AN ACTIVE MAN-IN-THE-MIDDLE ATTACK ON BLUETOOTH SMART DEVICES

TAL MELAMED
Tech Leader, AppSec Labs, Israel.

ABSTRACT
In the last years, the Internet of Things (IoT) has become integral part of our lives and its influence is expected to exponentially increase in the next years. For several reasons, however, the development of IoT has not gone hand in hand with an adequate reinforcement and consolidation of our security and privacy, despite the serious impact that IoT vulnerabilities may have on our digital and physical security. Bluetooth Low Energy (BLE), also known as Bluetooth Smart, is the most popular protocol for interfacing smart devices, wearables, and medical equipment. This contribution surveys the key security issues in the BLE protocol and discusses a possible architecture for BLE Man-in-the-Middle (MitM) attacks together with the related necessary equipment. In addition, after introducing some of the available tools for hacking BLE, a case-study based on their use was presented, which describes a MitM attack between a Bluetooth smart device and its designated mobile app. The case-study well exemplifies how easily, given the required proximity to the target, a possible hacker can control the data and, in some instances, even the mobile device itself, when connecting it to a BLE device.
Keywords: BLE security; Bluetooth Low Energy; Bluetooth Smart; IoT security

1 INTRODUCTION

Bluetooth technology was first introduced as 'Wibree' by Nokia in 2006. With the adoption of the Bluetooth Core Specification Version 4.0 in 2010, Bluetooth Low Energy (BLE) became the main Bluetooth standard, known under the names of 'BLE' and 'Bluetooth Smart' [1]. BLE was designed as a smaller, cheaper, and power-efficient technology aimed at supporting the increasing growth in the use of IoT devices, which is excepted to reach the number of 30.7 billion devices in 2020 [2]. Due to its low cost and ease of implementation, about one third of the existing smart devices use BLE. Bluetooth is broadly used with IoT devices/applications and smart products in the healthcare, fitness, beacons [3], security, and home-entertainment industries, and nowadays, we encounter BLE in almost every aspect of our lives (e.g. in wearables, sensors, medical devices, security products, etc.). Natively supported by the most popular Mobile operation systems (including iOS, Android, Windows Phone and BlackBerry), as well as by macOS, Linux, and Windows 8/10, by 2018, Bluetooth Smart is expected to be supported out-of-the-box by more than 90% of Bluetooth-enabled smartphones [4].

By underling some of the security vulnerabilities of the BLE, this paper aims to raise concern about the crucial need to devise new strategies for developing adequate IoT security protocols that will be able to keep up with the increasing diffusion of IoT devises and with the growth in the use of BLE. In my contribution, I will first examine some of the fundamental security issues in the BLE protocol mainly relying on Mike Ryan's survey presented in 2013 [5]. I will then discuss in details a possible architecture and the related equipment required to perform MitM attacks under BLE. After presenting two available tools that use this architecture, I will offer a case-study which exemplifies on a practical level the most important issues discussed in the paper. Based on the above-mentioned tools that use the MitM architecture, the chosen case-study describes a practical MitM attack attempted on a mobile application, which communicates with a BLE-based sport bracelet.

ISSN: 2041-9031 (paper format), ISSN: 2041-904X (online), http://www.witpress.com/journals
DOI: 10.2495/SAFE-V8-N2-200-211

2 KEY SECURITY ISSUES IN THE BLE PROTOCOL

BLE protocol has proved insecure against eavesdropping [5]. As intercepting and decrypting the BLE communication is relatively simple, potential hackers can easily perform Man-in-the-Middle (MitM) attacks, i.e. attacks in which the attacker manages to place himself in the middle of the communication between the smart device and the designated mobile app.

In the following paragraphs, I briefly survey some key concepts of Bluetooth Smart together with their impact on the global security level of the protocol. Further information about BLE encryption, pairing, key-generation, signing data and other security-related issues are described in the BLE Security specification [6].

2.1 Security manager

The Security manager is part of the Core System Architecture (Fig. 1) and is in charge of the security capabilities of the protocol: pairing integrity, authentication and encryption. BLE responds to MitM attacks with AES-CCM encryption. While AES encryption is considered secure, the way the encryption keys are exchanged is insecure and introduces some severe security vulnerabilities, which can be exploited by potential hackers to decrypt the data. In fact, to exchange the encryption key, Bluetooth Smart uses a custom key-exchange protocol, in which the devices exchange a temporary key (TK) and use it to create a short-term key (STK), which will be then used to encrypt the connection. The level of security of such process depends on the pairing method used to exchange the TK, for which see the following section.

