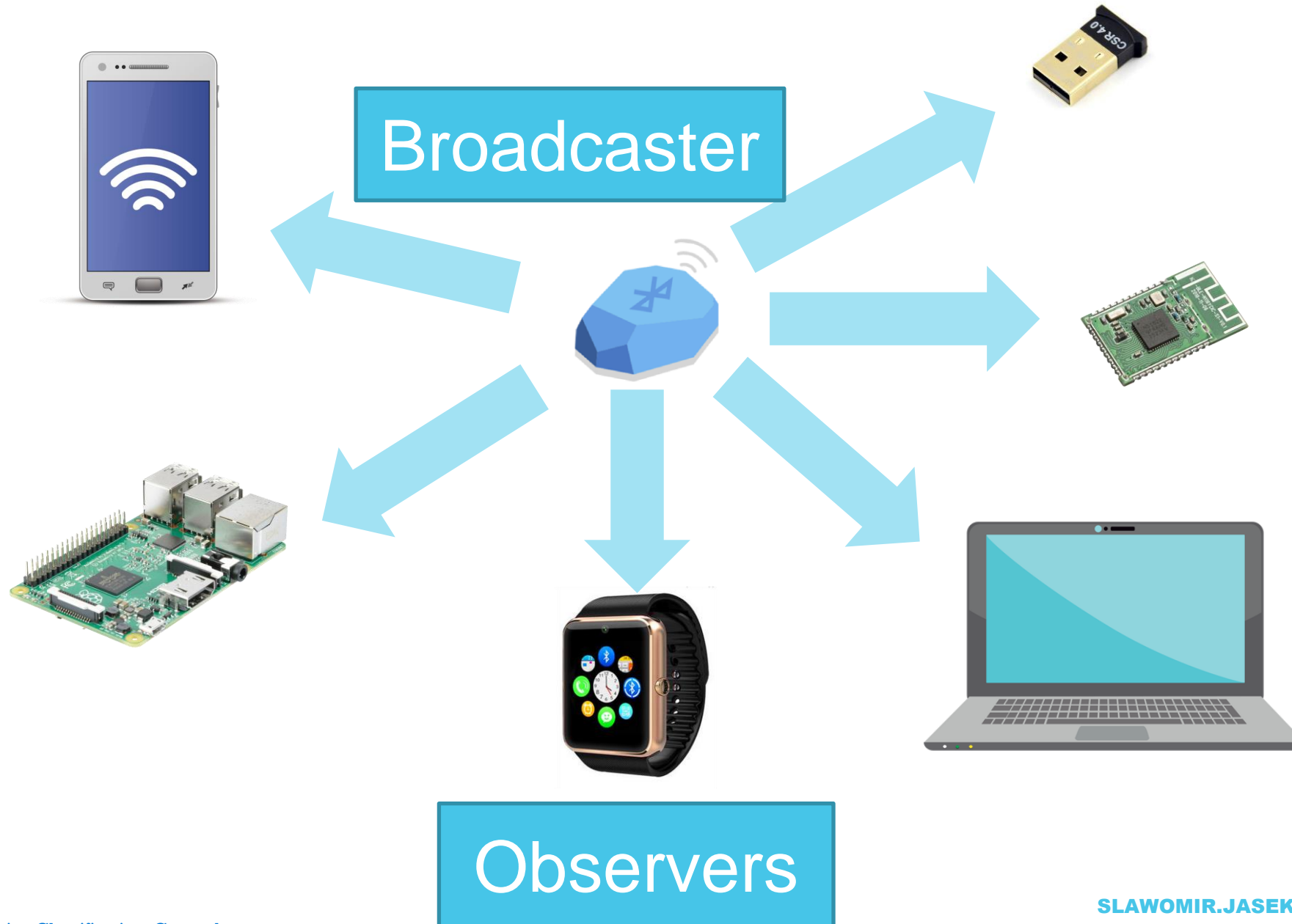
### Summary

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

# BLE advertisements



**Public packets\***  
**No pairing required**

\* except "targeted advertisements" (uncommon)



# BLE advertisements

☰ **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

# Raw hex data



**DESKTOP-L823ACP**

77:D7:EC:A3:E1:C9

NOT BONDED ▲ -49 dBm ↔ N/A

**CONNECT** ⋮

Device type: LE only

Advertising type: Legacy

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

Complete list of 16-bit Service UUIDs: 0x180A, 0x180F

Complete Local Name: DESKTOP-L823ACP

CLONE

**RAW**

MORE



Raw hex bytes transmitted by device

Raw data:

0x02011A05030A180F1810094445534B544F  
502D4C383233414350



Details:

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

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

OK

## 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>

Data Type Value ▲	Data Type Name ▲
-------------------	------------------

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

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

**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:

```
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>

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 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.

**🎯** 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:

Raw data:

