Short Paper: The Security Analysis of a BLE Connected Health Device

Paul L. R. Olivier[1], Florent Galtier[1], Guillaume Auriol[1,2], Vincent Nicomette[1,2]
[1]*LAAS-CNRS,* [2]*Université de Toulouse, INSA*

***Abstract*—IoT devices represent a prime target for security threats. Unfortunately, effective security practices are not widespread as they should be, in particular concerning the health sector. This paper conducts a security analysis of a connected blood pressure monitor, revealing six significant vulnerabilities. We carry out four attack scenarios to highlight the dangers they pose to its users.**

***Index Terms*—Security, Health, Bluetooth Low Energy**

1. Introduction

Wireless communications offer numerous benefits, particularly in enhancing user interaction with IoT devices. The seamless connectivity provided by technologies like Bluetooth Low Energy (BLE) has become a key selling point for manufacturers, facilitating a wide array of applications. For instance, it enables the integration of companion applications on smartphones, allowing for additional features and functionalities. This integration also enables manufacturers to optimize the device hardware to offload demanding computing tasks to smartphones or remote servers. This, in turn, improves the overall efficiency of the device. Moreover, the wireless connection allows manufacturers to introduce new features post-deployment without the need to wait for every functionality to be fully implemented before shipping the devices. This approach significantly reduces the time-to-market, ensuring a responsive product development cycle.

Despite these advantages, the connection link and the information it carries must be properly secured. Implementing robust security measures is a non-trivial task, as demonstrated by the multitude of attacks on wireless protocols. The impact of such attacks can be devastating, posing serious threats to user privacy and system integrity.

This underlines the necessity of external audits on the final product to identify potential security vulnerabilities that may have been overlooked during the development phase. Various frameworks were developed with this goal in mind [2]. Unfortunately, we still observe today poor security practices in implementing wireless communication for IoT devices.

In this paper, we conduct a security analysis of a connected blood pressure monitor. It is a health device for people with medical conditions such as hypertension, hypotension, diabetes, and other cardiovascular conditions. It can also be beneficial for people interested in prevention or lifestyle monitoring, such as pregnant women, athletes, fitness enthusiasts and seniors. Overall, such a device targets a large portion of the population and its compromise may present a risk for the user's health.

Through our study, we demonstrate the security implications resulting from an insecure implementation of the Bluetooth Low Energy protocol. In particular, we show that the lack of authentication and integrity checks leads to device firmware tampering.

We summarize our contributions as follows:

- Reverse engineer a health device monitoring the blood pressure and conduct a security analysis of its components.
- Exploit the 6 vulnerabilities in its BLE implementation and firmware Over-The-Air update.

2. Background

Bluetooth Low Energy (BLE) is a lightweight variant of Bluetooth, dedicated to devices needing low energy consumption. Every BLE-based application using the *connected mode* is built on top of the *ATT* (Attribute Profile) and *GATT* (Generic Attribute Profile) layers [5]. Both layers define a client / server model, providing a generic solution to exchange data between devices. Specifically, an *ATT* server is a database of *attributes*. Each *attribute* is composed of an identifier, a type and a value. An *ATT* client is able to interact with this database using some requests. For example, a *Read Request* allows the client to read a given *attribute*, while a *Write Request* allows modifying the value of an *attribute*. The *GATT* level provides an additional layer of abstraction to define some *services* including *characteristics* and creates generic profiles for a given type of device.

3. Analysis

The blood pressure monitor performs three distinct measures: the diastole, the systole and the pulse. It also displays if irregularities were observed during the measurements. The device works by inflating a cuff around the wrist, measuring the pressure exerted by blood on the artery walls as the cuff deflates. This type of device is particularly useful for people with medical conditions related to the heart or interest in monitoring their lifestyle in a preventive way.

The device can be used alone, however the companion app helps keeping track of the records. The synchronization happens wirelessly when the application connects to the device via BLE. The user also has the possibility to enter the measurements manually in the companion app.