2.2 Pairing

The pairing process is performed by two BLE – or BLE-supported – devices to establish keys in order to encrypt the communication. The keys can be also used to encrypt the communication in future reconnections, to verify received signed-data, or to perform random

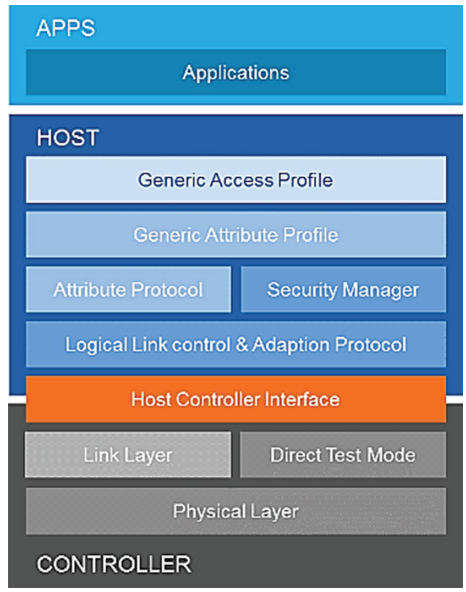

Figure 1: BLE Core System Architecture.

address resolution. BLE uses a three-phase pairing process. (i.) The initiating device sends a pairing request and, consequently, the devices exchange pairing capabilities over an insecure channel. (ii.) The devices exchange the TK and verify the use of the same TK, which is then used to generate a STK. Rather than TK and STK, newer devices (BLE 4.2) support LE Secure Connections, which use a long-term key (LTK) exchanged using Elliptic Curve Diffie-Hellman (ECDH) public key cryptography [7], a significantly stronger methodology than the standard BLE key exchange protocol. (iii.) The created key is then distributed between the devices over a secured connection and can be used to encrypt the data.

The TK is determined according to the pairing method. In version 4.0 and 4.1 of the Core Specification, BLE uses the Secure Simple Pairing model (known as LE Legacy), in which devices can choose to operate with one of the following methods, according to their specific I/O capabilities: (i.) Just Works™, (ii.) Passkey entry, and (iii.) Out-of-Bound (OOB), outlined below (2.2.1, 2.2.2, 2.2.3). It ought to be emphasized that mobile apps cannot control the pairing method, which is instead determined on the OS level.

2.2.1 Just Works™

In Just Works™, the TK is set to 0, thus offering no protection at all against MitM attacks. Despite its high level of insecurity, this is the most commonly used method today among the BLE devices. Furthermore, this is the only possible method that can be employed with devices lacking a display.

2.2.2 Passkey

In Passkey, the TK is a six-digit number combination, which the user actively inserts into one of the devices, thus manually exchanging it between the devices. For instance, if one of the devices generates a random six-digit number and shows it on its LCD display, the user will need to read the number and enter it into the other device using a keypad. Since it was proven that also this method can be bypassed [5], it is therefore recommended to follow either the OOB or Numeric Comparison pairing methods (the latter supported only by BLE 4.2).

2.2.3 Out-of-Band

In this method, the TK is exchanged using a different medium, such as Near Field Communication (NFC). The security level of OOB is determined by the security capabilities of the technology used to exchange the key. If the OOB channel is protected from MitM attacks, the BLE connection shall be also regarded as sufficiently protected. In other words, the pairing process will result immune to eavesdropping, if the OOB channel is immune. OOB pairing is the most secure method available in the BLE 4.0/4.1 protocol. Unfortunately, this method is still very uncommon: having tested dozens of smart devices, I have seen no implementation of OOB yet.

2.3 GATT

The Generic Attribute Profile (GATT) is used by BLE devices to communicate with each other using a standardized data schema (Fig. 2). The GATT describes the device roles and general behaviors employing hierarchy of services, characteristics, and attributes. According to Bluetooth Specifications, “a service is composed of characteristics or references to other services. A characteristic consists of a type (represented by a UUID), a value, a set of properties indicating the operations the characteristic supports and a set of permissions relating to security. It may also include one or more descriptors metadata or configuration flags relating to the

