Introduction

Nowadays the majority of electronic devices use the Bluetooth technology to communicate with other equipment.
For example smartband, headphones, car audio, body scale, medical devices, etc …

Bluetooth is very useful to make possible the interconnection of different devices, but at the same time it also presents various data security risks. While supporting the use of key authentication and encryption, many devices rely on a four-digit numeric PIN and not much more secure passwords.

Therefore, the goal of the following work is to exploit the vulnerabilities of BLE technology used by a body fat scale to communicate the detected parameters to a mobile device.

Bluetooth Low Energy (BLE)

Bluetooth Low Energy (Bluetooth LE, colloquially BLE, formerly marketed as Bluetooth Smart) is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group (Bluetooth SIG) for new applications in healthcare, fitness, and security , for the home entertainment industry and for the automotive and automation industries.
The difference to the "classic" Bluetooth technology: Bluetooth Low Energy is intended to provide significantly lower power consumption and cost while maintaining a similar communication range.
Bluetooth devices have the ability to implement security measures for pairing (called pairing) and subsequently for communication.
In the first case it is possible to implement a pin (usually 6 characters) to be typed on the device to be connected, in the second case the transmitted data can be symmetrically encrypted with the AES-128 algorithm.

Poweradd body fat scale data interpretation (BLE)

We bought a Poweradd Body Fat scale [1], this device supports height body parameters such as:

- Weight
- Body water
- Body fat
- Visceral fat
- Bone weight
- BMI
- BMR
- Muscle mass

This scale communicates the transmitted data with the **HealthU +**[2] app, available on the play store.
In our case (as is often the case) neither of the two security measures was put in place. In case the pairing was protected by the pin it would still be possible to decrypt the transmitted packets thanks to the crackle software.
In order to sniff the traffic it was necessary to have a bluetooth dongle available, which implements the monitor mode that allows sniffing of the surrounding traffic.
The device we have equipped ourselves with is the ubertooth one[3].

Uberthoot One

The Ubertooth One is an open source Bluetooth test tool from Michael Ossmann. It is the world's first affordable Bluetooth monitoring and development platform and is a fully open source product (both hardware and software).

Steps performed :

1. (terminal) `mkfifo /tmp/pipe`
2. (wireshark) add as a pipe interface /tmp/pipe and run
3. (teminal) `ubertooth-btle -f -c /tmp/pipe`
4. (wireshark) save file as .pcap or .pcapng

---

[1] https://www.amazon.it/Poweradd-composizione-analizzatore-Technology-idratazione/dp/B079BCYQSM
[2] https://play.google.com/store/apps/details?id=com.lefu.healthu&hl=en_US&gl=US
[3] https://greatscottgadgets.com/ubertoothone/