```
0x1AFF4C0002156B633468336D4269333463
```

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

Device type: UNKNOWN

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

## 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](#)

# Turn off Windows BLE advertisements

Settings -> Shared experiences

 Shared experiences

Disable both “Nearby sharing”  
and “Share across devices”.

 Nearby sharing

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

Off

Share across devices

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

Off

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

# iBeacon

Transmits

UUID

Two numbers:

- Major
- Minor

Signal strength



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 hex

0xFF

«Manufacturer Specific Data»



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

Raw data:

0x1AFF4C0002156B633468336D4269333463  
306E553144E3DEBC92C8

Details:


LEN.	TYPE	VALUE
26	0xFF	0x4C0002156B633468336D4269333463 306E553144E3DEBC92C8

LEN. - length of EIR packet (Type + Data) in bytes,

TYPE - the data type as in <https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile>

CLONE    RAW    MORE



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



**Bluetooth™**  
**4.0**   
*Low Energy*



# BLE GATT

Generic **AT**Tribute 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 with Battery Level Value.

You will feel it much better in practice!

☰ **Devices** STOP SCANNING ⋮

SCANNER BONDED ADVERTISER

-60 dBm



**DESKTOP-L823ACP**

77:D7:EC:A3:E1:C9

NOT BONDED

▲ -58 dBm ↔ 109 ms

**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

☰ **Devices** DISCONNECT ⋮

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

CONNECTED  
NOT BONDED

CLIENT SERVER ⋮

☰ **Devices** DISCONNECT ⋮

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

CONNECTED  
NOT BONDED

CLIENT SERVER ⋮

**Generic Access**

UUID: 0x1800  
PRIMARY SERVICE

List of services

**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

Characteristics  
in the service

**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



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

# Properties

Read



Notify



Write



Can be combined

The screenshot shows a mobile application interface for managing Bluetooth devices. At the top, there is a teal header with a hamburger 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' (highlighted in teal) and 'NOT BONDED'. Under 'CONNECTED', there are two sub-sections: 'CLIENT' (highlighted with a teal underline) and 'SERVER'. The 'CLIENT' section lists several services with their UUIDs and properties:

- Central Address Resolution**: UUID: 0x2AA6, Properties: READ. A downward arrow icon is to its right.
- 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%. A downward arrow icon is to its right.



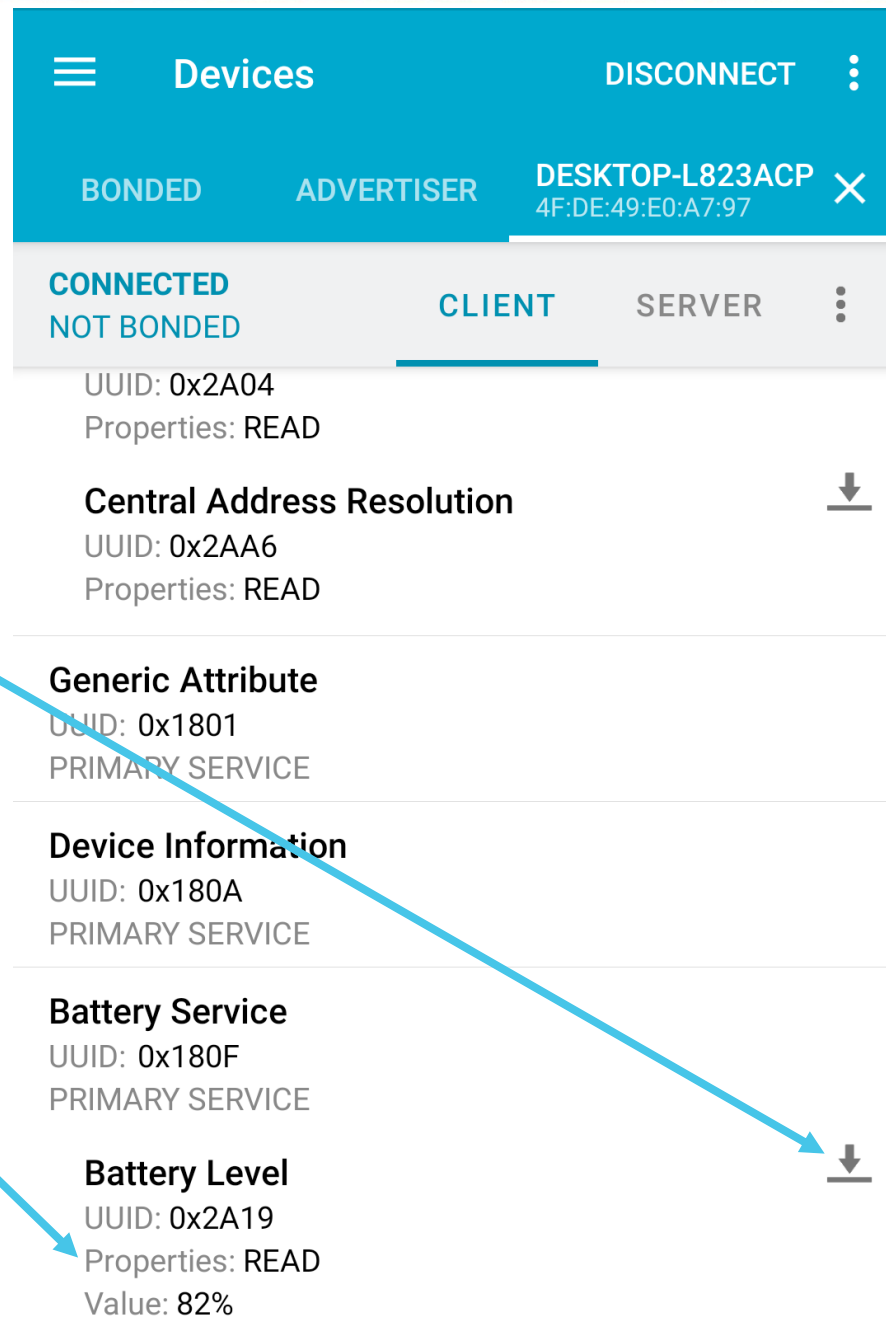
# Properties

Read 

Notify 

Write 

Can be combined




The screenshot shows a mobile application interface for managing Bluetooth devices. At the top, there is a blue header with a hamburger menu icon, the text 'Devices', and a 'DISCONNECT' button with a three-dot menu icon. Below the header, there is a list of device properties for a device named 'DESKTOP-L823ACP' with MAC address '4F:DE:49:E0:A7:97'. The list includes:

- CONNECTED** (status) and **NOT BONDED** (status)
- CLIENT** and **SERVER** (roles)
- UUID: 0x2A04** and **Properties: READ**
- Central Address Resolution** (feature) with a downward arrow icon, **UUID: 0x2AA6**, and **Properties: READ**
- Generic Attribute** (feature) with **UUID: 0x1801** and **PRIMARY SERVICE**
- Device Information** (feature) with **UUID: 0x180A** and **PRIMARY SERVICE**
- Battery Service** (feature) with **UUID: 0x180F** and **PRIMARY SERVICE**
- Battery Level** (feature) with **UUID: 0x2A19**, **Properties: READ**, and **Value: 82%**

Blue arrows from the 'Read' and 'Notify' text on the left point to the 'Properties: READ' labels in the device list.



Completion progress: 

 SMARTLOCKPICKING.COM

## BLE HACKME

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

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

## BLE Notifications

### Theory introduction


Besides reading characteristic value, it is also possible to subscribe for it. The device will automatically send a notification whenever the value update is available, eliminating the need for manual read. There are two types of notifications:


- **Notification** ("NOTIFY" property) - without receive confirmation
- **Indication** ("INDICATE" property) - the receiver confirms packet reception to sending device.

Characteristic can have both Notify and Indicate or just one of them. The difference is just in low level Bluetooth packets, the transmitted data is the same, and for application it makes no difference.

