Security Vulnerabilities in Bluetooth Low Energy

Rajib Dey and Debashri Roy
Department of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32826
{rajib, debashri}@eecs.ucf.edu

***Abstract*—Our research goal is to find security vulnerabilities in Bluetooth Low Energy (BLE). We forwarded towards this project by going through the actual Bluetooth specifications and state of the art research papers. First we represented our insights over probable vulnerabilities in BLE, in this paper, from our extensive research. We have presented an overview of collective flaws in security measures in modern day's BLE enabled devices. Later we demonstrated some possible countermeasures which could be considered to overcome those flaws. We also analyzed the trade-offs of implementing those possible countermeasure for low cost BLE devices. This term project report contains an educational survey of different software and hardware security vulnerabilities of BLE devices.**

I. Introduction

Bluetooth Low Energy (BLE) is now called Bluetooth Smart, and it is a part of Bluetooth 4 and was introduced in 2010. It was meant to be designed for power-efficient and significantly smaller and cheaper devices used in Internet of Things (IoT) among many others. These devices use BLE as an underlaying technology. It supports low cost and easy implementation, which led BLE to be widely used among IoT devices and applications. It could be deployed in wearables, sensors, lightbulbs, medical devices, and many other smart-products. 48 billion IoT devices are expected to be in market and in our daily lives by 2021. Bluetooth Low Energy is predicted to be in nearly one-third of those devices.

Now a days, maximum laptops and smartphones have BLE built in it. Bluetooth Smart Ready indicates a dual-mode device. Typically a laptop or smartphone, whose hardware is compatible with both Classic and Low Energy (LE) Bluetooth peripherals are termed as "Smart-Ready" device. Bluetooth Smart indicates an LE-only device, typically a battery-operated sensor, which requires either a SMART Ready or another SMART device in order to function. iPhone 4S was the first ever commercial device to include BLE.

A. Bluetooth Stack

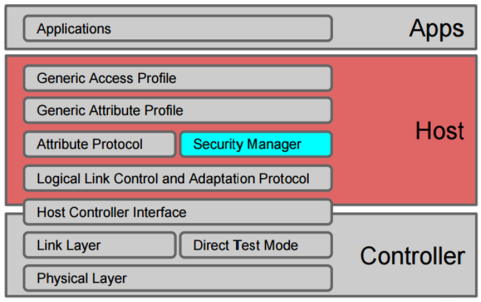

Fig. 1. Protocol Stack in BLE

In Fig. 1, we show the protocol stack of BLE. The PHY and Link layers are unique to Bluetooth Low Energy. The upper three layers have some similarities with classic Bluetooth. A very brief description of each layer is provided below:

1) ***Applications*** *:* Applications are built on top of all layers. It interacts with only the host layer. Depending on the application environment different API would be needed to control the application layer.

2) ***Generic Access Profile (GAP)****:* It enables a generic Profile to enable communication between BLE device. Which defines basic requirements. It ties all the layers together and helps maintaining all the layers in BLE. It manages the security aspects, including security modes and procedures and responsible for connection functionality like the access modes and procedures of the device, i.e., device discovery, link establishment, link termination, initiation of security features, and device configuration. It defines the roles the device would perform in a BLE environment (i.e: broadcaster,

observer, peripheral and central). Depending on the implementation, BLE device can operate under one role at a time. To support multi-role, underlying controllers have to support those roles.

3) ***Generic Attribute Profile (GATT)****:* It is built on top of the Attribute Protocol (ATT), which uses GATT data to define the way that two BLE devices send and receive standard messages. (GATT is not used in Classic Bluetooth). GATT defines how data is formatted, packaged and sent. GATT is used by the application for data communication between two connected devices. Data is passed and stored in the form of characteristics, which are stored in memory on the BLE device. It describes in detail how attributes (data) are transferred after the devices have a dedicated connection. A characteristic (eg. blood pressure) contains a single value ("attribute") Which Can be read, written to or subscribed for notifications. From a GATT standpoint, when two devices are connected they are each in one of the two roles: the GATT server, and the GATT client. Collection of characteristics designed to fulfill a particular task is a GATT service. For example, heart rate monitor service might include body sensor location, heart rate control point characteristics.