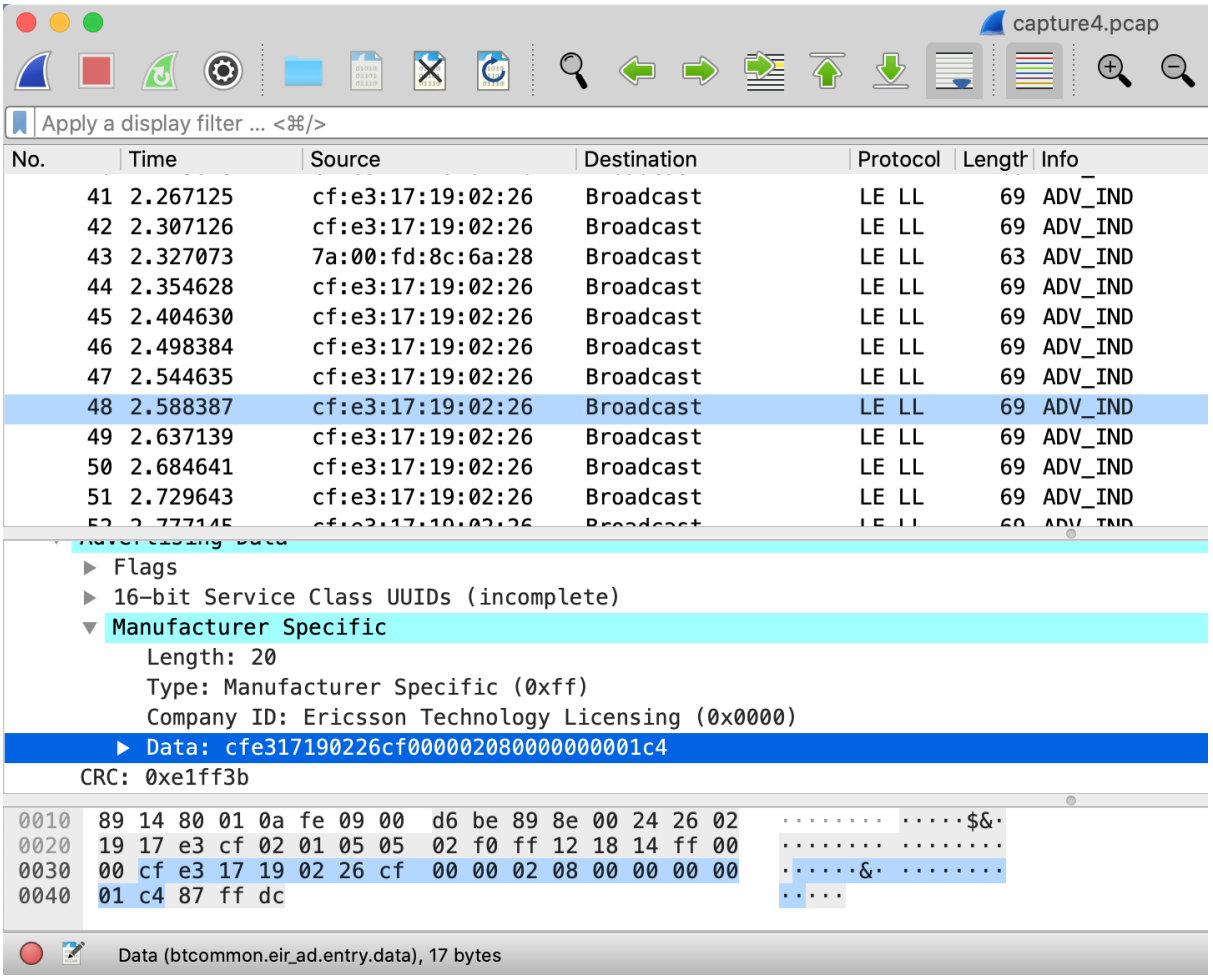

Figure 1. Some of the captured traffic.

We try to understand the hex string:

Type	Comp. ID	MAC	Scale Type	Weight	Impedance set	Impedance
FF	0000	CFE317190226	CF	0208	01	c4

To make the process of understanding the data easier it was necessary to retrieve the manual of the chip (Holtek Body Fat Scale MCU HT45F75)[4]

The parameters set to zero and always zero in the various measurements is due to the fact that the chip inside the scale is not fully used. Below some photos of the chipset.

[4] https://www.holtek.com/productdetail/-/vg/45F75

To understand the use of the data sent, it was necessary to reverse engineer the app, retrieving the .apk file through ApkPure.
In this phase, various tools have been used to have the possibility to debug a phone without root permissions.

Tools:

- Adb
- Jadx
- Dex2jar
- Android Studio

With adb it was possible to interface with the device and perform a backup of the data saved by the app.

```
adb backup -f mybackup.ab -apk com.myapp
```

The .ab file or the backup was converted to a tar using openssl

```
dd if=mybackup.ab bs=24 skip=1|openssl zlib -d > mybackup.tar
```

Unpacked in the following way

```
tar tvf mybackup.tar
```

In the extracted folder /apps/com.lefu.healthu/ we find the db folder where all the db created by the app are present.

Our Attention was focused on the nationalEnergy.db db which contains the tables shown in the screenshot, the records recorded with the scale are present in the BODY_FAT table.

