BlueDoor: Breaking the Secure Information Flow via BLE Vulnerability

Jiliang Wang[1], Feng Hu[1], Ye Zhou[1], Yunhao Liu[1,3], Hanyi Zhang[2], Zhe Liu[2]
[1]Tsinghua University, [2]Nanjing University of Aeronautics and Astronautics, [3]Michigan State University
jiliangwang@tsinghua.edu.cn,{hu-f17,ye-zhou16}@mails.tsinghua.edu.cn,
zhe.liu@nuaa.edu.cn,kyouichi@nuaa.edu.cn,yunhaoliu@gmail.com

ABSTRACT

Today's smart devices like fitness tracker, smartwatch, etc., often employ Bluetooth Low Energy (BLE) for data transmission. Such devices thus become our information portal, e.g., SMS message and notifications are delivered to those devices through BLE. In this study, we present BlueDoor, which can obtain unauthorized information from smart devices via BLE vulnerability. We thoroughly examine the BLE protocol, and leverage its intrinsic properties designed for low-cost embedded and wearable devices to bypass the encryption and authentication in BLE. By mimicking a low capacity device to downgrade the process of encryption key negotiation and authentication, BlueDoor can enforce a new key with the peripheral BLE device and pass the authentication without user participation. As a result, BlueDoor can extract BLE packets as well as read/write stored data on BLE devices. We show that BlueDoor works well on the fundamental design tradeoff of using BLE on diverse embedded and wearable devices, and thus can be generalized to various BLE devices. We implement the BlueDoor design and examine its performance on 15 COTS BLE enabled smart devices, including fitness trackers, smartwatch, smart bulb, etc. The results show that BlueDoor can break the information flow and obtain different types of information (e.g., SMS message, notifications) delivered to BLE devices. In addition to privacy threats, this further means traditional operations such as using SMS for verification in widely adopted authentication, are insecure.

CCS CONCEPTS

• **Networks** → **Network protocols**; • **Security and privacy** → **Mobile and wireless security**.

KEYWORDS

BLE, Bluetooth Low Energy, Security

ACM Reference Format:
Jiliang Wang, Feng Hu, Ye Zhou, Yunhao Liu, Hanyi Zhang and Zhe Liu. 2020. BlueDoor: Breaking the Secure Information Flow via BLE Vulnerability. In *The 18th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '20), June 15–19, 2020, Toronto, ON, Canada.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3386901.3389025

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MobiSys '20, June 15–19, 2020, Toronto, ON, Canada
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7954-0/20/06…$15.00
https://doi.org/10.1145/3386901.3389025

1 INTRODUCTION

Bluetooth Low Energy (BLE) is a widely used wireless personal area network technology targeting at communication for low power and smart devices. BLE provides considerably reduced power consumption and cost [1] while maintaining a data rate of about 100 kbps and a communication range of up to 250 meters [2]. Due to substantial reduction in energy consumption, BLE has become the primary choice of communication technology for personal wearable and healthcare devices [3] such as fitness trackers, smartwatches, smart home devices like smart locks, smart bulbs, etc. According to [4] [5], the shipment of BLE fitness trackers reached 115.4 million in 2017. The number is still increasing and is forecasted to reach 200 million by 2022.

BLE based smart devices are often connected to the user's mobile phone and act as the information portals. For example, SMS messages and different types of notifications, traditionally viewed as secure, are forwarded from mobile phones to connected smart devices through BLE so that user can read SMS message and other types of information on the fitness tracker. The application of BLE enabled smart, and wearable devices significantly increase convenience in our daily life.

Generally, BLE uses encryption to avoid data eavesdropping, and authentication to verify the identity of the device. Two BLE devices (i.e. a central device and a peripheral device), agree on a common encryption key and authentication information at the first time of connection, e.g., by out-of-band user input and confirmation on both devices. The devices, however, do not exchange the encryption key in the subsequent communication process. As a result, we usually assume that an eavesdropping device or a man-in-the-middle (MITM) attack device has no way of sniffing the encryption key and thus cannot decrypt packets to obtain information in the connection. Thus, traditional eavesdropping or MITM based methods[6] cannot work for BLE devices with both encryption and authentication.

To examine the above security assumptions, we conduct an in-depth analysis and measurement of the BLE protocol. Despite the encryption and authentication mechanism, we show that many practical BLE devices are vulnerable to data leakage and malicious control. We present BlueDoor as a technique for obtaining the information flow in BLE connection. The key idea in BlueDoor is to leverage two intrinsic properties of BLE and its application scenarios on low capability wearable devices.

First, many BLE enabled smart devices have limited input capability, and so the BLE mechanism is designed to be compatible with these kinds of devices. The limited input and output capabilities make it difficult and inconvenient to set up an encryption key with complex user interaction. For compatibility with various scenarios, the BLE mechanism provides different methods of generating an encryption key according to the device capability. Leveraging this feature, in BlueDoor we propose a method of mimicking a low input capacity device to circumvent the requirement of user input for generating the encryption key. As a result, BlueDoor can enforce a new encryption key with the target device in silence, without requiring any user input. It allows BlueDoor to bypass the encryption requirement for a BLE connection.

Second, for a BLE connection with encrypted authentication, we can downgrade the connection to authentication without encryption. One of the reasons is that many smart devices have different security levels of information. The BLE mechanism is compatible with accessing different levels of information. Thus, BlueDoor can downgrade the link with authentication and encryption to the link without encryption by rejecting the encryption request. Moreover, the rejection of encryption does not incur extra operations in BLE state management, and upper layer users are typically unaware of this. Thus, BlueDoor can pass the authentication by a man-in-the-middle attack.

As a result, BlueDoor can bypass the encryption and authentication in BLE, and obtain stored data on BLE devices as well as the communication data through the BLE connection. We implement BlueDoor and apply it on 15 different Commercial Off-The-Shelf (COTS) BLE devices from different manufacturers. We successfully read and write data on most BLE devices. For example, we can obtain the sleep information, heartbeat rate, walking steps, etc., from a fitness tracker. BLE has become the last mile for typical network connections in many scenarios, and so we need to pay more attention to the problem in BLE. We can obtain data packets such as the notification data including incoming call, SMS messages, etc. through BLE. Information such as SMS messages, used as a part of authentication, is usually considered private and sensitive. We can impersonate a legal user to log in a web account with the SMS authentication message. BlueDoor also enables operations on BLE devices, e.g., setting an alarm clock on a fitness tracker, turning on a smart bulb, and even unlock a smart lock.

We show that such a problem comes from the fundamental tradeoff for using BLE on diverse wearable and embedded devices. The design of BLE should be able to adapt to various kinds of devices and BLE applications in many scenarios, aim to introduce minimal overhead to the device user. Thus, the design of BlueDoor is not specific to a certain implementation but can be generalized to most BLE devices. The large and increasing number of BLE devices that are close to everyone's daily life may even exacerbate such a problem.

By analyzing the causes of the vulnerability, we present lessons learned from BlueDoor. To address the vulnerability, we exhibit an example of a defense mechanism for BLE without any modification to devices, which can be applied to most BLE devices by software updates. We also conduct a user survey to examine the impact of the defense method on normal BLE users.

The contributions of this work are as follows:

- **BLE vulnerabilities.** We present an in-depth analysis of BLE and find the common vulnerability for BLE due to intrinsic BLE mechanism and design choices in diverse embedded and wearable devices.
- **Break BLE connection.** We propose BlueDoor to obtain data on BLE devices and packets through BLE connection, bypassing encryption and authentication, while leveraging intrinsic properties of the BLE mechanism.
- **Lessons & defense.** We present the lessons learned from this work and design a secure defense mechanism to address the vulnerability.

