Weaknesses Improvement of Bluetooth Low Energy

Quentin Guardia
Université de Paris
Paris, France
quentin.guardia@etu.u-paris.fr

Ludivine Bonnet
Université de Paris
Paris, France
ludivine.bonnet@etu.u-paris.fr

Redouane Otmani
Université de Paris
Paris, France
redouane.otmani@etu.u-paris.fr

Ahmed MEHAOUA
Université de Paris
Paris, France
ahmed.mehaoua@parisdescartes.fr

Osman SALEM
Université de Paris
Paris, France
osman.salem@parisdescartes.fr

ABSTRACT

In an increasingly connected environment, we have to question the security of the objects around us. And in particular objects connected by Bluetooth Low Energy (BLE). Thus, as part of the tutored Master's 1 project, we studied the security of BLE. After dissecting the mechanisms behind this protocol, we identified the attacks known to date. We put some one in practice, which consists in snorting exchanged packets. Finally, in this report we propose various means of securing the objects connected by BLE and thus part of our environment.

1 INTRODUCTION

Context and motivation. The Internet of Things (IoT) refers to the connection of things to the Internet. Sensors occupy a growing place in our society. They are used in a variety of fields, from healthcare to geolocation to industry. The rise of connected objects implies an adaptation of modes of transmission. This is why some wireless networks like LoRa or BLE have emerged. We will focus on the latter throughout the project. After being developed by Nokia in 2006, it integrates with Bluetooth 4.0 under the name we know today, or also under the name of Bluetooth Smart. However, as with any technology, flaws exist. It is our will to understand them in order to master them.

Contributions and organization of the paper In the first part, we will present the generalities and technicalities of BLE. That is, on what modalities the devices communicate with each other. Then we will study in a second time the vulnerabilities of BLE known to date, with the possible security implementations. Finally, we will put an attack into practice, with proposals for securing to finish.

2 BLE OVERVIEW

BLE is a wireless, low-power, Bluetooth-based technology used to connect devices to each other. It consumes about half the energy than Bluetooth, because less data is exchanged. Also, devices can stay in "sleep mode" until the next interaction. It is for these advantages that it is used by sensors, which on the one hand need to send little data, on the other hand lightweight implementation and low consumption. In addition, thanks to these advantages, applications using BLE can run for long periods of time, months or even years. Bluetooth standards are managed by the Bluetooth Special Interest Group (Bluetooth SIG). Thus, we will detail how several devices can communicate with this technology.

2.1 Communication models

BLE has two types of possible connection model.
Master and slave. The master, also called "Central", aims to initialize the connection with one or more slaves. Conversely, a slave, or "Peripheral", cannot initiate a connection and can only connect to a single master. He only executes orders of it. He gets noticed thanks to the advertising packages he sends. The Central is generally associated with a computer or a smartphone. It consumes more than the Peripheral.
Broadcaster and Observer. The Broadcaster does not accept any connection, and detects Observers, to possibly send advertising packets. The Observer, like the Peripheral, is detected through advertising packets sent at regular intervals until a connection request is made.

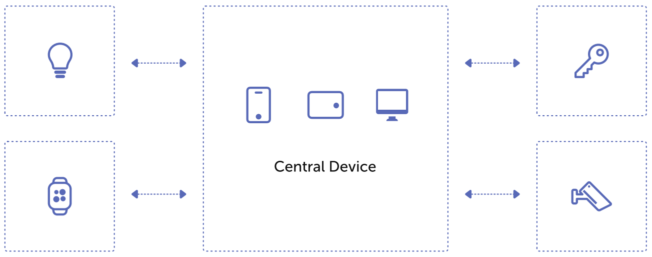


Figure 1: Communication between Central and Peripherals

2.2 The advertising

Thus, the Observer and Peripheral stop sending advertising packets when they receive a specific packet, indicating that they are connected to a Broadcaster or Central. Moreover, we speak of connected mode when the exchange is bidirectional, this which is the case for the Central and the Observer. And advertising mode for one-way connections, for the Peripheral and the Broadcaster.

BLE operates using ultra high frequency radio waves on a 2.4 GHz to 2.8 GHz band. This with 40 physical channels, against 80 for Bluetooth, for frequency and time multiplexing thanks to the L2CAP layer. By simple calculation, the difference between two channels is found to be 2 MHz. The devices in advertising mode send packets of 31 bytes at regular intervals. And this only on 3 of the channels: 37, 38 and 39. The other channels are reserved for data exchange between devices.[2]

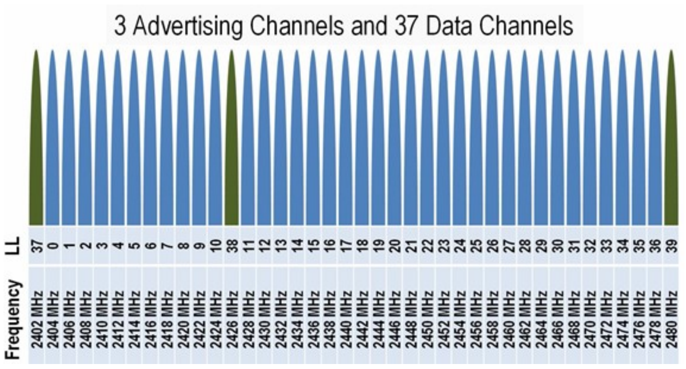
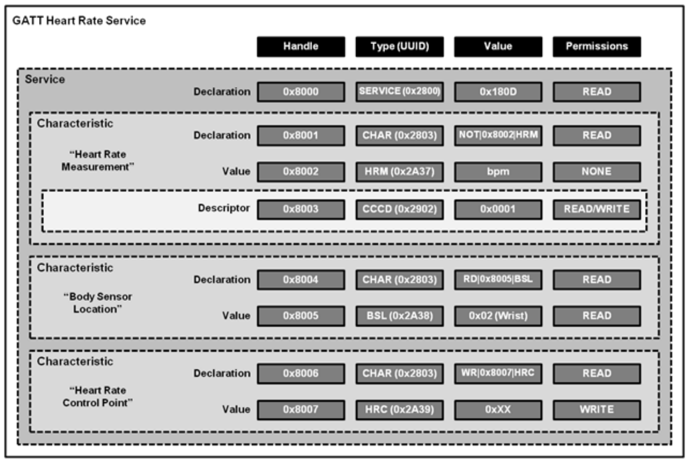