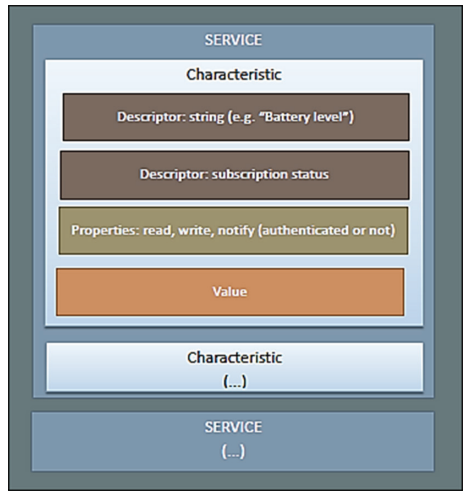

Figure 2: GATT Profile Hierarchy.

owning characteristic" [8]. Provided the proximity to a smart device, its GATT schema and generic data can be read by any BLE-supported app.

Each service contains properties that can be used by the connected app (certain properties require authentication). There are three supported events: read, write, and notify. While read and write requests transmit a single value, a notify event allows the mobile app (central) to subscribe for a certain characteristic and receive data asynchronously from the smart device (peripheral). All the communications are performed using integer-value handlers associated with the requested characteristics.

2.4 BLE connection flow

In a typical BLE connection flow (Fig. 3), a smart device continuously broadcasts an advertisement, in order to allow a mobile app to identify and connect to it. Once the mobile app is connected to the smart device, the BLE device advertises its GATT schema, providing the connected app with the necessary information.

This flow enables potential hackers to discover the device services, clone them and re-publish them, in order to lure the victim into their 'trap' (see section 3). Figure 4 illustrates in three steps a generic GATT-reader app (also referred to as BLE Scanners) reading the available services and the generic data published by the smart device (in this specific case, a 'Power Watch+' smart device).

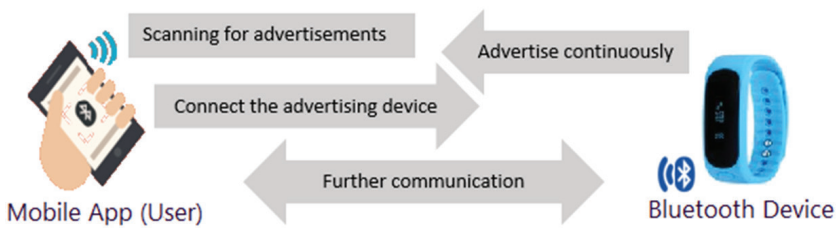

Figure 3: Typical BLE Connection Flow.

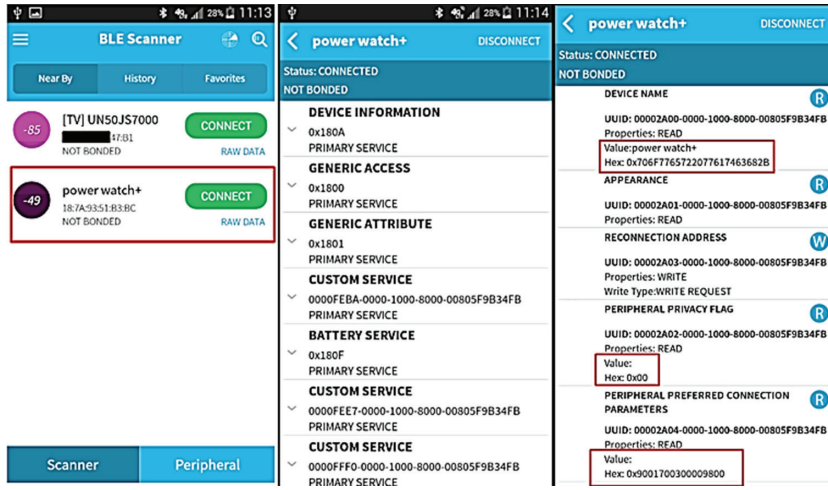

Figure 4: BLE Scanner app accessing data published by the 'Power Watch+' smart device.

3 PRACTICAL MITM ATTACK FOR BLE

A common MitM attack, in which the attacker 'sits' between two parties connecting to both ends, does not apply for Bluetooth Smart. As shown in Fig. 5, in the common MitM architecture, when one party (the client) sends the data, the MitM attacker acts as the receiving end (the server) and, vice versa, when the server sends the response back, the attacker acts as the client. This does not apply to BLE, due to a typical Bluetooth technology limitation for which the device can be connected to only one party and, therefore, cannot connect to both ends simultaneously.

