Blacktooth: Breaking through the Defense of Bluetooth in Silence

Mingrui Ai
University of Science and Technology of China, Hefei, Anhui, China
amr2016@mail.ustc.edu.cn

Kaiping Xue
University of Science and Technology of China, Hefei, Anhui, China
kpxue@ustc.edu.cn

Bo Luo
University of Kansas
Lawrence, KS, USA
bluo@ku.edu

Lutong Chen
University of Science and Technology of China, Hefei, Anhui, China
lutong98@mail.ustc.edu.cn

Nenghai Yu
University of Science and Technology of China, Hefei, Anhui, China
ynh@ustc.edu.cn

Qibin Sun
University of Science and Technology of China, Hefei, Anhui, China
qibinsun@ustc.edu.cn

Feng Wu
University of Science and Technology of China, Hefei, Anhui, China
fengwu@ustc.edu.cn

ABSTRACT

Bluetooth is a short-range wireless communication technology widely used by billions of personal computing, IoT, peripheral, and wearable devices. Bluetooth devices exchange commands and data, such as keyboard/mouse inputs, audio, and files, through a secure communication channel that is established through a pairing process. Due to the sensitivity of those commands and data, security mechanisms, such as encryption, authentication, and authorization, have been developed and adopted in the standards. Nevertheless, vulnerabilities continue to be discovered.

In the literature, few successful attacks against the Bluetooth connection establishment stage have been reported. Many attacks simply assume that connections are already established or use a compromised agent, e.g, a malicious app or a careless user, to initialize the connection. We argue that such assumptions are strong and impractical. A stealthily established connection is a critical starting point for any practical attack against Bluetooth devices. In this paper, we demonstrate that the Bluetooth Specification contains a series of vulnerabilities that will enable an attacker to impersonate a Bluetooth device and successfully establish a connection with a victim device. The entire process does not require any involvement of the device owner/user or any malicious app on the victim device. The attacker could further escalate permissions by switching Bluetooth profiles to retrieve sensitive information from the victim device and inject arbitrary commands. We name our new attack as the **Blacktooth Attack**. To demonstrate the effectiveness and practicality of the Blacktooth attack, we evaluate it against 21 different Bluetooth devices with diverse manufacturers and operating systems, and all major Bluetooth versions. We show that the newly proposed attack is successful on all victim devices.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3560668

CCS CONCEPTS

• **Security and privacy → Mobile and wireless security**.

KEYWORDS

Blutooth security; spoofing; permission escalation

ACM Reference Format:
Mingrui Ai, Kaiping Xue, Bo Luo, Lutong Chen, Nenghai Yu, Qibin Sun, and Feng Wu. 2022. Blacktooth: Breaking through the Defense of Bluetooth in Silence. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS '22), November 7–11, 2022, Los Angeles, CA, USA*. ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3548606.3560668

1 INTRODUCTION

The Bluetooth technology has been widely deployed in billions of electronic devices for low-power, short-range wireless communications [11]. The Bluetooth specifications are standardized by the Bluetooth Special Interest Group (SIG), which currently maintains two partly incompatible variants of Bluetooth protocols: Bluetooth BR/EDR and Bluetooth Low Energy (BLE). Since Bluetooth is designed for and well-adopted by personal computing and mobile/wearable devices, a large amount of sensitive information is being transmitted over Bluetooth connections. Hence, significant efforts have been devoted to Bluetooth security. However, vulnerabilities continue to be discovered [17, 23, 34]. In particular, protocol vulnerabilities were exploited by several high-profile attacks, which eventually led to substantial revisions of the specifications [7, 22, 30]. Recently, a wide spectrum of attacks have been discovered in different stages of the Bluetooth stack [8, 32], e.g., Antonioli *et al.* presented the BIAS attack [5] to spoof Bluetooth authentication and the KNOB attack [4] to downgrade the Bluetooth session key entropy. Xu *et al.* [36] discovered that a Bluetooth device is still trusted by a paired Android phone even if its profile has changed.

However, to the best of our knowledge, there does not exist an effective attack that autonomously triggers a Bluetooth connection

without interacting with the user. In practice, connection establishment is the first step of many attacks that target the later stages of Bluetooth communications. In the literature, attacks often assume a pre-established connection, e.g., [3, 21, 31], or set up the connection with a user misoperation or a pre-installed malicious application on the victim device, e.g., [33, 36]. Such assumptions could be too strong in real-world settings. While some existing works focus on other stages of the connection establishment process, they neglect the initial connection trigger stage [6, 21]. We argue that an effective attack that is capable of stealthily establishing a Bluetooth connection with the victim device is a significant threat to billions of Bluetooth users, because such an attack would make all the subsequent attacks more practical and dangerous.

In this paper, we present **Blacktooth**, an autonomous and stealthy attack against the connection establishment phase of Bluetooth communications. We first demonstrate that the Bluetooth Specification contains a series of vulnerabilities, which can be exploited by the attacker to (1) *impersonate* a peripheral Bluetooth device and establish a connection with a victim device without prompting the device owner for confirmation, (2) *bypass authentication* by downgrading to unilateral Legacy Authentication and only "authenticating" the victim device, (3) *break encryption* by forcing the victim device to use a short encryption key and brute-forcing the key, and (4) *escalate privileges* by switching Bluetooth profiles, and retrieve sensitive information from the victim device or inject arbitrary commands. In this attack, the attacker (or attacking device) does not need to be present when the impersonated device (e.g., a benign Bluetooth headset) and the victim device (e.g., a smartphone) are paring, does not need to observe any previous communication between the impersonated device and the victim device, does not know the link key of the connection, and does not need to have any pre-shared secret between them. The proposed attack is completely stealthy such that no notification or prompt dialog would pop up on the victim device during the whole attack process.