Figure 2: 3 channels used for advertising

In general, the devices send their services, term explained in the next part, their name and the manufacturer's information.
For its part, the Central alternates between scanning (listening to) advertising packets and sending them. He scans to see if he finds the Peripheral he is looking for and then sends to begin the exchange with the Peripheral. The scanning process is expensive so the scan usually does not run indefinitely.

2.3 GATT protocol and data structures

The GATT protocol is used during communication between a master and a slave. Specifically, the slave is a GATT server, and the master is a GATT client connecting to the server. Thus, the client makes a request to obtain data named attributes, to the server. The attributes are in the form of characteristics, services or even descriptor. The characteristics group together descriptors. One of the descriptors, optional, is the final requested value, such as the pulse. There are necessarily two other descriptors, one to declare the characterestic and another to provide more information on the value sent, such as its unit for example.
Services are sets of characteristics. The client waits for the server to return its completed data. It can be a group of characteristics giving information on several parameters of the body for example. Each attribute is recognized by a UUID (Universally Unique IDentifier) which makes it possible to universally define the nature of the information; by a value that instantiates the descriptor, such as the unit of the pulse, any permissions (read, write, both or none) and the handle, a hexadecimal value that recognizes each attribute within the packet. [17]


GATT Heart Rate Service

		Handle	Type (UUID)	Value	Permissions
Service	Declaration	0x8000	SERVICE (0x2800)	0x180D	READ
Characteristic "Heart Rate Measurement"	Declaration	0x8001	CHAR (0x2803)	NOT|0x8002|HRM	READ
	Value	0x8002	HRM (0x2A37)	bpm	NONE
	Descriptor	0x8003	CCCD (0x2902)	0x0001	READ/WRITE
Characteristic "Body Sensor Location"	Declaration	0x8004	CHAR (0x2803)	RD|0x8005|BSL	READ
	Value	0x8005	BSL (0x2A38)	0x02 (Wrist)	READ
Characteristic "Heart Rate Control Point"	Declaration	0x8006	CHAR (0x2803)	WR|0x8007|HRC	READ
	Value	0x8007	HRC (0x2A39)	0xXX	WRITE

Figure 3: GATT Diagram Heart Rate Service

Basically, a GATT client can, via the characteristics, make two types of request: write and read. Respectively to get confirmation or value from the server. The client can also have write permission with write. For its part, the server can send a notification to the client, without waiting for a return. However, if an ACK is required, then it will be referred to as an indication.

To conclude, BLE devices respect a hierarchy. There is indeed a client and a server, a master and a slave. The client is usually the master. By way of illustration, a smartphone can connect in BLE to a weather station to receive the temperature. In this case, the smartphone is the master, or GATT client, and the weather station is the slave, or GATT server. Communication by BLE is special. Indeed, only a few variables are exchanged in what are called characteristics. What makes BLE light. Of course, BLE offers a certain level of security. This is what we will see in the next section.

3 SECURITY

It is known that a malicious user, if close enough to BLE devices, can intercept certain packets. To avoid attacks, the protocols have been made more complex to ensure the integrity, authenticity and confidentiality of the data exchanged. We will present here the methods of encryption, MAC addresses randomization and how the whitelist works.

3.1 Encryption

3.1.1 LE Legacy Connections and LE Secure Connections. For devices equipped with Bluetooth 4.0 and 4.1, we speak of LE Legacy Connections. One of the devices sends a pairing request to the other. From there, the devices exchange their information: their input / output ressources, such as the presence or not of a keyboard, the methods for the authentication of the first pairing and the connection during future re-pairings (bonding) and the maximum size of the key that will be used. Note that for the moment, nothing has yet been quantified.
Then, still during pairing, one of the devices generates a temporary key (TK) which will be known from both devices. Confirmation of the key is made through the exchange of random values, exchanged and encrypted and then decrypted. With the TK and a random value, a new key is generated: the STK. The connection will be encrypted with this key, at the link layer level. Eventually, an LTK can be exchanged for bonding.
For devices armed with Bluetooth 4.2 or higher, which can however use the previously described method, we are talking more about LE Secure Connections. There are no more TK and STK keys, but an LTK key, the computation of which is based on key pairs generated from a Diffie-Hellman elliptical curve. This method is much safer.[13]

Key Generation for LE Legacy Connections

- Just Works. With this method, the TK is initialized to 0. However, it is easy to find the STK by brute force.
- Out of Band (OOB) Pairing. The TK is exchanged by another technology like NFC. There are two advantages here. First, the exchanged key can be longer, up to 128 bits. Then, BLE and the other means of connection are independent and therefore much more secure. This is the most reliable method. As long as the second medium is secure, then all the exchange is secure.
- Passkey. A 6-digit string is exchanged by various possible means. For example, it can be displayed on the screen and validated by both users. This method is quite safe. However, she has already shown to be vulnerable.[14]