### Task

Connect to your device, find Heart Rate service and its characteristic. Try reading current Heart Rate

Measurement value using single down arrow  . You can try it multiple times to see if the value changes in time.

Next, subscribe to the Heart Rate Measurement characteristic notification using the subscribe button:  . The button will change its status and characteristic value will be updated every second.

Your task is to submit current beats per minute value.

If needed, you can unsubscribe from notifications by tapping again on the same button.

### Submit

Enter the current heart rate (decimal bpm) indicated (+/- 5):

### Hints

# Subscribe 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

Tap to subscribe  
for value change



**Heart Rate**  
UUID: 0x180D  
PRIMARY SERVICE

**Heart Rate Measurement**  
UUID: 0x2A37  
Properties: NOTIFY, READ  
Value: Heart Rate Measurement: 123 bpm,  
Contact is Detected

**Descriptors:**  
Characteristic User Description  
UUID: 0x2901  
Client Characteristic Configuration  
UUID: 0x2902  
Value: Notifications enabled

Value updates  
automatically



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](#)


## 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!

# Write

Read



Notify



**Write**



Can be combined

☰ Devices DISCONNECT ⋮

BONDED	ADVERTISER	DESKTOP-L823ACP 4D:86:E5:D5:2A:B0	✕
CONNECTED	CLIENT	SERVER	⋮
NOT BONDED			
PRIMARY SERVICE			

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

Status:

**Congratulations!** Proceed to the next task

☰ Completion progress:  SMARTLOCKPICKING.COM

**BLE HACKME**

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

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

## Various writes

**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.

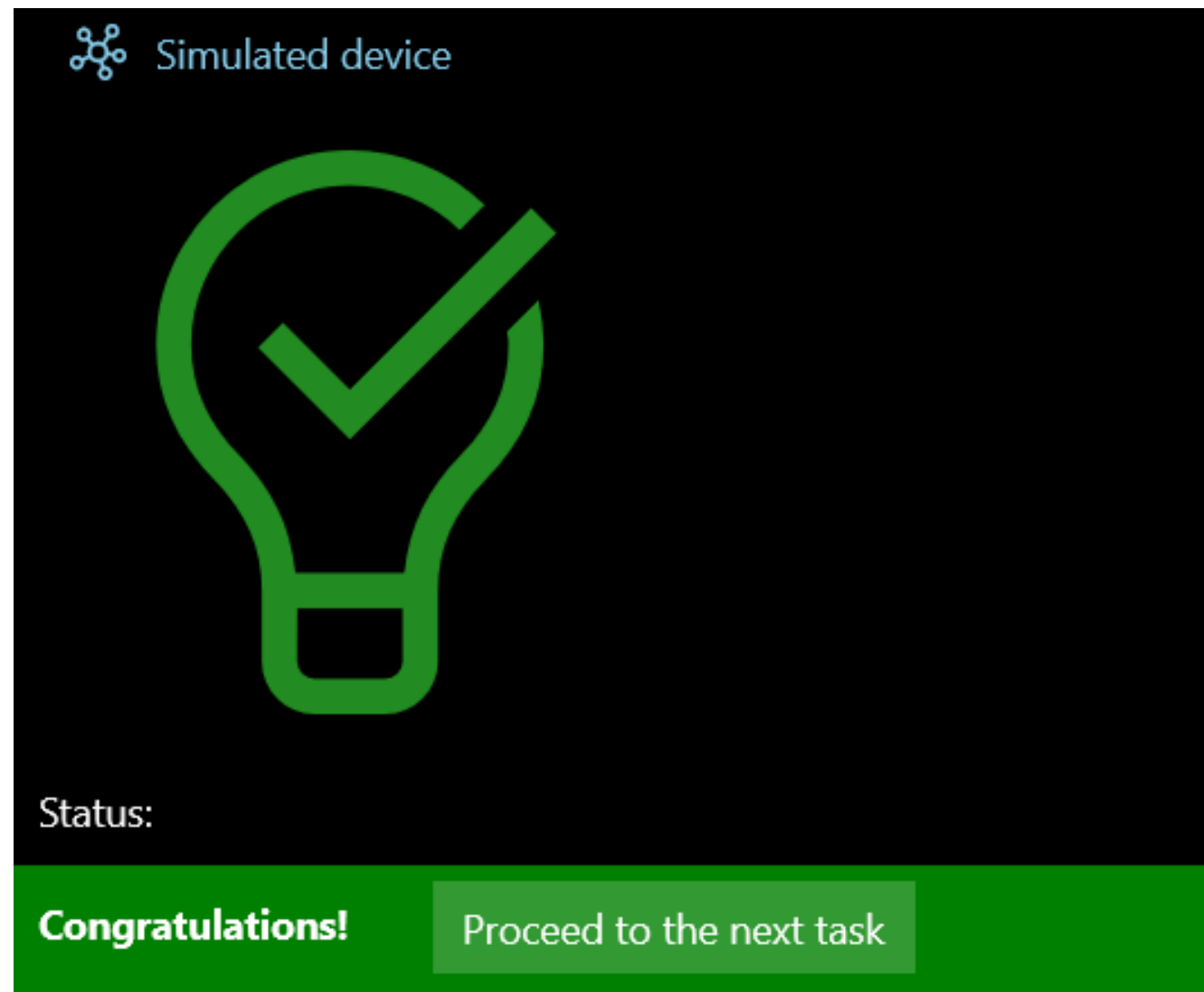
The job consists of few tasks:

- 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".

# Talking BLE smart light bulb





Completion progress:  SMARTLOCKPICKING.COM

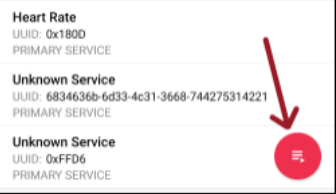
- 1) Start
- 2) First steps
- 3) BLE Advertisements
- 4) Beacons
- 5) Manufacturer Specific Advertisements
- 6) Connections, services, characteristics
- 7) Characteristic read
- 8) Notifications
- 9) Descriptors
- 10) Characteristic write
- 11) Various writes
- 12) Write automation**
- 13) Protocol reverse-engineering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

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