In a proof-of-concept implementation, we demonstrate that the Blacktooth attack is effective against commodity Bluetooth devices. We test smartphones and tablets with different Bluetooth chips and versions, manufactured by the major vendors, including Apple, Google, Samsung, Lenovo, HUAWEI, HONOR, OnePlus, OPPO, Xiaomi, etc. The attacks are successful on all the tested devices.

We summarize our main contributions as follows:

- We articulate a series of vulnerabilities in the Bluetooth Specification, including novel vulnerabilities. We demonstrate that a malicious device can initiate a connection due to the unfixed master-slave roles in a piconet. Combining the new vulnerability with two known vulnerabilities, we are able to establish a malicious connection with the victim device in an utterly stealthy manner. This attack also makes many previously reported attacks practical, since it eliminates the main obstacle in the initial connection establishment stage.
- We further demonstrate that the attacker can exploit two more vulnerabilities in the Bluetooth profile authentication process to escalate the connection to sensitive profiles. We show that the newly proposed attack is completely stealthy, i.e., no alert or notice is presented to the device owner and no user confirmation is needed to launch the attack. The attack is also very practical that it does not assume any prior knowledge or require any pre-installed malicious agent on the victim devices.
- To demonstrate the practicality of the proposed Blacktooth attack against real-world devices, we successfully launch the attack on 21 different Bluetooth devices. Our device samples include a wide range of smartphone and PC manufacturers, Bluetooth chips, operating systems, and all major Bluetooth versions. Finally, we discuss possible defenses against the new attack.

Ethical Considerations and Responsible Disclosure. The objective of this research is to investigate unknown vulnerabilities in a widely adopted technology. The research may potentially benefit billions of end-users. All the experiments presented in this paper were conducted in research labs. We have never attacked any real-world device outside of the lab. In April 2022, we disclosed the technical details of the Blacktooth attack and potential countermeasures to the manufacturers of the tested devices. We also disclose our attacks to the Bluetooth SIG. As of the publication of this manuscript, we have received responses from several companies, who acknowledged our findings, successfully replicated the attack, and/or planned to further explore this issue. We are in close collaboration with some manufacturers in developing effective patches. We have not disclosed the attack to anyone other than the device manufacturers and the Bluetooth SIG.

The rest of the paper is organized as follows: in Section 2, we briefly introduce Bluetooth BR/EDR. In Section 3, we present the system and adversary model, followed by a series of vulnerabilities to be exploited. We present the details of the Blacktooth attack in Section 4, followed by the implementation and evaluation results in Section 5. We further discuss several practical issues of the attack and propose potential defenses in Section 6, review the literature in Section 7, and finally conclude the paper in Section 8.

2 BACKGROUND

2.1 Bluetooth BR/EDR Overview

Bluetooth Basic Rate/Extended Data Rate (Bluetooth BR/EDR), also known as Bluetooth Classic, is a low-power wireless technology widely used in short-range communications. Bluetooth BR/EDR uses the ISM band at 2.4GHz [10]. One device, called the Master, provides the reference clock signal, and all other devices (no more than seven), called the Slave(s), are synchronized to the clock. All synchronized devices build a network called *piconet*. Every Bluetooth device has a 48-bit Bluetooth device address (BD_ADDR).

The Bluetooth system consists of the Bluetooth core stack and the applications. The Bluetooth core stack contains the Physical, Logical, and Logical Link Control Adaptive Protocol (L2CAP) layers, and some protocols to support applications. The Bluetooth devices implement the physical and logical layers in Bluetooth chips, known as the *Bluetooth Controller*, and implement the L2CAP layer and the application-orientated protocols in the device operating system (OS), known as the *Bluetooth Host*. The Bluetooth Host communicates with the Controller through the Host Controller Interface (HCI). The Bluetooth devices broadcast the supported services and the associated parameters to other devices using Service Discovery Protocol (*SDP*). The application layer implements the Bluetooth service interfaces and functions offered to users in userspace.

2.2 Bluetooth Connection and Pairing

To establish a secure connection between two Bluetooth devices, the Master first queries the Slave to get the device name and features. Then, the Master sends a Connection Request to initiate connection establishment. Note that the device roles are not fixed–the specification only defines the Master as the device that initiates a connection. That is, any arbitrary Bluetooth BR/EDR device can initiate a connection and become the Master regardless of its functionality or previous role. For example, Bluetooth headsets, which are usually slave devices, can also take master roles and proactively initiate connections with smartphones. The devices can switch the Master-Slave roles after establishing a piconet.

The Bluetooth Specification provides link layer mechanisms to authenticate devices to avoid adversarial connections. *Pairing* is to negotiate a link key between two Bluetooth devices with the Link Manager Protocol (LMP). Secure Simple Pairing (SSP) is the most popular pairing mechanism, which uses Elliptic Curve Diffie-Hellman (ECDH) for key agreement. User confirmation is *required* when a new device attemps to pair with the user's smartphone. When the are paired, two Bluetooth devices share the common *link key* to be used for future authenticates. Once authentication is successful, the devices derive an *encryption key* from the link key and public parameters. The entropy of an encryption key is in the range of 1 to 16 bytes according to the key length negotiation procedure. Note that the procedures mentioned above are neither integrity protected nor encrypted. If both devices support Secure Connections, Secure Authentication is used for authentication, and AES CCM is used for encryption. Otherwise, Legacy Authentication and E_0 stream cipher are used for authentication and encryption.

2.3 The Bluetooth Profile

Different Bluetooth devices produced by different manufacturers may provide the same functionalities; hence, protocols, known as Bluetooth profiles, are developed to regulate all kinds of communications between the devices. The Bluetooth profiles define the necessary functions in the Bluetooth core stack and other required settings. As of now, the Bluetooth SIG has specified about 40 profiles for the Bluetooth BR/EDR. There are some commonly used profiles in Bluetooth devices, like the Advanced Audio Distribution Profile (A2DP) used for audio communication between Bluetooth headsets and smartphones, the Human Interface Device Profile (HID) used to send keystrokes from keyboards to smartphones, and the Phone Book Access Profile (PBAP) used to get contacts. A Bluetooth device can implement multiple profiles simultaneously and indicate all available services to other devices using SDP.