Key Generation for LE Secure Connections

- Just Works. After the devices have exchanged their public keys, the device that did not initiate the connection produces a nonce, which is a random value, and uses it to create a Cb confirmation value. Then, Cb and the nonce are sent to the machine which initiated the connection. At the same time, the latter also produces a nonce before sending it. It also uses the nonce it received to create its own Ca confirmation value. If both values match, then the connection can take place. Diffie Hellman elliptical curve provides additional security. However, this method remains the most vulnerable.
- Out of Band (OOB) Pairing. Public keys, nonces and confirmation values are exchanged by means other than Bluetooth, such as NFC.
- Passkey. Both devices use the exchanged public keys, the 6-digit code displayed on both screens and the 128-bit nonces to authenticate the connection. Here, each bit of the passkey is exchanged and then confirmed as in the Just Works method. Once each bit is mutually confirmed, the secure exchange begins.
- Numeric Comparison. This method takes Just Works and adds another step. After the Just Works verified, each machine generates 6 digits using the nonces. These 6 digits are then displayed and the user must verify that they correspond to the expected value.

To recap, here are the association models used depending on the hardware.

D2 / D1	Display	Display, Yes/No	Keyboard	No input/ output	Keyboard, display
Display	Just works	Just works	Passkey	Just works	Just works
Keyboard only	Passkey	Passkey	Passkey	Just works	Passkey
No input/output	Just works	Just works	Just works	Just works	Just works
Keyboard, display	Passkey	Numeric comparison	Passkey	Just works	Numeric comparison

Table 1: Hardware-based-association-models [18]

3.2 MAC addresses Randomization

As with other types of connection, a BLE device is identified by a 48-bit MAC address. Nowadays, four formats are identified

- IEEE public format: The 24 most significant bits designate the company, and the other 24 are to be defined by the company. Here the address cannot change, which does not provide maximum security.
- Random static: The MAC address is updated randomly during each new start. Thus, some protection is provided if the device is restarted regularly.
- Resolvable private random:This method is used, provided that, an Identity Resolving Key (IRK) is exchanged during pairing. Here, a device creates a random MAC address from the IRK, which is transmitted in an advisory packet. The other device can deduce the original address from it in order to continue the communication. A new address is generated periodically, which provides some protection.
- Non-resolvable private random: This time, the address is converted to a random number. This number can be renewed very regularly, which provides strong security.

In fine, the fact that only the paired machine can guess the original MAC address from a randomized address avoids the phenomenon of tracking.[16]

3.3 Whitelist

The whitelist is a method that allows devices to filter other devices according to their MAC addresses. The filtering policy determines how an advertiser can handle the scan or connection request. When advertising filtering policies are applied, devices can only allow whitelist scan or connection requests. To do this, a device must maintain a locally whitelist, which stores the address and its type. Thus, a device does not need to respond to every connection request, as most are filtered.

To conclude, the data is end-to-end encrypted. The hacker must therefore intercept the data when it is still likely to be transmitted in clear, ie when pairing. Then, two devices connected in BLE are generally quite close. A The hacker would therefore need to physically interfere between both devices, which is rarely possible. For these reasons, it does not seem easy to carry out an attack on BLE. In practice, not all BLE devices necessarily apply the previously mentioned security methods. Indeed, it may be deemed unnecessary in some cases. For example on certain cashier payment systems. Or difficult to set up. To illustrate, it is conceivable that it is difficult to set up a very secure pairing system when several Peripherals are involved.
In addition, it is possible to secure the application layer. Especially since, even the connection between two devices encrypted, data once on the device is available on all apps.
We will see in the next section the persistent vulnerabilities despite all the security mentioned.

4 POSSIBLE ATTACKS AND PROTECTION PROPOSALS

Despite the security measures previously stated, some attacks are known to date. They range from simple passive data interception to identity theft and denial of service. As this is no longer a zero day vulnerability, answers have already been offered to these attacks. We present them below the attacks with their solutions.

4.1 Attacks on advertisements

4.1.1 Principle. The advertising packets sent by connected devices can be spoofed and deceive the targeted smartphone or the Central in general. In the interest of battery optimization, devices often increase the interval between two advertisements. Thus, a hacker can jam the device and retrieve his advertising packets which he sends at a much higher frequency and when he wants to Central. In fact, the first packet received by the Central comes from the pirate machine for the attack to be successful. In addition, when the machine is in online mode, with the pirate machine for example, then it no longer sends out advertisements. Which can favor the pirate's task, which aims to send packets to Central instead of the original Peripheral. So the simplest attack is denial of service. All the hacker has to do is pretend to be the Central, in order to intercept the packets from the Peripheral. In this way, we easily prevent the connection between the Peripheral and the Central.

4.1.2 Attacks.

- Denial of service in home automation: More and more, smartphones are connected to objects in the house: light bulb, lock, etc. The smartphone remembers the MAC address of the connected objects. A hacker just needs to spoof the MAC address of a connected object, establish the connection with the smartphone and send false packets (containing "on" instead of "off" for example). Thus, the smartphone can no longer control the original connected object.[1]
- Compromise of an anti-theft device: On the same model, there are luggage whose lock is connected by BLE to the smartphone. If a hacker spoofs the anti-theft MAC address and sends packets containing the opposite information to the real one, then the security of the baggage is compromised.[7]
- Beacon abuse: A beacon is a small BLE box that detects the proximity of a smartphone. There is an app that allowed you to earn points by visiting places. The points were redeemable for gifts. The application encountered problems because it was possible to spoof packets sent by the beacons. This had the effect of earning points for stationary smartphones. This required knowing the value of some original beacon descriptors: UUID, GPS coordinates, etc.[8]

4.1.3 Countermeasures. To counter this, additional signature and encryption systems have been introduced for broadcast values. For example, some vendors have made the packages contain a predefined value that is changed periodically. Only the seller's smartphone app can check the value of the package. However, this kind of system has its limits. It must be compatible with both hardware and software. And sometimes an internet connection is required. Finally, the seller does not necessarily have his encryption algorithm verified by experts because these are unofficial implementations. So there is always a risk. The safest thing is not to trust advertising packages for important things.

