

Open Bar at the Playground

Nullcon Goa 2023 | @SamZorSec

Agenda

- Introduction
 - The Application
 - The Wireless Communications
 - Conclusion
 - Q & A
-

Introduction

Who I am

- Samuel M.
- Fribourg, Switzerland
- Sr. Security Engineer @ Kudelski IoT
- Interests
 - Home automation
 - Hardware security
 - Bluetooth Low Energy
- Twitter / X: @SamZorSec

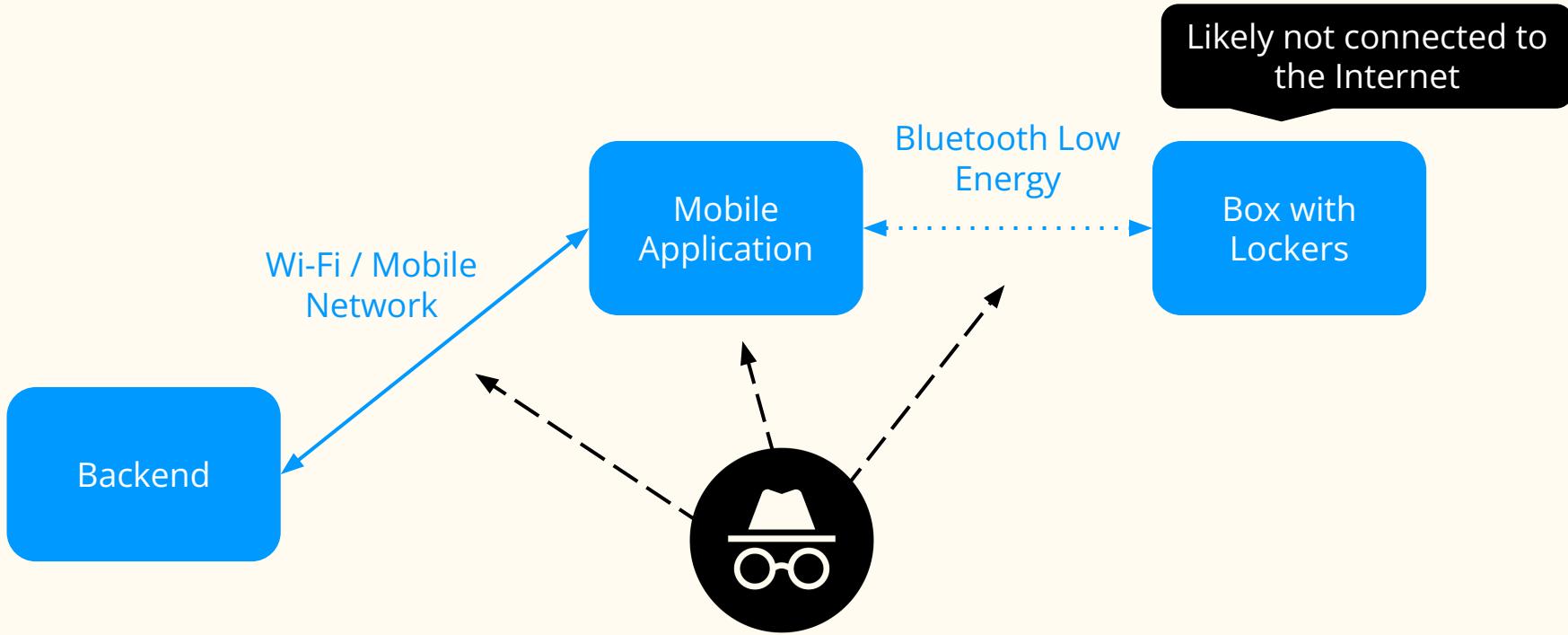


KUDELSKI
IOT THINGS

The Solution

- An **Android** or an iOS application is required
- An account is **mandatory** and the registration requires to **scan an ID card**
- Free
- A box has several lockers
- Rental conditions
 - Maximum duration of **3 hours**
 - Opening hours from **8AM to 10PM**
 - **1 article** at a time

The Solution



Is it possible to
open a locker
without an
account?

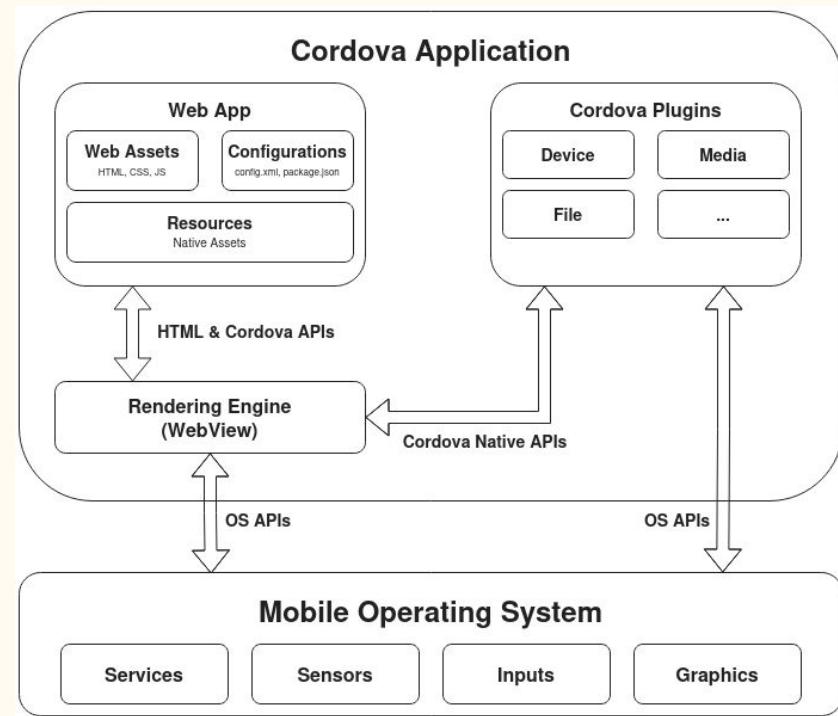
The Application

The Application

- Introduction to Cordova Applications
 - Introduction
 - Security
 - Tools
 - Data Acquisition
 - Data Analysis
 - Findings
-

Cordova > Introduction

- **Mobile development framework** to create cross-platform applications (Android, iOS and Electron)
- Open-source
- Development with **Web technologies** (HTML5, CSS3 and JavaScript)
- Extensible with plugins
 - Battery Status
 - Camera
 - Geolocation
 - Etc.



Cordova > Security



Do not assume that your source code is secure

- Since a Cordova application is built from HTML and JavaScript assets that get packaged in a native container, you should not consider your code to be secure. It is possible to reverse engineer a Cordova application.

Certificate Pinning

Cordova does not support true certificate pinning. The main barrier to this is a lack of native APIs in Android for intercepting SSL connections to perform the check of the server's certificate. (Although it is possible to do certificate pinning on Android in Java using JSSE, the webview on Android is written in C++, and server connections are handled for you by the webview, so it is not possible to use Java and JSSE there.) Since Apache Cordova is meant to offer consistent APIs across multiple platforms, not having a capability in a major platform breaks that consistency.

There are ways to approximate certificate pinning, such as checking the server's public key (fingerprint) is the expected value when your application starts or at other various times during your application's lifetime. There are third-party plugins available for Cordova that can do that. However, this is not the same as true certificate pinning which automatically verifies the expected value on every connection to the server.

There are also plugins that can do true certificate pinning for some platforms, assuming your app is able to do all of its network requests using the plugin (i.e.: no traditional XHR/AJAX requests, etc).

When running Cordova on Android, using `android:debuggable="true"` in the application manifest will permit SSL errors such as certificate chain validation errors on self-signed certs. So you can use self-signed certs in this configuration, but this is not a configuration that should be used when your application is in production. It is meant to be used only during application development.

Cordova > Tools

- Apktool
- Frida
- Medusa
- Chrome DevTools

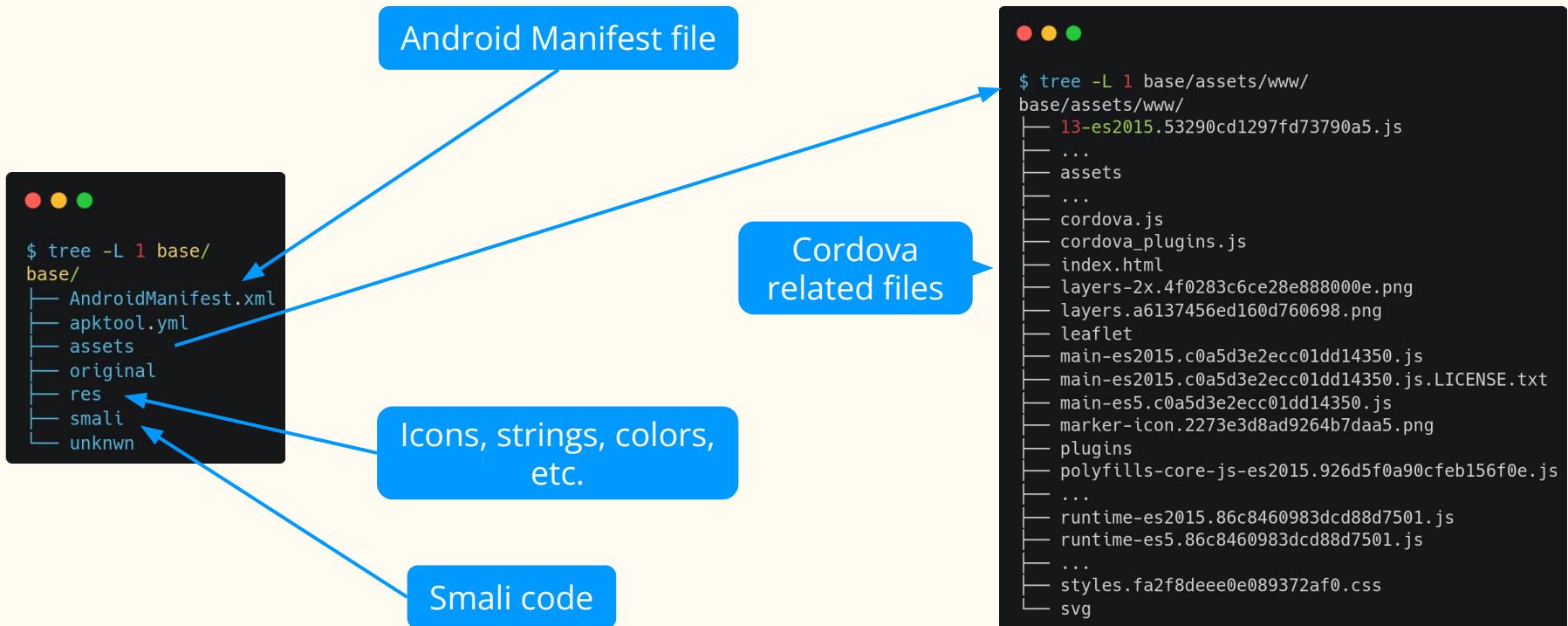
Note: These tools are not limited to Cordova applications