3 SECURITY ANALYSIS OF BLUETOOTH BR/EDR

The existing Bluetooth security mechanisms are deployed in the Bluetooth Controller and Host. The Controller authenticates the remote device in pairing and ensures communication confidentiality through encryption. The Host identifies the Class of Device (CoD) and restricts the capabilities of a remote device. These mechanisms work under the assumption that they are well-coordinated. However, we find that this assumption is unreliable, especially, the security mechanisms do not fully understand the behaviors and threats from their counterparts. An adversary can take advantage of such fundamental design flaws in each mechanism, break through the defense step-by-step and finally control the target device.

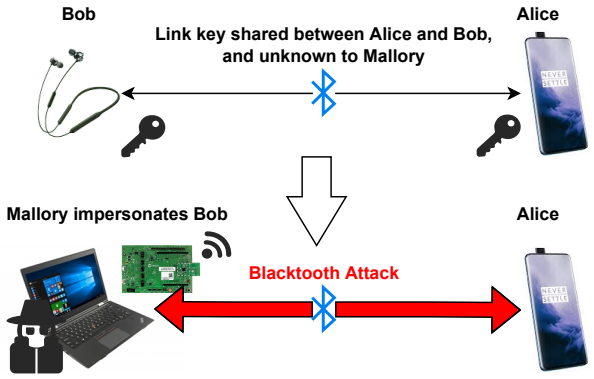

Figure 1: Blacktooth attack overview: Alice and Bob share a link key that is unknown to Mallory. Mallory's goal is to establish a Bluetooth connection with Alice and get the privilege through sensitive profiles while impersonating Bob. The attack is utterly stealthy without the victim's interactions.

3.1 The System Model

We consider two victim devices, Alice and Bob, who securely communicate with each other over Bluetooth BR/EDR (Figure 1). Since both victims are not required to be present at the time of the attack, we only assume that two legitimate devices exist and have communicated in the past. That is, Alice and Bob have already shared and stored the link key, which is a long-term key that is unknown to the attacker. The key has been established through Bluetooth's Secure Simple Pairing (either using Legacy Authentication or Secure Authentication). These standardized security mechanisms are expected to protect Bluetooth communication against eavesdropping, impersonation, and man-in-the-middle attacks.

Without loss of generality, we assume that Alice is the Bluetooth Master, e.g., a smartphone, and Bob is the Bluetooth Slave, e.g., Bluetooth earbuds. Alice can initiate and establish a secure connection with Bob using the existing link key, while Bob is willing to accept a connection from Alice using this key. We assume all the security primitives in use, such as FIPS and AES, are perfectly secure.

3.2 Attacker Model

The attacker, Mallory, aims to impersonate Bob to establish a secure Bluetooth connection with Alice and control Alice through advanced permissions (such as keyboard input) obtained from sensitive profiles (such as the HID profile). In Section 4, we present the technical details of all the steps.

Mallory must be physically in the victim device's Bluetooth connection range. Mallory does not observe the secure pairing process between Alice and Bob, nor does it know the link key. Mallory is capable of eavesdropping, decoding, and manipulating unencrypted packets, and jamming the Bluetooth spectrum. Mallory knows the public information about Alice and Bob, such as Bluetooth names and addresses, protocol versions, capabilities, etc. Since connection establishment is not encrypted, Mallory can gather the exchanged

attributes by eavesdropping the communication. After a secure connection is established between Alice and Bob, Mallory could force them to disconnect by jamming the Bluetooth spectrum. Mallory could attempt to re-establish a secure connection with Alice.

3.3 Security Vulnerabilities in Bluetooth BR/EDR

Here we articulate 5 vulnerabilities in the connection/pairing process described in Section 2.2. In particular, Vulnerability #1 is newly discovered in this research, while Vulnerability #4 was previously reported on Android but we confirmed it on iOS, iPadOS, macOS, HarmonyOS, and the latest version of Android as well.

Vulnerability #1 (new): Unfixed Roles (Inquiry). Alice and Bob discover each other in the inquiry procedure and then connect over Bluetooth BR/EDR. Alice (Master) is expected to send a Connection Request to Bob to start establishing a connection. This process usually requires a user confirmation on the Master (Alice) side to avoid misconnection to unintended (malicious) devices. However, Bluetooth BR/EDR does not bind Alice and Bob to the Master-Slave roles; hence, either device can switch to the other role. In practice, the paging device always becomes the Master, even if the device takes the functionality of a traditional Slave device (e.g., Bluetooth headset). The role change can be performed anytime after the connection is complete. That is, Mallory could first impersonate Bob, who has paired with Alice before. Mallory then makes full use of the Master definition rule and the role switching mechanism to send a Connection Request to Alice. Since Mallory (impersonating Bob) now serves as the Master and Alice becomes the Slave, user confirmation is NOT prompted on Alice.

Please note that some commodity Bluetooth devices on the market, especially some headsets, utilize this vulnerability as a feature to implement the "automatic reconnect" function. However, we demonstrate that this design introduces a serious security vulnerability in the *Bluetooth Specification*, which causes severe consequences when it is exploited in an attack. The Master/Slave roles are symmetric in most link layer communications, so that the roles are considered switchable from a communication perspective, which is the current practice in the Bluetooth standard and in the products. However, the two roles are *not* symmetric in the security design. In particular, the authentication and handshake process is dominated by the Master. Moreover, the protocol does not bind the device roles with the upper layer functions, while the functions of the paired devices are also *not symmetric*. Therefore, role switching introduces a serious security vulnerability in the Bluetooth standard. We consider the vulnerability of unfixed roles as an amplifier of other (known or unknown) threats, since it may enable other attacks to be launched silently. We are concerned that new vulnerabilities may be discovered in future research that can exploit the unfixed roles vulnerability to perform a complicated and silent attack.