4.2 Passive interception

4.2.1 Principle. Unencrypted packets can easily be spied on. There is a lot of accessible material that allows for such an attack. This is the case with the Ubertooth key sold by Great Scott Gadget [5] or a dongle using nRF51822 Nordic Semiconductor [11]. The hardware, accompanied by the corresponding software system, allows packets to be sniffed using applications such as Wireshark. This is still possible through active attacks as we will see in the next section.

4.2.2 Attacks.

- Smart finder: "smart finder" is a connected object that has 6-digit authentication. These 6 digits are sent unencrypted by the smartphone to the connected object, for security. It was possible to see its 6 digits in clear thanks to the GATTacker tool.
- Beacons management: Usually, beacons have a static password. Specifically, each device has its own configured password. To connect, the associated smartphone application may therefore send the password in clear text.
- OTP authentication token: "One Time Password" allows a device to connect with a one-time password, with 6 digits in BLE. However, it has been observed with sniffers that sometimes passwords are not encrypted. This is not necessarily dangerous in a passive attack, because once used the password is no longer valid. In contrast, during an active attack, a third party can recover the password to impersonate the client.[9]

4.2.3 Countermeasures. The attacks mentioned are similar and therefore require similar solutions. Already, as one can trivially guess, it is important to encrypt all the characteristics, in particular with the security tools offered by the Bluetooth link layer. Encryption can also be implemented on the higher layers according to the proprietary protocol. It would then be necessary to test the robustness of the implementations for more security. At the same time, it would be necessary to ensure the identity of each device before pairing, in addition to the confidentiality of the message.

4.3 Active interception

4.3.1 Principle. Unlike passive interception, the hacker impersonates both the Central to receive the packets from the Peripheral, and the Peripheral to send the packets to the Central. For this, it uses the modification of its MAC address, for example possible with the bdaddr program. Thus the packets pass through the pirate machine, and the pirate can modify them and re-send them. The GATTacker software makes it possible to do this, here is a diagram where we see the IoT communicating with the Central, thinking that it is the smartphone, and the smartphone exchanging with the Peripheral instead of the IoT. We are talking about a Man In The Middle (MITM) attack.[6]

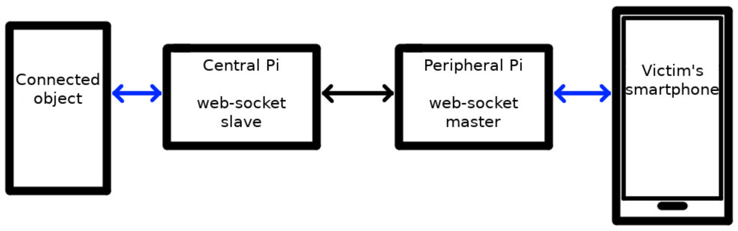


Figure 4: MITM with GATTacker

4.3.2 Attacks.

- Data manipulation: This flaw exists on certain payment terminals connected to a smartphone. The data exchanged between the terminal and the smartphone is encrypted. However, some commands, especially those for displaying text on the terminal, are not. By analyzing the cyclic redundancy check on the smartphone using clear text, it is possible to display any text on the terminal by MITM. Coupled with social engineering, it is theoretically possible to rip off the seller, by displaying "Payment accepted" on the terminal after payment when this is not the case.
- Command injection: It has happened that a smart car system has been hacked by command injection [12]. More precisely, the victim's smartphone connects to the car, to lock it, for example, in a challenge-response authentication model. Then, the application communicates unencrypted with the car through the session. A hacker can, without tampering with the authentication process, intercept the session and send commands to the car, including changing session keys to take full control of the car. This is facilitated by the fact that the implementation of security is often carried out by the manufacturer itself, which does not guarantee robustness. In addition, the mobile app sends its data automatically to the car through background processes. So if the smartphone is too far from the car for the car to detect it, but the hacker is halfway there and can connect to both, then it is a risky situation. We talk about Bluesnipping when the hacker seeks to extend his Bluetooth range, which can be used in this kind of attack.
- Replay: This attack can be used with smart locks, which have been its victims before, such as the early generations of August Smart Lock. Here, the exchanges are well encrypted. On the other hand, there is no security against replay, that is to say the resending of packets already sent, even encrypted. Concretely, the victim may receive a packet indicating that the lock is properly locked when it is not. To do this, a weakness is used. For each given challenge, the smartphone sends the same encryption key. This makes it easy for the hacker to listen to the exchange between the lock and the vulnerable application, first keys, then communication. The next time, during the authentication process, the hacker device can pretend to be the lock and send the same challenges the lock would send. Since the hacker knows the key, he can decipher the entire exchange. And that's when it can play the replay, sending a packet back to the user saying the lock is locked, when it isn't.

4.3.3 Countermeasures. These are mainly the same countermeasures advocated against passive interception. In addition, a feature can be added to verify that there has been no attempt to intercept or modify the data. For this, there are Cyclic Redundancy Check (CRC) or checksum algorithms, ensuring data integrity. In other words, it is necessary to ensure the authenticity of the characteristics.

4.4 Free access devices

4.4.1 Principle. It is common for a device to be available for any attempt to connect with an unknown device. Which is even more risky in the absence of prior encryption. However, a hacker can find loopholes and exploit them, although they are not generic.