3.1 BLE MitM architecture

In MitM attacks for BLE, a common MitM cannot connect simultaneously to both ends (smart device; mobile app). Therefore, a BLE MitM needs to make use of two BLE

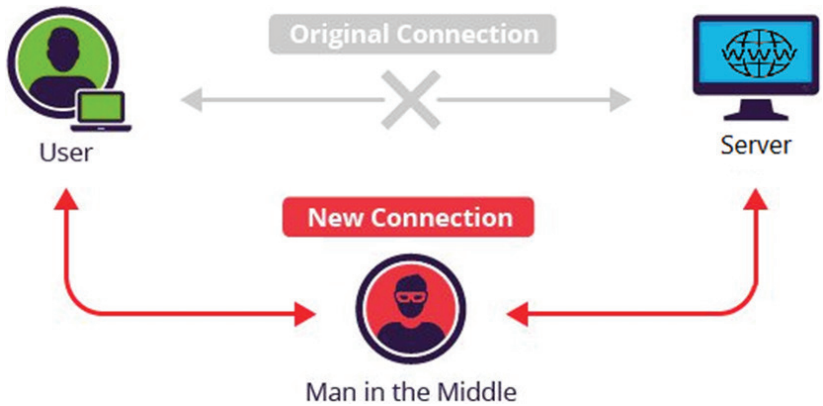

Figure 5: Common MitM architecture.

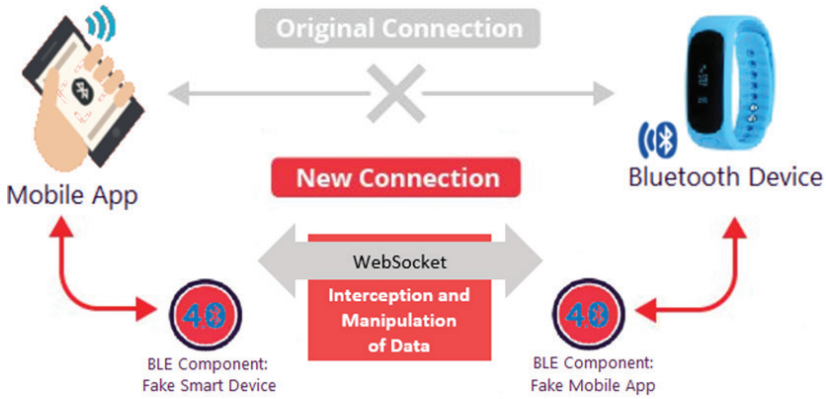

Figure 6: Practical man-in-the-middle architecture for BLE.

components capable of acting together: one connects to the mobile app acting as the smart device, while the other connects to the smart device acting as the mobile app. Once the smart device is connected to one of the components (fake mobile app), it can no longer publish itself or enable any other application to connect to it. At this point, it is required a different protocol to enable the two components – the fake mobile app and the fake smart device – to exchange the data that each of them received from, respectively, the smart device and the mobile app. A useful protocol to complete the MitM attack is WebSocket [9], which enables a two-way communication between two hosts, one running code in a controlled environment and the other opting-in to communications from that code. As shown in Fig. 6, by using the discussed architecture, the MitM attack can succeed, thus allowing the hacker to intercept and even modify the data sent over the BLE channel.

3.2 Necessary equipment

To support the described MitM architecture for BLE, the attacker requires two BLE 4.0 components. The most popular component used for this purpose is the CSR 8510-based USB dongle [10], which supports BLE (Fig. 7).

By connecting both dongles to the computer – or, better, to two different machines – the attacker can easily perform the MitM attack according to the architecture outlined above (Section 3.1; Fig. 6). Specifically, one of the dongles (acting as the fake smart device) is used to connect to the mobile app, while the other (acting as the fake app) to the smart device. In order to intercept and manipulate the data transferred between the mobile app and the smart device, the two dongles communicate with each other using a WebSocket code.

Figure 7: Typical CSR 4.0 adapters.

4 AVAILABLE TOOLS BASED ON THE BLE MITM ARCHITECTURE

Although there are other possible MitM architectures for BLE, the one described in this paper (Section 3) has the advantage that has been already tested and proved to be stable on both Linux and Windows operations systems. In what follows, I introduce two available tools on which the above-mentioned architecture is implemented: GATTacker (Section 4.1) and BtleJuice (Section 4.2). Both tools produce excellent results enabling a MitM attack for BLE and, among other many advantages, support two important features: (i.) Hooking, which enables hackers to write a code that can be executed upon a BLE event (read, write or notify), and Replay attack, which enables hackers to re-transmit an event that was previously sent [11]. Both (and other) tools can be accessed from my GitHub page (available at https://github.com/nu11p0inter). The case-study described in Section 5 uses both the above-mentioned tools and features to hack the designated mobile app.