Our primary goal is to identify potential **vulnerabilities in the implementation of the wireless protocol**. It is essential to assess the device's ability to secure the data it handles, particularly when it is related to the health. Next, **analyzing the insight of the firmware** allows a better understanding in the vulnerabilities present on the device. This involves recognizing the instruction set used,

recovering the program load address, and identifying the memory layout including the I/O mapping. This can be achieved through intercepting the firmware update image for instance. Finally, the most impactful goal is to **gain arbitrary code execution** on the device. This not only serves to validate the identification and exploitation of vulnerabilities, but also demonstrates their security implications. Furthermore, it aids in comprehending and mitigating the associated risks.

Considering the goals mentioned earlier, we divided our analysis in four aspects. We started our analysis with a network reconnaissance and the companion app reverse. This allowed us to pinpoint the first flaws in the device related to its BLE implementation. Then, we looked at the hardware to get a better picture of the device's internals. With this knowledge, we focused on the firmware to get closer to code execution. While we haven't fully achieved this goal, the presented results provide encouraging prospects for future work.

Throughout the steps of our analysis and the discovery of novel insights, we systematically searched online for publicly available information.

3.1. Network

The device uses BLE to synchronize its records with the companion app. Using an Android Smartphone and a computer with the Mirage framework [2], we analyzed the structure of the GATT server on the device, along with its communications. The GATT server exposed several services and characteristics such as the firmware version, name and model. We identified three main manufacturer specific services for records synchronization, control data and firmware update.

We have examined the events that occur when the application connects to the device. For this purpose, we set up a Man-in-the-Middle (MitM) between the smartphone and the device. We identify two events triggered on the connection. First, the application sends the name of the current user, to which the device acknowledges good reception. The message has a total size of 20 bytes: two bytes flag, the entered name and the padding with null bytes. Second, the application queries the GATT server for information about the device status. In particular, to determine if an update is available for the device.

Then, if unsent records are stored, the device sends them in the form of `Handle Value Notifications` on a dedicated characteristic. The record messages contain a flag, the measured values (systole, diastole, pulse), a boolean indicating pulse irregularity, and the date when it was taken.

The MitM was possible because neither authentication nor encryption are employed during the communication. The application requests first to pair the device. However, we did not observe any standard pairing such as described by the BLE specification [5]. Instead, the application registers the device from its link-layer address. Moreover, it seems the device does not log any registration information, allowing anyone to connect and access saved records without restrictions. We found that the use of the term "pairing" in the application is confusing, as it could suggest this security mechanism is implemented in the communications, while it is not.

3.2. Companion Application

The blood pressure monitor comes with a companion app designed for recording measures and keep track of user health over time. For the analysis we focus on the latest version (2.2.2.5) dating from October 2023.

The APK contains two firmware images corresponding to the BLE chip. Further details about these images are provided in Section 3.4.

For GATT services and characteristics, we use the UUID to identify code sections related to the device communication. The majority of those sections are contained in the `BleService` class. We identified the code responsible for managing the firmware update, handling the device records and executing various control actions (e.g., modifying the username).

By manually inspecting the reversed code, we identified an out-of-bounds read during the parsing of health record messages. The main cause is the control exerted by the flags in the frame header over the size read, leading the application to read beyond its actual size. It may expose sensitive information or compromise the companion app's integrity.

3.3. Hardware

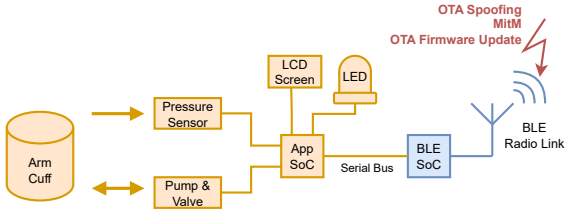

Figure 1. System Architecture Overview

The hardware is composed of a main PCB board which controls the LED, the LCD screen and the pump, and handles the wireless communication. In this analysis, we are more interested in the two main System on Chip (SoC). The application SoC is `SH79F6488P` and manufactured by Sino Wealth while the BLE SoC is a `CC2541 F256` from Texas Instruments. Both SoC implements different derived versions of the 8-bits 8051 microcontroller.

The board offers several test points. Using the datasheet of the application SoC, we managed to trace its JTAG pins. However, after analyzing its signals on an oscilloscope, we hypothesize that it may be disabled.