4.4.2 Attacks.

- Non-secure interfaces: Modules can connect to certain interfaces without authentication, and send GATT requests like write or notify. Depending on the interface, some requests may disrupt the interface and interfere with its proper functioning, depending on the on-board system. We can detect those interfaces which do not require a connection. They are therefore susceptible to attacks.
- Brute force: The majority of Peripherals and beacons do not limit the number of connection attempts. On 6 digits, a pirate can perform a brute force attack without difficulty.[15]
- Random deficiency: Some modules do not include a random number generator. Developers are forced to rely on some pre-existing variables to generate low random numbers. For example, some modules multiply their Serial Number with the ambient temperature [4]. It can be critical. Indeed, always during a challenge-response authentication, if a hacker guesses the value of the challenge, then he can carry out an attack by MITM.
- Too many services available without authentication: Too many services may be accessible. While individually accessing a single service is safe, accessing all services can reveal information that is believed to be confidential.
- Poor entries management: This goes back to the first point. In some cases, sending outliers as characteristics can compromise the behavior of the device in question. This happens when the received variables are not checked.
- More generally, all the flaws in logic: Here is just one example of many. It may happen on a device that stores multiple keys for different connections. A user wishing to connect must then give the index of the key as well as its value. If a malicious user enters an out of range index and predicts the value written to memory at that address, then they can log into the device.

4.4.3 Countermeasures. Here you just need to be conscientious and to test the systems well during their design. Using the examples above, we can limit access to certain data or check each entry. Likewise, you can limit the duration of the device's activity, to reduce the range of possible attacks.

4.5 Attacks during pairing

4.5.1 Principle. As we have seen, encrypting an exchange is essential. To encrypt an exchange, you must first pair both systems. However, there may be loopholes in this step, which can even be exploited by social engineering.

4.5.2 Attacks.

- Just Works flaws: Just Works is frequently used during pairing. When linking a connected object to a smartphone, the hacker can approach the device to try to access the characteristics and thus create a new link. Then the hacker can create a clone of the object to impersonate him. From there, the smartphone connects to the clone. If he doesn't verify the presence of the MAC address in his list, it's good. The smartphone does not have LTK keys associated with the MAC address to encrypt, but being well connected to the device, it will communicate by encrypting with what it has: nothing. On the other hand, if the presence of the MAC address is verified, then the attacker must clone it. But it ends there as the attacker lacks LTK, which quickly disconnects him.[10]

- Pin code: As explained above, a hacker can clone a device while linking. As the exchange is not secure, the smartphone is automatically disconnected. The hacker then deactivates his clone and can snort the exchange during the pairing process. During the next pairing, the hacker will be able to crack the PIN code using Crackle in order to recover the LTK [3]. This is how an active interception can begin.
- SCO mode: The Secure Connection Only mode disconnects the device when the connection is not secure. However, BLE has already been compromised at multiples by a flaw using SCO.[19]

4.5.3 Countermeasures. To counter this, it is simply recommended to use the most secure pairing mechanisms, which we saw in the previous section. In addition to this, you can have the pairing initiated by manual action on the connected device, such as pressing a button. Finally, mobile applications should warn the user when an attempt is made to intercept data, so that the user can take action, such as strengthening security, changing the password or even finding which device is trying to hack.
Finally, we can add security at the application layer to verify the existence of the identity of the people connecting.

4.6 Bypassing the whitelist

This is a fairly straightforward flaw to understand. It has already been explained that some devices have a list of trusted MAC addresses. If a hacker learns about it by some means, then he can change his own MAC address to be part of it and connect to the victim. All that is needed here is other means of verification, such as an LTK.

4.7 Confidentiality breach

We can also approach things in the following way. Data contained in public advertising packets may be collected. This poses a problem if this data is to be kept confidential. This is specifically what we will look at as we practice in the next section.
Beyond that, the ability to see which devices are connected and when is a threat to user privacy. We are talking about Bluetracking. The hacker can know the log of connections of every device within his reach. This can also help him to anticipate a potential attack.

To conclude, there are several categories of malicious acts: passive eavesdropping, active attack by MITM, denial of service, etc. Some vulnerabilities require special equipment, such as a sniffer. An attacker should also know the vulnerabilities of the devices they are trying to hack, because not all devices have the same vulnerabilities. Manufacturers have adapted their devices to circumvent these attacks. One of the weaknesses lies in the encryption mechanisms that are implemented by these manufacturers, which are not necessarily robust. In addition to encryption, security should ideally be holistic.

5 BLE ATTACK BY SNIFFING

After reviewing a large part of the attacks, we put one into practice. This is a passive data interception. For this, we have a smartphone and a Raspberry Pi 4 with BLE. We also have a computer with a sniffer BlueFruit nrf51822 V2. The Raspberry will act as Peripheral, and Central's smartphone.

The Peripheral must send a heart rate, obviously simulated, without encryption. This data can be considered sensitive in the fields of medicine and confidentiality. The heart rate must be read by the Central. As part of the attack, the computer will intercept data through the sniffer, and read heart rates without other devices noticing.

Figure 5: Sniffer BlueFruit

5.1 Preparing the GATT server

First, we will prepare the GATT server. For this, we retrieve a code available on Github. It is based on BlueZ technology and is written in C. Here are the commands to download and compile it:

```
git clone https://github.com/evanslai/bluez-gatt.git
```

```
cd bluez-gatt
mkdir build
cd build
cmake ../
make
```

If cmake is not installed, then write the command below before recompiling.

```
sudo apt-get install build-essential cmake
```

GATT server is ready.

5.2 Preparing the GATT client

On the smartphone side, many applications allow the device to behave like a Central BLE. On Google PlayStore, we opted for “BLE Scanner”.

5.3 Setting up the pirate device

5.3.1 Drivers. First, you need to install the necessary drivers for the sniffer to work properly. We have a "Black Board" model, so the drivers of SiliconLabs are needed regardless of the OS. For a "Blue Board", you need the drivers of FTDI Chip.