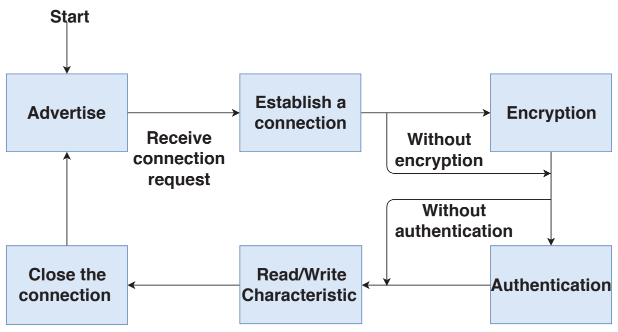


Figure 1: BLE workflow.

The rest of the paper is organized as follows. Section 2 introduces the BLE overview. Section 3 introduces the weakness in the workflow of BLE and the main design of BlueDoor. Section 4 introduces the implementation of our attack method. Section 5 describes some cases under our method. Section 6 shows the evaluation results on different COTS BLE devices. Section 7 presents the implications and suggestions. Section 8 introduces related works. Section 9 concludes this work.

2 BLE OVERVIEW

Overall, there are two main roles, central and peripheral, for devices in the BLE communications, where the peripheral acts as the data holder and the central acts as the data requester. A typical central device can be a smartphone or tablet with relatively high CPU and memory resources. A typical peripheral device, on the other hand, can be a fitness tracker or a smart lock with relatively low CPU and memory resources compared with central devices. Figure 1 shows the workflow of BLE communication.

Advertise and establish a connection. To set up a connection, the peripheral device first advertises packets to enable central devices to discover it. The advertising packets usually contain the basic information of the advertising device, including the MAC address, information of the manufacturer, and the connectivity status. A central device acquires the basic information of the peripheral device from advertising packets and uses the MAC address to identify a specific device.

A central device can establish a connection after receiving advertising packets from a connectable peripheral device. The central device initiates the connection by sending a connection request (`CONN_REQ`) packet with the parameters of frequency-hopping, including hop interval, hop increment, access address, and CRC init. Upon receiving a connection request from the central device, the peripheral device negotiates a common frequency hopping sequence with the central device and establishes a connection with the central device. The central and peripheral devices can communicate using the frequency hopping sequence.

Encryption. The BLE specification provides the encryption mechanism to avoid eavesdropping. The central device can encrypt the communication packets. For encryption, the central device first sends an encryption request (`LL_ENC_REQ`) packet to the peripheral device. If the peripheral device accepts the encryption request, it sends back an `LL_ENC_RSP` packet to the central device; otherwise, it returns an `LL_REJECT_IND` packet to reject the encryption request. If the peripheral accepts the encryption request, the central and peripheral negotiate a common encryption key. This process of generating an encryption key is also known as pairing.

The pairing process incorporates different levels of user participation. BLE should adapt to different types of devices with limited input and output capabilities, which is difficult and inconvenient to setup an encryption key by the user's input. Thus, BLE provides the following four pairing methods, in increasing order of security, according to the capabilities of the peripheral and the central device.

(1) Just Works: Two devices negotiate the encryption key by exchanging wireless packets without the user's authorization, e.g., the user does not need to confirm anything on the devices.
(2) Numeric Comparison: The two devices negotiate to generate a 6-digit code. The code is displayed on both devices for users so that users can confirm those two pairing devices. For example, a user can click the fitness tracker to confirm the pairing.
(3) Passkey Entry: One device (e.g, peripheral) randomly generates and displays a 6-digit code. A user records this code and enters it into the other device (e.g., central device). The encryption key is then generated based on the shared 6-digit code, which cannot be intercepted by other wireless eavesdroppers.
(4) Out-of-Band: The two devices exchange some additional information using an out-of-band channel such as NFC, and use this information for key generation.

The central and peripheral devices choose the pairing method with the highest security level based on their capabilities. After pairing, the central and peripheral devices start to use the key to encrypt the transmitting data. Once the two devices are paired, they can use the key for subsequent data communication and do not need to pair again.

Authentication. Besides encryption, the central device also needs authentication to visit data with high security requirements in the peripheral device. The process of authentication is to verify the legality of the central device. BLE specification [7] does not provide specific application layer authentication methods, but suggests to manufacturers to implement authentication of their own in the application layer. Thus, different BLE devices may implement different authentication methods. Generally, like the authentication in Web applications, the central and peripheral devices should first agree on a certificate. The central device uses a series of read or write packets to send the certification to the peripheral device.

Read/write characteristics. In a peripheral device, data is stored in Generic Attribute Profile (GATT) [8]. The central device can send read/write requests to require data in the peripheral device. GATT contains a list of characteristics, which are the basic units for data storage in a peripheral device. Each characteristic contains the following attributes.

- UUID: The unique identifier of the characteristic.
- Handle: The access address of the characteristic.
- Property: A list of permitted operations on this characteristic, which can be *read only*, *write only* or *read and write*.
- Secure: The access control on this characteristic, which can be *no security* (NS), *encryption required* (ER), *authentication required* (AR) or *encryption and authentication required* (EA). For example, the encryption required (ER) means encryption is required to read or write this characteristic.
- Value: Data stored in the characteristic.

Encryption and authentication are defined by the secure field of the characteristics in the peripheral devices. Table 1 shows the result of read/write request by different connections. For a characteristic with no security (NS), it can be accessed by any connections. For a characteristic of ER or EA, it can be accessed only by encrypted connections, but returns an error code *Encryption Insufficient* for unencrypted connections. The result of accessing a characteristic requiring authentication is similar to that of encryption. On one hand, in normal communication, the central device should satisfy the encryption and authentication requirements of characteristics, which is the main way to protect BLE data. On the other hand, if an attack can setup a connection with the peripheral device while satisfying the encryption and authentication requirements of a certain characteristic, the attack can access the characteristic and the communication data of the connection.

Close the connection. After finishing data communication, the peripheral or central device can close the connection, and the peripheral device turns back to advertising.

3 BLUEDOOR DESIGN

3.1 Traditional attack methods

BLE protects its data characteristics and connection by ensuring the confidentiality of the communication (through encryption) and the legitimacy of the device (through authentication). More specifically, BLE uses characteristics as the basic data storage unit and protects each characteristic by using the Secure field, which defines the security requirements for the connection.

As a peripheral device in advertising mode accepts any connection request from a central device, a traditional attack method can mimic a central device, and establish a connection with the target peripheral device, e.g., using nRF Connect [9] on a smartphone or gatttool [10]. However, as the attack device does not have the encryption key or authentication certification of the target device, it cannot pass the encryption and authentication. Thus, this method can only use the characteristics without security requirement (NS). To access information from a peripheral device, an attack needs to set up a connection with the peripheral device while satisfying the security requirements.

In order to pass the authentication, gattacker [6] can establish a man-in-the-middle (MITM) connection between a central device

result / connection / char	non-encry, non-auth	encry, non-auth	non-encry, auth	encry, auth
NS	response	response	response	response
ER	*Encryption Insufficient*	response	*Encryption Insufficient*	response
AR	unknown	unknown	response	response
EA	*Encryption Insufficient*	unknown	*Encryption Insufficient*	response

Table 1: Access control of characteristics.

and a peripheral device. If packets between the central and the peripheral device are not encrypted, the MITM device can read and forward the authentication packets between central and peripheral device. Thus, it can pass the authentication. Then, the MITM can construct read or write request to the characteristics required authentication (AR) and obtain the data packets in BLE connection. However, the MITM does not know the encryption key prebuilt between the central and the peripheral device. If the central and the peripheral device enables encryption, the MITM cannot decrypt the packets from the central device and cannot re-encrypted the packets to send to the peripheral device; thus fails the authentication. Therefore, we cannot apply traditional MITM to the scenario where the peripheral device requires encryption and authentication.

3.2 Assumptions and goals

If a device can capture the connection request CONN_REQ packet the first time two devices are establishing a connection, it can obtain parameters of frequency-hopping. The device can then receive all subsequent packets in the connection. If the device captures all packets during the initialization of encryption and authentication, it can obtain the encryption key and the authentication certification. Thus, the device can understand the subsequent data packets in the connection. However, this requires the attack device to capture all packets during the first time the two devices connect, which is not common in practice.