Vulnerability #2: Unilateral Legacy Authentication (Connection) [5]. During connection, Alice and Bob use the shared link key for authentication. Secure Authentication (SA) is used only if both devices support Secure Connections. Meanwhile, the Specification only requires unilateral authentication in Legacy Authentication (LA), i.e., mutual authentication is not required in LA. Antonioli *et al.* [5] confirm that only the Master authenticates the Slave during LA. Therefore, the attacker without the link key can impersonate the Master and complete secure connection establishment without authenticating the Slave. Even if both devices support Secure Connections, the attacker can downgrade SA to LA by declaring that the impersonated device does not support Secure Connections.

Vulnerability #3: Potential Low Encryption Key Entropy (Connection) [4]. During Connection, Alice and Bob compute an encryption key K_C and negotiate the entropy of the encryption key, which is in the range of 1 byte to 16 bytes. Since the encryption key K_C is derived from the link key, Mallory cannot compute the encryption key directly. However, K_C is not directly used to encrypt messages. The actual encryption key K'_C is computed by reducing the entropy of K_C to N bytes. N is the result of the Bluetooth encryption key negotiation protocol. Thus, the attacker can impersonate one device and force the other one to accept 1 byte of entropy and then attempt to brute force the encryption key K'_C. It is easy to exhaust all the possibilities since the key length of K_C is 8 bits. The attacker can use well-known Bluetooth fields (e.g., L2CAP headers) as oracles to launch a known-plaintext attack on K'_C.

Vulnerability #4: Inconsistent Profile Authentication Procedure (Communication) When a profile change is requested during communication, Alice is expected to take additional security validations before connecting to Bob's new profile and allowing the new functionality. However, what Alice should do to verify the profiles is *unclear*. Bluetooth Specification does not give any exercisable suggestions about profile authentication. Bob is trusted to make arbitrary profile changes as long as Alice and Bob have paired. Even worse, no alert appears on Alice's screen to warn her about the profile changes, nor does a confirmation dialog pop up to request permission from Alice. As such, if Mallory successfully impersonates Bob and establishes a secure connection with Alice, Mallory can use new (malicious) profiles that are inconsistent with those used by Bob. For instance, Mallory can impersonate Bob, which used to be a headset, and enable a Human Interface Device Profile (HID)–a profile for keyboard and mouse.

The whole process described above is silent. On Android devices, Alice can only notice the profile and permission changes if she checks the details of Bluetooth devices in phone settings [36]. Moreover, Alice is less likely to notice the changes on Apple devices, since iOS does not always display the device information (e.g., device type), and hides most of the permissions from the user.

Vulnerability #5: Default Profile Services Authorization (Communication) During communication, the Host accepts arbitrary profile requests from the Bluetooth device. By default, all the permissions associated with the requested profile are granted, even when the requested profile is inconsistent with the capabilities of the actual device. For example, when a headset adds the HID (Human Interface Device) profile, the victim phone will accept the request and grant permissions without prompting the user for authorization. Therefore, as long as Mallory impersonates Bob and connects to Alice, Mallory can practically obtain any arbitrary permission without user consent. We consider this a severe vulnerability that allows privilege escalation. Even worse, in the Android family of operating systems, the new profile's status is shown as "Denied" in System Settings, while the new profile has indeed obtained the permissions. In iOS, the profile statuses are invisible.

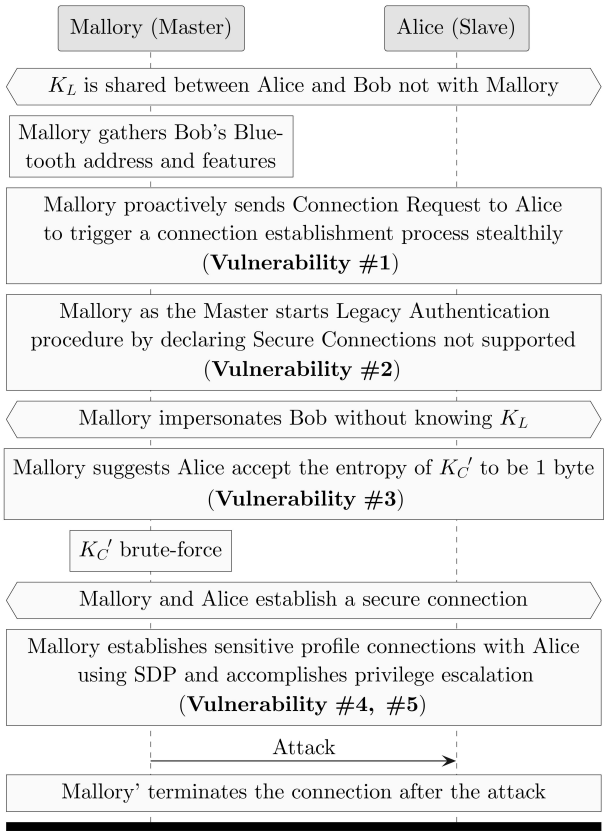

Figure 2: The attack process of the Blacktooth attack.