5.3.2 Installing AdaFruit API. We install the API provided by AdaFruit which allows to exploit the capabilities of the key. This API developed in Python allows you to retrieve the data sent on the Serial Port of the sniffer and to structure them in a pcap file. Thus, we can open the intercepted packets with Wireshark.

```
git clone
https://github.com/adafruit/Adafruit_BLESniffer_Python.git
cd Adafruit_BLESniffer_Python
```

You must of course have Wireshark on the machine. In Linux, the appropriate command is as follows:

```
sudo apt-get install wireshark
```

While on Mac or Windows you have to go to the Wireshark download page.

5.3.3 Determining the Serial Port. We now determine the Serial Port associated with the sniffer. On Linux, it is of the form /dev/ttyXX-XX. We can use the following command:

```
dmesg | grep tty
```

On Windows, it is of the form COMX. We open the Device Manager and search in the “COM & LPT” category. Normally this is a COM port.
On Mac, its like /dev/tty.X. the command below is needed to find it:

```
ls /dev/tty.*
```

5.3.4 Python et pySerial. We tried running the API using Python 3.6.9, but it didn’t work, but Python 2.7 did. We therefore recommend using this latest version. In addition, the six and pySerial packages are indispensable. It is therefore necessary to ensure their installation.

5.4 Data interception (sniffing)

The pirate machine is preferably placed between the client and the GATT server.

5.4.1 Launch of the GATT server (Raspberry). On the Raspberry Pi, we look at the Bluetooth MAC address using the command

```
hciconfig
```

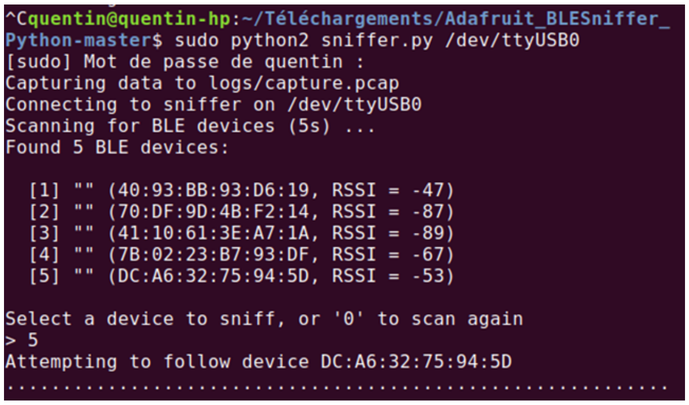
We keep it in mind.Now we are in the bluez-gatt/build folder. We start the server with the -r argument to activate the service "Heart Rate".

```
chmod +x btgatt-server
./btgatt-server -r
```

5.4.2 API launch (hacker computer). To launch the API, you must first be in its folder. We then use the command:

```
python2 sniffer.py <serial-port>
```

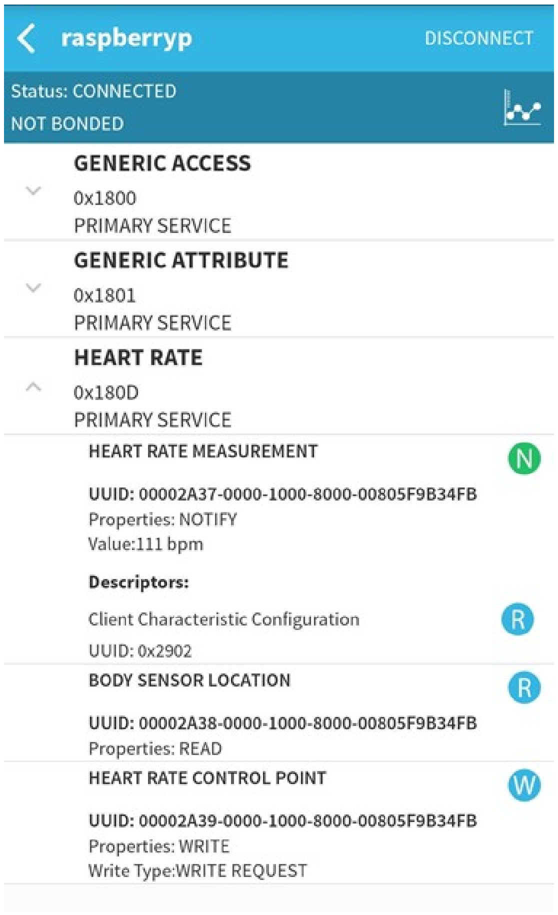
You must add sudo before if you are on Linux, and simply call python if the version is adequate.
Note the path to the logs and captured packets, which is displayed in the first lines. The API first connects to the sniffer. Then, it scans all surrounding BLE devices. After identifying them with their Bluetooth MAC address and RSSI, it numbers and lists them. The closer the machine, the greater the RSSI. We must then choose the right device. As we are both hacker and user, our task is simplified, we know which MAC address to choose. The GATT server is now bugged.



```
^Cquentin@quentin-hp:~/Téléchargements/Adafruit_BLESniffer_
Python-master$ sudo python2 sniffer.py /dev/ttyUSB0
[sudo] Mot de passe de quentin :
Capturing data to logs/capture.pcap
Connecting to sniffer on /dev/ttyUSB0
Scanning for BLE devices (5s) ...
Found 5 BLE devices:

  [1] "" (40:93:BB:93:D6:19, RSSI = -47)
  [2] "" (70:DF:9D:4B:F2:14, RSSI = -87)
  [3] "" (41:10:61:3E:A7:1A, RSSI = -89)
  [4] "" (7B:02:23:B7:93:DF, RSSI = -67)
  [5] "" (DC:A6:32:75:94:5D, RSSI = -53)

Select a device to sniff, or '0' to scan again
> 5
Attempting to follow device DC:A6:32:75:94:5D
...........................................................
```

Figure 6: AdaFruit Sniffer API

5.4.3 Client connection (smartphone). On the application, we connect to the Raspeberry. It can be found when scanning using RSSI, name and/or MAC address. Once connected, we open the “Heart Rate” characteristic. To display the heart rate, you simply have to authorize notifications, by pressing "N" for example.