4.1 GATTacker

GATTacker is a Node.js package for BLE MitM developed by Slawomir Jasek (Securing) and presented at the Black Hat USA conference in 2016 [12]. After installing the prerequisite libraries (i.e. noble, a NodeJS BLE central module; bleno, a Node.js module for implementing BLE peripherals), GATTacker can scan and copy BLE advertisements and services which can then be used to run a cloned (fake) version of the smart device (peripheral). Once the fake smart device is provided with the IP address of the machine to which the dongle acting as fake mobile app (central) is connected, the WebSocket connection is established and functions as an active MitM, which enables the hacker to intercept and manipulate the transmitted data.

4.2 BtleJuice

BtleJuice is a framework to perform MitM attacks on Bluetooth Smart devices, which was developed by Damien Cauquil (Digital Security) and first presented at the DefCon 24 conference in 2016 [13]. BtleJuice includes two components, which require to be run on independent machines in order to operate two Bluetooth 4.0+ adapters simultaneously. Also in this case, the two components communicate with each other over WebSocket protocol. BtleJuice includes a web interface and – among other useful features – presents Replay GATT operations (Replay attack) and On-the-fly data modification capabilities (Hooking).

5 CASE-STUDY: HACKING A BLUETOOTH SMART MOBILE APP

In what follows, I attempt to demonstrate on an empirical base the notions surveyed so far, by describing in details and step-by-step a practical MitM attack which I performed on a BLE device for research purposes. The BLE device chosen for the MitM attack and described in the case-study is a smart sport bracelet (Dax-Hub SW-28 Smart Bracelet) designed to monitor sport activities and health-related data (e.g. calories, distance, etc.). The data detected by the smart bracelet are synchronized with the designated app, called PowerSensor, which can be downloaded from the App Store and Goole Play. The case-study shows that, giving the proximity to the device, an attacker can easily intercept and manipulate the data transmitted between the mobile app and the BLE device, operation that can result in the modification of the data and in a partial mobile takeover from remote (limited to the BLE protocol range, which is ~100m).

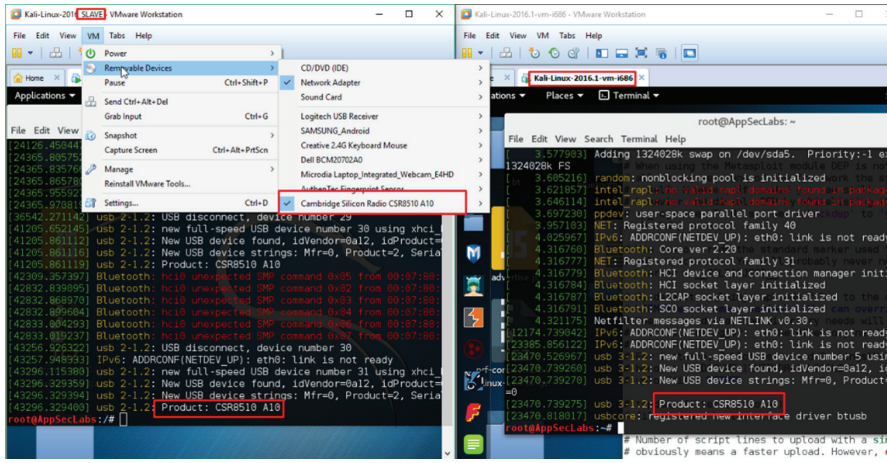

Figure 8: Each CSR dongle is connected to a different VM.

5.1 Preparing the environment

Before carrying out the planned MitM attack, I downloaded a Kali Linux virtual image (available at https://kali.org) and installed all the required libraries and tools (see Section 4). The next step was that of inserting the CSR 4.0 dongle (Section 3.2) and connecting it to the Virtual Machine (VM). Since most tools require two different machines to run the BLE components, I decided to clone the VM, making sure that each dongle was connected to a different VM, as shown in Fig. 8.