Cordova > Tools > Apktool

- Allows to **(un)pack Android applications** in order to edit the AndroidManifest.xml or to access code and resources
- Features
 - Unpack an Android Package Kit (APK)
 - apktool d <base.apk> -o <output_dir>
 - Repack an APK
 - apktool b <output_dir>
- Tampered APKs require to be aligned (`zipalign`) and signed (`apksigner`)
- <https://apktool.org/>

Advice: On Ubuntu, do not install Apktool from the package manager ;-)

Cordova > Tools > Apktool



[Bonus] Cordova > Tools > Apktool

- Custom script to
 - Build,
 - Align,
 - Sign, and
 - Install an APK



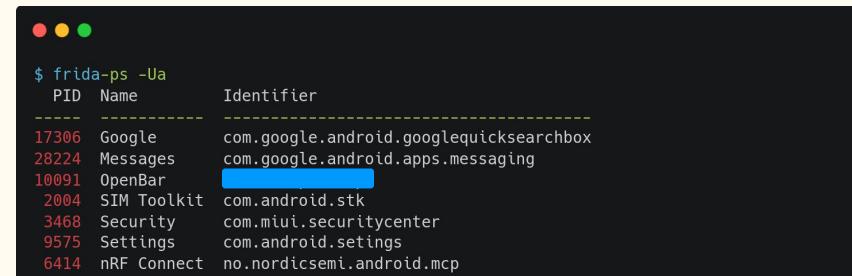
```
1#!/bin/sh
2# sudo apt install adb zipalign apksigner
3# apktool: https://apktool.org/
4
5# generate a keystore
6# keytool -genkey -v -keystore my.keystore -keyalg RSA -keysize 2048 -validity 10000 -alias app
7# disassemble the APK and perform the modifications (AndroidManifest.xml for example)
8# apktool d Original/base.apk -o Rebranded
9
10 APP_NAME="Rebranded"
11 OUT_DIR="${PWD}/${APP_NAME}/dist"
12 APP_BASE="${OUT_DIR}/base.apk"
13 APP_ALIGNED="${OUT_DIR}/base_aligned.apk"
14 APP_SIGNED="${OUT_DIR}/base_signed.apk"
15
16 # clean the destination folder
17 rm "$APP_BASE"
18 rm "$APP_ZIPPED"
19 rm "$APP_ALIGNED"
20
21 # repack the application
22 apktool b "$APP_NAME"
23
24 # optimize the application
25 zipalign -p -v 4 "$APP_BASE" "$APP_ALIGNED"
26
27 # sign the application
28 apksigner sign --ks-key-alias app --ks my.keystore --out "$APP_SIGNED" "$APP_ALIGNED"
29
30 # install the application
31 adb install "$APP_SIGNED"
```

Cordova > Tools > Frida

- Allows to perform **dynamic code instrumentation** by injecting snippets of JavaScript into native applications (Android, iOS, Windows, etc.)
- Features
 - Bypass root detection
 - Bypass SSL pinning
 - Trace Java classes and functions
 - Hook functions
 - Etc.
- Requirements
 - **Rooted Android phone** running the frida-server binary, or
 - Altered application embedding frida-gadget
- <https://frida.re/>
- Tools built on top of **frida**
 - Objection
 - Command-line interface (CLI)
 - <https://com/sensepost/objecton>
 - House
 - Web GUI
 - Not maintained anymore
 - <https://github.com/nccgroup/house>
 - Medusa
 - CLI
 - <https://github.com/Ch0pin/me>
 - [dusa](https://github.com/Ch0pin/me)

Cordova > Tools > Frida

- List running processes
 - `frida-ps -Ua`
- Attach a JS script to a running process
 - `frida -U -n <process> -l <script>`
- Trace native functions from a specific library
 - `frida-trace -U -n <process> -I <library>`



```
$ frida-ps -Ua
  PID  Name           Identifier
-----+-----+
 17306 Google         com.google.android.googlequicksearchbox
 28224 Messages       com.google.android.apps.messaging
 10091 OpenBar        com.google.android.apps.chrome
 2004 SIM Toolkit     com.android.stk
 3468 Security        com.miui.securitycenter
 9575 Settings        com.android.settings
 6414 nRF Connect     no.nordicsemi.android.mcp
```

Cordova > Tools > Frida

setWebContentsDebuggingEnabled

Added in API level 19

```
public static void setWebContentsDebuggingEnabled (boolean enabled)
```

Enables debugging of web contents (HTML / CSS / JavaScript) loaded into any WebViews of this application. This flag can be enabled in order to facilitate debugging of web layouts and JavaScript code running inside WebViews. Please refer to WebView documentation for the debugging guide.

In WebView 113.0.5656.0 and later, this is enabled automatically if the app is declared as `android:debuggable="true"` in its manifest; otherwise, the default is `false`.

Enabling web contents debugging allows the state of any WebView in the app to be inspected and modified by the user via adb. This is a security liability and should not be enabled in production builds of apps unless this is an explicitly intended use of the app. More info on [secure debug settings](#).

Parameters

enabled	boolean: whether to enable web contents debugging
---------	---

```
1 Java.perform(function() {
2     // https://developer.android.com/reference/android/webkit/WebView
3     var Webview = Java.use("android.webkit.WebView")
4     Webview.loadUrl.overload("java.lang.String").implementation = function(url) {
5         console.log("\nWebview.loadUrl()");
6         console.log("> URL: " + url);
7         this.setWebContentsDebuggingEnabled(true);
8         this.loadUrl(url)
9     }
10 }
11 })
```

```
$ frida -U -n "OpenBar" -l Frida/cordova.js
...
[M2004J19C::OpenBar ]->
WebView.loadUrl(
> URL: javascript:(function() { window.WEBVIEW_SERVER_URL = 'http://localhost';})();
```

Cordova > Tools > Medusa

- Allows to perform **dynamic code instrumentation** with frida within a **CLI**
- Extensible and modular
- Features (`medusa.py`)
 - Same as with frida
- Features (`mango.py`)
 - Patch an APK (`android:debuggable="true"`)
 - Set up a MITM / proxy
 - Wrap ADB commands
 - Etc.
- <https://github.com/Ch0pin/medusa>

Cordova > Tools > Medusa

- List all packages
 - `list`
- Search for a module
 - `search <keyword>`
- Select a module
 - `use <module>`
- Compile modules into a JS script
 - `compile`
- Initiate a frida session and attach to the selected package
 - `run -f <package_name>`, or
 - `run -n <package_number>`

Cordova > Tools > Medusa

```
$ python3 medusa.py
[!] Loading modules...
[!] Done....
[!] Total modules available 103
...

medusa>search cordova
cordova/get_loaded_plugins
cordova/cordova_enable_debugging

medusa>use cordova/cordova_enable_debugging
Current Mods:
① cordova/cordova_enable_debugging

medusa>compile
Script is compiled
```

Search for a module

Select the module

Compile the script

Attach the script to the package

```
$ medusa>run -f com.<REDACTED>
Spawned package : com.<REDACTED> on pid 23667
[in-session] |c:clear |e:exit |r:reload |
| rs:reset scratchpad |i:info |t:trace |?:help |-----Cordova Enable Webview
Debugging loaded-----

-----Application Info-----
- Frida version: 16.1.3
- Script runtime: QJS
- Application Name: android.app.Application
- Files Directory: /data/user/0/<REDACTED>/files
- Cache Directory: /data/user/0/<REDACTED>/cache
- External Cache Directory: /storage/emulated/0/Android/data/<REDACTED>/cache
- Code Cache Directory: /data/user/0/<REDACTED>/code_cache
- Obb Directory: /storage/emulated/0/Android/obb/<REDACTED>
- Package Code Path: /data/app/~~5rxkEdghUzCj8cadEE__lQ==/<REDACTED>--FSVNFQs-G0TRZeb6oFLXg==/base.apk

-----EOF Application Info-----
[+]Loading URL from https://localhost/index.html
[+]Setting the value of setWebContentsDebuggingEnabled() to TRUE

[+]Loading URL from javascript:(function() { window.WEBVIEW_SERVER_URL = 'http://localhost'; })()
[+]Setting the value of setWebContentsDebuggingEnabled() to TRUE
```

Cordova > Tools > Medusa

- Interesting modules
 - cordova/get_loaded_plugins
 - **cordova/cordova_enable_debugging**
 - **bluetooth/bluetooth**
 - http_communications/universal_SSL_pinning_bypass
 - http_communications/multiple_unpinner

Cordova > Tools > Chrome DevTools

- Allows to **remotely debug** an application with **Web developer tools** directly into the Google Chrome browser
- Features
 - Access source code
 - Monitor network
 - Set breakpoints
 - Etc.
- Requirements
 - The application needs to be (un)packed with Apktool*
 - Add `android:debuggable="true"` in the `AndroidManifest.xml`
 - Google Chrome / Chromium
 - Navigate to `chrome://inspect`