4 THE BLACKTOOTH ATTACK

In this section, we present the technical details of the Blacktooth attack, which exploits all five vulnerabilities introduced in Section 3.3. As shown in Figure 2, the attack consists of the following steps: **(1)** Mallory (the attacker) impersonates Bob (the Slave) by changing its Bluetooth address to Bob's device address (BD_ADDR$_B$). Mallory further proactively sends a Connection Request to Alice in order to trigger the connection process over Bluetooth BR/EDR and force Alice to the Slave role, so that the connection will not be noticed by the user (**Vulnerability #1**). **(2)** Mallory establishes a secure connection with Alice using Legacy Authentication, so that Alice (the Slave) does not authenticate Mallory (the Master) (**Vulnerability #2**). **(3)** Mallory reduces the entropy of the encryption key to 1 byte. Without knowing any pre-shared secret including the link key (K_L), Mallory launches a brute force attack to obtain the encryption key K_C'. (**Vulnerability #3**). **(4)** Finally, Mallory connects to Alice's sensitive profiles, injects malicious commands (using the HID profile), and retrieves Alice's sensitive data (using the PBAP profile) (**Vulnerability #4, #5**). As a result, Mallory impersonates Bob and escalates his privileges to obtain full control of Alice. Mallory can send arbitrary keystrokes, click arbitrary buttons to change settings, take and retrieve screenshots, read the phone book and messages, and even power off the smartphone. The whole attack procedure is totally silent and does not need any user's operation.

4.1 Identity Forging and Proactive Connection Request (Vulnerability #1)

The first step for every Bluetooth BR/EDR connection is the inquiry procedure. Conventionally, Alice (the Master) pages Bob (the Slave) to get the remote device name and features before establishing a connection. The device owner can modify the device name, which means the attacker can change its device name to Bob's (e.g., Steve's Headset). In this way, even if the owner notices that his phone is connected to a Bluetooth device, he will not observe any anomalies, since the device name is familiar to him. At the same time, Mallory has to change its Bluetooth address to Bob's (BD_ADDR$_B$). Alice (the smartphone) then takes Mallory as Bob, since Bluetooth devices are identified by Bluetooth addresses. It is trivial for Mallory to obtain Bob's device information by eavesdropping on the connection establishment procedure (unencrypted) between Bob and any device, or exploiting the inquiry procedure.

Next, Mallory expects to trigger the connection process with Alice. It is the most critical step in the entire attack process where Mallory needs to stay stealthy and avoid alerting the user. In particular, previous researches fall short in autonomously and silently establishing a connection between Alice and Mallory. For instance, Xu *et al.* [36] requires a malicious application installed in the victim phone to send the Connection Request to the malicious Bluetooth device, while many other attacks neglect this process by assuming a careless user or an already-established connection. None of the existing attacks that we are aware of can silently establish a Bluetooth connection with the target device without exploiting a malicious agent or requiring confirmation from the user.

To make the attack practical in the real world, we take advantage of a flaw in the protocol design of device roles. In particular, the Bluetooth Specification defines that the device that initiates a connection is the Master and Master-Slave roles may change after a piconet has been established. That means the roles of Alice and Bob are not static, and Bob can initiate a connection proactively as the Master before switching his role to the Slave. While this mechanism is utilized by Bluetooth headsets to automatically reconnect to the phones they are paired with, it has not caught the attention of the security community. Mallory can impersonate Bob and send a Connection Request to Alice in order to trigger the connection establishment process without any operation by Alice. The connection initiation phase is completely silent without triggering any notification or confirmation or requiring any manual intervention. Furthermore, the connection initiation phase cannot be restricted by Alice unless the user turns off Bluetooth manually, since the definition of the Master and Slave is the basic concept, and the role switch procedure is a basic design of Bluetooth BR/EDR.

4.2 Authentication Spoofing (Vulnerability #2)

After sending the Connection Request, connection establishment moves into the authentication procedure. The Bluetooth Specification defines two authentication methods: Legacy Authentication and Secure Authentication. Note that Secure Authentication is used only if both devices support Secure Connections, which is first proposed in Bluetooth 4.1.

The Legacy Authentication process is illustrated in Figure 8. It works as follows: the verifier generates a random number called

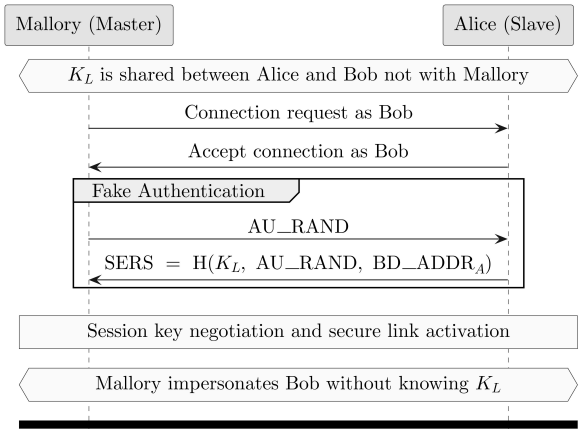

Figure 3: Authentication spoofing on Bluetooth Legacy Authentication process. Mallory establishes a connection with Alice while impersonating Bob. Mallory sends AU_RAND to Alice and receives SERS from Alice. Since the Bluetooth Specification does not require mutual authentication, Alice does not have to verify that Mallory knows K_L.

AU_RAND and sends it to the claimant. The claimant computes the response SERS = H (K_L, AU_RAND, BD_ADDR$_B$) and sends it back to the verifier. Then the verifier computes the SERS' using the same hash function and the same parameters to compare with SERS. The verifier believes the claimant has the same link key K_L if the value of SERS and SERS' are equal.

According to Bluetooth Specification, in Legacy Authentication, the verifier does not have to be the Master and the application defines which device has to be authenticated. In practice, the verifier is almost always the Master and the claimant is the Slave. Moreover, the Legacy Authentication procedure only provides unilateral authentication. To achieve mutual authentication, the Master and the Slave must run the Legacy Authentication procedure respectively. The Bluetooth Specification does not require mutual authentication, and in practice, few devices require mutual authentication. Antonioli *et al.* [5] confirms that only the Master authenticates the Slave unilaterally during secure connection establishment. Therefore, Mallory can establish a secure connection with Alice without being authenticated by Alice if Mallory keeps its role as the Master.