Thus, we consider the scenario of two connected BLE devices, one central device and one peripheral device, which are not the first time connected, i.e., they have already shared the encryption key and authentication certifications, and usually do not need to exchange this information anymore. We assume the attacker cannot physically touch these two devices, i.e., cannot read/input anything on both devices. Meanwhile, we assume the attacker can send and receive packets on a specified BLE channel, which is very common, e.g., any BLE enabled mobile phone or laptop with BLE adapter.

Without loss of generality, the goal of BlueDoor is that, for a general BLE device and a given characteristic in this device, we can establish a connection with the target device, which satisfies the security requirements of the target characteristic. Meaning, we can pass the encryption and authentication required by the given characteristic of the device in this connection. Therefore, we can read/write the target characteristic and obtain all the data of the connection. For example, we can obtain the historical data stored in the characteristic of the peripheral device, and the real-time packets through the connection, e.g., read/write/notification packets.

3.3 Design Overview

The basic idea of BlueDoor is also to leverage the MITM attack. We use a BLE enabled device to build a MITM between the peripheral device and the central devices. Meanwhile, the MITM can pass the encryption and authentication by exploiting BLE properties, i.e., the weaknesses. Finally, we can read/write the characteristics and obtain all data packets through the connection. Here we first briefly introduce the main weaknesses used in BlueDoor design.

Weakness 1: Identity spoofing. A central device uses advertising packets to discover a peripheral device and uses the MAC address to identify a specific device. Thus, a BLE device can pretend to be a legitimate peripheral device by changing its MAC address to that of the peripheral device and advertise as the peripheral device. As a result, the BLE device can deceive the central device, which is the basic requirement for the attack. The central device can check the existence of the legitimate peripheral device through its advertising packets. To suppress legitimate advertising packets, the attack device can first connect to the legitimate peripheral device by sending it a connection request (CONN_REQ) packet. Upon receiving the connection, the legitimate peripheral device stops advertising, and thus it cannot be discovered anymore. In the case that two devices are already connected, we can interrupt the connection (e.g. using a signal jammer) to let the peripheral device go back to the advertising state.

Weakness 2: Disable encryption for central device. Normally, to satisfy the encryption requirement of the characteristic in the peripheral device, a central device tries to enable the encryption by sending an encryption request (LL_ENC_REQ) packet to the peripheral device after establishing a connection. If the peripheral device accepts the encryption request, it sends back an LL_ENC_RSP packet to the central device. However, if the central device does not receive the LL_ENC_RSP packet or it receives the LL_REJECT_IND packet, it will disable the encryption without notification to upper layer users. We think that this is probably due to the practical design considerations from two aspects. First, the central device needs to adapt to possible changes of characteristics on peripheral devices. Second, the peripheral devices are in charge of all the security requirements of all characteristics. The central device usually trusts the security information from the peripheral devices; otherwise, it is also very difficult for the central device to verify the security requirements of characteristics. In practice, we find that most central devices do not use encryptions when the peripheral devices reject the encryption request. Thus, an attack device can either respond with an LL_REJECT_IND packet or ignore the encryption request to reject the encryption request from the

central / peripheral	OOB=1	OOB=0	MITM=1	MITM=0
OOB=1	Use OOB	Check MITM	*NaN*	*NaN*
OOB=0	Check MITM	Check MITM	*NaN*	*NaN*
MITM=1	*NaN*	*NaN*	Check IO capabilities	Check IO capabilities
MITM=0	*NaN*	*NaN*	Check IO capabilities	Just works

Table 2: Check OOB & MITM in pairing.

central / peripheral	DisplayOnly	DisplayYesNo	KeyboardOnly	NoInputNoOutput	KeyboardDisplay
DisplayOnly	Just works	Just works	Passkey Entry	Just works	Passkey entry
DisplayYesNo	Just works	Just works or Numeric Comparison	Passkey Entry	Just works	Passkey Entry or Numeric Comparison
KeyboardOnly	Passkey Entry	Passkey Entry	Passkey Entry	Just works	Passkey Entry
NoInputNoOutput	Just works	Just works	Just works	Just works	Just works
KeyboardDisplay	Passkey Entry	Passkey Entry or Numeric Comparison	Passkey Entry	Just works	Passkey Entry or Numeric Comparison

Table 3: Check I/O capabilities in pairing.

central device. Then the attack device can build a connection with the central device without encryption requirement.

Weakness 3: Silent pairing with peripheral device. Different from the central device, the peripheral device knows the security requirements of all characteristics and thus enforces the encryption or authentication according to the *secure* field of each characteristic. Though an attack device cannot change the security requirement of characteristics, it can modify its pairing parameters to mimic a low capability device. Normally, BLE supports four different pairing methods of setting up the encryption key, i.e., Just Works, Numeric Comparison, Passkey Entry, and Out-of-Band. The highest level of security is Out-of-Band. A peripheral device selects the pairing method with the highest security level according to the capabilities of two pairing devices. When the attack device mimics a low capability device, the peripheral device will select the pairing method of the lowest security, i.e. Just Works. This feature of silent pairing is because BLE is usually used for different types of devices. For devices with limited input and output capabilities, it is usually difficult or inconvenient to set up secure encryption keys by the user's input. So, BLE provides different pairing methods according to the capabilities of peripheral and central devices. Based on silent pairing, two devices using Just Works can negotiate the encryption key without any user participation. Therefore, the attack device can enforce a new encryption key with the peripheral.

Weakness 4: Weak authentication method. The authentication is usually implemented on the application layer in BLE connection. Most devices have the authentication and encryption decoupled, and so even when the encryption is disabled (e.g., by rejecting encryption request), the authentication process still works. Therefore, the BlueDoor device can forward the authentication packets between these two devices to bypass the authentication process. The peripheral then authenticates the MITM device as the legal central device.

We summarize the complete process of BlueDoor in figure 2 as follows:

- Interrupt a connection. BlueDoor first needs to interrupt an existing BLE connection between the target device and a legitimate central device.
- Establish a connection. The BlueDoor device can send a `CONN_REQ` packet to establish a new connection with the peripheral device. At this time, the BlueDoor device can read/write characteristics of NS in the non-encrypted non-authenticated connection.
- Silent Pairing. The BlueDoor device leverages silent pairing method (**Weakness 3**) to pair with target peripheral device. Then BlueDoor can build a shared encryption key with the target peripheral devices. The BlueDoor can thus build an encrypted non-authenticated connection with the peripheral device, and access the characteristics of ER on the target devices.
- Connection degradation. For characteristics requiring authentication, BlueDoor needs to build an authenticated connection with the target devices. The BlueDoor device first builds a connection with the central device and then forwards its authentication information to the peripheral device for authentication. In this process, the BlueDoor device first pretends to be a legitimate peripheral by manipulating its MAC address (**Weakness 1**) to deceive the central device. The BlueDoor device rejects the default encryption request (**Weakness 2**) from the central device (**Weakness 4**) and only allows authentication packets without encryption.
- Read/Write Characteristics. Till now, BlueDoor can read/write all the characteristics on the target device and obtain all data packets in the BLE connection.

In the following part, we mainly focus on the detailed design of silent pairing and connection degradation.

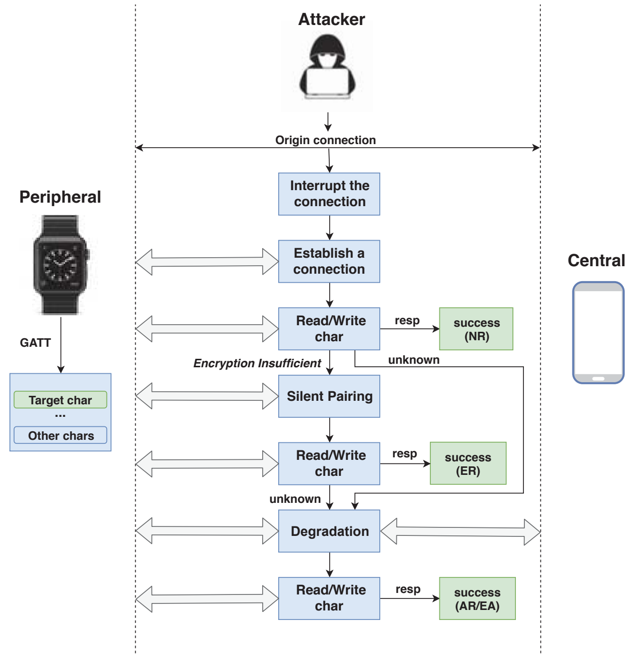


Figure 2: Main workflow of BlueDoor.