*Hooking the `WebView` with frida is an alternative

Cordova > Tools > Chrome DevTools

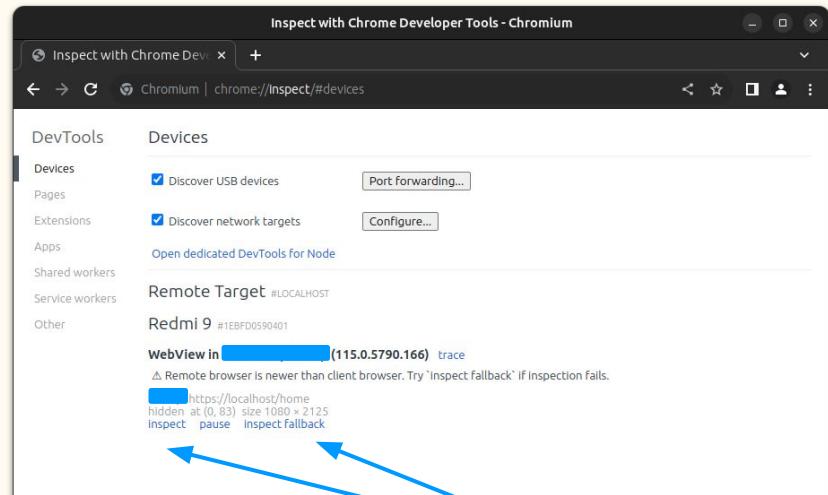
Chrome Remote Debugging [🔗](#)

Virtually the same as the Safari version, this works with Android only but can be used from any desktop operating system. Once connected, you get the same Chrome Dev Tools experience for your mobile applications as you do with your desktop applications. Even better, the Chrome Dev Tools have a mirror option that shows your app running on the mobile device. This is more than just a view - you can scroll and click from dev tools and it updates on the mobile device.

To inspect, simply open up the URL `chrome://inspect` in Chrome on your desktop. Here you will see a list of connected devices and inspectable apps. Your device must be set up for USB debugging for this to work. Full instructions on getting set up can be found at <https://developers.google.com/chrome/mobile/docs/debugging>.

If you can see your device in the inspect devices section, but you can't see the Cordova webview you may need to add `android:debuggable="true"` in the `<application>` node of your [AndroidManifest.xml](#).

It is also possible to use Chrome Dev Tools to inspect iOS apps, through a WebKit proxy: <https://github.com/google/ios-webkit-debug-proxy>



Launch the Chrome DevTools

DevTools - localhost/home

Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder

Preserve log Disable cache No throttling

Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party requests

Name	Status	Type	Initiator	Size	Time	Waterfall
arrow-back-sharp.svg	200	Fetch	polyfills-es2015.747d769...js:2	0 B	37 ms	
boxes	200	xhr	polyfills-es2015.747d769...js:2	22.6 kB	33 ms	
probe-login	200	xhr	polyfills-es2015.747d769...js:2	416 B	17 ms	
user	200	xhr	polyfills-es2015.747d769...js:2	759 B	94 ms	
user	200	xhr	polyfills-es2015.747d769...js:2	759 B	25 ms	
current-rent-data	404	xhr	polyfills-es2015.747d769...js:2	460 B	23 ms	
current-rent-data	404	xhr	polyfills-es2015.747d769...js:2	460 B	43 ms	
person-circle-sharp.svg	200	Fetch	polyfills-es2015.747d769...js:2	0 B	134 ms	
today-sharp.svg	200	Fetch	polyfills-es2015.747d769...js:2	0 B	127 ms	
en.json	200	xhr	polyfills-es2015.747d769...js:2	0 B	74 ms	
probe-login	200	xhr	polyfills-es2015.747d769...js:2	416 B	14 ms	
user	200	xhr	polyfills-es2015.747d769...js:2	759 B	21 ms	
person-circle.svg	200	Fetch	polyfills-es2015.747d769...js:2	0 B	63 ms	
current-rent-data	404	xhr	polyfills-es2015.747d769...js:2	460 B	24 ms	
boxes	200	xhr	polyfills-es2015.747d769...js:2	22.6 kB	142 ms	
arrow-back-outline.svg	200	Fetch	polyfills-es2015.747d769...js:2	0 B	45 ms	
delete-account	200	xhr	polyfills-es2015.747d769...js:2	419 B	35 ms	
current-rent-data	404	xhr	polyfills-es2015.747d769...js:2	460 B	21 ms	
boxes	200	xhr	polyfills-es2015.747d769...js:2	22.6 kB	35 ms	

19 / 274 requests | 73.1 kB / 399 kB transferred | 95.9 kB / 7.7 MB resources

Access the source code (HTML, CSS, JS)

Access the HTTP traffic

Control the application

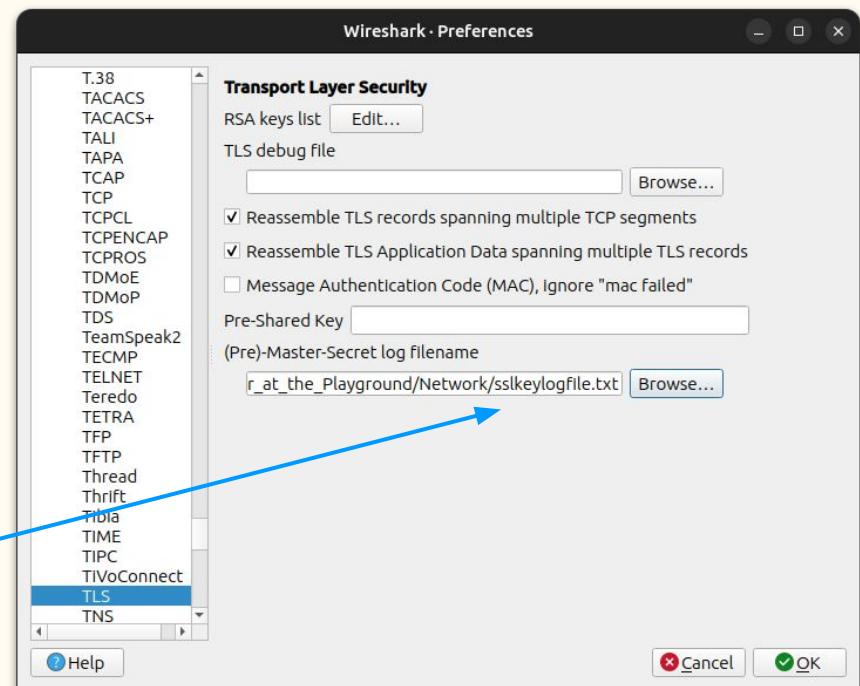
Data Acquisition

- Setup (host computer)
 - Wi-Fi AP
 - **Mitmproxy** / mitmweb
 - SSLKEYLOGFILE environment variable set to log TLS handshakes (\$SSLKEYLOGFILE=<file>)
 - Wireshark
 - Captures on the Ethernet interface
 - Allows to decrypt HTTPS packets with the exported TLS handshakes
- Setup (phone)
 - Xiaomi Redmi 9
 - Android 13 / LineageOS 20
 - Allows to run ADB as root
 - Rooted with Magisk
 - MagiskFrida module installed to launch frida at boot
 - Mitmproxy CA certificate installed
 - Target application ("OpenBar")

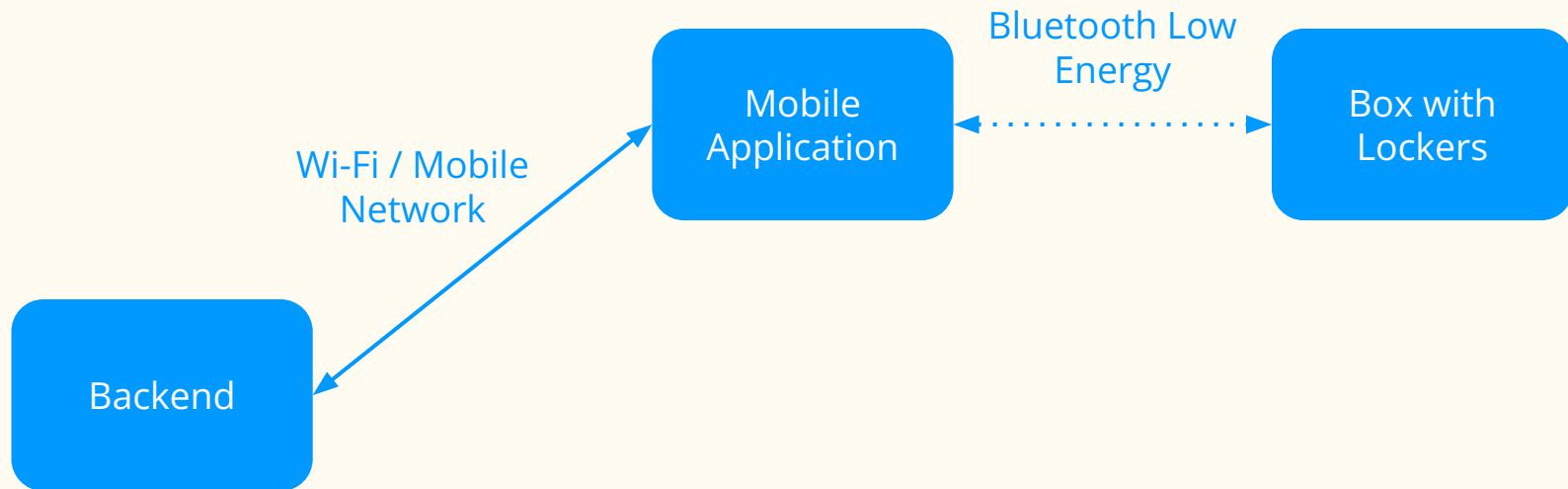
Data Acquisition

```
$ SSLKEYLOGFILE="./sslkeylogfile.txt" mitmproxy
```