The authentication spoofing process for Legacy Authentication is illustrated in Figure 3. Alice accepts the Connection Request from Mallory and treats it as Bob. Mallory then sends AU_RAND to Alice, and Alice computes SERS using the tuple AU_RAND, BD_ADDR$_A$ and the link key K_L and sends the result to Mallory. In this way, Mallory can perform the rest of the authentication steps (e.g., key negotiation and secure link activation) as Bob without having to prove to Alice that it knows K_L.

Secure Authentication is only used to provide mutual authentication in Secure Connections. The vulnerability that the Master unilaterally authenticates the Slave no longer exists in Secure Authentication. However, we can manage to downgrade Secure Authentication to Legacy Authentication to launch the spoofing attack. The Bluetooth Specification does not require two devices that were paired using Secure Connections to keep using Secure Authentication subsequently. That is, Alice and Bob can use Legacy

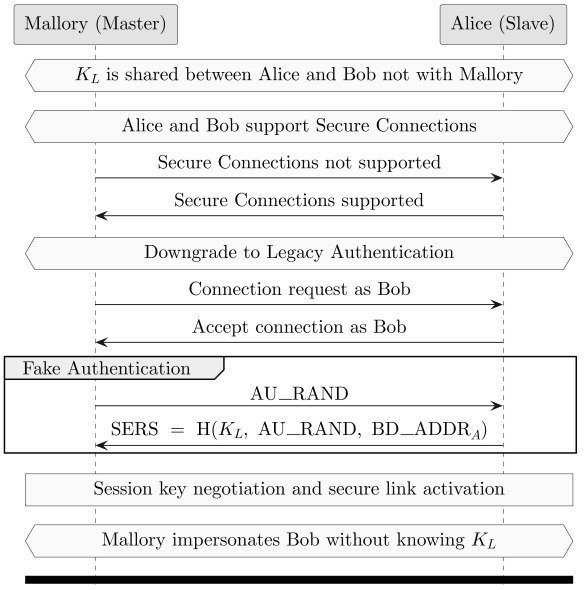

Figure 4: Authentication spoofing on Bluetooth Secure Authentication process. During feature exchange, Mallory (impersonating Bob) declares that Secure Connections is not supported, and Alice declares that Secure Connections is supported. Secure Authentication is downgraded to Legacy Authentication. Mallory sends AU_RAND to Alice and receives SERS from Alice. Mallory establishes a connection with Alice without knowing the link key.

Authentication for connection establishment even though they were paired using Secure Connections. Mallory can take advantage of this vulnerability to downgrade Secure Authentication.

Assuming that Alice and Bob have previously paired using Secure Authentication, the new spoofing attack process is shown in Figure 4. Mallory needs to declare to Alice that it does not support Secure Connections in order to downgrade the authentication procedure to Legacy Authentication. As a result, Mallory acts as the Master and "authenticates" Alice using unilateral Legacy Authentication, while Alice does not authenticate Mallory.

4.3 Encryption Key Negotiation and Brute Force (Vulnerability #3)

Once the authentication is successful, the encryption procedure starts. The Bluetooth Specification defines two key generation methods for Bluetooth BR/EDR, for Secure Connections and approaches prior to Secure Connections, respectively. Since Secure Connections has already been downgraded in the authentication phase, our attack never uses the encryption key generation method for Secure Connections. We focus on the encryption key generation method for prior to Secure Connections in the rest of this section.

In encryption, a key K_C is generated from tuple EN_RAND, AU_RAND, BD_ADDR$_A$ and K_L. K_L is the link key shared between Alice and Bob, and the other parameters are known to the public. The entropy of K_C is always 16 bytes if the key generator is implemented according to the Bluetooth Specification. Note that K_C

cannot be used to encrypt messages directly. A key shortening function is used to compute the actual encryption key K_C' by reducing the entropy of K_C to N bytes, where $N \in [1, 16]$ is negotiated by the two devices. The entropy of the encryption key is irrelevant to the Bluetooth security level–the entropy could be 1 byte even if the connection is at the highest security level.

The key negotiation process is implemented in the Bluetooth Controller, and it is transparent to the Bluetooth Host and Bluetooth applications. Therefore, Mallory (the attacker) can persuade Alice to accept 1 byte of entropy for the encryption key K_C' without being noticed by the victim. In the following, we describe how Mallory and Alice negotiate N equal to 1 in detail (as shown in Figure 5). The Bluetooth Specification defines two parameters L_{max} and L_{min}, where $1 \leq L_{min} \leq L_{max} \leq 16$, to declare the range of the allowed key length. After the two devices agree to initiate Bluetooth link layer encryption, the Master (now still Mallory) will send a suggested value, L_{sug}, to the Slave (now Alice). L_{sug} is supposed to equal to L_{max} initially. Hence, if Mallory wants to negotiate N equal to 1, it has to set $L_{max} = L_{min} = 1$. Then Alice will compare L_{sug} and its own L_{min}. If $L_{min} \leq L_{sug}$, which means Alice supports the length, Alice will take the value as the length of the encryption key K_C' and the negotiation is successful.

The success of the negotiation process described above relies on Alice's $L_{min} = 1$. There is a theoretical possibility that Alice sets L_{min} larger than 1 and does not accept Mallory's suggestion, which means Mallory cannot convince Alice to accept the entropy of the encryption key as low as 1 byte. Indeed, if Alice sets the value of L_{min} properly, it could prevent the attack. However, Antonioli *et al.* [4] found that L_{min} and L_{max} are stored in the firmware of the Bluetooth Controller (chip) and are fixed generally. The Bluetooth Host and the Bluetooth application cannot check and set the value of L_{min} and L_{max} nor decide whether to accept the suggestion. Even worse, Antonioli *et al.* [4] found that most devices on the market can accept 1-byte entropy keys. That is, it is unavoidable for Alice to accept the entropy of 1 byte in practice.