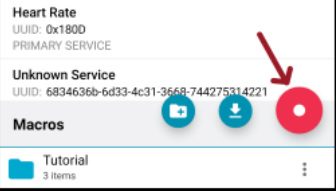
## Write automation

**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.

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

Now send something to device like previously

**Macros**

Add delay  
200 ms

optional delay between requests

Start recording

Stop recording

Save macro as...

Name  
Blink

Choose icon:

CANCEL SAVE



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](#)



Task

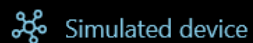
The light bulb has yet another characteristic, which allows to change its color and brightness level. You will surely find it in the light bulb service. The valid data format to send via write is however unknown. Fortunately, there was a mobile application possible to decompile. The decompiled source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

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

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

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

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



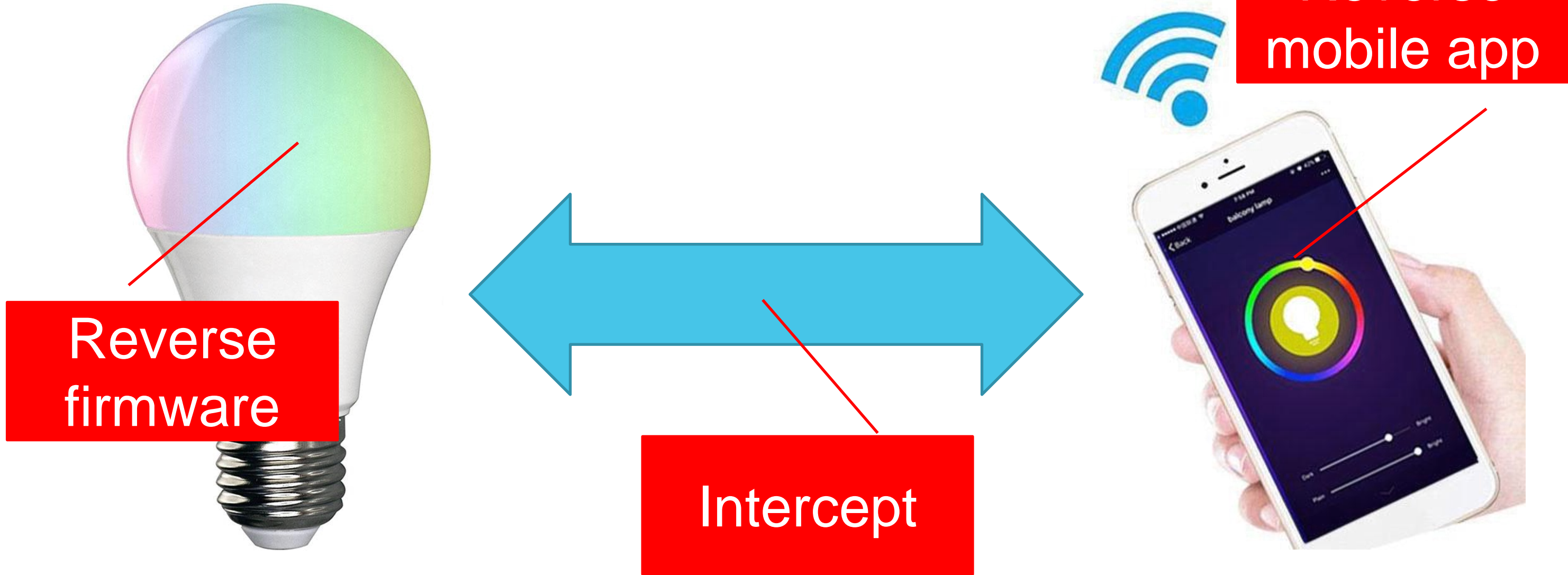
Simulated device



Hints

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

# Data format?



# Mobile app reversing?

Grab the “apk” binary



apk download

Decompile:

- JADX

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

- BytecodeViewer

<https://github.com/Konloch/bytecode-viewer>

- 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;
    private String mTempName = "";
    private 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

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:

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

**Congratulations!**

Proceed to the next task



Completion progress: 

 SMARTLOCKPICKING.COM

## BLE HACKME

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

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

## Password brute force

### Theory introduction

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

### Task

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

Here is the relevant decompiled source code fragment:

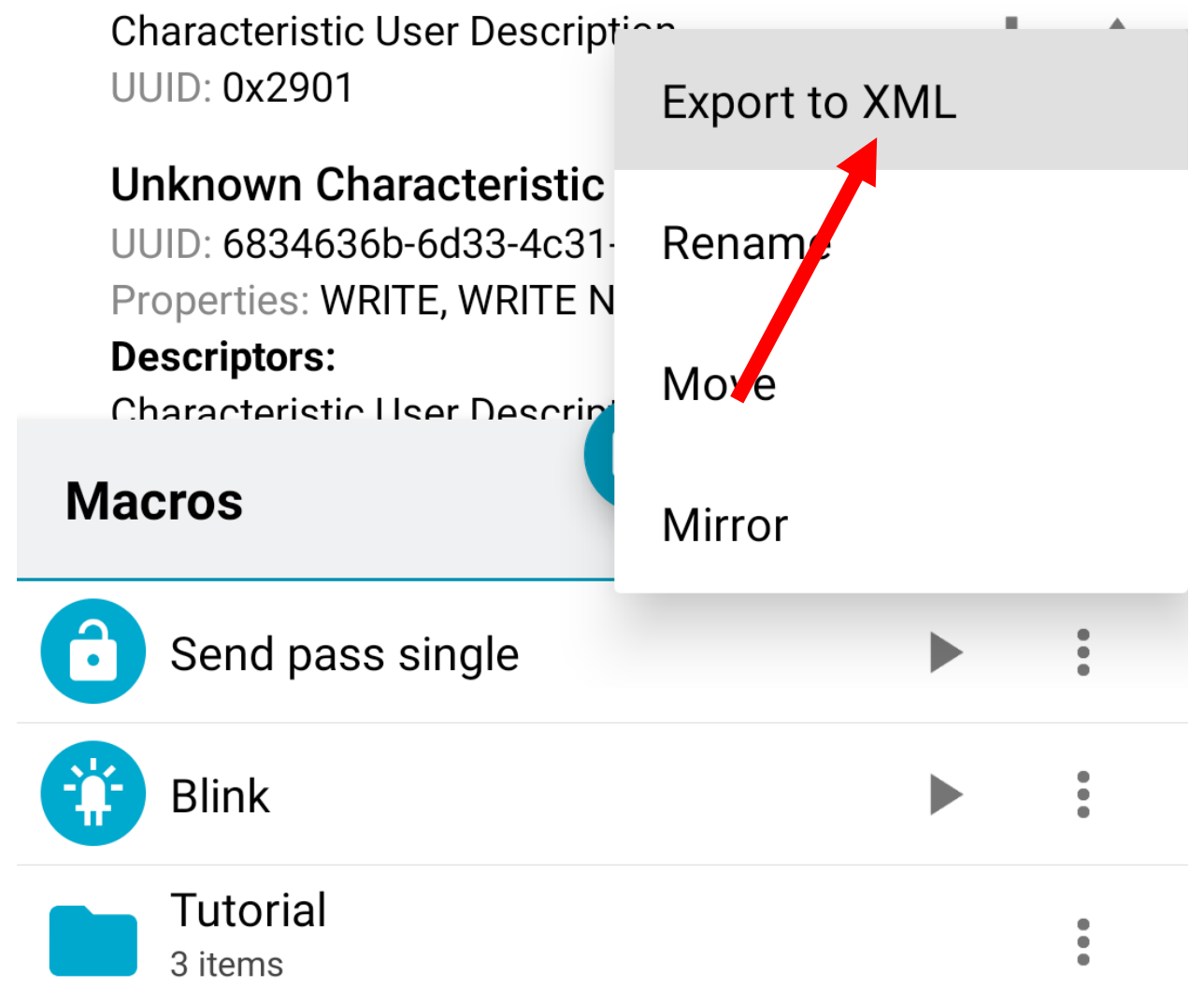
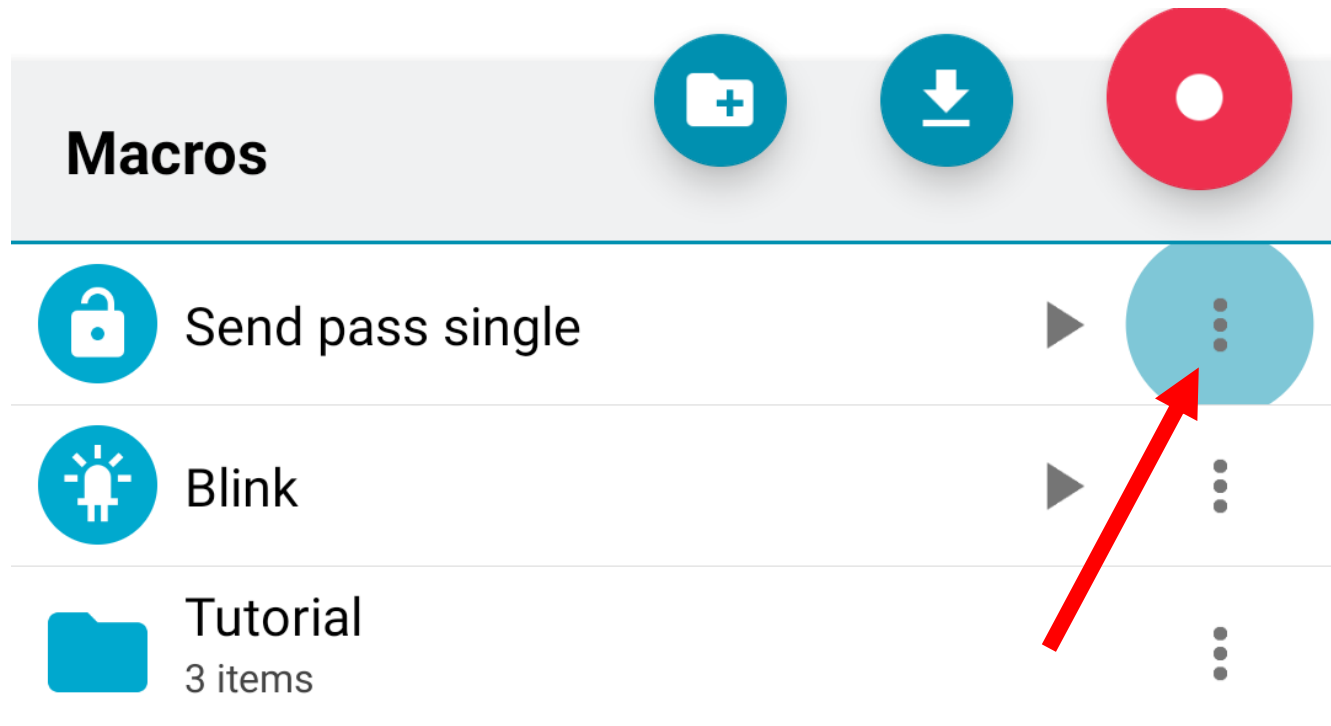
```
public static final byte FRAME_SUFFIX = (byte) -1;
public static final byte FX_FRAME_PREFIX = (byte) -66;
public static final byte FX_ON = (byte) 1;
public static final byte FX_OFF = (byte) 0;