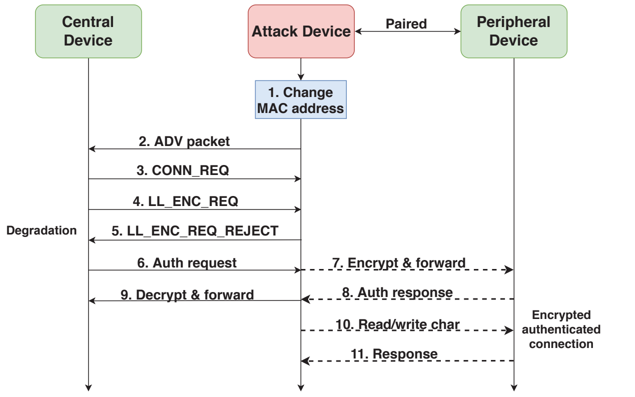


Figure 3: Connection degradation.

3.4 Silent Pairing

The BlueDoor device can pair with the target device without the user's participation by pretending to be a low capability device to enforce the Just Works pairing method. When a central and a peripheral device start pairing, they exchange pairing parameters to determine the pairing method, including SC bit (LE secure connection indicators), OOB Data Flag bit, MITM bit, IO Capabilities. Overall, BLE selects the highest security method based on the parameters of both sides. First, the central and peripheral check SC bit in pairing feature. If both devices have the SC bit set to 1, BLE uses LE secure connection; otherwise, it uses LE legacy pairing. LE Secure Connection is an enhanced security feature introduced in Bluetooth 4.2, which uses a Federal Information Processing Standards (FIPS) compliant algorithm called Elliptic Curve Diffie Hellman (ECDH) for key generation. Therefore, to avoid using LE Secure Connections, the BlueDoor device needs to set the SC flag into 0 to use LE legacy pairing.

For LE legacy pairing, as shown in table 2, if both devices have the OOB bit set to 1, BLE will use an Out-of-Band method for pairing, which generates key with additional input, e.g., user input of a password. Otherwise, if the OOB bit is set to 0 on any side, BLE will ignore the OOB flag and check the MITM bit. Therefore, to avoid using Out-of-Band, the BlueDoor device sets the OOB flag to 0 and the MITM bit to 0. As shown in table 2, if both the central device and the peripheral device set the MITM bit to 0, both devices agree on using the Just Works pairing method. Otherwise, if the MITM bit on the target device is set to 1, BLE will further check the I/O Capabilities. After exchanging the I/O capability, two pairing devices will select a pairing method according to table 3. The BlueDoor device set its I/O Capability to *NoInputNoOutput*, claiming to be a low capability device. According to table 3, no matter what I/O capability the target device has, BLE will use the Just Works pairing method. The reason is that BLE needs to adapt to the device with lower I/O capability devices. As a result, the BlueDoor device can pair with the target peripheral device using Just Works, and they will exchange a new encryption key in silence. Then the BlueDoor device can encrypt the connection with the new encryption key, and visit characteristics required encryption (ER).

3.5 Connection Degradation

Now the BlueDoor device has a new encryption key with the target peripheral device. The next step is to pass the authentication. The authentication packets from central device are encrypted by a pre-built key (with the targeted device), and thus they cannot be decrypted by the BlueDoor device. The goal of connection degradation is to make the central device send authentication packets without encryption. We find that the encryption process is independent with the authentication process on most BLE devices. The reason is that the encryption process is usually implemented in the protocol layer by the BLE specifications, but the authentication process is implemented at the application layer by different manufacturers. Therefore, the BlueDoor device degrades the connection by rejecting the encryption request from the central device. Then the central device usually chooses to send authentication packets without encryption, and thus the BlueDoor device can forward the authentication to the peripheral device.

As shown in figure 3, the connection degradation method consists of the following steps. (1) The BlueDoor device first changes its MAC address to that of the target peripheral device. (2) Then the BlueDoor advertises the same packet with that of the target peripheral device. The BlueDoor device can record the advertising packet of the target peripheral device while it is broadcasting. The target peripheral device stops advertising after it connects to the BlueDoor device during silent pairing. When the central device tries to connect with the target device, the central device receives advertising packets from the BlueDoor device. (3) The central device will treat the BlueDoor device as the target device and establish a connection with it. (4) If the central device has paired with the target peripheral device before, it will send an encryption request using the pre-built key. (5) The BlueDoor device sends an `LL_REJECT_IND` packet to refuse the encryption request. As the process of encryption and authentication decouples, the central device will stop encryption, and send authentication packets without encryption. (6)-(9) For authentication, when the attack device receives an authentication request from the central device, it forwards the requests to the peripheral device. It also forwards the responses from the target device to the central device. If the connection between the attack device and peripheral device is encrypted, the BlueDoor device needs to encrypt or decrypt the requests to and responds from the peripheral device.

The connection degradation is due to the following two fundamental reasons rather than implementation bugs. (1) The lack of interaction across different layers in BLE. At the protocol layer, the BlueDoor device can reject the encryption request from the central device. If the BlueDoor rejects the encryption request, the central device stops using encryption anymore. Meanwhile, such a change is usually not captured at the application layer. Thus, the application layer is unaware of the change, and most central devices still send authentication information without encryption. (2) Usually, there exist characteristics (NS) that do not require encryption. To access those characteristics, the upper layer has to continue the authentication process even when the encryption is not enabled.

Note that if the central device does not send authentication packets after the BlueDoor device rejects the encryption request, we can perform silent pair with the central device to enable the encryption process between the central device and the BlueDoor device.

After the central device has finished the process of authentication, the BlueDoor device is also authenticated by the peripheral device. Then the BlueDoor device can access the characteristics required by both encryption and authentication (EA) on the peripheral device.

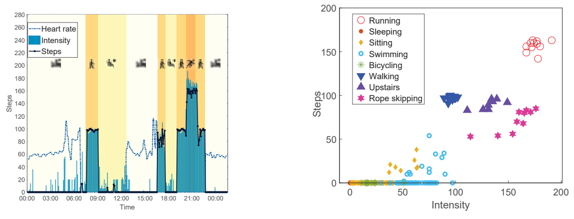
4 IMPLEMENTATION

In our implementation, we use a laptop with Bluetooth adaptor nRF51822 [11] as our attack device. To interrupt a connection, we use a Bluetooth signal jammer [12]. The signal jammer can interrupt the connection between the central and the peripheral devices in less than 10 seconds. Therefore, BlueDoor can establish new connections as a MITM between the peripheral and the central devices.

To establish a connection with the peripheral device and send read/write requests, we use the *gatttool* in *blueZ* [10]. Thus, we can establish and maintain a connection with a BLE peripheral device. We can also inquire about the characteristic list of GATT, and generate read/write requests by BLE commands using the tool.

For silent pairing, we use the *bluetoothctl* in *blueZ*. *bluetoothctl* allows users to modify the capabilities of the device. Based on the tool, we can pair with a connected device and encrypt the connection. We can also use the tool to send read/write requests in the encrypted connection.

For connection degradation, we first use the tool *bdaddr* [13] to change the MAC address of the Bluetooth adapter. Therefore, to pass the authentication, we can use *gattacker* [6] to establish a MITM between the central device and the target peripheral device. If packets between the central and the peripheral device are not



(a) Intensity, steps and heart rate changes during one day.