Next to the BLE SoC, we identified the serial bus used to communicate between both SoC. A simple custom protocol is used, and we managed to partially reverse it. Each packet is formed of a header, the payload and, at the end, the checksum of all bytes except the first, modulo 8 bits. The header is composed of a preamble (indicating if it is a request or a response), an operation code and the payload size. The operation may be initiated by any of the two SoC. We managed to identify the codes corresponding to initiating a wake-up call, responding with device information (including firmware revision and model number), sending the username, and transmitting the measures (systole, diastole, pulse) along the time.

3.4. Firmware

Because we couldn't get any access to the running program on the hardware, we focused the firmware analysis on the two binaries embedded in the Android companion app. The binaries follow the format described in the Texas Instruments BLE Over-the-Air Download (OAD) [4]. The images start with 2 bytes CRC-16, the image header and continue with the firmware content. The image header follows a public format [4] composed of 2 bytes CRC-shadow, 2 bytes user-defined image version number, the image length on 4 bytes, and the 4 bytes user-defined image identification. In summary, the OAD mechanism is composed of three images: the boot image manager (BIM) and two different images called "A" and "B". The BIM maps into the 8051 interrupt vectors to intercept all resets and jump to a valid image, B or A. The image A is cut in half where the upper segment maps to `0x800` and runs the proprietary OAD Target BLE Profile which is in charge to handle the firmware update. The image B maps to `0x4000` and implements the BLE stack.

We explored the firmware update procedure as a noteworthy entry point for code execution. On each connection, the companion app checks whether the firmware requires an update, according to the number of the running version. The outline of the procedure is described by Texas Instruments [4]. The companion app sends the image header to the device, and upon acceptance, initiates the transfer of the firmware image. On completion, the device validates the image's CRC-16 against the one sent at the beginning and finally resets the connection. Despite being explained in the documentation [4], no encryption or signature are used for the OAD mechanism.

At the time of writing, we have not extracted or located the firmware used by the application SoC.

The firmware analysis is an ongoing task. We aim to focus on better understanding its internal work to be able to patch it and get code execution on the BLE SoC (Section 4).

3.5. Vulnerabilities

During the security analysis, we identify 6 vulnerabilities in the different components of the device.

Communication Protocol: the BLE implementation on the device suffers from vulnerabilities on the pairing, authentication and communication protocols.

- The *pairing* mechanism is *not* implemented at all, and therefore fails to establish trust between the two devices. It is responsible for providing authentication, enabling key distribution and negotiating the shared secrets.
- *No authentication* is used: both the companion app and the device do not authenticate to each other. This enables spoofing attacks for both sides.
- The *communication is not encrypted*. Data exchanged during communication is in clear text and can easily be obtained.
- The *communication does not present any integrity protection*. Data exchanged during the communication can be manipulated.

Firmware Update

- *No encryption* is used during the OAD mechanism despite being available.
- *No signature* is implemented for protecting the firmware integrity.

4. Attack Scenario

We describe 4 over-the-air attacks to demonstrate the severity of the vulnerabilities presented in Section 3.5.

The system model is similar to what was described in Section 3. It is composed of the blood pressure monitor and the companion app communicating over BLE. We do not assume the victim has the monitor device around the wrist during the attacks, because it records the measures to download them later to the companion app.

The attacker only knows public information advertised by the blood pressure monitor over BLE such as the BLE address. For the OTA spoofing and MitM, we assume the attacker does not have physical access to the target devices, hence cannot tamper the devices' operating system and firmware. For the OTA firmware update, the attacker needs a way to put the device in the update mode (long press on/off button).

The attacker has the following four goals:

1) Spoofing the blood pressure monitor to the companion app.
2) Spoofing the app to the blood pressure monitor.
3) Establishing a MitM between the blood pressure monitor and the app
4) Pushing a custom firmware on the BLE SoC

4.1. Attacks

OTA Spoofing. These attacks are straightforward as no security mechanism is implemented.

To impersonate the companion app, the attacker simply connects to the device using BLE. It is then possible to read the different GATT services, and the records made by the device.

In the same way, the attacker can advertise fake blood pressure measures knowing the BLE address, and forge fake records to poison the history in the companion app.