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

Your job is to:

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

# Macros can be exported and edited





# Sample macro (XML) file

Ensure matching characteristics available (optional)

```
<macro name="Blink" icon="LED_ON">  
  <assert-service description="Ensure 6834636b-6d33-4c31-3668-744275314221 service" uuid="(...)"> <property name="WRITE" requirement="MANDATORY"/></assert-characteristic> </assert-service>  
  <write description="Write 0x01" 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 a value to specific characteristic

# Brute password

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




Unknown Characteristic

**Macros**

pass brute

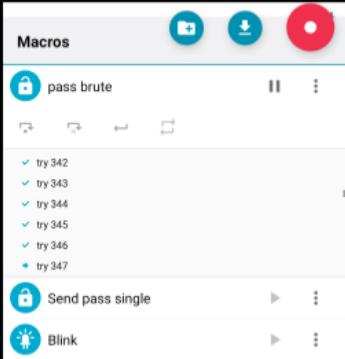
- ✓ try 023
- ✓ try 024
- ✓ try 025
- ✓ try 026
- ✓ try 027
- ➔ try 028

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

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!**

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



Completion progress:

SMARTLOCKPICKING.COM

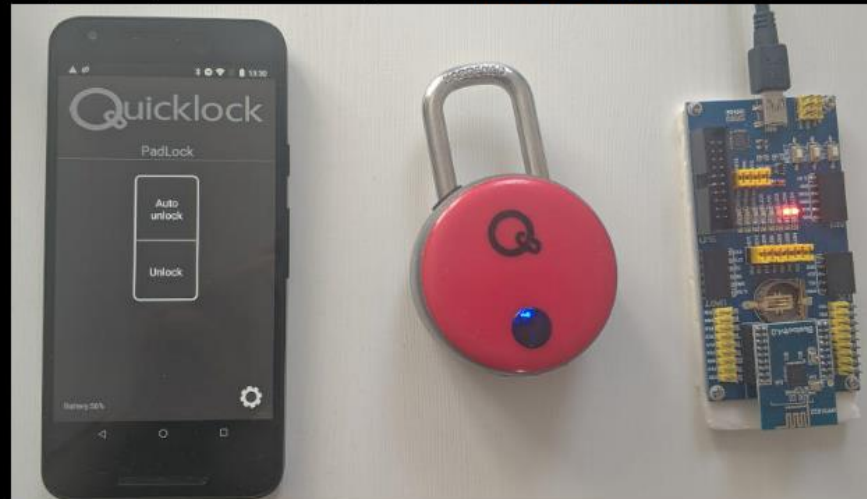
## BLE HACKME

- 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

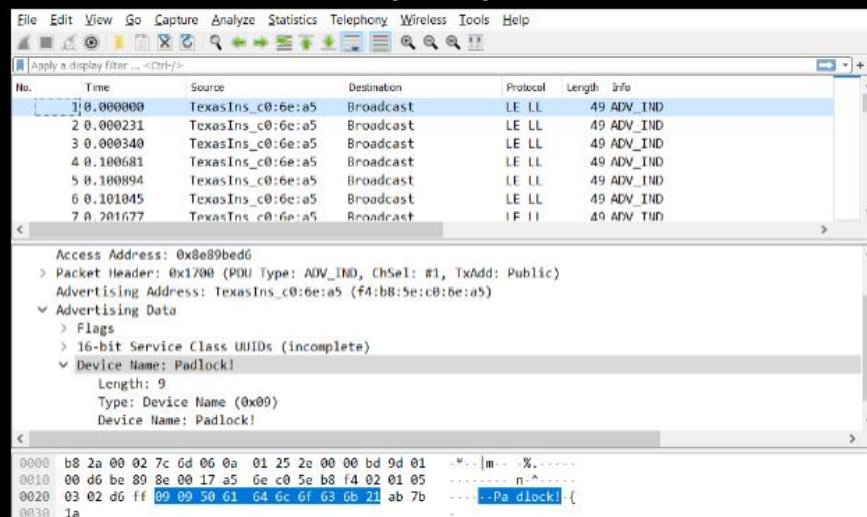
© smartlockpicking.com  
[More information](#) | [FAQ](#) | [Source code](#) | [Issues](#)

### 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:





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

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

Anthony Rose & Ben Ramsey



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

## >>> Plain Text Passwords

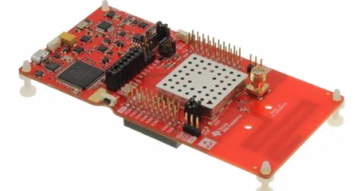
- \* Are they even trying?
- \* Found on 4 separate locks
  - Quicklock Doorlock
  - Quicklock Padlock
  - iBluLock Padlock
  - Plantraco Phantomlock

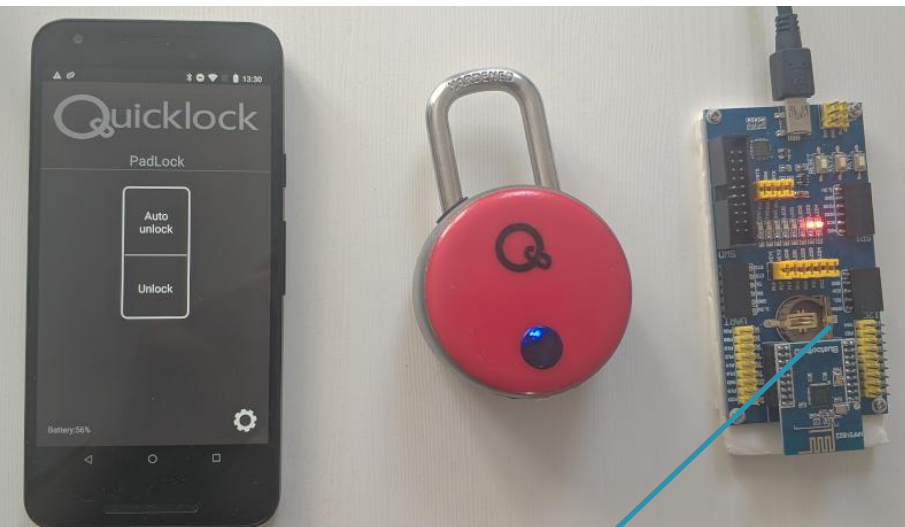