4) ***Attribute Protocol (ATT)****:* Protocol to access attributes is called the Attribute Protocol (ATT). It accesses the attributes via reading and writing attributes. It performs basic operations: request, response, command, indication, and confirmation. ATT defines the communication between two devices playing the roles of server and client, respectively, on top of a dedicated L2CAP channel. Read and write requested, and initiated by client; while notification and indication are initiated by server. Client access the server's attributes by sending requests, which trigger response messages from the server. Server can send two types of unsolicited messages that contain (1) attributes: notifications, which are unconfirmed; (2) indications: which require the client to send a confirmation. Client may also send commands to server to write attribute values. Client might request/response, and indication/confirmation transactions follow a stop-and-wait scheme. This allows the application to know what data packets have been successfully transmitted and can be used to design extremely reliable applications.

5) ***Security Manager****:* This layer is responsible for implementing all the security features. This implements a number of cryptographic functions and responsible for data encryption and decryption. It follows 3 phase process on connection (1) pairing feature exchange, (2) short term key generation, and (3) transport specific key distribution. Depending on the version of BLE and implementation these can vary greatly. Details about what happens in this layer is described broadly later on in this paper.

6) ***Logical Link Control and Adaptation Layer Protocol (L2CAP)****:* This layer Transfers data between the Upper layers and lower layer protocol stack. It takes multiple protocols from the upper layers and encapsulates them into the standard BLE packet format (and vice versa). It does services like fragmentation and recombination by taking large packets from the upper layers and breaks them up into chunks that fit into the 27 bytes maximum payload size of the BLE packets on the transmit side, and vice versa. It also receives multiple packets that have been fragmented and recombines them into a single large packet that can then be sent to the upper Layer.

7) ***The Link Layer(LL)****:* Provides first level of control and data structure over raw radio operations, bit stream transmission and reception. It defines BLE State machine, state transitions, Data and advertisement Packets format, Connections, packet timings, retransmissions. LL has mostly 3 major operations: advertising, scanning and connection establishment.

8) ***Physical Layer****:* It operates in 2.4 GHz ISM band using 1Mbps GFSK modulation scheme, which is larger modulation index than Bluetooth BR (which means better range). It has 40 Channels with 2 MHz spacing between them, where

$$f = 2402 + 2kMHz$$

Fig. 2 shows the 3 advertisement channels in *green* and 37 data channels in *blue*.

B. General Operating Procedure

The advertiser, actually the BLE device, sends out advertisements. It provides a way for devices to broadcast their presence so that connection can be established between software and devices. It broadcasts data like the list of supported services,

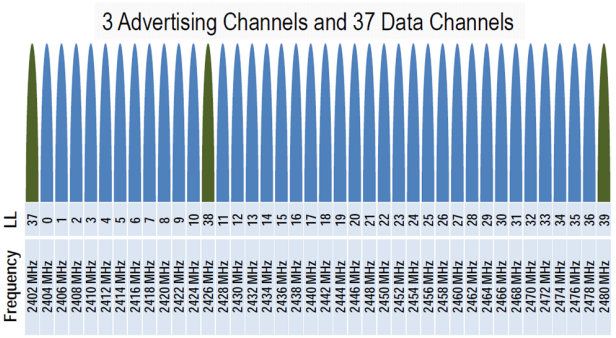

Fig. 2. Data and Advertisement channels in BLE

device name, Input/Output (I/O) capabilities, TX power level etc. The app looks for a particular advertisement and get paired with the peripheral device based on some trivial security features. Once a connection is made with the master (here, the app), informs BLE device of hopping sequence and when to wake up. All subsequent transactions are performed in the 37 data channels. Transactions can be encrypted. Both devices can go into deep sleep between transactions.

C. Security in BLE

Five main features for security in BLE are:

- Pairing: Process for generating shared keys (STK).
- Bonding: Subsequent process of pairing where they store STK to form trust.
- Encryption: For message confidentiality.
- Authentication: Two device identify if they have the same keys.
- Message Integrity: To avoid attacker to forge data.

After the devices find each other through advertisement they would go through some security procedures. Pairing would be the first one, where Short Term Key (STK) would be generated. Then this STK is used to generate a few more security keys. Most important of all of those keys is the Long Term Key (LTK). After the STK is generated the communication is encrypted using the STK for the first time. After that the LTK is generated to encrypt all future communications.