(b) Classification of different activities by intensity and steps.

Figure 4: User activity data.

encrypted, the MITM can read and forward the authentication packets between central and peripheral device. But *gattacker* does not support pairing with a peripheral device. Thus, to implement our attack method, we write a new tool based on *bluetoothctl* and *bleno*. We use the tool to pair with the user's peripheral device and build an encrypted connection with it. Meanwhile, we use the tool to reject the encryption request from the central device. Our new tool can also record and modify the transmitting packets for characteristic analysis.

We use nRF51822 [11] and Wireshark [14] to sniff the communication for non-encrypted connections. The nRF51822 can get the hopping parameters from the `CONN_REQ` packet and follow the frequency-hopping sequence of a connection.

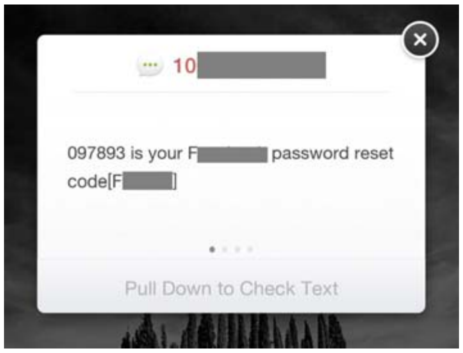
5 CASE STUDY

In this section, we show three attack cases with BlueDoor. We use Mi Band 2, a fitness tracker supporting the BLE 4.0, as the peripheral device. We use an Android-based smartphone (Smartisan U2 Pro), and an iOS-based smartphone (iPhone 8) as the central devices, which support BLE 4.2 and 5.0, respectively.

5.1 Obtain sensitive personal data

A typical peripheral device such as fitness trackers normally records personal activity data of the user, including user activity, heart rate, sleeping status, etc. As the data in fitness tracker cannot be real-time synchronized with the connected central device (e.g., a smartphone), part of the data is cached locally in the fitness tracker. With BlueDoor, we can obtain the historical data stored in peripheral device and the real-time data packets in the process of communication. It should be noted that those personal data, once stored on your mobile device, are considered sensitive personal data and require privilege to access those data. For example, APPs in iOS to access the data require specific privileges explicitly confirmed by data owner. However, once we have obtained the read and write permission of the corresponding characteristic, we can obtain the historical data in the fitness tracker, which circumvents the security requirements of the sensitive data.

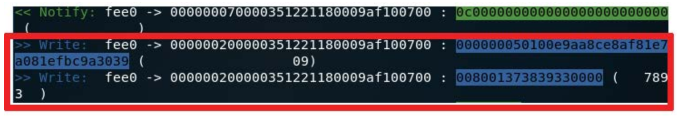
We initially connect the fitness tracker with the central device of iPhone 8, as the standard usage scenario. We use BlueDoor to read the historical data stored in the fitness tracker. Figure 4(a) shows the results of the experiments. The obtained data contains the heart rate, intensity, and steps of the users of the fitness tracker. Here intensity is calculated by the fitness tracker to indicate the



10[redacted]

097893 is your F[redacted] password reset code[F[redacted]]

(a) Received SMS message.



```
<< Notify: fee0 -> 000000070000351221180009af100700 : 0c000000000000000000000000
 (               )
>> Write:  fee0 -> 000000020000351221180009af100700 : 000000050100e9aa8ce8af81e7
a081efbc9a3039 (                        09)
>> Write:  fee0 -> 000000020000351221180009af100700 : 008001373839330000 (   789
3  )
```

(b) Obtain the verification code.

Figure 5: Obtaining the verification code in SMS messages via BlueDoor.

intensity of activity of the user. Based on the raw data, we can further analyze the detailed activity and living habits for the user of the fitness tracker.

Figure 4(a) shows the intensity, steps, heart rate information, and activity type of a volunteer user for one day. We can see that the data reflect the main activity of the user for the entire day. From 00:00 to 07:00, the fitness tracker user is sleeping. The heart rate is recorded automatically to detect sleep quality when the user is sleeping. There are no steps, and the intensity is very low. From 07:00 to 09:00, the user is walking outside. During this time, the fitness tracker stops recording the heart rate. The intensity and steps are visibly increasing. From 09:00 to 12:00, the user is sitting in front of his desk and working. The intensity and steps are much lower than those in walking. Then, the user is sleeping from 12:00 to 16:00, and from 16:00 to 17:00, the user is walking. From 18:00 to 19:00, the user is eating. In the period between 20:00 to 22:00, the user is running, and the intensity and steps are much higher than those in walking. At 23:00, the user begins to sleep.

We further show the result of the activity inference based on the obtained data. Figure 4(b) shows the result. We can see that those activities can be classified based on intensity and steps. From the result, we can see that running leads to the highest intensity in all activities. Compared with walking, climbing stairs and rope skipping lead to higher intensity, but similar steps. That is due to more arm swings in climbing stairs and rope skipping. Similarly, swimming leads to a higher intensity than sitting and bicycling. By adding more features such as heart rate, we can obtain a higher classification accuracy.

5.2 Obtain secure SMS messages

Most of the fitness trackers support notification of an incoming call, SMS messages, and SNS messages from the central devices. The notification is delivered from the central device to the peripheral device through the BLE connection.

It is very common to use SMS messages as part of the authentication in many scenarios, such as log in a website, confirm a transaction, etc. SMS authentication is one of the major forms of 2FA(two-factor authentication) [15], and 71.5% of respondents having experienced authentication via SMS message [16].

As an example, we show how to use BlueDoor to obtain the verification code from the SMS message. Once with the SMS verification code, we can further break the secure information flow. Note that we are not discussing the security of the SMS message itself. It has been shown that SMS in the old generation of mobile communication is considered insecure [17][18]. We show that even if the SMS message is secure itself, BlueDoor can obtain the information through the step it is delivered to the peripheral device, and thus the secure flow cannot be assumed anymore.

We use Mi band 2 as an experimental device that supports the notification of the incoming call, short messages, SMS messages, etc. We use another smartphone as the central device. Usually, to prevent users from forgetting their passwords, most websites offer multiple options to log in or reset their passwords. A common method is to use an SMS verification code. Based on the account name, we can input the username of a legitimate user and then ask the website to send an SMS verification code to reset the password. Once the user's phone (central device) received the verification message, the central device will send packets to the peripheral device to notify the user. By using BlueDoor, we can capture these real-time communication packets.

Figure 5(a) shows the SMS message that the user received on the smartphone. The SMS message contains the verification code and the website identity (e.g., the name or the URL of the website, or both). Figure 5(b) shows the packets captured by BlueDoor. In this case, Mi band 2 receives the SMS that contains the verification code. Then, BlueDoor can obtain this verification code, which we can further use to reset the password (or bypass authentication process in other cases).

5.3 Prerequisites for further attacks

After bypass the authentication process, the attack device is not only authenticated by the peripheral device as a legitimate device but also authenticated by the central device. We can further obtain information from the central device, which can be the prior steps for further BLE based attack method [19].

6 EVALUATION

To further understand those BLE devices, we bought 15 different BLE devices, such as wearable devices of commonly used fitness trackers and smart home devices like smart locks and smart bulbs. We choose fitness trackers from the major markets according to the IDC Report [20]. These include Mi Band 1, 2, 3 from Xiaomi, Flex 2, charge 2 and Alta from Fitbit, and honor Band 3 from Huawei. Figure 6 some of the devices.

BlueDoor can be applied to many other types of devices, such as thermometer from Xiaomi, smart weight scale (SENSSUN FAT), etc., as long as they use BLE for connection. We have also tested our method on other types of BLE devices including thermometer,

Figure 6: BLE Devices.

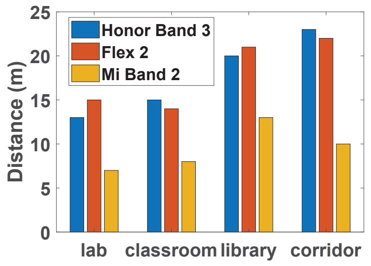
smart bulb, and mobike (sharing bike that supports using BLE to connect to user mobile phone).