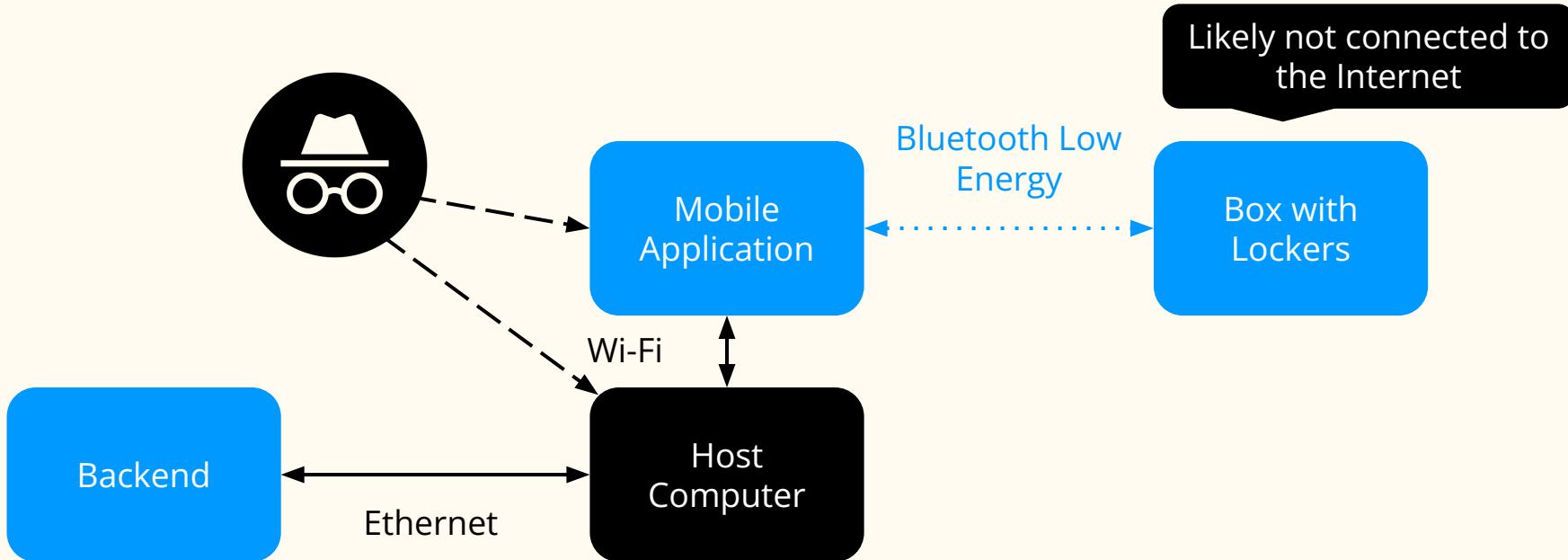
TLS handshakes



Data Acquisition



Data Acquisition



Data Analysis

- Application
 - No root detection
 - No instrumentation detection
 - No hardcoded secrets
- Network communications
 - **HTTPS**
 - **Token-based authentication**
 - SSL pinning
 - Could be easily bypassed with frida / medusa ;-)

Data Analysis

- API
 - GET /api/boxes
 - Returns the boxes' information (name, location, etc.)
 - GET /api/box-content/<BOX_ID>
 - Returns the lockers' information (availability, ID, etc.)
 - GET /api/items-in-locker?lockerId=<ID>
 - Returns the content of the locker
 - POST /api/rent + {"lockerID: <ID>"}
 - POST /api/end-rent + {"lockerID: <ID>"}
 - GET /api/user
 - Returns the user's information
 - POST /api/user/reset-password
 - POST /api/user/**update-identity** + {"firstName, lastName, cardID, etc."}
 - Etc.

Findings

```
curl 'https://<REDACTED>.infomaniak.ch/api/user/update-identity' \
-H 'Accept: application/json, text/plain, */*' \
-H 'Accept-Language: fr' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/json' \
-H 'Cookie: PLAY_SESSION=<REDACTED>' \
-H 'Origin: https://localhost' \
-H 'Referer: https://localhost/' \
-H 'Sec-Fetch-Dest: empty' \
-H 'Sec-Fetch-Mode: cors' \
-H 'Sec-Fetch-Site: cross-site' \
-H 'User-Agent: <REDACTED>' \
-H 'X-Requested-With: com.<REDACTED>' \
--data-raw '{"firstName": "Samuel", "lastName": "M.", "cardId": "C1234567", "sex": "M", "birthDate": "25/6/2023"}' \
--compressed
```

HTTP POST
/api/user/update-identity

Payload
name, ID number, birthdate, ...

Findings

HTTP GET
/api/user

```
curl 'https://<REDACTED>.infomaniak.ch/api/user' \  
-H 'Accept: application/json, text/plain, */*' \  
-H 'Accept-Language: fr' \  
-H 'Connection: keep-alive' \  
-H 'Content-Type: application/json' \  
-H 'Cookie: PLAY_SESSION=<REDACTED>' \  
-H 'Origin: https://localhost' \  
-H 'Referer: https://localhost/' \  
-H 'Sec-Fetch-Dest: empty' \  
-H 'Sec-Fetch-Mode: cors' \  
-H 'Sec-Fetch-Site: cross-site' \  
-H 'User-Agent: <REDACTED>' \  
-H 'X-Requested-With: com.<REDACTED>' \  
--compressed
```

● ○ ●

```
{  
    "id": 27642,  
    "email": "<REDACTED>@gmail.com",  
    "phoneNumber": "<REDACTED>",  
    "cardId": "C1234567",  
    "password": "$2a$10$Ww...cVYW7X0u",  
    "firstName": "Samuel",  
    "lastName": "M.",  
    "birthDate": "25/6/2023",  
    "sex": "M",  
    "address": "",  
    "signupDate": "2023-06-17T00:00:00Z",  
    "blocked": false,  
    "preferredLanguage": "fr",  
    "blockedDate": "  
}
```

Anonymized
information

Is it possible to
open a locker
without an
account?

The Wireless Communications

The Wireless Communications

- Introduction to Bluetooth Low Energy
 - Introduction
 - Topology
 - GATT and ATT
 - Security
 - Tools
- Data Acquisition
- Data Analysis
- Findings
- Demos
 - Replay
 - Device Spoofing

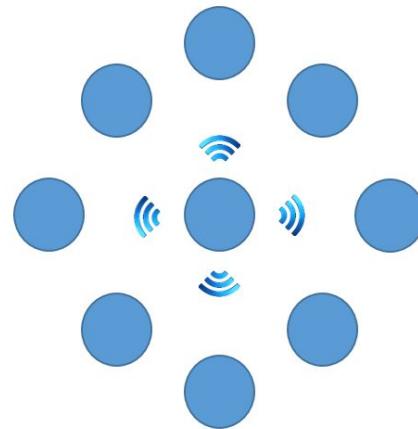
BLE > Introduction

- Wireless communication technology
 - Spectrum: 2.4 GHz ISM band
 - Channels: 40x 2 MHz channels
 - Bit rates: 125 & 500 Kb/s, 1 & 2 Mb/s
- Designed for **low power** consumption
 - Enables battery-operated devices
 - Ideal for IoT devices and wearables
- Used for **short-range** communication (up to 100 meters)
 - Suitable for indoor applications
- Use cases
 - Health and fitness
 - Wearable fitness trackers
 - Health monitoring devices
 - Smart home
 - Connected lights, thermostats, locks
 - Home automation systems
 - IoT devices
 - Sensors and actuators
 - Asset tracking

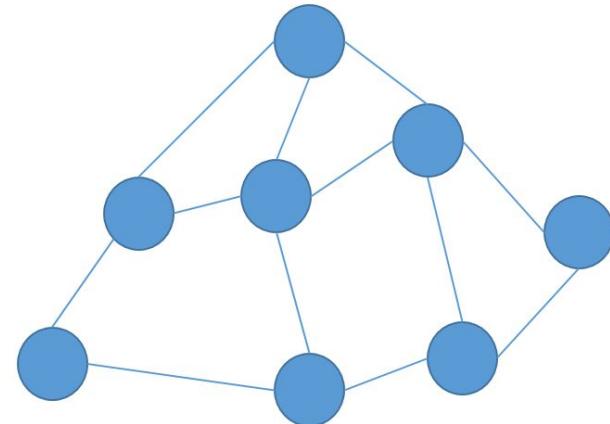
BLE > Topology



Point-to-Point (1:1)
Connection-Oriented Communication



Data Broadcast
One-to-Many (1:m)
Connection-less Communication



Mesh
Many-to-Many (m:m)
Large-scale Device Networks

BLE > Generic Attribute Profile (GATT)

- Defines how data is **organized and exchanged** between BLE devices
- Establishes a **hierarchy of services and characteristics** for data exchange
- Services
 - Represent **functionalities** offered by the device
 - **Logical containers** for related data
- Characteristics
 - **Data elements** within services
 - Include values, descriptors, and properties
- Services, characteristics and descriptors have a UUID