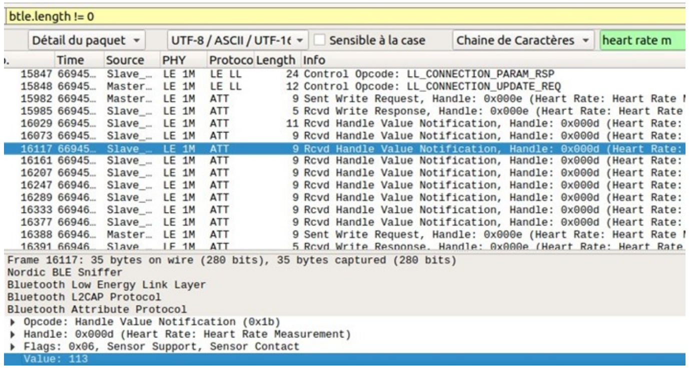


```
raspberryp                                DISCONNECT
Status: CONNECTED
NOT BONDED

GENERIC ACCESS
0x1800
PRIMARY SERVICE

GENERIC ATTRIBUTE
0x1801
PRIMARY SERVICE

HEART RATE
0x180D
PRIMARY SERVICE

  HEART RATE MEASUREMENT                            N
  UUID: 00002A37-0000-1000-8000-00805F9B34FB
  Properties: NOTIFY
  Value:111 bpm
  Descriptors:
  Client Characteristic Configuration               R
  UUID: 0x2902

  BODY SENSOR LOCATION                              R
  UUID: 00002A38-0000-1000-8000-00805F9B34FB
  Properties: READ

  HEART RATE CONTROL POINT                          W
  UUID: 00002A39-0000-1000-8000-00805F9B34FB
  Properties: WRITE
  Write Type:WRITE REQUEST
```

Figure 7: BLE Central

5.4.4 Passive listening to the server (hacker computer). Despite the error-free exchanges between the client and the GATT server, the API captures all packets in transit. You can stop listening at any time using the Ctrl+C system call. We open the Wireshark file named "capture". Remember, its path was indicated when the API was launched. Moreover, in the event of a problem, we can refer to the logs in the same folder.

Once on Wireshark, we look for packages containing the "Heart Rate Measurement", the heart rate measurement. We will find them in the packets including the protocol associated with ATT for Attribute, which leave from the Slave to the Master. For better readability, empty packets can be removed with the “btle.length != 0” filter. We look for the packages that interest us with the keyboard shortcut Ctrl+F. The packages that interest us contain the character string "Heart Rate Measurement", in "Package details".



```
btle.length != 0
Détail du paquet ▾   UTF-8 / ASCII / UTF-1( ▾   ☐ Sensible à la case   Chaine de Caractères ▾   heart rate m
).         Time      Source    PHY    Protoco Length Info
   15847 66945…  Slave_…  LE 1M  LE LL      24 Control Opcode: LL_CONNECTION_PARAM_RSP
   15848 66945…  Master…  LE 1M  LE LL      12 Control Opcode: LL_CONNECTION_UPDATE_REQ
   15982 66945…  Master…  LE 1M  ATT         9 Sent Write Request, Handle: 0x000e (Heart Rate: Heart Rate !
   15985 66945…  Slave_…  LE 1M  ATT         5 Rcvd Write Response, Handle: 0x000e (Heart Rate: Heart Rate
   16029 66945…  Slave_…  LE 1M  ATT        11 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16073 66945…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16117 66945…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16161 66945…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16207 66945…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16247 66946…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16289 66946…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16333 66946…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16377 66946…  Slave_…  LE 1M  ATT         9 Rcvd Handle Value Notification, Handle: 0x000d (Heart Rate:
   16388 66946…  Master…  LE 1M  ATT         9 Sent Write Request, Handle: 0x000e (Heart Rate: Heart Rate !
   16391 66946…  Slave_…  LE 1M  ATT         5 Rcvd Write Response, Handle: 0x000e (Heart Rate: Heart Rate
Frame 16117: 35 bytes on wire (280 bits), 35 bytes captured (280 bits)
Nordic BLE Sniffer
Bluetooth Low Energy Link Layer
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
 ▸ Opcode: Handle Value Notification (0x1b)
 ▸ Handle: 0x000d (Heart Rate: Heart Rate Measurement)
 ▸ Flags: 0x06, Sensor Support, Sensor Contact
   Value: 113
```

Figure 8: Wireshark displaying heart rate

Thus, all heart rate measurements are accessible. There can be no privacy without encryption.

To conclude, ultimately, a malicious person can intercept data for only a few tens of euros. The procedure to follow is extremely simple if the data is transmitted in the clear. All it takes is a tool like crackle if they are encrypted with a method like Just Works. You can imagine that with more time and knowledge than we do, it is possible to have a much bigger action, like doing replay. In our case, an encryption algorithm, at the link layer level for example, would already have been sufficient to avoid such listening. Every vulnerability has a suitable solution.

6 CONCLUSION

The risks associated with BLE have increased in proportion to the growth of the protocol. Despite standardized security mechanisms, many flaws remain exploitable. They can be responsible for passive or active interceptions, data tampering or denial of service. It is the responsibility of the manufacturers of hardware and developers implementing the systems to prevent risks. As we have seen, there are many ways to ensure the security of a BLE terminal and an exchange, from the robustness of the encryption to the verification of entries through a secure pairing.