Besides those commercial devices, to examine the performance of BlueDoor in a more controlled environment, we also implement a special device (Charlie) with characteristics of all four different security requirements. Thus, we used a total of 16 different devices in our evaluation.

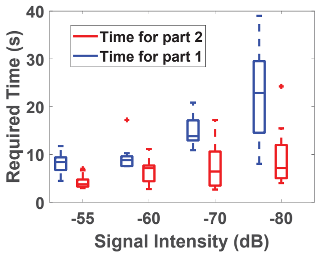
These peripheral devices used in our evaluation use different versions of BLE from 4.0 to 4.2. The central devices used in our evaluation support BLE from 4.2 to 5.0.

Attack range. The attack range of BlueDoor is related to the range of BLE communication. We measured the range of BlueDoor in different environments, including the laboratory, classroom, library, and corridor. Figure 7(a) shows the evaluation results. Usually, BLE can achieve a minimum receiving sensitivity of -70 dBm to -82 dBm [21]. Thus, we chose -80 dBm as a threshold to determine whether it is a valid BLE connection and calculated its corresponding range. We chose three devices from different manufacturers, Mi Band 2 from Xiaomi, Honor Band 3 from Huawei, and Flex 2 from Fitbit in our evaluation. In the library and the corridor, Honor Band 3 and Flex 2 could achieve a range of more than 20 m. But the range drops to 15 m in the laboratory and the classroom, due to more walls and reflective surfaces in those environments. The signal intensity drops sharply after going through the wall. Mi Band 2 could only achieve a range of 7 m in the laboratory and 13 m in the library.

Attack time required. We evaluate the duration needed for the attack. The result is shown in Figure 7(b). We use Xiaomi 3 as the central device and put it 0.5 m away from the peripheral device, Mi Band 2, which is a common distance in practical usage scenarios. We use a laptop with Bluetooth adaptor as our attack device. We repeat the attack 20 times and divided the attack duration into two parts: (1) the time for establishing a connection with the peripheral device, and (2) the time for establishing a connection with the central device and bypass the authentication process. The average time for part (1) is about 22.49 s, and the average time for part (2) is about 9.45 s when the signal strength is -80 dB, and the attacker is



(a) Attack range in different places.



(b) Time required for two parts.

Figure 7: Range and required time of the BlueDoor.

7 m away from the peripheral device. We also evaluate the required time when the signal strength is -70 dB, -60 dB and -55 dB. We can see a drop of the required time needed for both parts. For example, the average required time for part (1) is 8.03 s and for part (2) is 4.25 s when the signal intensity is -55 dB. This is because higher signal strength leads to better connection quality, less packet loss, and less retransmission.

Silent Pairing. Then we evaluate the effectiveness of the main steps of BlueDoor. We mainly examine the practical performance for silent pairing and connection degradation in real devices and their success ratio. We first test the silent pairing method on 16 different types of devices. Table 4 shows that silent pairing method has succeeded on all those 16 devices. This indicates that we can pair with these devices without user's authorization. Then BlueDoor is able to visit the characteristics require encryption (ER) in these devices.

Connection degradation. We also evaluate the success ratio of connection degradation on 16 different types of devices. In our evaluation, for all devices except Honor Band 3, we can successfully degrade the connection and pass the authentication. We find that for the connection between Honor Band 3 and the mobile phone, the central device (mobile phone) does not send authentication packets when the peripheral device rejects the encryption request. In such a case, the BlueDoor device can still perform silent pairing with the central device to obtain the encryption key. Then the central device will send authentication packets, and the attack device can forward the received authentication packets.

Read performance. We further evaluate read performance to characteristics on those devices. First, BlueDoor will query the peripheral device and obtain the list of all characteristics on the device. Then for each characteristic, we can evaluate the read and write performance, i.e., the ratio of characteristics that can be successfully read and written.

Table 5 shows the attack results on readable characteristics and their corresponding security requirements. Overall, we can see that different devices have different numbers of characteristics. The number of characteristics ranges from 6 to 24 for devices in our experiment. Meanwhile, we further notice that devices from the same manufacturer tend to have the same number of characteristics. For example, Flex 2, Charge 2, Alta from Fitbit all have eight

Device Name	Silent pairing	Connection degradation
Mi Band 1	√	√
Mi Band 2	√	√
Mi Band 3	√	√
Flex 2	√	√
Charge 2	√	√
Alta	√	√
Watch 9	√	√
honor Band 3	√	×
MiSwi (clock)	√	√
Mi Thermometer	√	√
SENSSUN FAT	√	√
Smart Lock	√	√
LifeSmart (bulb)	√	√
fmxy (beacon)	√	√
mobike	√	√
Charlie	√	√

Table 4: Result of readable characteristics.

Device Name	# of readable chars	# of successful attacks				
		NS	ER	AR	EA	total
Mi Band 1	23	23	0	0	0	23
Mi Band 2	24	22	0	2	0	24
Mi Band 3	24	22	0	2	0	24
Flex 2	8	7	1	0	0	8
Charge 2	8	7	1	0	0	8
Alta	8	7	1	0	0	8
Watch 9	7	7	0	0	0	7
honor Band 3	17	17	0	0	0	17
MiSwi (clock)	6	6	0	0	0	6
MI Thermometer	14	14	0	0	0	14
SENSSUN FAT	14	14	0	0	0	14
Smart Lock	14	13	0	1	0	14
LifeSmart (bulb)	8	8	0	0	0	8
fmxy (beacon)	21	21	0	0	0	21
mobike	8	3	0	5	0	8
Charlie	4	1	1	1	1	4

Table 5: Attack result of readable characteristics.

characteristics. As shown in Table 5, we can see that BlueDoor can read all characteristics on all devices.

Among those readable characteristics, we observe that the security requirements for most of them are *no security* (NS), which indicates that any connected central devices can read these characteristics without encryption or authentication. Only Flex 2, Charge 2, and Alta have one characteristic that requires encryption. Only Mi Band 2 and 3, the smart lock, and mobike have several characteristics that require authentication. None of those characteristics requires both encryption and authentication, which shows that the security requirement of data on those devices should be more carefully considered. For the Charlie device, which has characteristics of all four different security requirements, BlueDoor can also read all characteristics on this device.

Write performance. We also evaluate the write performance of BlueDoor on different characteristics of those devices. Different from readable characteristics, it is more difficult evaluating write performance. First, we need to infer the format (e.g., parameters) of the write request; otherwise, the write request cannot succeed. We should, however, note that inferring the format is not related to our attack method. Even we do not know the format, BlueDoor can still build a connection while bypassing the encryption and authentication. Second, in practice, it is difficult to check whether a write request to a specific characteristic is successful or not. Normally, if a write request changes the value of characteristic, we can check the result by reading the characteristic after the write request. However, not all the write requests will change the value of the corresponding characteristics. For example, a write request to trigger operations like vibrating the fitness tracker will not change the value of the characteristic.

To infer the format of the request, we use different methods. (1) Many peripheral devices have official mobile APPs provided by the device manufacturer to control the devices. We first install those APPs on a mobile phone and use those APPs to control the devices. Meanwhile, we record the corresponding write request packets in the BLE connection, and then use those packets in our evaluation. (2) There are some open-source code for some types of devices. We can obtain the format of the write request from the source code. (3) We can infer the format according to the features of the data stored in the characteristics, like the RGB color of the bulb. To check the result of the write request, we take a conservative method. We only consider the request with a response from the peripheral device. The measure is conservative as no response from the peripheral device does not necessarily mean the write request is not successful.

Table 6 shows the overall result. First, we can see that most devices have only a very limited number of writable characteristics, which coincides with the fact that those devices usually act as data providers (e.g., providing fitness data to central devices). We successfully infer the format for write requests to all characteristics. Meanwhile, we can obtain a response for most writable characteristics, except for two characteristics on Mi Band 2 and Mi Band 3. Thus, for those two characteristics, we cannot determine whether the write request is successful or not.