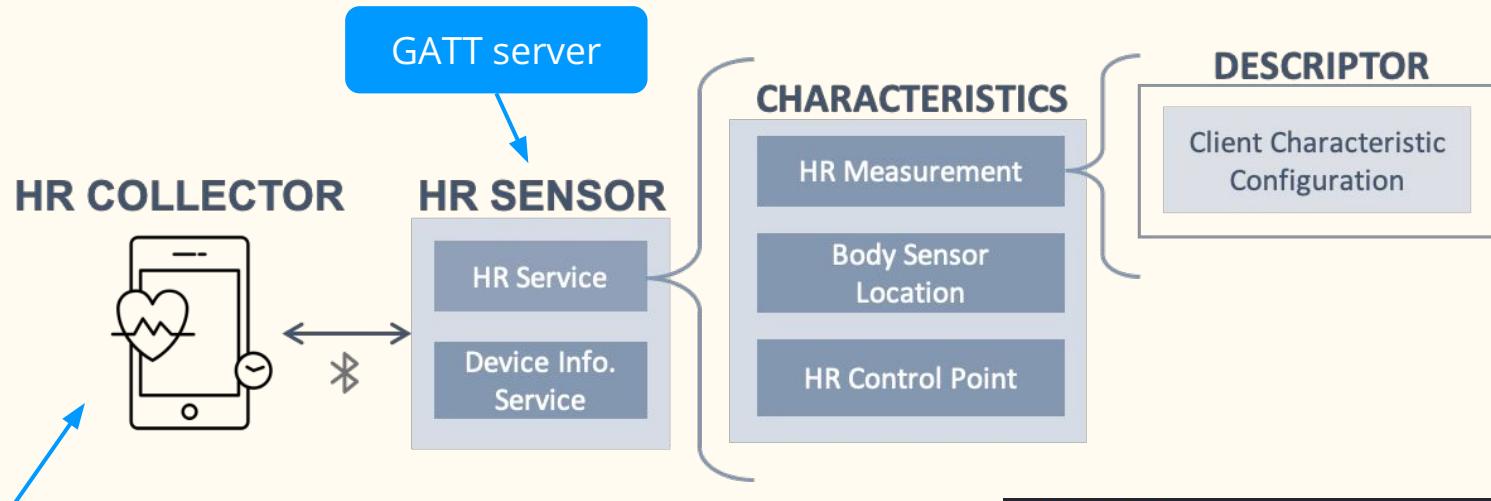
BLE > Generic Attribute Profile (GATT)

- Roles
 - GATT server
 - Responds to client requests
 - Contains attributes and data to be exchanged
 - → Embedded device
 - GATT client
 - Initiates requests to read or write data
 - → Phone, tablet or computer
- Data exchange in GATT
 - a. GATT server advertises its services
 - b. GATT client scans and connects to the server
 - c. Client discovers services and characteristics
 - d. Client reads or writes characteristic values

BLE > Attribute Protocol (ATT)

- Defines the **format and rules** for reading and writing attributes
- Read and write operations
 - Read: retrieve the value of an attribute
 - Write: set the value of an attribute
- Characteristic properties
 - Define how a characteristic can be accessed
 - Common properties
 - Read
 - Write
 - Notify
 - Indicate

BLE > GATT and ATT



Characteristic	Property
Heart Rate Measurement (HRM)	Notify
Body Sensor Location	Read
Heart Rate Control Point	Write

BLE > Security

- Pairing and bonding
 - Establish secure connections
 - Link key generation for future authentication / **encryption** (AES-CCM)
- Privacy
 - **Random device addresses** to prevent tracking
 - Resolving addresses to protect user identity
- Challenges
 - Eavesdropping
 - Unauthorized interception of data during transmission
 - MITM attacks
 - Attacker intercepts and possibly alters communication between devices
 - Device impersonalization
 - Attackers pretend to be legitimate devices to gain access
 - Data tampering
 - Unauthorized modification of data exchanged between devices

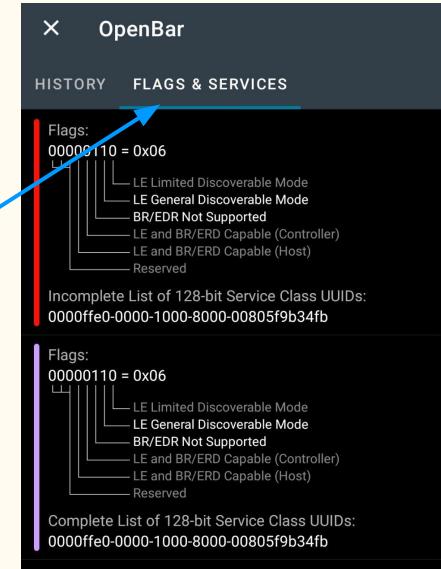
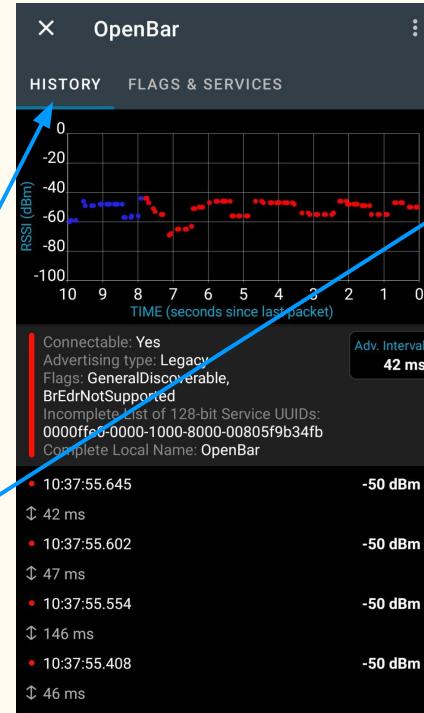
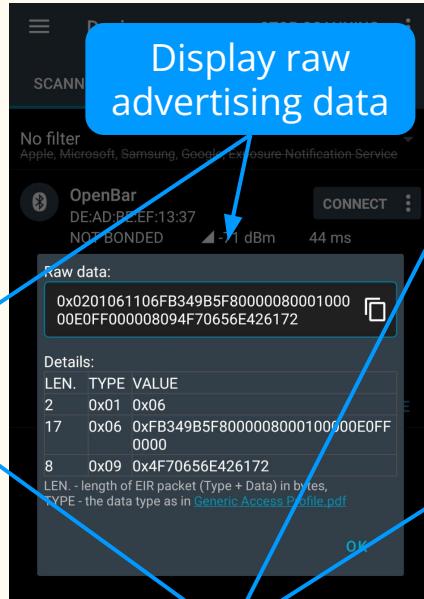
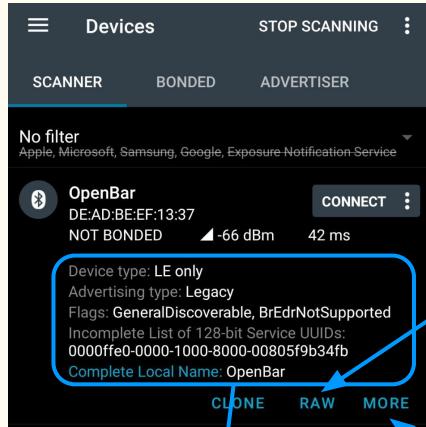
BLE > Tools

- nRF Connect for Mobile
- Android Bluetooth HCI snoop log
- Sniffle
- Mirage

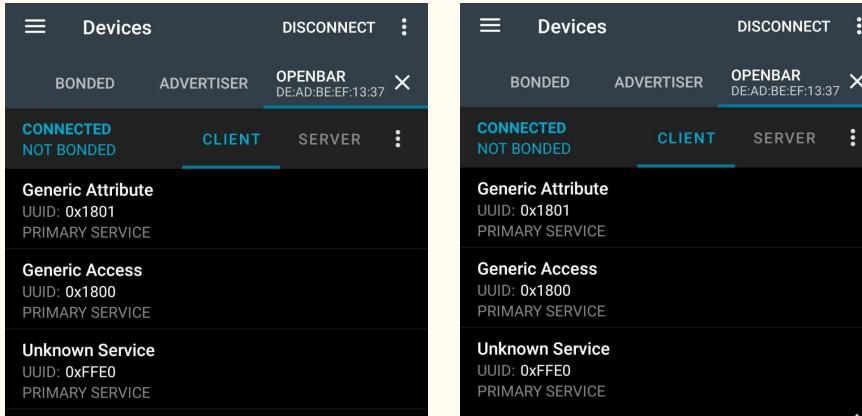
BLE > Tools > nRF Connect for Mobile

- Allows to scan for BLE devices
- Allows to connect to a device and to discover its services and characteristics
- Allows to interact with a device (read, write, subscribe to notifications, etc.)
- Features
 - **Create macros**
 - Clone advertising data
 - Parse advertising data, services and characteristics
 - Perform Device Firmware Update (DFU)
- <https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-mobile>

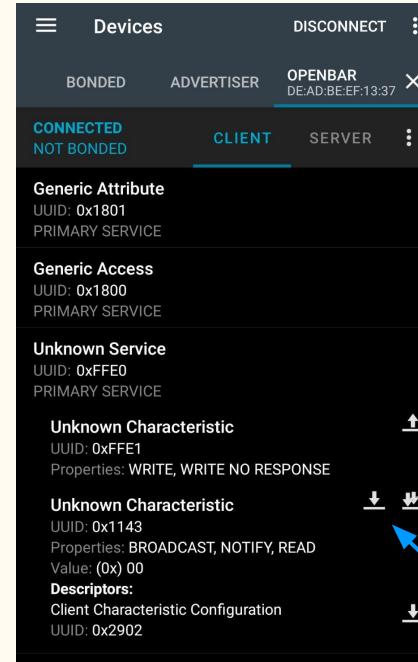
BLE > Tools > nRF Connect for Mobile



BLE > Tools > nRF Connect for Mobile



Create or trigger
a macro



Write to the
characteristic

Subscribe to
notifications

Read the
characteristic

BLE > Tools > Android Bluetooth HCI snoop log

- Allows to retrieve all **Bluetooth** Host Controller Host interface (HCI) **logs into a single file**
- Features
 - Available by default
 - Settings > System > Developer options > Enable Bluetooth HCI snoop log
 - Logging happens before the BLE Link Layer encryption
- HCI logs are generally stored in /data/misc/bluetooth/logs

Method #1 to retrieve the HCI logs

```
$ adb bugreport  
$ adb shell  
$ su  
$ cp /data/misc/bluetooth/logs/btsnoop_hci.log /sdcard/Download  
$ exit  
$ adb pull /sdcard/Download/btsnoop_hci.log  
  
$ adb root  
$ adb pull /data/misc/bluetooth/logs/btsnoop_hci.log
```