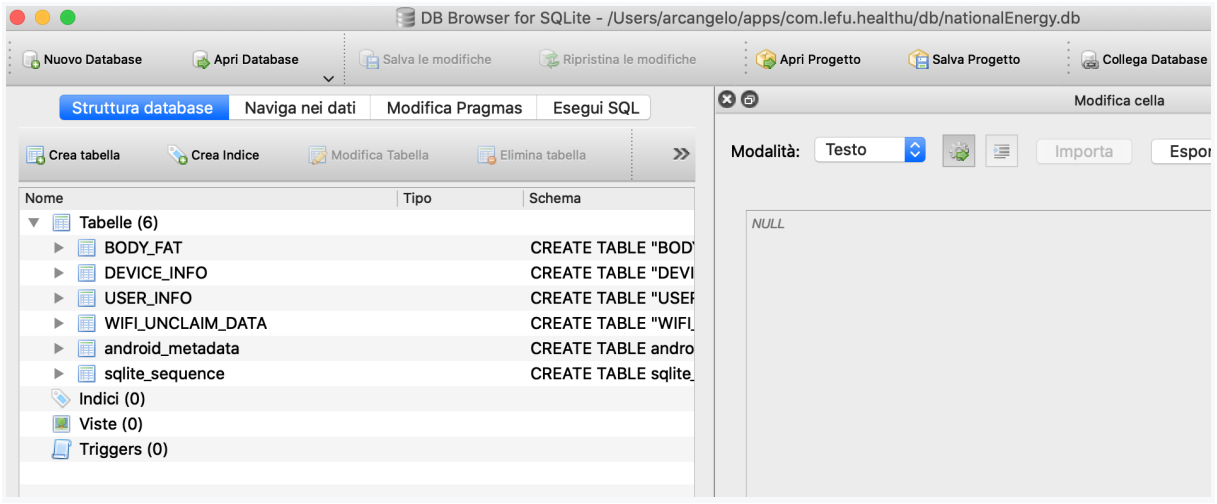

Figure 2. Available Tables

Fields of BODY_FAT table:

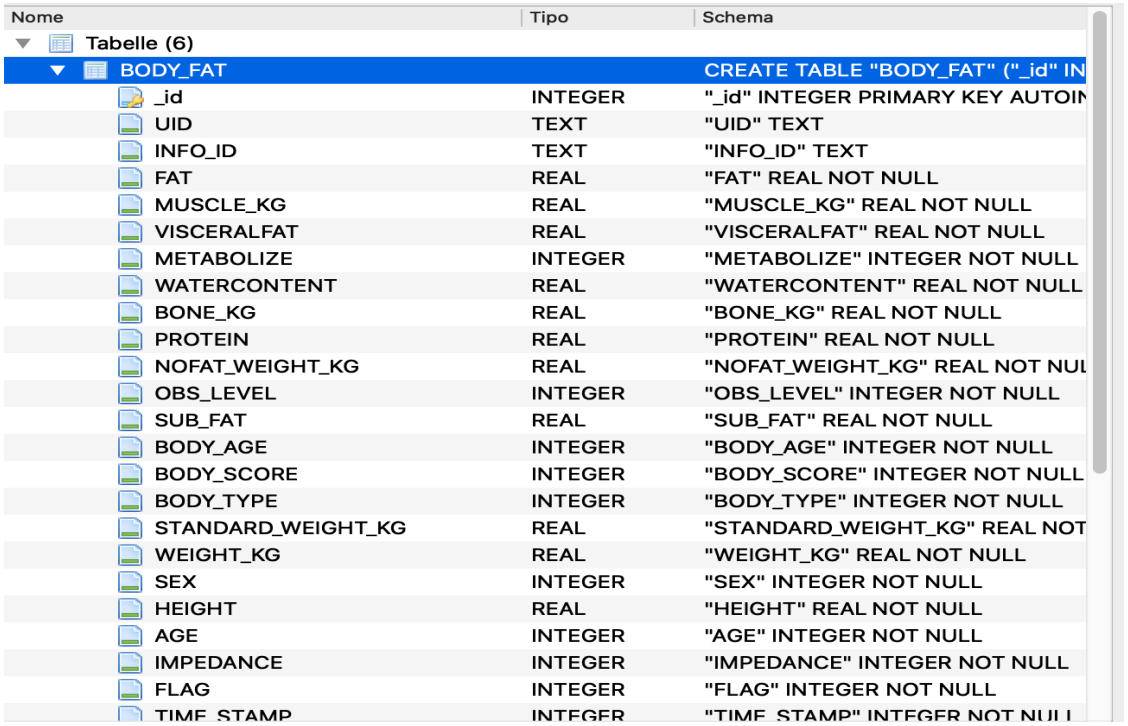

Figure 3.

Much of this data was useful in understanding what values sent were and how they were calculated.

Following the logs produced by the application when the data was received, this string was logged: "locked-weight --------- 82.3 impedanceInt = 2.3"

This log does not refer to the hexadecimal string explained above and allowed us to find the class that dealt with the interpretation of the data sent in the form of hexadecimal converted into bytes and then interpreted.