Similar to readable characteristics, none of those characteristics requires both encryption and authentication. Therefore, we also evaluate BlueDoor on Charlie for characteristics with both encryption and authentication. Therefore, to check the result, we read characteristics on Charlie to see if the value is changed. The result shows that BlueDoor can successfully write to characteristics on Charlie.

Some devices (e.g., MiSwi) mainly use BLE for information collection, and they use other technology instead of BLE to control the device. So we cannot collect write requests to infer the format.

In summary, for all representative BLE devices, we find 204 readable characteristics and 12 writable characteristics in total. BlueDoor can read all of the readable characteristics, and write to all of the writable characteristics, except two uncertain writable characteristics with no responses.

7 LESSONS AND DEFENSE

From the design and evaluation of BlueDoor, we have learned the following lessons. First of all, the initialization of BLE devices is very important. It is always important to ensure the initial pairing and authentication are secure. Usually, the initial pairing and authentication are assumed to be secure by many other applications.

Device Name	# of inferred chars	# of successful attacks				
		total	NS	ER	AR	EA
Mi Band 1	2	2	1	0	1	0
Mi Band 2	4	3	1	0	2	0
Mi Band 3	4	3	1	0	2	0
Flex 2	0	0	0	0	0	0
Charge 2	0	0	0	0	0	0
Alta	0	0	0	0	0	0
Watch 9	0	0	0	0	0	0
honor Band 3	0	0	0	0	0	0
MiSwi (clock)	0	0	0	0	0	0
MI Thermometer	0	0	0	0	0	0
SENSSUN FAT	0	0	0	0	0	0
Smart Lock	1	1	0	0	1	0
LifeSmart (bulb)	1	1	1	0	0	0
fmxy (beacon)	0	0	0	0	0	0
mobike	0	0	0	0	0	0
Charlie	4	4	1	1	1	1

Table 6: Attack result of writable characteristics.

If the initialization is not secure as assume, subsequent transmissions between the central and the peripheral devices can be easily overheard and interpreted.

A simple and straightforward method of defense is to enable user participation in the process of pairing, such as using numeric comparison and passkey entry for pairing, requiring user confirmation during pairing, using side-channel information (e.g., user gesture), notifying users of new pairing, etc. Those features certainly introduce a tradeoff between convenience and security. We conduct a survey on typical methods to improve security by incorporating user participation. A total of 84 users participate in the survey, of which 72 are males and 12 are females. Among them, 67 users have their own BLE devices or have used BLE devices before (77.61% have used fitness trackers), while the others do not have any BLE devices. 59.52% of the volunteers prefer to enable SMS reminder on a BLE peripheral smart device, and 72.62% care about the security of the SMS data. Almost all volunteers are willing to use encryption to protect their data, except for four people. Therefore, to protect the data on BLE, 61.9% of the volunteers agreed to confirm the status of the connections to ensure data security each time the device reconnected. The other participants, however, prefer not to be interrupted.

An attack device can mimic a low I/O capability device and easily build an encrypted connection with a peripheral device. It is better to consider the I/O capability of the peripheral in the pairing process. We recommend that the BLE device manufacturers consider the pairing process more carefully and restrict the pairing methods. For example, we usually connect fitness trackers to a smartphone. We can assume that the central device (i.e., a smartphone) usually has a high I/O capability. If an attack device mimics a low I/O capability device, the peripheral device can reject the pairing request or just cancel the connection.

In practice, we find that many manufacturers force encryption at the software level instead of the BLE protocol level. It is better to enable both encryption and authentication in BLE connections, i.e., using EA secure requirement. Unfortunately, in our evaluation, we find none of the existing devices uses EA secure requirement to protect its characteristics.

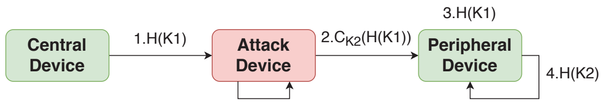


Figure 8: Defense method in the MITM scenario.

The success of MITM also relies on the state check mechanism in the BLE protocol. Assuming a legitimate central device is connected to a legitimate peripheral device with a shared encryption key. When the connection is interrupted, BLE usually assumes the previous state with the shared encryption key. A MITM device can refuse the encryption from the central device to set up a connection without encryption, leading to inconsistency in the BLE state. Such a change, however, is not sent to the upper layer protocol in BLE design. Thus, the BLE protocol should check the consistency of states of connection and notify the upper layer protocol with the state change.

Note that some vulnerabilities come from the design choice of BLE applications. There should be different ways to address this problem. We propose a mechanism to prevent EA characteristics from attacks. In the mechanism, we use unique and unforgeable information (encryption key) instead of a random number used in the weak authentication method. Before authentication, the central and peripheral device should negotiate a common encryption key first.

(1) The central device applies a Hash algorithm on the encryption key K (or/and adds additional information) and obtains $H(K)$ as the authentication packet. Then it encrypts the authentication packet and sends the encrypted packet $C_k(H(K))$ to the peripheral device.
(2) The peripheral device decrypts the received authentication packet and it can obtain the authentication information A_r. Normally, we have $A_r = H(K)$.
(3) The peripheral device applies the same Hash algorithm on the encryption key K locally to calculate the hash value A_c. Normally, we have $A_c = H(K)$.
(4) The peripheral device compares the calculated hash value A_c with the received authentication information A_r. If $A_c = A_r$, the connection from the central device is authenticated.

If there is a MITM device in the communication process, as shown in figure 8, the attack device negotiates a new encryption key $K2$ with the peripheral device, which is different from the previous key $K1$. The central device stores $K1$, which is negotiated by the central and the peripheral devices. As the attack device does not know the authentication method (Hash method in this case) and the additional information, the attack device cannot generate an authentication packet of $H(K2)$. Then the peripheral device will be aware of the difference between the received authentication information and local calculated information.

8 RELATED WORK

Bluetooth Low Energy (BLE) is a widely adopted wireless personal area network technology used for communication in low power and smart devices [22]. Recently, there have many engineering efforts and research works on Bluetooth security issues. Xing et al. [23] propose a method to predict the frequency hopping sequence without overhearing the CONN_REQ packet. Ryan et al. [24] propose a tool [25] to calculate the encryption key by overhearing packets during pairing. These works rely on overhearing the communication packets between the central and the peripheral devices. Meanwhile, if the packets are encrypted, they cannot interpret the overheard packets. There are also many studies on the vulnerabilities of BLE in different situations [26][27][28], including beacons network [29], telemedicine devices [30], Internet of Things systems [31], etc. They do not propose a systematic attack method for common BLE devices. The work in [32] introduces a method to make two (or more) victims agree on an encryption key with only 1 byte of entropy, which enables the attacker to exhaustively search for the negotiated encryption keys.

Some research works have also shown the attack method to mimic a device with a low capability to circumvent the requirement of user input in traditional Bluetooth 2.0 [33] [34]. Compared with the pairing process in traditional Bluetooth, BLE takes more parameters (e.g., SC flag bit) as well as more security features (e.g., LE secure connection) for devices with different capabilities. Thus, the silent pairing method need more operations to achieve the goal. Our work can also bypass application-level authentication, which is mentioned as a possible defense in [33][34].

There are also a series of tools for analyze the security of BLE, which facilitate the implementation of BlueDoor. For example, the *Ubertooth* project [35] and USB dongle nRF51822 [11] implement a BLE sniffer, which has been used in many BLE related works. BlueZ [10] is a Bluetooth stack for the Linux kernel-based family of operating systems. It contains the basic scripts to scan the BLE channel, connect and pair with a BLE device, and send read/write requests with any BLE devices. The tool *gattacker* [6] provides a method to establish a MITM attack on BLE devices. However, it cannot build a MITM between BLE devices requiring encryption.