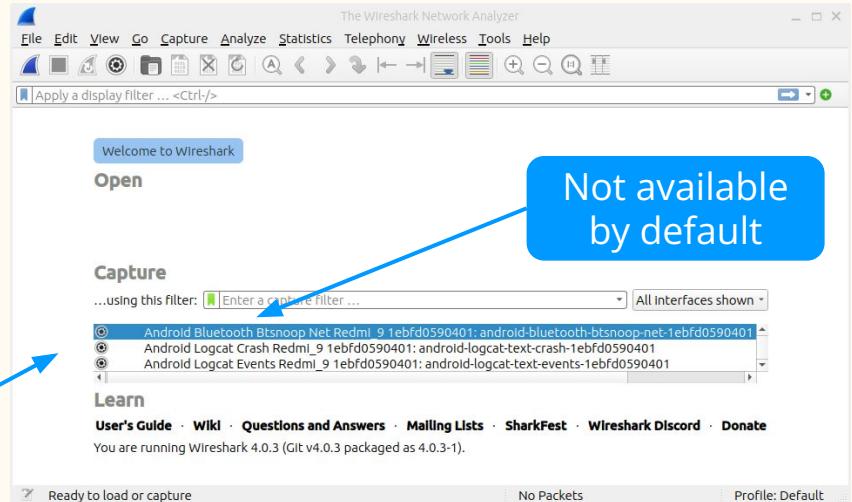
Method #2

Method #3

BLE > Tools > Android Bluetooth HCI snoop log

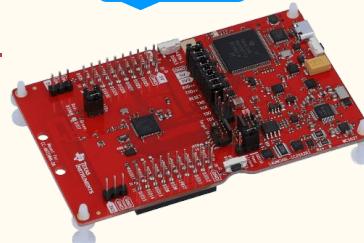
```
$ adb shell su -c "'nc -s 127.0.0.1 -p 8872 \
-L system/bin/tail \
-f -c +0 data/misc/bluetooth/los/btsnoop_hci.log'"
```

Allows to capture
HCI logs in realtime



BLE > Tools > Sniffle

- Allows to **sniff Bluetooth 5.X and 4.X** using TI CC1352/CC26X2 hardware
- Features
 - Scan for BLE devices
 - `scanner.py`
 - Follow and capture BLE packets for a specific connection
 - `sniff_receiver.py -m <MAC> -o <output_file>`
- <https://github.com/nccgroup/Sniffle>



- Alternatives
 - Ubertooh
 - Not maintained anymore
 - <https://github.com/greatscottgadgets/ubertooh>
 - nRF Sniffer for Bluetooth LE
 - Similar to Sniffle
 - <https://www.nordicsemi.com/Products/Development-tools/nrf-sniffer-for-bluetooth-le>
 - ICE9 Bluetooth Sniffer
 - All-channel Bluetooth sniffer based on a SDR
 - <https://github.com/mikeryan/ice9-bluetooth-sniffer>

99 \$

10 \$

540 \$

39 \$

BLE > Tools > Mirage

- Allows to interact with a BLE device, to perform a MITM, to jam or hijack a connection, etc. → **BLE Swiss army knife**
- Supports other RF protocols (Wi-Fi, Zigbee, etc.)
- Features
 - Interact with a BLE device (discover, read, write, etc.)
 - Clone a device
 - Sniff packets
 - Perform a MITM
 - Create scenarios
 - Etc.
- <https://homepages.laas.fr/rcayre/mirage-documentation/>

The screenshot shows a web browser window displaying the 'Overview — Mirage 1.2' page at <https://homepages.laas.fr/rcayre/mirage-documentation/overview.html>. The page has a dark theme. On the left, a sidebar lists various modules and protocols: Direct execution, Modules, Chaining operator, Configuration file, Supported Devices, Emitters and Receivers, Scenarios, Bluetooth Low Energy protocol, Enhanced ShockBurst protocol, Mosart protocol, IEEE 802.11 protocol, Zigbee protocol, Infrared Radiations protocols, List of Bluetooth Low Energy Modules, List of Enhanced ShockBurst Modules, List of Mosart Modules, List of 802.11 Modules, List of Zigbee Modules, List of Infrared Radiations Modules, mirage.libs package, and mirage.core package. The main content area is titled 'Getting started' and contains instructions for configuring and running modules. It includes a table titled 'list' showing module details:

Name	Type	Description
ble_hijack	attack	Hijacking module for Bluetooth Low Energy Connection
ble_pair	action	Pairing module for Bluetooth Low Energy devices
ble_adv	spoofing	Spoofing module simulating a Bluetooth Low Energy advertisement
ble_mitm	attack	Man-in-the-Middle module for Bluetooth Low Energy devices
ble_adb	monitoring	Sniffing module monitoring an HCI Android log
ble_sniff	sniff	Sniffing module for Bluetooth Low Energy devices
bt_scan	scan	Scan module for Bluetooth Devices
ble_scan	scan	Scan module for Bluetooth Low Energy devices
ble_discover	discover	Discovery module for Bluetooth Low Energy ATT / GATT
ble_jam	attack	Jamming module for Bluetooth Low Energy advertisements
ble_connect	action	Connection module for Bluetooth Low Energy devices
bt_info	info	Information module for Bluetooth interface
ble_slave	spoofing	Spoofing module simulating a Bluetooth Low Energy slave
ble_crack	bruteforce	Enumerates all possible values of PIN in order to find it
ble_master	spoofing	This module permits the User to interact with Bluetooth Low Energy interfaces
ble_info	info	Information module for Bluetooth Low Energy interfaces

If you want to load the module `ble_info` from the Command-Line Interface, just type the following command.

BLE > Tools > Mirage

- Scan for BLE devices
 - `mirage ble_scan`
- Connect to a device and discover its services and characteristics
 - `mirage ble_master`
 - `connect <MAC> [<connection type>]`
 - `discover`
- Perform a MITM
 - `mirage ble_mitm TARGET=<MAC>`
- Monitor BLE communications from an ADB interface
 - `mirage ble_monitor`

[Bonus] BLE > Tools > EXPLIoT

- Allows to interact with a BLE device, to discover its services and characteristics, **to fuzz a characteristic**, etc.
- Supports other protocols (Zigbee, MQTT, CoAP, etc.)
- Features
 - Scan for BLE devices
 - run ble.generic.scan --timeout <timeout>
 - Connect to a device and discover its services and characteristics
 - run ble.generic.enum --addr <MAC> --randaddrtype --services --chars
 - Fuzz a specific characteristic
 - run ble.generic.fuzzchar --addr <MAC> --handle <handle> --value <x>
- <https://exploit.readthedocs.io/en/latest/index.html>

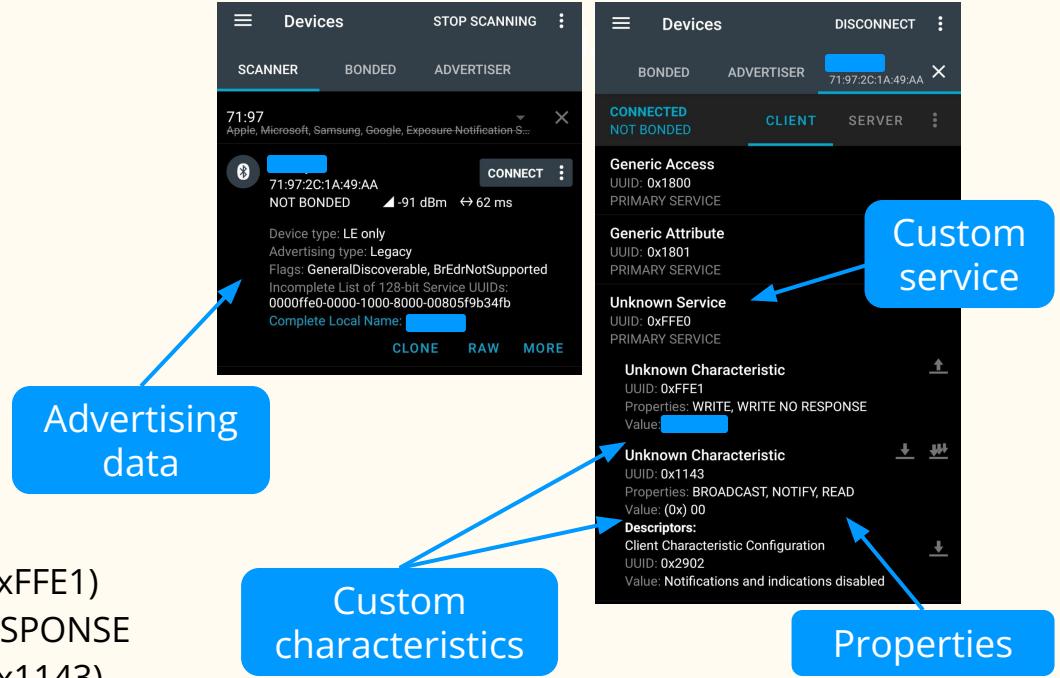
Data Acquisition

- Setup (phone)
 - nRF Connect for Mobile
 - Reconnaissance
 - Advertising data
 - Services
 - Characteristics
 - Android Bluetooth HCI snoop log
 - BLE packets
- Scenario
 - Start / stop renting locker 1
 - Start / stop renting locker 5