MitM. Since it is trivial to impersonate both the device and the companion app, a MitM attack is no more difficult. Multiple approaches can be considered, but the easiest is BTLEJuice [1].

The idea consists of first connecting with the blood pressure monitor to make it stop advertising, and then using a second BLE device spoofing its address to wait for a connection request from the companion app. Once the MitM established, the attacker can easily modify the transmitted data, such as the device records to tamper the user's health history.

OTA BLE Firmware Update. To update the firmware, the attacker needs a way to put the device in the firmware update mode. A solution for that is to long press the on/off button on the device. From that, the attacker can connect to the device and push the firmware update with the desired modifications (e.g., backdoor). Neither encryption nor signing is present in the firmware images.

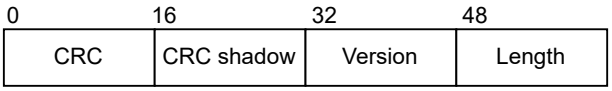

Figure 2. Firmware Update Image Header Structure

4.2. Attacks' Impact

The severity of these attacks are high for several reasons. First, these attacks are cheap and low efforts. No security mechanisms are used to protect the link between the blood pressure monitor and the companion app. Moreover, the firmware update attack potentially allows full execution on the BLE SoC. To a greater extent, the device interacts with humans and report their health status for storing records. These records may later be used to decide whether the users should take specific medical treatment. Therefore, one could use these attacks to falsify the records, or even change the device's behavior with a malicious update. This may lead to erroneous diagnosis and dangerous medical decisions for the user. Finally, the absence of cryptography can lead to breaches of the user's privacy, especially concerning their medical condition.

5. Experiments

The experiments were carried out using the Braun wrist blood pressure monitor iCheck 7 BPW4500WE running the firmware 1.0.15, an Android smartphone with the companion app version 2.2.2.5, and a computer with two Bluetooth interfaces.

The Mirage framework [2] includes modules to launch standard wireless attacks such as MitM and device spoofing. In particular, we used the `ble_master`, `ble_slave`, and `ble_mitm` modules.

Application Spoofing. Any device supporting BLE can connect to the device and run commands, as no authentication and encryption are implemented in the device. For instance, we were able to connect and read the records with a BLE-debugging application such as *nRF Connect For Mobile*.

Device Spoofing. First, we cloned the GATT server of the device to replicate all the services and characteristics that the companion app may need or request. Then, Mirage is used to generate a new instance of the GATT server to which the companion app will connect.

MITM. We spoofed both roles to perform the BTLEJuice [1] MitM attack. In addition, we implemented a scenario to alter the packets containing health records, thereby tampering with the companion app history.

Firmware Update. We extended the application spoofing attack with a Mirage scenario for faking the firmware version and uploading our modified variant following the OAD process. To confirm our success, we modified strings in the firmware accessible via the GATT server, particularly the device information such as the version and model. We also had to adjust the leading CRC-16 to align with our changes before uploading the firmware. Indeed, as represented on Figure 2, each firmware file begins with a CRC, computed over the whole file [4].

6. Countermeasures

The presented attacks rely on the lack of basic security mechanisms such as authentication, integrity and encryption. The main countermeasure against the spoofing and MitM starts with the implementation of a secure pairing procedure[1]. Additionally, other protections described by the Bluetooth Core Specification [5] Volume 3 Part H and Volume 6 Part E, should be applied. The firmware update process should follow the BLE OAD guidelines [4] to use firmware image encryption and signing.

7. Conclusion & Future Work

We performed a security analysis of a connected blood pressure monitor and showed that unfortunately, it suffers from several serious vulnerabilities. Some of them arise from the fact the device does not implement the security measures recommended by the BLE specification [5]. Sadly, this is not an isolated case in the more general world of IoT devices. Indeed, as shown in several works [6], [7], a huge majority of recent devices do not implement secure pairing, and by extension encryption, or contain vulnerabilities. This issue, especially in health-related devices, is a big concern for the users' safety. Moreover, it has been shown through the last years that security has not been a priority for manufacturers, and despite evolution in the standards, most devices remain unsecure to this day. A way to solve this problem would be to make it compulsory to implement appropriate security mechanisms, for instance by the mean of a certification that all connected objects should undergo.