Since Alice and Mallory have already negotiated the entropy equal to 1 byte, it is easy for Mallory to enumerate all 256 possible K_C' and brute force it. It is convenient for Mallory to utilize some well-known Bluetooth packet fields (e.g., L2CAP headers) as oracles.

$$K_C'(x) = g_2^{(N)}(x)(K_C(x) \ mod \ g_1^{(N)}(x)) \qquad \text{(EQ 1)}$$

K_C' is computed based on the E_0 cipher using Equation (EQ 1), where N, an integer in the range of 1 to 16, is the negotiated encryption key length. $g_1^{(N)}(x)$ is an $8N$-degree polynomial used to reduce the entropy of K_C to N bytes. The reduction output is encoded with a block code $g_2^{(N)}(x)$, which is a polynomial of degree less or equal to $128 - 8N$. The polynomials depend on N, and they are defined in [10, p. 969]. When $N = 1$, $g_1^{(1)}(x)$ equals to 0x0000000000000000000000000000011d and $g_2^{(1)}(x)$ equals to 0x00e275a0abd218d4cf928b9bbf6cb08f.

After learning the actual encryption key, Mallory can utilize K_C' to send messages to Alice and decrypt packets from Alice.

4.4 Profile Change (Vulnerabilities #4, #5)

As of now, Mallory has successfully convinced Alice that it is Bob and established a secure connection. Mallory can take advantage

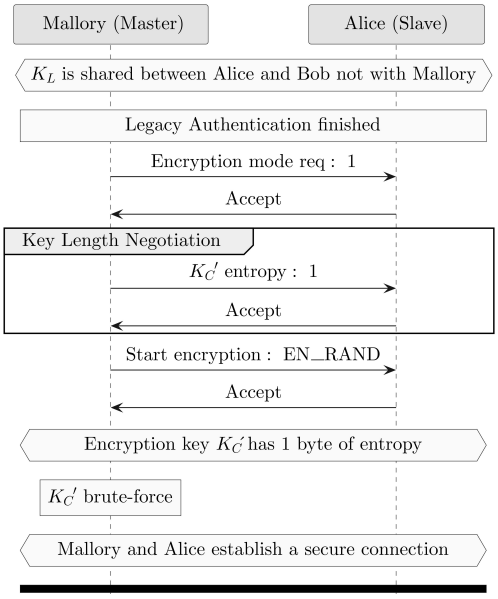

Figure 5: Mallory and Alice encryption key length negotiation. The attacker persuades Alice to accept 1 byte of entropy for the encryption key K_C'. Mallory does not need to know the link key or observe SSP between Alice and Bob. After negotiation, Mallory brute forces K_C' using some well-known Bluetooth packet fields as oracles.

Table 1: Blacktooth attack involved profiles

Profile name	Description	Usage
HID	Human Interface Device	Mouse, Keyboard
PBAP	Phone Book Access	Smartwatch
MAP	Message Access	Smartwatch
HFP	Hands-Free	Wireless Headset
A2DP	Advanced Audio Distribution	Wireless Headset
AVRCP	Audio/Video Remote Control	Wireless Headset
OPP	Object Push	File Transfer

of Alice's trust to perform the next step of the Blacktooth attack: privilege escalation through profile change. In practice, Mallory does not change Bob's profiles. Instead, it adds new profiles that it wants to utilize using SDP and asks Alice to accept them. As we have mentioned in vulnerabilities #4 and #5 in Section 3.3, Alice automatically accepts Mallory's requests to establish connections to these profiles and give corresponding permissions. Since the Bluetooth Specification is ambiguous on profile authorization, mobile operating systems do not properly implement this function–all the Android and iOS devices we have tested authorize the new profiles without asking for the user's confirmation for most profiles. The victim will not notice the service changes of the Bluetooth device unless he examines the Bluetooth settings. Table 1 shows some profiles in [1] that can be utilized to launch the Blacktooth attack.

Mallory can get permission to input by spoofing as a keyboard and/or a mouse and connecting the HID profile [9]. Mallory then controls Alice just like a real user using a keyboard and a mouse. For example, Mallory can open the camera to take photos (clicking

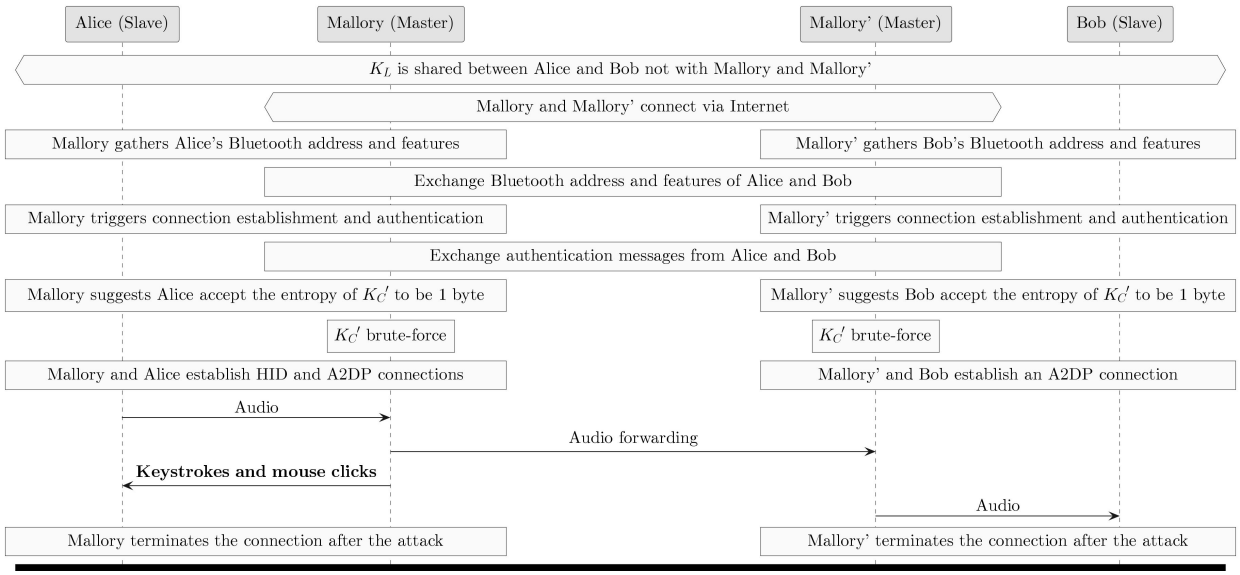

Figure 6: Stages of the MITM Blacktooth attack.