Data Analysis

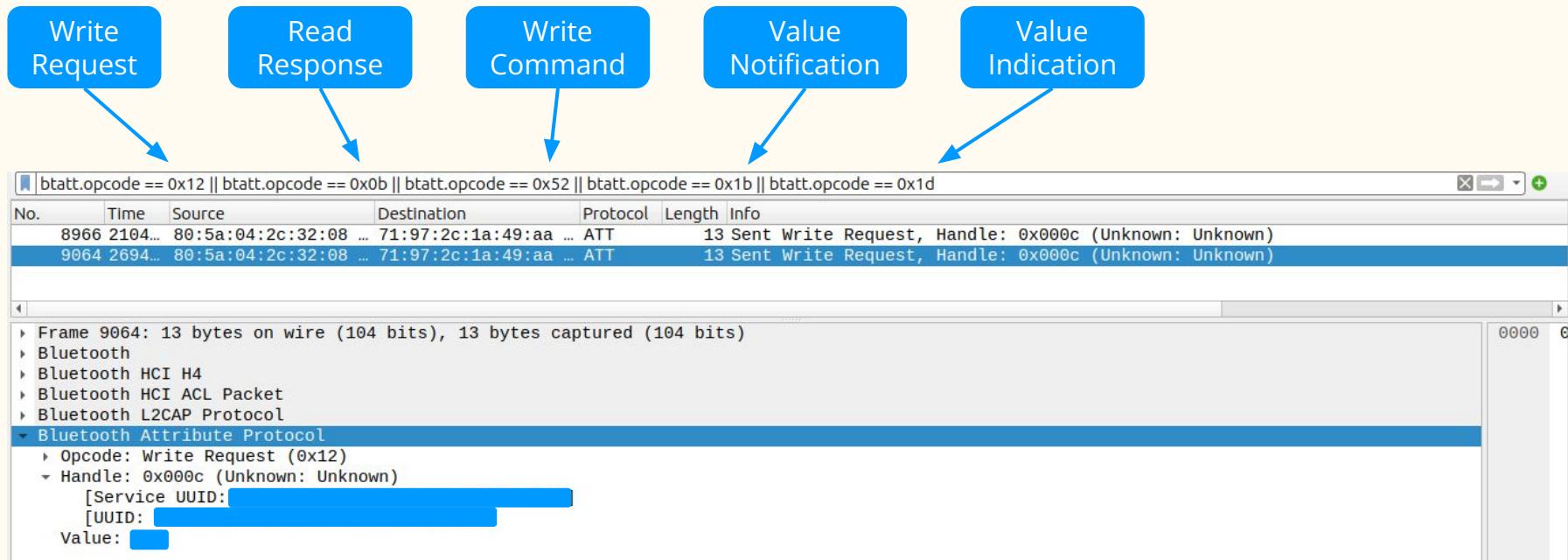
- Advertising data
 - Flags
 - Local Name
 - Service UUID
- Services and characteristics
 - Generic Access
 - Device Name
 - Appearance
 - Generic Attribute
 - Service Changed
 - Unknown Service (0xFFE0)
 - Unknown Characteristic (0xFFE1)
 - WRITE, WRITE NO RESPONSE
 - Unknown Characteristic (0x1143)
 - BROADCAST, NOTIFY, READ



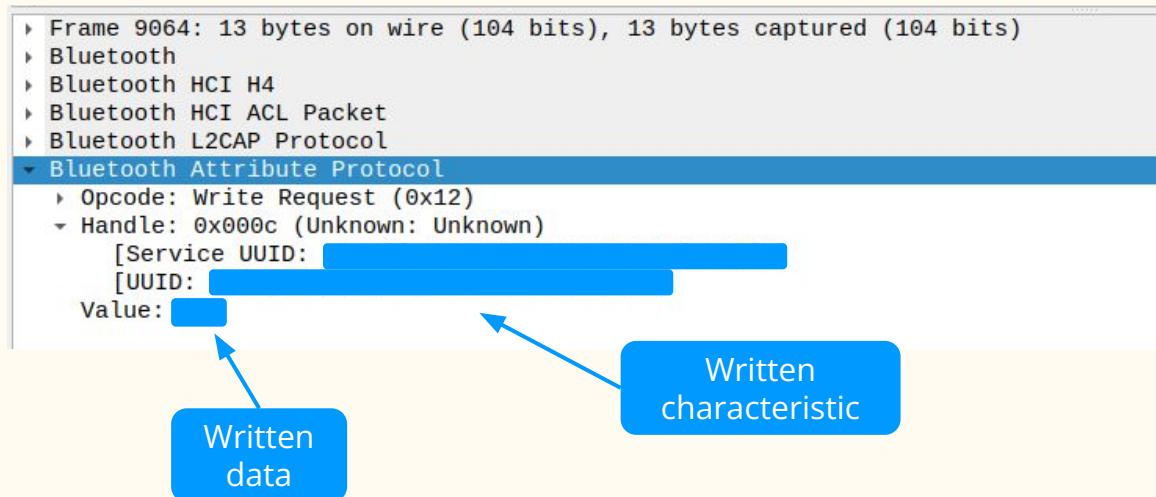
Data Analysis

No.	Time	Source	Destination	Protocol	Length	Info
9034	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	14	Rcvd Error Response - Attribute Not Found, Handle: 0x0005 (Generic Access Pr
9035	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Include Declaration, Handles: 0x0006..0x0009
9037	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	14	Rcvd Error Response - Attribute Not Found, Handle: 0x0006 (Generic Attribute
9038	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Characteristic Declaration, Handles: 0x0006.
9040	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	18	Rcvd Read By Type Response, Attribute List Length: 1, Service Changed
9041	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Characteristic Declaration, Handles: 0x0008.
9043	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	14	Rcvd Error Response - Attribute Not Found, Handle: 0x0008 (Generic Attribute
9044	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	14	Sent Find Information Request, Handles: 0x0009..0x0009
9046	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	15	Rcvd Find Information Response, Handle: 0x0009 (Generic Attribute Profile: S
9047	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Include Declaration, Handles: 0x000a..0x000f
9049	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	14	Rcvd Error Response - Attribute Not Found, Handle: 0x000a (Unknown)
9050	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Characteristic Declaration, Handles: 0x000a.
9052	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	32	Rcvd Read By Type Response, Attribute List Length: 1, Unknown
9053	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Characteristic Declaration, Handles: 0x000c.
9055	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	32	Rcvd Read By Type Response, Attribute List Length: 1, Unknown
9056	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	16	Sent Read By Type Request, GATT Characteristic Declaration, Handles: 0x000e.
9058	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	14	Rcvd Error Response - Attribute Not Found, Handle: 0x000e (Unknown: Unknown)
9059	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	14	Sent Find Information Request, Handles: 0x000f..0x000f
9061	2694...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	15	Rcvd Find Information Response, Handle: 0x000f (Unknown: Unknown: Client Cha
9064	2694...	80:5a:04:2c:32:08	71:97:2c:1a:49:aa	ATT	13	Sent Write Request, Handle: 0x000c (Unknown: Unknown)
9067	2695...	71:97:2c:1a:49:aa	80:5a:04:2c:32:08	ATT	10	Rcvd Write Response, Handle: 0x000c (Unknown: Unknown)

Data Analysis



Data Analysis



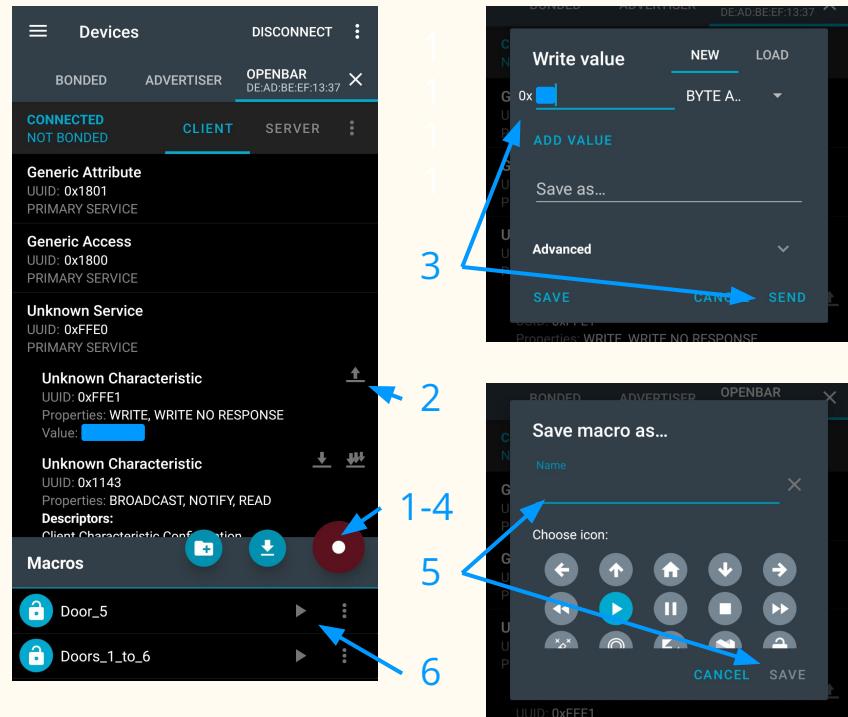
Findings

- **No pairing / bonding**
- **No encryption** (Link-Layer or Application Layer)
- A single **Write Request** is enough to open a locker
 - A single byte  permits to select the locker (1-6)

Is it possible to
open a locker
without an
account?

Demo #1 > Replay

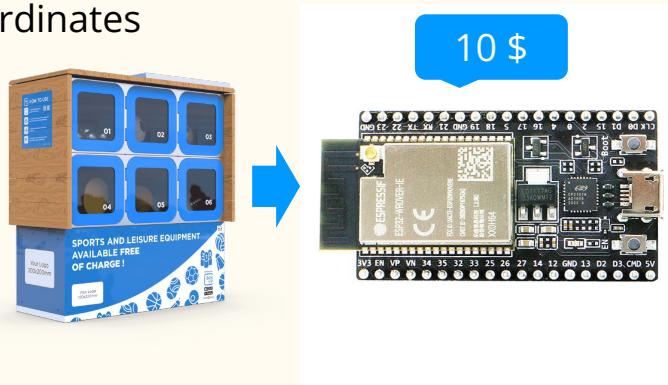
- Scenarios
 - Open the **fifth locker** by sending  to the characteristic 
 - Open all the lockers at the same time
- Tools
 - nRF Connect for Mobile
 - Macros
 - Alternatives
 - Bleak
 - Python library
 - https://github.com/hbldh/b_leak
 - Mirage



Is it possible to
virtually rent an
article?