To further our analysis on this device, we plan to continue the analysis of the BLE firmware to reach the third goal on code execution. For instance, inserting a backdoor or a function which modify the records before sending them by BLE to the companion app. We also have to recover and analyze the application firmware responsible for displaying the measures on the LCD screen and commanding the pressure exercised on the wrist by the pump. Another possibility is to look for deeper vulnerabilities in the firmware by using dynamic analysis techniques. For example, the serial bus could be fuzzed, or the firmware could be rehosted [3] in an emulator allowing full control over its state.

Responsible Disclosure. In adherence to responsible disclosure practices, the vulnerabilities identified during the course of this research were reported to the respective vendor on two separate occasions, in January and March 2024. Despite these efforts to engage in constructive dialogue towards mitigating potential security risks, no acknowledgments were received from the vendor.

8. Acknowledgment

We want to thank Sebastien Di Mercurio for his precious help in the hardware analysis. This work has been partially supported by the French National Research Agency under the France 2030 labels (Superviz ANR-22-PECY-0008 and REV ANR-22-PECY-0009).

1. It is recommended to avoid the "Just-Works" version which relies on a fixed key.

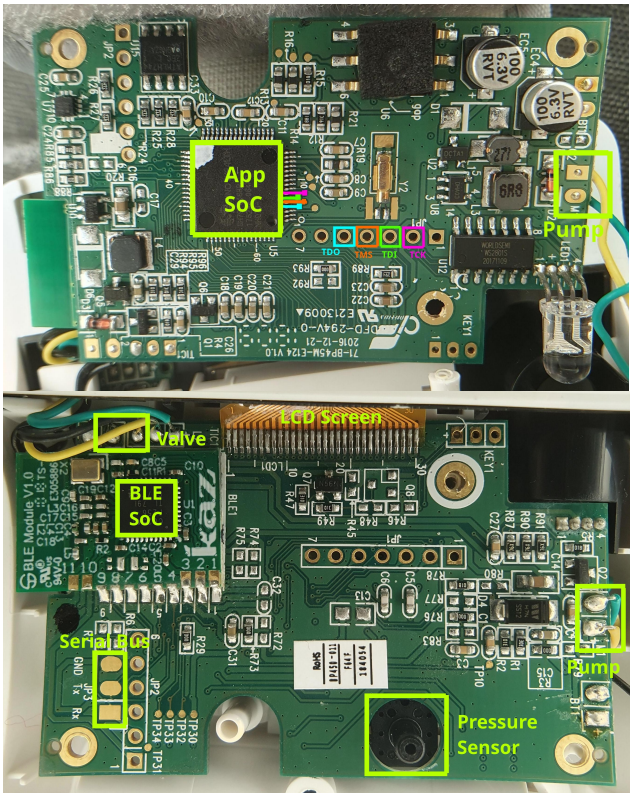

Figure 3. Both sides of the PCB

References

[1] Damien Cauquil. Btlejuice: The bluetooth Smart MitM Framework, 2016.

[2] Romain Cayre, Vincent Nicomette, Guillaume Auriol, Eric Alata, Mohamed Kaaniche, and Geraldine Marconato. Mirage: towards a metasploit-like framework for iot. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019.

[3] Andrew Fasano, Tiemoko Ballo, Marius Muench, Tim Leek, Alexander Bulekov, Brendan Dolan-Gavitt, Manuel Egele, Aurélien Francillon, Long Lu, Nick Gregory, et al. Sok: Enabling security analyses of embedded systems via rehosting. In *Proceedings of the 2021 ACM Asia conference on computer and communications security*, 2021.

[4] Texas Instruments. BLE Developer's Guide for Over-the-Air Download for CC254x Version 1.2.

[5] Bluetooth SIG. Bluetooth Core Specification v5.4.

[6] Pallavi Sivakumaran, Chaoshun Zuo, Zhiqiang Lin, and Jorge Blasco. Uncovering vulnerabilities of bluetooth low energy iot from companion mobile apps with ble-guuide. In *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, 2023.

[7] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019.