```
▶ Frame 278: 49 bytes on wire (392 bits)
▶ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low Energy)
▶ Bluetooth Low Energy Link Layer
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Request (0x12)
    Handle: 0x002d
    Value: 001234567812345678
```

001234567812345678  
Opcode Current Password New Password

# BLE sniffers


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

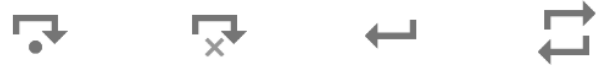


File for “Smart lock replay”  
task analysis was sniffed with  
nRF51822 (nRF sniffer)



# Smart lock 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

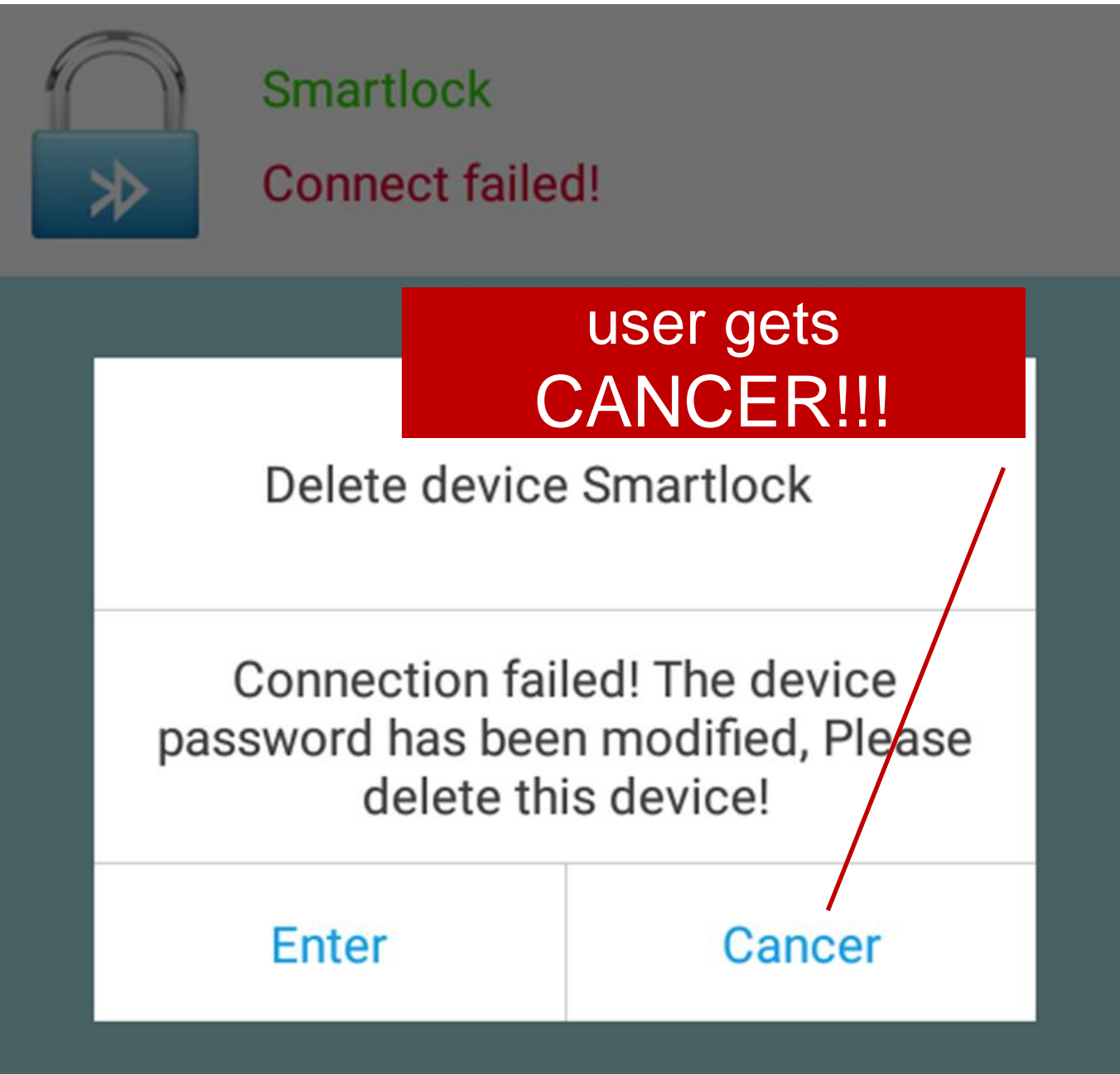
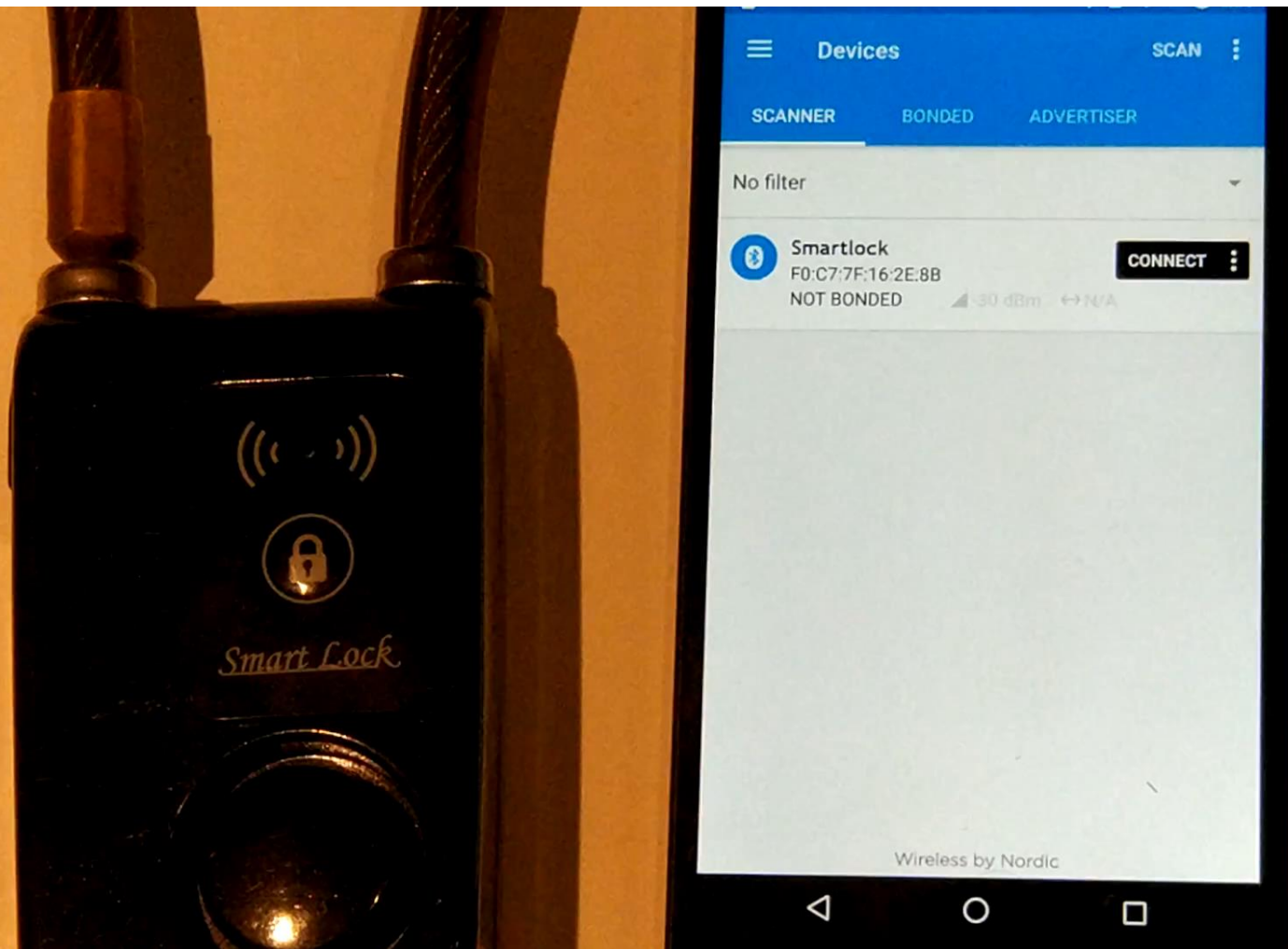


Status:

**Congratulations!**

Proceed to the next task

# Other locks



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

# Tapplock: pass=MD5(MAC)



BLOG: INTERNET OF THINGS

## Totally Pwning the Tapplock Smart Lock

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

Forbes

Jun 13, 2018, 05:25am EDT

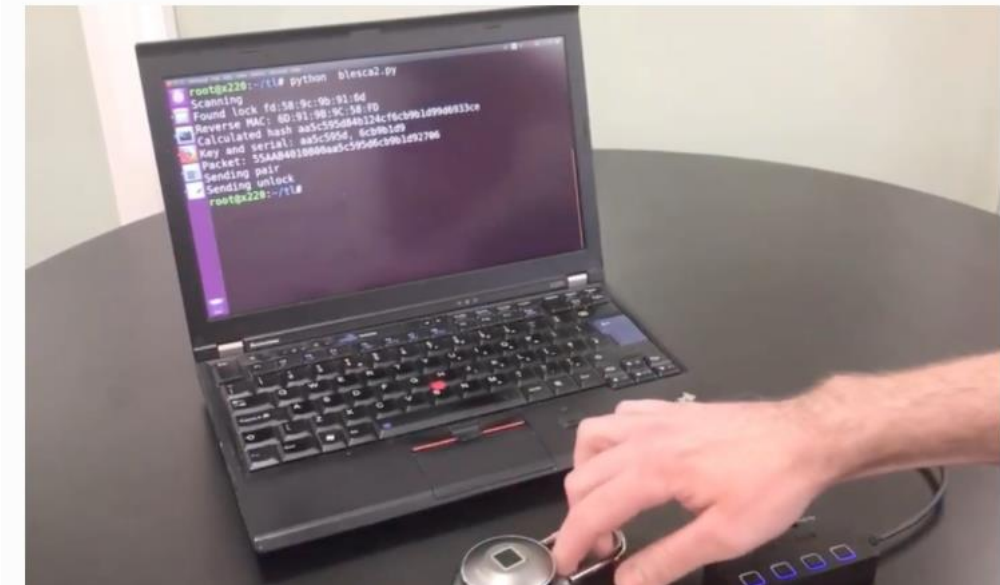
## Tapplock: This \$100 'Smart Lock' Can Be Hacked Open In 2 Seconds



**Thomas Brewster** Forbes Staff  
Cybersecurity


Associate editor at Forbes, covering cybercrime, privacy, security and surveillance.

This article is more than 2 years old.