Demo #2 > Device Spoofing

- Scenario
 - Create a **clone of a box** to try to fool the application and so, to simulate fake rentals
- Tools
 - Arduino IDE + arduino-esp32 + Espressif ESP32
 - Frida
 - Fake GPS coordinates
- Alternatives
 - Mirage
 - GATTacker
 - Not maintained anymore
 - <https://github.com/securing/gattacker>
 - BtleJuice
 - Not maintained anymore
 - <https://github.com/DigitalSecurity/btlejuice>



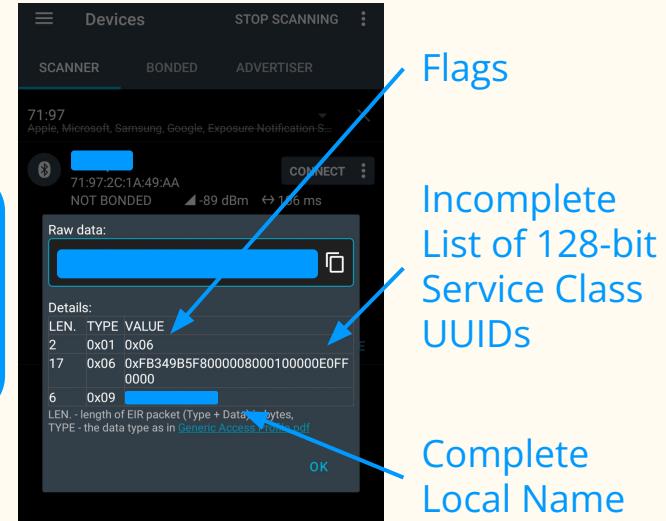
Demo #2 > Device Spoofing

```
1 uint8_t BLE_ADDRESS[6] = {0xDE, 0xAD, 0xBE, 0xEF, 0x13, 0x37};  
2 #define SERVICE_UUID "0000ffe0-0000-1000-8000-00805f9b34fb"  
3 ...  
4 void setup() {  
5   BLEDevice::init("Arduino");  
6   ...  
7   BLEAdvertisementData advData = BLEAdvertisementData();  
8   advData.setFlags(0x06); // 0x01: Flags  
9   // 0x06: Incomplete List of 128-bit Service Class UUIDs  
10  advData.setPartialServices(BLEUUID::fromString(SERVICE_UUID));  
11  
12  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();  
13  pAdvertising->setAdvertisementData(advData);  
14  
15  BLEAdvertisementData scanData = BLEAdvertisementData();  
16  scanData.setName("OpenBar"); // 0x09: Complete Local Name  
17  
18  pAdvertising->setScanResponseData(scanData);  
19  pAdvertising->setDeviceAddress(BLE_ADDRESS)  
20  pAdvertising->setScanResponse(true);  
21  ...  
22  BLEDevice::startAdvertising();  
23 }
```

Flags

Incomplete List of 128-bit Service Class UUIDs

Complete Local Name



Flags

Incomplete List of 128-bit Service Class UUIDs

Complete Local Name

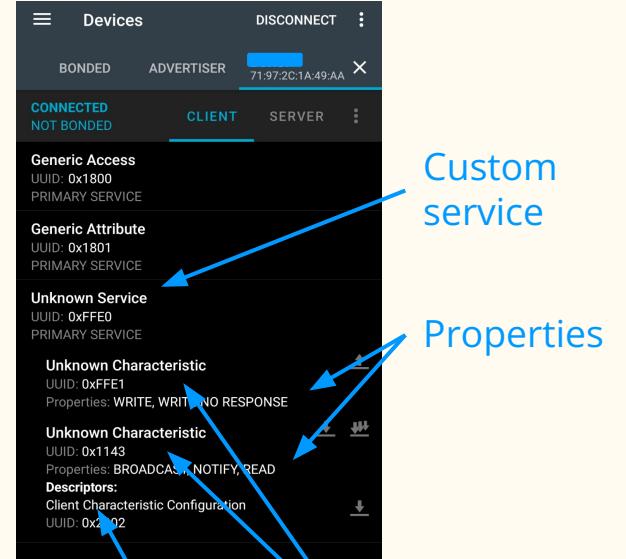
Demo #2 > Device Spoofing

```
1 #define SERVICE_UUID      "0000ffe0-0000-1000-8000-00805f9b34fb"  
2 #define CHARACTERISTIC_UUID_1  "0000fffe1-0000-1000-8000-00805f9b34fb"  
3 #define CHARACTERISTIC_UUID_2  "00001143-0000-1000-8000-00805f9b34fb"  
4  
5 void setup() {  
6     ...  
7     BLEServer *pServer = BLEDevice::createServer();  
8     BLEService *pService = pServer->createService(SERVICE_UUID);  
9  
10    BLECharacteristic *pCharacteristic_1 = pService->createCharacteristic(  
11        CHARACTERISTIC_UUID_1,  
12        BLECharacteristic::PROPERTY_WRITE |  
13        BLECharacteristic::PROPERTY_WRITE_NR  
14    );  
15  
16    BLECharacteristic *pCharacteristic_2 = pService->createCharacteristic(  
17        CHARACTERISTIC_UUID_2,  
18        BLECharacteristic::PROPERTY_BROADCAST |  
19        BLECharacteristic::PROPERTY_NOTIFY |  
20        BLECharacteristic::PROPERTY_READ  
21    );  
22    pCharacteristic_2->addDescriptor(new BLE2902());  
23  
24    pService->start();  
25    ...  
26 }
```

Custom service

Custom characteristic 1

Custom characteristic 2



Custom service

Properties

Descriptor

Custom characteristics

Demo #2 > Device Spoofing

```
1 Java.perform(function() {
2     const SIMULATED_LATITUDE = 46.79735;
3     const SIMULATED_LONGITUDE = 7.15337;
4
5     // https://developer.android.com/reference/android/location/Location
6     var Location = Java.use("android.location.Location");
7
8     Location.getLatitude.overload().implementation = function() {
9         console.log("\nLocation.getLatitude()");
10        console.log("> Old: " + this.getLatitude());
11        console.log("> New: " + SIMULATED_LATITUDE);
12
13        return SIMULATED_LATITUDE;
14    }
15
16    Location.getLongitude.overload().implementation = function() {
17        console.log("\nLocation.getLongitude()");
18        console.log("> Old: " + this.getLongitude());
19        console.log(" New: " + SIMULATED_LONGITUDE);
20
21        return SIMULATED_LONGITUDE;
22    }
23});
```

[Bonus] Could I
have used
ChatGPT to
generate the BLE
clone?

[Bonus] ChatGPT > Device Spoofing

- Request

*"Generate an arduino sketch for the ESP32 that emulates a device with one service ("000ffe0-0000-1000-8000-00805f9b34fb") and two characteristics. The first characteristic ("0000ffe1-0000-1000-8000-00805f9b34fb") has the permissions **PROPERTY_WRITE** and **PROPERTY_WRITE_NR**. The second characteristic ("00001143-0000-1000-8000-00805f9b34fb") has the permissions **PROPERTY_BROADCAST**, **PROPERTY_NOTIFY** and **PROPERTY_BREAD**. The Complete Device Name is OpenBar and it has the following BLE address DE:AD:BE:EF:13:37."*

[Bonus] ChatGPT > Device Spoofing

```
1 #include <BLEDevice.h>
2 #include <BLEUtils.h>
3 #include <BLEServer.h>
4
5 BLEServer *pServer;
6 BLEService *pService;
7 BLECharacteristic *pCharacteristic1;
8 BLECharacteristic *pCharacteristic2;
9
10 void setup() {
11   Serial.begin(115200);
12   BLEDevice::init("OpenBar");
13   BLEDevice::setAddress(BLEAddress("DE:AD:BE:EF:13:37"));
14
15   pServer = BLEDevice::createServer();
16   pService = pServer->createService(BLEUUID("000ffe0-0000-1000-8000-00805f9b34fb"));
17
18   pCharacteristic1 = pService->createCharacteristic(
19     BLEUUID("000ffe1-0000-1000-8000-00805f9b34fb"),
20     BLECharacteristic::PROPERTY_WRITE | BLECharacteristic::PROPERTY_WRITE_NR
21 );
```

error: 'setAddress'
is not a member of
'BLEDevice'

[Bonus] ChatGPT > Device Spoofing

```
22     pCharacteristic2 = pService->createCharacteristic(  
23             BLEUUID("00001143-0000-1000-8000-00805f9b34fb"),  
24             BLECharacteristic::PROPERTY_BROADCAST | BLECharacteristic::PROPERTY_NOTIFY |  
25             BLECharacteristic::PROPERTY_READ  
26         );  
27     pService->start();  
28  
29     BLEAdvertising *pAdvertising = pServer->getAdvertising();  
30     pAdvertising->addServiceUUID(pService->getUUID());  
31     pAdvertising->setCompleteServices(BLEUUID(pService->getUUID()));  
32     pAdvertising->start();  
33 }  
34  
35 void loop() {  
36     // Do nothing here  
37 }
```

error: 'class
BLEAdvertising' has no
member named
'setCompleteServices'

Conclusion

Conclusion

- A forgotten HTTP endpoint allows to **anonymize** the user's personal information
- The link between the phone and the box is weak, allowing to **open the lockers without an account**
- **Medusa** is a great tool for debugging / analyzing Android applications
- **Mirage** is a Swiss army knife for BLE, including modules to discover, to interact or to clone devices, as well as sniffing or performing a MITM

Conclusion

- The company has been contacted in **July 2022** (responsible disclosure) and June 2023
- Patches should be applied until the end of the year
- **OpenBar** is not the real name of the solution ;-)

Q & A

References

- <https://fribourg.ch/wp-content/uploads/2021/12/vue-Fribourg-top-Banner3.jpg>
- <https://www.thetechdevocate.org/wp-content/uploads/2023/05/incognito.png>
- <https://cordova.apache.org/docs/en/dev/guide/overview/index.html>
- <https://developer.android.com/reference/android/webkit/WebView>
- <https://www.bluetooth.com/learn-about-bluetooth/tech-overview/>
- <https://i0.wp.com/embeddedcentric.com/wp-content/uploads/2019/05/BLE-Communication-Options.png>
- <https://mariam.qa/post/hr-ble/>
- <https://www.ti.com/content/dam/ticom/images/products/ic/microcontrollers/simplelink/evm-board/launchxl-cc26x2r1-angled.png>
- <https://chat.openai.com>