In a few months it has been possible, with some difficulties, to compromise the confidentiality of BLE data. In this way, we could understand how this precise protocol could be secured. This is all the more important as BLE is used in various important fields, including medicine. With a little more time, we could have looked at another network, such as LoRa. It also has a significant place in the world of telecommunications and IoT, but it works very differently.

REFERENCES

[1] Biren Benchoff "The Terrible Security Of Bluetooth Locks". August 8, 2016. https://hackaday.com/2016/08/08/the-terrible-security-of-bluetooth-locks/

[2] Bluetooth SIG. « Bluetooth Radio Versions ». https://www.bluetooth.com/learn-about-bluetooth/radio-versions/.

[3] "crackle, crack Bluetooth Smart (BLE) encryption". April 24, 2014. http://lacklustre.net/projects/crackle.

[4] Reilly Grant et Ovidio Ruiz-Henríquez. "Web Bluetooth, Draft Community Group Report". May 4, 2021. http://webbluetoothcg.github.io/web-bluetooth.

[5] Projets IMA. "Evaluation des attaques sur protocole Bluetooth". 2017. https://projets-ima.plil.fr/mediawiki/index.php/Evaluation_des_attaques_sur_protocole_Bluetooth

[6] Sławomir Jasek. "GATTACKING BLUETOOTH SMART DEVICES". https://raw.githubusercontent.com/securing/docs/master/whitepaper.pdf.

[7] Sławomir Jasek. "Blue picking –hacking Bluetooth Smart Locks". March 3, 2017. https://conference.hitb.org/hitbsecconf2017ams/materials/D2T3%20-%20Slawomir%20Jasek%20-%20Blue%20Picking%20-%20Hacking%20Bluetooth%20Smart%20Locks.pdf

[8] Perm Soonsawad Kang Eun Jeon James She et Pai Chet Ng. "BLE Beacons for Internet of Things Applications : Survey Challenges and Opportunities". 2018. https://www.researchgate.net/publication/322201975_BLE_Beacons_for_Internet_of_Things_Applications_Survey_Challenges_and_Opportunities.

[9] Kwangsu Cho Hyunhee Jung Dongryeol Shin et Choonsung Nam. "BLEOTP Authorization Mechanism for iBeacon Network Security". August 2015. https://www.researchgate.net/publication/283191016_BLE-OTP_Authorization_Mechanism_for_iBeacon_Network_Security.

[10] Karim Lounis et Mohammad Zulkernine. "Bluetooth Low Energy Makes "Just Works" Not Work". April 2, 2020. https://hal.archives-ouvertes.fr/hal-02528877/document

[11] Nordic Semiconductor. "nRF51822". https://infocenter.nordicsemi.com/pdf/nRF51822_PS_v3.4.pdf.

[12] Maya Posch. "Hands-On : Wireless Login With The New Mooltipass Mini BLE Secure Password Keeper". July 23, 2020. https://hackaday.com/2020/07/23/hands-on-wireless-login-with-the-new-mooltipass-mini-ble-secure-password-keeper/.

[13] Kai Ren. "Bluetooth Pairing Part 3 – Low Energy Legacy Pairing Passkey Entry". August 25, 2016. https://www.bluetooth.com/blog/bluetooth-pairing-passkey-entry/.

[14] Tomas Rosa. "Bypassing Passkey Authentication in Bluetooth Low Energy". 2013. https://eprint.iacr.org/2013/309.pdf.

[15] Anthony Rose et al. "Securing bluetooth low energy locks fromunauthorizedaccess and surveillance". June 20, 2018. https://hal.inria.fr/hal-01819142/document

[16] Martin Woolley. "Bluetooth Technology Protecting Your Privacy". April 2, 2015. https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/.

[17] Martin Woolley. "Advertising Works Part 2". February 9, 2016. https://www.bluetooth.com/blog/advertising-works-part-2/

[18] Yue Zhang, Jian Weng et Xinwen Fu. « B Bluetooth Low Energy (BLE) Security and Privacy ». October 2019. https://www.researchgate.net/publication/336360887_B_Bluetooth_Low_Energy_BLE_Security_and_Privacy.

[19] Yue Zhang et al. "Breaking Secure Pairing of Bluetooth Low Energy Using Downgrade Attacks". August 14, 2020. https://www.usenix.org/system/files/sec20-zhang-yue.pdf.

7 GLOSSARY

ACK	ACKnowledgement
API	Application Programming Interface
ATT	ATTribute
BLE	Bluetooth Low Energy
Bluetooth SIG	Bluetooth Special Interest Group
CRC	Cyclic Redundancy Check
GATT	Generic ATTribute
IoT	Internet of Things
IRK	Identity Resolving Key
LTK	Long Term Key
MAC	Media Access Control
MITM	Man In The Middle
NFC	Near Field Communication
OOB	Out of Band
OTP	One Time Password
PIN	Personal Identification Number
RSSI	Received Signal Strength Indication
STK	Short Term Key
TK	Temporary Key
UUID	Universally Unique IDentifier

Advertising	Sending packets to notify terminal's presence
Bluesnipping	Technique used by the attacker to increase his Bluetooth range
Bluetracking	Bluetooth information tracking
Bonding	Used to plan future connections after pairing
Central	Terminal that initiates connection with one or more Peripherals
Characteristic	Element containing a variable and possibly descriptors
Descriptor	Variable giving information about a characteristic
GATT Client	Terminal that receives the characteristics of a one or several servers
GATT Server	Terminal that sends its characteristics to a client
Master	Terminal initiating the connection to obtain characteristics
Pairing	Pairing two Bluetooth devices
Peripheral	Can accept the connection of a Central
Service	Set of characteristics
Slave	Terminal sending its characteristics to the master