D. Pairing

Pairing is done using keys to encrypt the communication. The keys can be used to encrypt future re-connections. They can also verify signed data, or perform random address resolution. There are 3

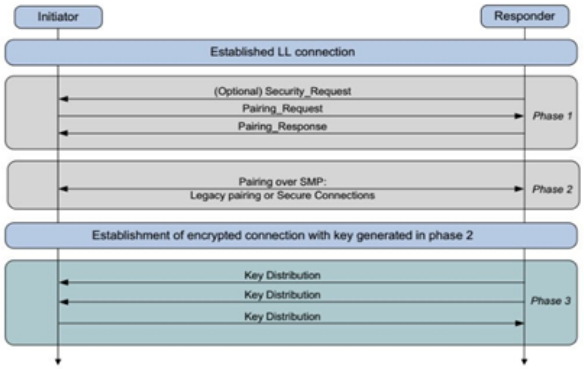

Fig. 3. Overall Pairing Procedure

phases for pairing (shown in Fig. 3):

1) Pairing Feature Exchange: It depends on I/O Capabilities. Based on that method for phase 2 is chosen.
2) STK Generation. It is actually called legacy pairing.
3) Transport Specific Key Distribution: LTK used for LL encryption and authentication

If we just focus on pairing, it happens in 3 phases. Overall, first phase determines the TK, seconds phase determines the STK and the third phase determines the LTK.

E. Pairing in BLE

How to determine the Temporary Key (TK)?

- **Just Works**^{TM}: Legacy, most common Devices without display cannot implement other Its actually a key of zero, thats why it just works.
- **6-digit PIN**: In case the device has a display, it gives 1 million options (BF-able).
- **Out of band (OOB)**: Here the secret key is not shared over the 2.4 GHz band. Rather it makes use of other mediums (e.g. NFC). Once secret keys are exchanged, encrypted data is transmitted though the channel. But it is not common and barely used in most of the BLE devices in market.

The first phase of pairing, can happen in one of these three method to determine the Temporary key.

It mainly depends upon the I/O capabilities of both devices. If one of the device does not have any I/O capabilities, then they would have to choose

the "Just Works" method. Which is where we think Bluetooth Low-Energy is most vulnerable. Because in this case the TK is always zero. In the last 2 phases STK and LTK is generated.

II. Related Works

We did literature survey on several published papers from 2016 to state of the art. We amazingly found that most of the work tried to address the security issues by providing some light-weighted cryptographic algorithms or proposing new application layer protocol, however none implemented in modern off-the-shelf BLE devices. We categorized our study in 2 parts, researches targeted to find the vulnerabilities, and researches proposed some solutions to those vulnerabilities.

A. Vulnerability Oriented

- In [4], the authors introduced with a special type of attack for BLE devices, called Denial of Sleep attack. BLE devices work with two main status: active and sleep. Devices enters into sleep mode when it is inactive for a certain amount of time, which is generally vender and device type specific. However, this special type of attack is build on leveraging this concept. The authors have presented thorough analysis on how devastating this attack could be to BLE sensing network, by giving an example implementation. Denial of Sleep attack poises threat to BLE device's lifespan by several orders of magnitude, resulting the network largely unusable. Ultimately, the authors have modified different BLE protocols to allow a malicious actor to rapidly drain battery lifetime of a targeted BLE sensing node. The authors have given corresponding simulation results, including power analysis, in the paper.
- Now a day, there are many wearable devices which works in synergy with different BLE enabled Internet of Things devices. Lotfy et al. have presented an empirical analysis on data exchange mechanism in wide range of major commercial wearable products in [2]. The authors have mainly focused on how the products live up to the security constraints for users' privacy. Three different types of Bluetooth LE pairing strategies were discussed here at a packet and protocol level. In the paper, the authors have examined three commercial wearables, where each use the different pairing processes. First, Jawbone UP (Jawbone), uses just works. Second, Pebble Steel smart watch (Pebble), uses numerical comparison. Lastly, the Fitbit Charge HR (Fitbit) uses passkey entry. The authors also stated specific methodology to test each of the devices and their associated pairing process, during their experimentation.The experimental results have shown that presumably secure pairing strategies have convincing security vulnerabilities that affect all of the devices considered.
- In [6], the authors have presented the concept of selective jammer for BLE devices. The proposed jammer can be programmed for BLE advertising in such a manner that only specific beacons, e.g. those with a particular MAC address, can be jammed. This exhibits a potential security loophole in BLE architecture. The proposed jammer leverages the technique that BLE beacons are sent on different advertisement channels. The jammer scans all advertisement channels in order to get estimate which channels are actually being used by the beacon sources, by its discovery component. Later those channels are attacked by the jammer to be interfered. The mentioned jammer transmits short jamming signal only on the attacked channel being used by the beacon frame, making it hard to detect, and energy efficient. The authors have also prototypically implemented the proposed jammer using low-cost, off-the-shelf and small-sized hardware. They were successful in showing the feasibility and efficiency of those jammers through experimental analysis.
- MAC addresses are unique to any electronic devices containing a NIC. In [5], the authors have show how bluetooth MAC address of BLE devices could be used to increase the security of pairing processes. As mentioned earlier, pairing process of any BLE devices poises maximum vulnerability. The authors claimed that assignment of random MAC address in most of modern generation BLE enabled devices can make those susceptible to higher