Below is a screen of the class that deals with interpretation.

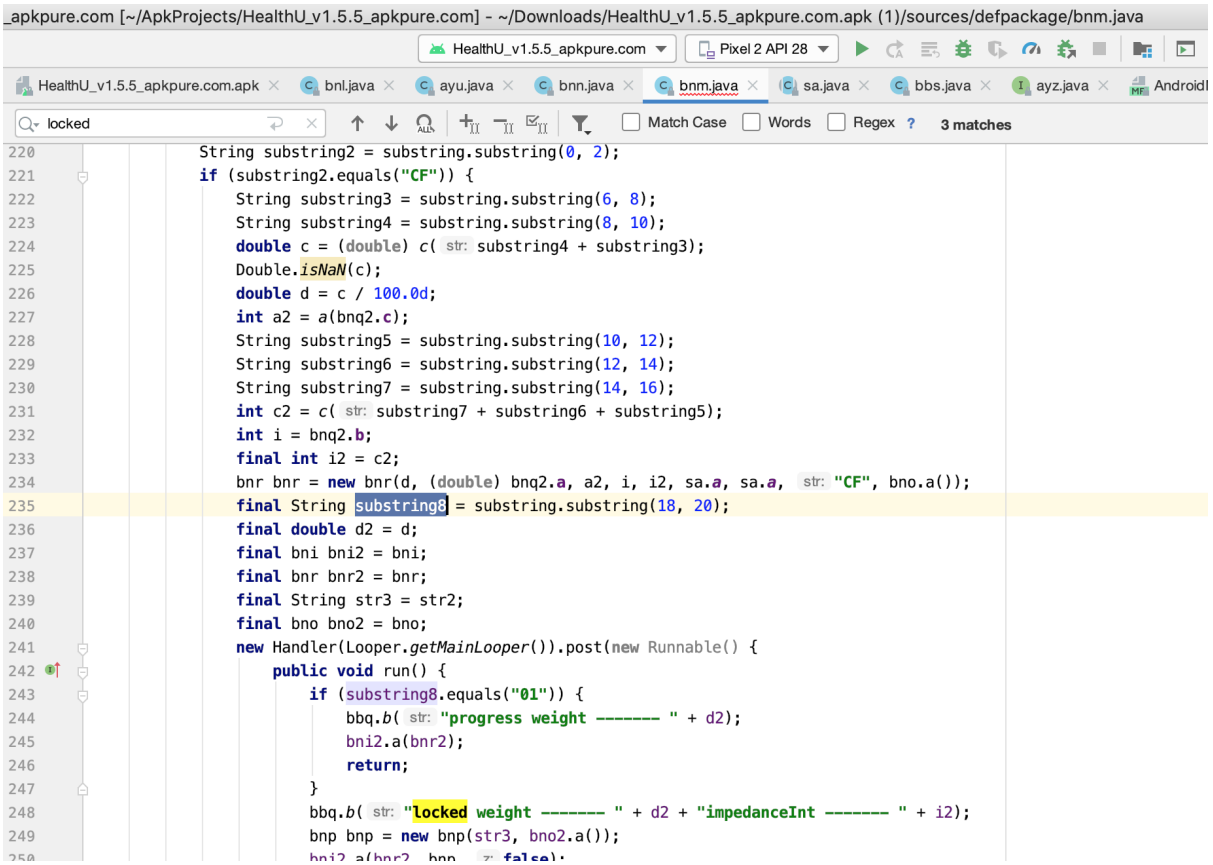

Figura 4.

The attention was focused on to the function c (string param) on line 224 which retrieves the bytes (inverting them) and performs the following operation.

```
/* access modifiers changed from: private */
public static int c(String str) {
    if (TextUtils.isEmpty(str)) {
        return 0;
    }
    return Integer.valueOf(str,  radix: 16).intValue();
}
```

Figure 5.

As well as the interpretation of a number in hexadecimal and corresponds to our weight. Then, starting from this and the data saved in the application database, all the rest are calculated. The last quartet of bits is used for the impedance or the parameter that allows you to divide the fat mass from the lean one.

Now we will be able to send any weight we want, without needing a scale.

The analysis of the body fat scale is documented below. This step allowed us to see the chip used, and the fact that some values were always 0 due to the partial use of this by the scale manufacturers.

It would also be possible to retrieve the firmware and understand it, since the documentation is available on the manufacturer's website.