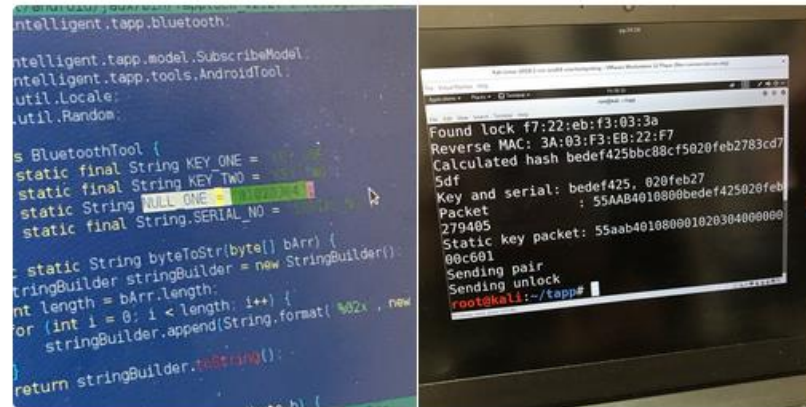
<https://www.forbes.com/sites/thomasbrewster/2018/06/13/tapplock-smart-lock-hacked-in-2-seconds>

# Tapplock (early fw, static password)

 **Luca Bongiorno** @LucaBongiorno Following

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

CC:@cybergibbons



5:16 AM - 29 Jun 2018

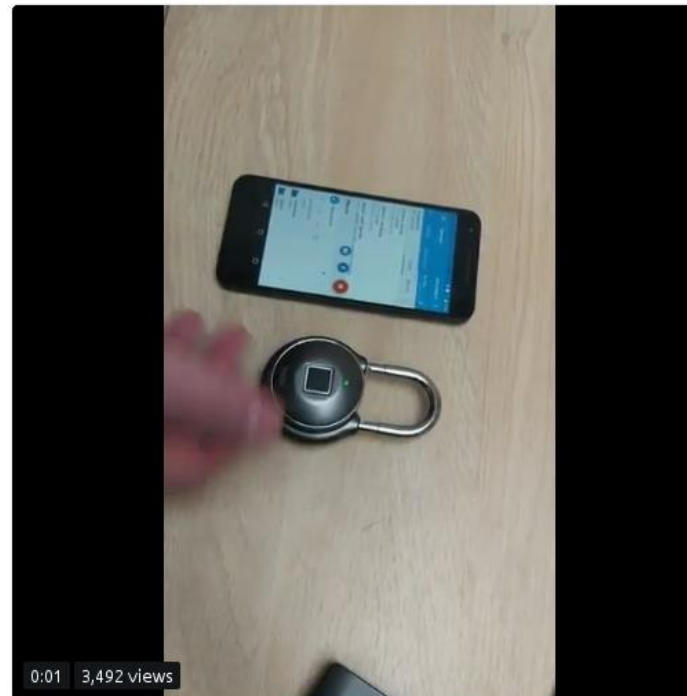
28 Retweets 59 Likes



9 28 59

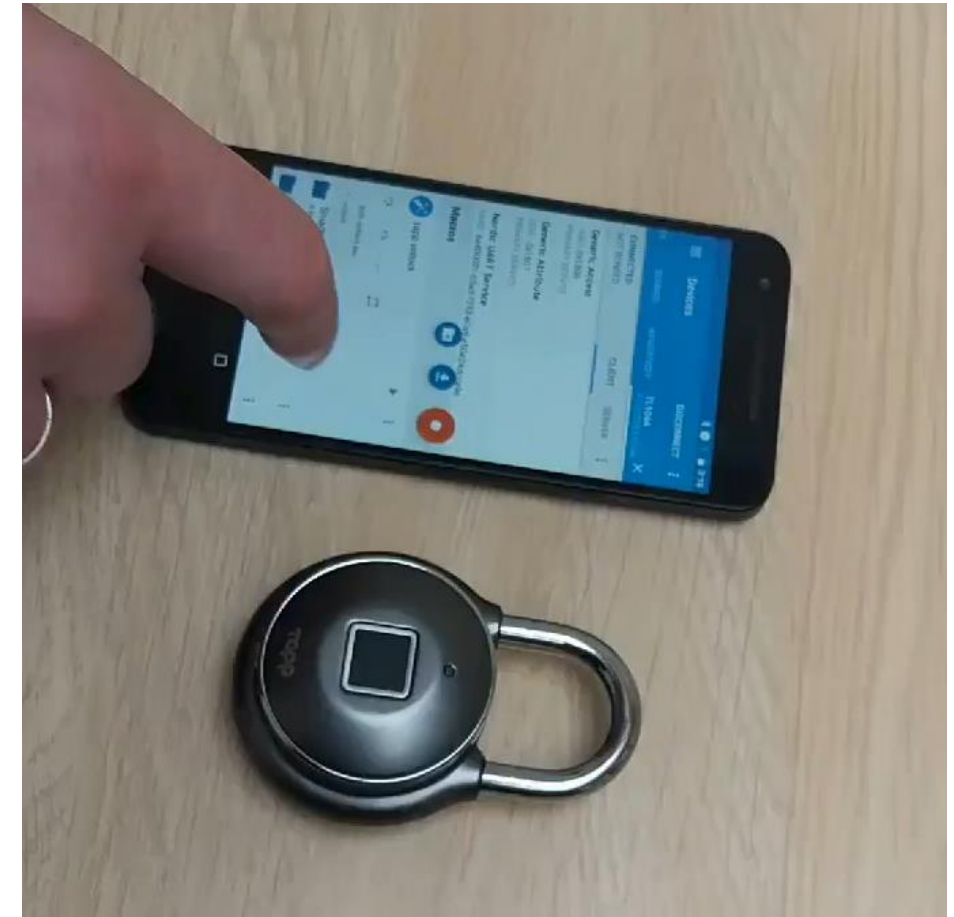
 **Slawomir Jasek** @slawekja

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



6:22 AM - 29 Jun 2018

49 Retweets 93 Likes



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

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

## What else?

BLE CTF running on ESP32 by Ryan Holeman @hackgnar

[https://github.com/hackgnar/ble\\_ctf](https://github.com/hackgnar/ble_ctf)

My old “hackmelock” (linux/rpi + android mobile app)

<https://smartlockpicking.com/hackmelock/>



Check [www.smartlockpicking.com](http://www.smartlockpicking.com)



new tutorials, trainings, hacking smart locks...

  
**black hat**<sup>®</sup>  
EUROPE 2020

DECEMBER 9-10  
A R S E N A L

# Questions?

Slawomir.Jasek@smartlockpicking.com @slawekja

[https://smartlockpicking.com/ble\\_hackme](https://smartlockpicking.com/ble_hackme)

#BHEU @BLACKHATEVENTS