security risk. However, the idea behind random MAC address for BLE enabled devices were to provide better security by barring the easiness of tracking those devices. The authors bolstered their claim by a study using a smart vehicle from a major Taiwanese brand. The authors have also provided some generalized security requirements for the example scenario.

B. Tool Oriented

- We have found few published researches which actually proposed some tools and solution for achieve better security for BLE enabled devices. One such approach was presented in [1], where the author presented a set of cryptographic protocols to minimize the BLE vulnerabilities. The author identified that security mechanisms provided in BLE specification sometime fails to provide suitable protection during accessing or modifying sensitive sensor parameters, or exchanging messages. He proposed Bluetooth Low Energy Application Layer Security Add-on (BALSA). BALSA specified set of cryptographic protocols, a BLE service and a guideline about usage architecture targeting to provide a suitable level of security. He also defined and analyzed these components and demonstrated the feasibility and benefits of BALSA by presenting an experiment with 4 BLE enabled smartphones.
- However, in [3], the authors have designed an extension during networking of BLE enabled devices. The authors have first extended a calculation formula for *Authentication* to accurately assess the vulnerability of BLE enabled IoT devices. The *Authentication* variable was first used by the National Infrastructure Advisory Council to determine conventional base score equations of the Common Vulnerability Scoring System (CVSS) v2 [7]. Later they have proved the superiority of their proposed extension over the current CVSS v2 base score by an experiment on network based shopping cart IoT system.

III. System Architecture

A. Pairing Method

As mentioned earlier, Bluetooth 4.0 specification [8] defines three pairing methods for BLE: Just Works, Passkey Entry, and Out-of-Band (OOB). Pairing is a actually a key exchange procedure between the sensor and the mobile device, Those keys are later used to encrypt or encode information during communication. Basically Short Term Key (STK) and Long Term Key (LTK) are generated during this paring procedure, as mentioned in Section I-D.

LTK is an AES key that is used to encrypt and authenticate messages at the L2CAP layer of BLE network architecture. LTK is generated by the Sensor and sent wrapped to the mobile device using Short Term Key (STK), which is actually derived from Temporary Key (TK). TK is directly generated from the association process during paring.As already discussed in Section I-D, TK is all zeros, or a six decimal digit string, or some exchanged information using other channel when architecture is Just Works, or Passkey Entry, or OOB, respectively. is exchanged It is clear that Just Works pairing method provides no security at all. From the other two alternatives, Passkey Entry has two disadvantages: (i) it ensures very limited security due to the low entropy of passkey field, which is 6 digits only. In fact it has been proved that the LTK can be cracked within seconds [9]. Second, the users need to intervene the automatic handling of pairing process, which makes it vulnerable to attacker inception. However, OOB can be used to share a full 128 bit long key, for STK and LTK generation, between the mobile device and the sensor. But, so far there are no APIs in recent versions of iOS or Android to use this pairing mechanism. If consumer mobile devices needs to be modified with customized micro controller to be able to use OOB. But all these efforts become less appealing with the market's demand of cheap BLE enabled devices only. Therefore, after reading a wide varieties of research papers on BLE security and probable measures, we found the protocol enhancement proposed in [1], to be quite interesting and feasible with respect to both cost and security measures. BALSA provides an additional security mechanism to be appended with

existing BLE implementations irrespective of the pairing method in place.