5.2 Data Interception via MitM

After the preliminary procedure described above (Section 5.1), I began the actual MitM attack by running the GATTacker central (ws-slave) on one VM and by running the GATTacker peripheral configured to the ws-slave's IP address. In order to find the advertisements published

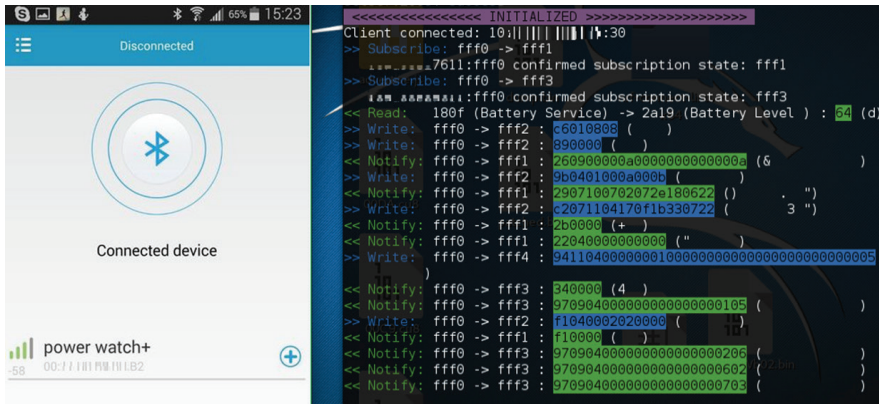

Figure 9: MitM attack based on the use of GATTacker.

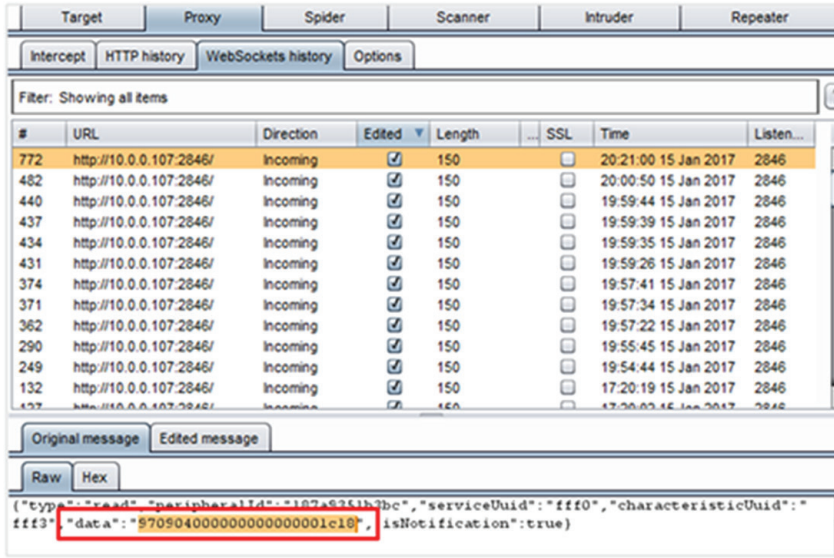

Figure 10: The BLE communication data transmitted via Burp.

by all the nearby BLE devices, it is necessary to run the scan command. However, to carry out the MitM attack, it is mandatory to detect and copy also the services of the targeted smart device. Therefore, to achieve this result, I ran the scan command specifying the MAC address of the BLE sport bracelet. This operation enables GATTacker to copy and store the discovered advertisements and services on the machine, where they will be identified by their specific MAC address. Once I obtained the advertisements and services of the BLE bracelet, I used them to simulate a fake device. In fact, by advertising the name of the targeted smart device (in this case, 'power watch+'), GATTacker can lure the victim to connect to the fake device and,