There are also many works on using BLE data for user activity analysis and prediction. Aveek et al. [36] sniffs the data packets and heartbeat packets in the BLE communication. They use the size and frequency of data packets to predict user activity. There are some works [37][38] using data from accelerometer, gyroscope, and compass on a smartwatch to capture user motion. Our work focuses more on how to obtain data packets and stored information on peripheral devices via BLE. Our work can be used as the basis for some of those works.

9 CONCLUSION

Nowadays, more and more devices use Bluetooth Low Energy (BLE) for data transmission. We thoroughly examine the BLE protocol and present a general attack method, which can pass the access control of BLE devices without touching the device, to obtain packets through BLE connection and information stored on BLE devices. The key idea of BlueDoor is to mimic a low capacity device to downgrade the process of encryption key negotiation and authentication. We apply our method to different types of 15 COTS BLE devices, and show that most BLE devices are vulnerable to information leakage and even malicious control. Therefore, this introduces significant vulnerability to existing secure information flow by obtaining different types of information delivered to BLE devices, e.g., SMS message, notifications. Besides privacy threat, this further makes traditional operations insecure, e.g., using SMS for verification in authentication.

ACKNOWLEDGEMENTS

We sincerely thank our shepherd, Rijurekha Sen, and the anonymous reviewers for their valuable feedback. This work is in part supported by National Natural Science Fund for Excellent Young Scholars (No. 61722210), and National Natural Science Foundation of China (No. 61932013, 61532012, 61802180), Natural Science Foundation of Jiangsu (No.BK20180421), the National Cryptography Development Fund (No. MMJJ20180105), and the Fundamental Research Funds for the Central Universities (No. NE2018106).

REFERENCES

[1] Artem Dementyev, Steve Hodges, Stuart Taylor, and Joshua Smith. Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario. In *Proceedings of IWS*, pages 1–4. IEEE, 2013.

[2] Shamsaa Hilal Al Hosni. Bluetooth low energy: a survey. *International Journal of Computer Applications*, 162(1), 2017.

[3] Bluetooth smart & smart ready market analysis by technology, by application (transportation, consumer electronics, home automation, medical), by region, and segment forecasts, 2018 - 2025. https://www.grandviewresearch.com/industry-analysis/bluetooth-smart-and-smart-ready-market.

[4] Global wearables market grows 7.7the leader position, says idc. https://www.idc.com/getdoc.jsp?containerId=prUS43598218.

[5] New wearables forecast from idc shows smartwatches continuing their ascendance while wristbands face flat growth. https://www.idc.com/getdoc.jsp?containerId=prUS44000018.

[6] Sławomir Jasek. Gattacking bluetooth smart devices. *Black Hat USA*, 2016.

[7] Bluetooth specification (core_v4.2.pdf). https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439.

[8] Robert Davidson, Akiba, Carles Cufí, and Kevin Townsend. Getting started with bluetooth low energy, chapter04. https://www.oreilly.com/library/view/getting-started-with/9781491900550/ch04.html.

[9] nrf connect github. https://github.com/NordicSemiconductor/Android-nRF-Connect.

[10] Official site of bluez. http://www.bluez.org/.

[11] nrf51 dongle. http://infocenter.nordicsemi.com/pdf/nRF51_Dongle_UG_v1.0.pdf.

[12] Bluetooth jamming: What for and how to? https://www.jammer-store.com/bluetooth-jamming-what-for-and-how-to.html.

[13] Change bluetooth address. https://kasiviswanathanblog.wordpress.com/2017/03/28/change-bluetooth-address/.

[14] Wireshark user documentation. https://www.wireshark.org/docs/.

[15] The 2019 state of the auth report: Has 2fa hit mainstream yet? https://duo.com/blog/the-2019-state-of-the-auth-report-has-2fa-hit-mainstream-yet.

[16] State of the auth. https://duo.com/assets/ebooks/state-of-the-auth-2019.pdf.

[17] Bradley Reaves, Luis Vargas, Nolen Scaife, Dave Tian, Logan Blue, Patrick Traynor, and Kevin R. B. Butler. Characterizing the security of the sms ecosystem with public gateways. *ACM Transactions on Privacy and Security*, 22(1), December 2018.

[18] Neetesh Saxena and Narendra S. Chaudhari. A secure approach for sms in gsm network. In *Proceedings of the CUBE International Information Technology Conference*, page 59–64, 2012.

[19] Fenghao Xu, Wenrui Diao, Zhou Li, Jiongyi Chen, and Kehuan Zhang. Badbluetooth: Breaking android security mechanisms via malicious bluetooth peripherals. In *Proceedings of NDSS*, 2019.

[20] Idc report: Xiaomi tops apple and fitbit with 21.5share in q3 2018. https://www.wearable-technologies.com/2018/12/idc-report-xiaomi-tops\-apple-and-fitbit-with-21-5/global-wearable-market-share-in-q3-2018/.

[21] Key factors that determine the range of bluetooth. https://www.bluetooth.com/blog/3-key-factors-that-determinethe-range-of-bluetooth/.

[22] Junjie Yin, Zheng Yang, Hao Cao, Tongtong Liu, Zimu Zhou, and Chenshu Wu. A survey on bluetooth 5.0 and mesh: New milestones of iot. *ACM Transactions on Sensor Networks*, 15(3), May 2019.

[23] Wahhab Albazrqaoe, Jun Huang, and Guoliang Xing. Practical bluetooth traffic sniffing: Systems and privacy implications. In *Proceedings of ACM MobiSys*, pages 333–345, 2016.

[24] Mike Ryan et al. Bluetooth: With low energy comes low security. *WOOT*, 13:4–4, 2013.

[25] Ryan. Crackle - cracking bluetooth smart encryption. http://lacklustre.net/projects/crackle/.

[26] Harry O'Sullivan. Security vulnerabilities of bluetooth low energy technology (ble). *Tufts University*, 2015.

[27] Angela Lonzetta, Peter Cope, Joseph Campbell, Bassam Mohd, and Thaier Hayajneh. Security vulnerabilities in bluetooth technology as used in iot. *Journal of Sensor and Actuator Networks*, 7(3):28, 2018.

[28] Ashwath Anand Pammi. *Threats, Countermeasures, and Research Trends for BLE-Based IoT Devices.* PhD thesis, Arizona State University, 2017.

[29] Hui Jun Tay, Jiaqi Tan, and Priya Narasimhan. A survey of security vulnerabilities in bluetooth low energy beacons. *Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-16-109*, 2016.

[30] Wondimu K Zegeye. Exploiting bluetooth low energy pairing vulnerability in telemedicine. In *International Telemetering Conference Proceedings*, 2015.

[31] Yanzhen Qu and Philip Chan. Assessing vulnerabilities in bluetooth low energy (ble) wireless network based iot systems. In *Proceedings of IEEE 2nd International Conference on Big Data Security on Cloud*, pages 42–48. IEEE, 2016.

[32] Kasper B. Rasmussen Daniele Antonioli, Nils Ole Tippenhauer. The knob is broken: Exploiting low entropy in the encryption key negotiation of bluetooth br/edr. In *Proceedings of USENIX Security Symposium*, pages 1047–1061, 2019.

[33] Keijo MJ Haataja and Konstantin Hypponen. Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures. In *Proceedings of IEEE ISCCSP*, pages 1096–1102, 2008.

[34] Konstantin Hypponen and Keijo MJ Haataja. "nino" man-in-the-middle attack on bluetooth secure simple pairing. In *The 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pages 1–5. IEEE, 2007.

[35] Great Scott Gadgets. Ubertooth one. https://greatscottgadgets.com/ubertoothone/.

[36] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers. In *Proceedings of the ACM International Workshop on HotMobile*, pages 99–104, 2016.

[37] Sheng Shen, He Wang, and Romit Roy Choudhury. I am a smartwatch and i can track my user's arm. In *Proceedings of ACM Mobisys*, pages 85–96. ACM, 2016.

[38] Chao Xu, Parth H Pathak, and Prasant Mohapatra. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the ACM International Workshop on HotMobile*, pages 9–14, 2015.