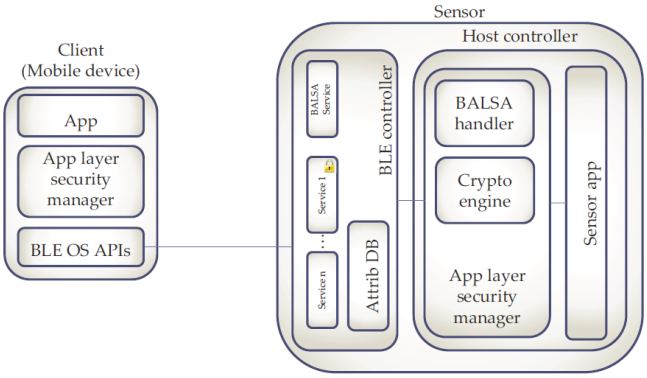

Fig. 4. System Architecture Proposed in BALSA [1]

B. Proposed Architecture in [1]

The proposed system architecture is presented in Fig. 4. The principal modules of the architecture are *BLE controller*, *Host controller* and *client* device. The implementation protocols are done by communication between *BLASA session* and *BALSA service*. Client directly interacts with BLE controller for regular operations. *BLE controller* communicates with local *Attribute Database* or *Host controller* to response Clients requests.However, when clients asks for sensitive informations, then it establish a *BALSA Session* by interacting with the *BALSA Service*. The encrypted characteristics values are transported by Attribute Protocol (ATT) operations after the the *BALSA Session* is established. The ATT operations on the *BALSA Service* can also be used to transport the corresponding MACs.Application layer security manager handles all the BALSA Sessions, generation/verification of MACs and encryption/decryption of characteristic values. It also helps to communicate between *OS BLE APIs* and the mobile device *App* in *Client* side. Whereas, it handles commands received from the *BLE controller* and *Crypto engine* by generating appropriate responses to the *Client* via the *BLE controller*, or to the *Sensor* app. The Application layer security manager provides the much needed security by imposing different security protocols proposed in the paper. The Sensor app is responsible to provide a frontend to the users to select the security configuration they want. However, Client side, each information is retrieved from the Backend. BALSA provides more security constraints by imposing the rule to create new services including the *BALSA Service* as a secondary service to show its usage is mandated. The advantage is that no configuration beyond the service definitions will be able to do any drastic changes to the new services for the user sensitive information. The overall approach is aimed to provide a robust security layout over existing BLE specification.

IV. Probable Security Flaws

After some detailed research, we found some vulnerabilities in BLE Security, they are enlisted below. We have found out these flaws, how they can be exploited, and how to avoid them, during our extensive research.

- Use of “Just Works” to generate TK for devices with limited I/O Capabilities.
- Not using Out of band (OOB) medium to generate the TK.
- Not implementing any kind of encryption.
- Poor implementation of existing security protocols.
- Sending sensitive information through plaintext without encryption.
- Using Password based authentication which is just waiting to be brute-forced, if most common passwords like 123456 is not being used that is. But unfortunately most BLE devices uses such easy passwords and they do not use any kind of encryption on them either.
- BLE beacons generally uses static passwords. As these devices interact with mobile phones they become extremely vulnerable due to this.
- Lack of OTA capabilities: Most if not all BLE capable peripheral devices lack OTA (Over-the-Air Programming) capabilities. So, if any vulnerabilities were found and it is fairly easy to fix it, these devices will remain vulnerable.
- According to the BLE Specifications, against passive or active eavesdropping “Just Works” and Passkey Entry do not provide any kind of protection. This is mainly because Secure Simple Pairing (SSP) uses Elliptic Curve Diffie-Hellman (ECDH) for the key exchange and BLE does not.

- Not enough protection against Man-in-the-middle (MITM) attacks. Which is prone to data leakage, data manipulation by the active interceptor.
- *Poor implementation of Random Number Generator:* As our computational capabilities are limited, the process to randomize a number is not that random and can be reverse engineered if it is not implemented well. High randomization can be achieved by processing and sampling a source of entropy (such as temperature of the device). But many peripheral devices are cooled to low temperatures which leads to a lower degree of randomization.
- To save energy, BLE devices optimize the advertisement interval. They try to send advertisement packets in large intervals. If the attacker sends out his/her own advertisement packet at higher frequency then that can lead to jamming of the BLE device. This can also lead to Denial of Service (DoS) attacks.