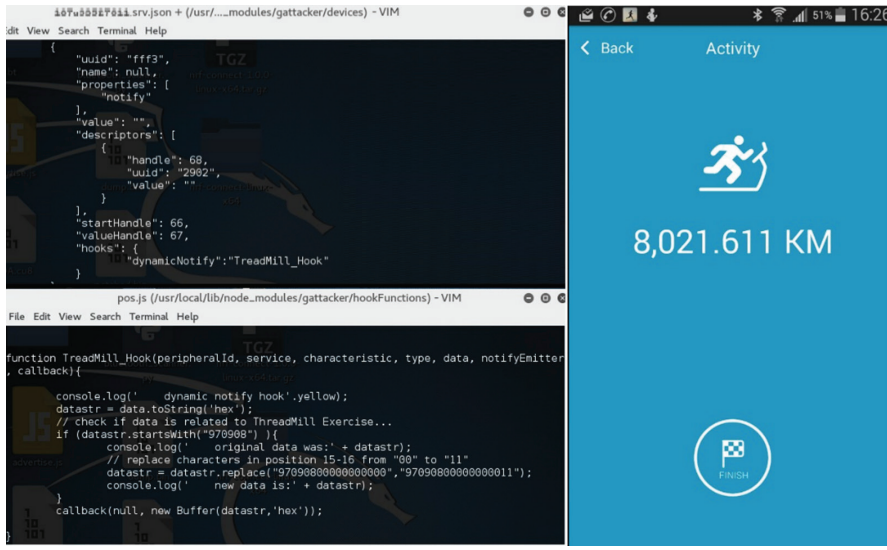

Figure 11: Hooking notification event using GATTacker.

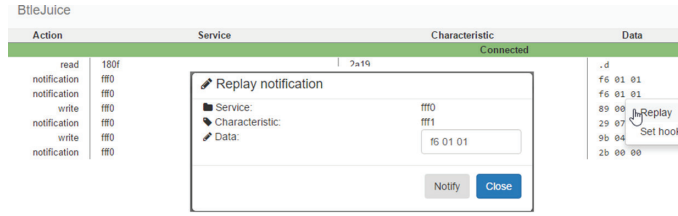

Figure 12: Replay Attack feature in BtleJuice.

when the mobile app scans for a nearby device (in the case-study PowerSensor mobile app) it locates the fake advertisement. At this point, the mobile app was successfully connected to the bracelet through MitM, thus allowing me to view all the communication data, which in this specific case were represented by long hexadecimal values, as shown in Fig. 9. In general, it ought to be noted that, if the victim has previously paired the mobile app with the smart device, the mobile app will automatically connect to the fake device, provided that the latter presents the MAC address of the real smart device using MAC spoofing [14].

Considering that setting a proxy in the middle of the WebSocket connection of the BLE components can provide additional capabilities (e.g. on-the-fly data modifications, brute-force, replay attacks, etc.), I set the peripheral component to go through Burp (i.e. a proxy tool that supports WebSocket communication, available at https://portswigger.net/burp/) rather than directly to the central component. This step allowed me to intercept and modify the data using Burp Intercept feature, as shown in Fig. 10.

5.3 Data manipulation via notification hooking

Unfortunately, Burp does not support hooking, providing only the on-the-fly ability, which can modify only one intercepted request at a time. Therefore, during the simulated MitM attack for the case-study, I had to make use also of the hooking feature of GATTacker which allows to execute custom code upon a received event (i.e. read, write, notify). The analysis of the data exchanged between the smart bracelet and the PowerSensor app during the simulated attack has shown that when a sport exercise activity is launched in the mobile app (e.g. jumping rope, treadmill, etc.), the app subscribes to a notification event – identified as fff3 (see Fig. 9) – in which the bracelet monitors the exercise activity and sends the data to the PowerSensor app in real-time. The GATTacker hooking feature enabled me to add a hook to the notification event identifier and, consequently, to execute code capable of automatically modifying the data on every request sent from the bracelet to the mobile app (see, Fig. 11). As a result of the hook, the values sent from the bracelet to the PowerSensor – which at the time was used for a treadmill activity – were modified to an exaggerated kilometrage. Obviously, modifying exercise data poses no actual threat to the targeted victim. Yet, while I deliberately chose to simulate in the case-study a harmless attack as proof of capabilities, in a real-world scenario hackers can target different life-depending devices, such as diabetic/ blood-sugar monitors, as well as security devices, such as electronic locks and alarms.

5.4 Mobile camera hijacking via replay attack

Besides the data monitoring features mentioned above, the smart bracelet chosen for the case-study supports the use of the mobile camera and music system directly from the wristband. Therefore,

using BtleJuice, I was able to identify such requests, after invoking the functionality from the bracelet. As mentioned in Section 4, BtleJuice includes Replay Attack capabilities, which allowed me to replay a request previously transmitted between the bracelet and the mobile app. After initializing the MitM attack using BtleJuice, the web interface showed a notification on the fff0 service that the data f60101 was used by the smart bracelet to take a picture on the victim's mobile device. Using the Replay function, I was then able to take pictures with the mobile camera directly from my own computer at any given time (Fig. 12). With the same method, I was able also to play music directly on the victim's mobile (for obvious reasons, I cannot supply concrete proof in the paper that I actually succeeded to take pictures and play music as a result of the MitM attack).