V. Possible Countermeasures

Throughout the course of our research, we have found some possible steps that can make BLE more secure. They are mentioned below:

- *Preventing advertisement abuse:* Need to use encryption and authentication while advertising broadcast values. Values in the advertisement should not remain the same, values should change which can only be decoded from the predefined devices. Services should not solely be performed using advertisement only (like heart-rate monitor broadcasting hear rates in advertisement packets). All services should go through proper encryption and authentication process. Using other advertisement channels other than *channel 37*, increasing advertisement interval can also help us avoid any kind of attack using the advertisement.
- While devices with limited capabilities does not have any other option than to use "Just works" as the method to choose the TK, we need to implement OOB as the method to share the initial keys whenever possible. As implementing OOB can increase the cost, we can at-least have strong dynamic passwords. Exchanging Keys with Elliptic Curve Diffie Hellman Key Exchange (ECDH) would be one of the best way to secure all the keys. But it is hardly implemented in low cost and low powered BLE devices. One other method could be, having a physical button on the device that needs to be pressed to authenticate ownership of the device, without which the pairing process should not proceed. In this way, the attacker would not be able initiate a pairing process. Which prevent the attacker to discard old keys from the device and capturing new keys that he just forced the device to generate.
- To prevent any replay attack, sequence number should be used. In this case, if the attacker is trying to replay back old commands to the device then the sequence number would not match and the command would be rejected. Each message should have a sequence number which would be obtained from the previous sequence number by incrementing a certain pre-defined value Δ to it.
  $NewSeq.Number = Prev.Seq.Number + \Delta$
- Always implement existing encryption protocol in a robust manner. Encrypt all useful data being sent to prevent passive monitoring data leaks.
- Implement OTA capabilities into each device. This includes leaving enough room in the device memory so that codes can be updated easily if there is any vulnerabilities found. Without an OTA capable device we would have no other option than to just through away the device to avoid being hacked.

VI. Conclusions

We have presented an educational survey about possible security vulnerabilities in BLE devices. We surprisingly found that Bluetooth Low Energy, being mostly different from standard Bluetooth architecture, still lags behind, in case of providing a robust security mechanism, as a trade-off for cost. However those vulnerabilities could be fatal and poise severe threat to the users privacy. This comprehensive study will help the reader to understand how crucially the BLE enabled devices need to give extra attention towards implementing security features, and how modern market is mostly ignorant about that. We have also presented our insights

about some possible future development to render security for BLE enabled devices in affordable cost.

REFERENCES

[1] D. A. Ortiz-Yepes, *BALSA: Bluetooth Low Energy Application Layer Security Add-on.* 2015 International Workshop on Secure Internet of Things (SIoT), Vienna, 2015, pp. 15-24.

[2] K. Lotfy and M. L. Hale, *Assessing Pairing and Data Exchange Mechanism Security in the Wearable Internet of Things.* 2016 IEEE International Conference on Mobile Services (MS), San Francisco, CA, 2016, pp. 25-32.

[3] Y. Qu and P. Chan, *Assessing Vulnerabilities in Bluetooth Low Energy (BLE) Wireless Network Based IoT Systems.* 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), New York, NY, 2016, pp. 42-48.

[4] J. Uher, R. G. Mennecke and B. S. Farroha, *Denial of Sleep attacks in Bluetooth Low Energy wireless sensor networks.* MILCOM 2016 - 2016 IEEE Military Communications Conference, Baltimore, MD, 2016, pp. 1231-1236.

[5] S. C. Cha, C. Y. Dai and J. F. Chen, *Is there a tradeoff between privacy and security in BLE-based IoT applications: Using a smart vehicle of a major Taiwanese brand as example.* 2016 IEEE 5th Global Conference on Consumer Electronics, Kyoto, 2016, pp. 1-4.

[6] S. Brauer, A. Zubow, S. Zehl, M. Roshandel and S. Mashhadi-Sohi, *On practical selective jamming of Bluetooth Low Energy advertising.* 2016 IEEE Conference on Standards for Communications and Networking (CSCN), Berlin, 2016, pp. 1-6.

[7] Common Vulnerability Scoring System (CVSS) v2, https://www.first.org/cvss/.

[8] Bluetooth 4.0 Specifications, https://www.bluetooth.com/specifications.

[9] B. Schneier, *The Internet-of-Things is wildly insecure and often unpatchable.* Wired Magazine., January 2014. http://www.wired.com/opinion/2014/01/theres-no-good-wayto-patch-the-internet-of-things-and-thats-a-huge-problem/

[10] Common BLE security vulnerabilities in IoT and countermeasures. https://www.simform.com/ble-iot-security-vulnerability-countermeasures/