6 CONCLUSIVE NOTES

The present paper confirms that BLE is insecure and vulnerable against passive eavesdropping, as demonstrated already in 2013 [5]. In particular, I have shown (Sections 3 and 4) that a passive eavesdropping can easily become an active MitM attack that enables a possible hacker not only to listen to the communication, but also to intercept and manipulate the data. Furthermore, with the case study presented in Section 5, I have demonstrated that, in some instances, by performing a MitM attack, hackers can even control from remote the mobile device used to communicate with the Bluetooth smart device.

It ought to be noted that with the release of the Bluetooth Core Specification version 4.2, BLE Security has been significantly improved by the new LE Secure Connections pairing model, which includes the Elliptical Curve Diffie-Hellman (ECDH) algorithm for key exchange and, in addition to the three standard pairing models (outlined in §2.2), the numeric comparison method. Similarly, additional security and privacy-related features are added in the Bluetooth Core Specification v5 [8], recently released by Bluetooth.

Despite these important improvements in BLE Security, it is vital to be aware and fully understand the limitations of the smart devices that we use rather than blindly relying on them. In particular, it is of essential importance to implement security protections on the application-side to guarantee users a sufficient protection against malicious activity. Ensuring full protection against complex hacking attacks may prove difficult and perhaps utopic at the present stage. Yet, a valuable step in this direction can be achieved by implementing additional and updated security controls, such as data encryption and signature, strong authentication and authorization mechanisms, and other security best practices.

ACKNOWLEDGEMENTS

The study presented in this paper is part of a broader research on IoT security carried out at AppSec Labs, a leading Israeli application security consulting company specialized in mobile and IoT security (https://appsec-labs.com/iot-security). A preliminary version of this study was originally presented at OWASP Israel January 2017 [15].

REFERENCES

[1] Bluetooth, S.I.G, SIG introduces bluetooth low energy wireless technology, the next generation of Bluetooth wireless technology, *press release*, 2009.

[2] Columbus, L., Roundup of internet of things forecasts and market estimates. *Forbes*, 2015.

[3] Dudhane, N.A. & Pitambare, S.T., Location based and contextual services using bluetooth beacons: new way to enhance customer experience. *Lecture Notes on Information Theory,* **3**(1), 2015.

[4] Janiak, S., *Three ways bluetooth smart technology enables innovation for the internet of things*, available at https://blog.bluetooth.com/three-ways-bluetooth-smart-technology-enables-innovation-for-the-internet-of-things (accessed 30 April 2017).
[5] Ryan, M., Bluetooth: with low energy comes low security. *WOOT*, 2013, available at https://www.usenix.org/system/files/conference/woot13/woot13-ryan.pdf (accessed 30 April 2017).
[6] Bluetooth S.I.G., *Proprietary information security, bluetooth low energy*, available at https://www.bluetooth.com/~/media/files/specification/bluetooth-low-energy-security.ashx (accessed 30 April 2017).
[7] Lauter, K., The advantages of elliptic curve cryptography for wireless security. *IEEE Wireless Communications,* **11**(1), pp. 62–67, 2004. https://doi.org/10.1109/mwc.2004.1269719
[8] Bluetooth Specifications, available at https://www.bluetooth.com/specifications (accessed 30 April 2017).
[9] Fette, I. & Melnikov, A., The WebSocket Protocol. *Internet Engineering Task Force*, RFC 6455, 2011, available at https://tools.ietf.org/html/rfc6455 (accessed 30 April 2017).
[10] CSR8510 Specification, available at http://www.csr.com/sites/default/files/csr8510.pdf (accessed 30 April 2017).
[11] Syverson, P., A taxonomy of replay attacks, computer security foundations workshop VII, CSFW 7, *IEEE*, pp. 187–191, 1994.
[12] Jasek, S., *GATTacking Bluetooth Smart devices*, BlackHat USA, 2016, available at http://gattack.io/whitepaper.pdf (accessed 30 April 2017).
[13] Cauquil D., *BtleJuice: the Bluetooth Smart MitM Framework, DEF CON 24 Internet of Things Village*, 2016, available at https://www.youtube.com/watch?v=lcn07TclnS0 (accessed 30 April 2017).
[14] Cardenas, E.D., Mac Spoofing–an introduction. *GIAC Security Essentials Certification*, 2013.
[15] Melamed, T., R U aBLE? BLE Application Hacking. *OWASP*, 2017, available at http://sl.owasp.org/melamed17 (accessed 30 April 2017).