the Camera button), take a screenshot (sending a keystroke named KEY_SYSQR), text arbitrary messages (sending regular keystrokes), copy (KEY_CTRL+KEY_C), paste (KEY_CTRL+KEY_V), grant permissions (clicking the Setting button and opening permissions), and take any other operations using a keyboard and a mouse. After launching the attack, Mallory can delete all the messages, photos, and screenshots to destroy the evidence. Moreover, Mallory can steal sensitive information using PBAP and MAP profiles. PBAP is a profile used to access the contacts, and Mallory can use it to download the names and corresponding phone numbers stored in Alice. MAP is a profile used to access the message, and Mallory can take advantage of it to download messages. Mallory may be able to obtain the SMS verification code to log in to the victim's account.

Fortunately, the profile that grants permission to read arbitrary files from phone storage will require a manual confirmation from the user, as we have discovered in our experiments. Hence, Mallory cannot simply download files from the phone using the compromised Bluetooth connection. However, Mallory can still exploit the HID profile to control the apps on the phone and send out arbitrary files or images using SMS, email, or cloud storage.

Even if Mallory does not escalate its permissions, it can still cause serious damage. Assuming Bob is a headset, the most common Bluetooth BR/EDR device. Alice has to grant Phone audio and Media audio permissions to use the headset function. Since Mallory impersonates Bob, it obtains these permissions, so that it can make phone calls and record them, or inject voice commands to Alice using the HFP and A2DP profiles.

Finally, we would like to emphasize that the Blacktooth attack does not require any pre-installed (normal or malicious) agent or any user interaction. Mallory only exploits the vulnerabilities in Bluetooth Specification and the built-in functions of Alice's OS. More importantly, all of Mallory's operations are perfectly allowed by Bluetooth Specification. Therefore, regular security software is unlikely to identify or prevent this attack.

4.5 The MITM Blacktooth Attack

Though our Blacktooth attack itself is completely silent, the victim still has a chance to notice he is under attack. For example, if an attacker launches the Blacktooth attack when the victim wants to use his Bluetooth headset, the victim will find he cannot connect to his headset and have a chance to discover the attack.

In order to make the attack more concealed, we can modify the Blacktooth attack to a MITM version. The MITM Blacktooth attack is more harmful than the original Blacktooth attack. The MITM Blacktooth attack can be regarded as a combination of two basic Blacktooth attacks with some extra message forwarding. The MITM Blacktooth attack stages are shown in Figure 6.

In particular, Mallory consists of two Bluetooth devices. The first device impersonates Bob, just like the original Mallory in the Blacktooth attack, and the second device impersonates Alice. These two devices communicate via the Internet. The first device implements Bob's functions (e.g., A2DP sink) as well as the attack profiles (e.g., HID and PBAP), and the second device implements Alice's function (e.g., A2DP gateway). Assuming Alice and Bob are not connecting with each other, the two parts of Mallory impersonate Alice and Bob respectively and try to establish secure connections with them. When Alice authenticates Bob, the first device forwards the challenge to the second device via the Internet and the second one sends the challenge to Bob, then Mallory forwards Bob's response to Alice. In this way, the first device of Mallory successfully passes Alice's authentication and impersonates Bob. Mallory can also pass Bob's authentication and impersonate Alice in the same way. For the encryption key, Mallory can take advantage of the KNOB attack [4] to make Alice and Bob negotiate an encryption key with 1 byte of entropy and try to brute force it. Therefore, Mallory can eavesdrop and forward all messages between Alice and Bob. When the victim is not using Alice and Bob, Mallory can launch the Blacktooth attack to inject commands or steal data. If the victim happens

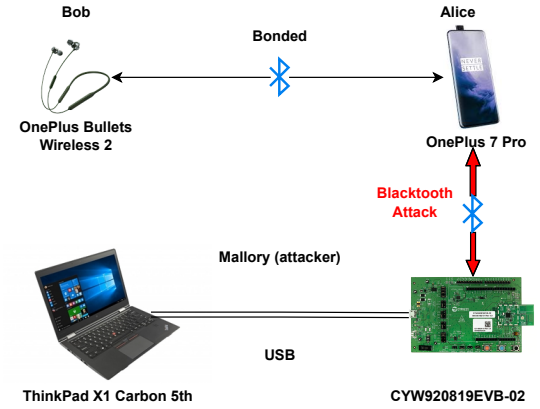

Figure 7: Blacktooth attack scenario. Alice is a OnePlus 7 Pro smartphone, Bob is a OnePlus Bullets Wireless 2 headset and Mallory (attacker) is a CYW920819EVB-02 development board connected via USB to a ThinkPad X1 Carbon laptop. Alice and Bob have paired in the absence of Mallory.

to use the device, e.g., to listen to music, Mallory can forward data between Alice and Bob to ensure normal operations.

In summary, the new *MITM Blacktooth* attack improves the stealthiness of the original Blacktooth attack by allowing the victim device (Alice) to perform its regular operations with the impersonated device (Bob). More importantly, Mallory can further eavesdrop on the communication between Alice and Bob, which may lead to more security and privacy concerns, e.g., when the user is making a phone call using the compromised Bluetooth headset (Bob), Mallory eavesdrops on the content of the call.

5 IMPLEMENTATION AND EVALUATION

In this section, we present our in-lab implementation of the Blacktooth attack. We then evaluate the attack on a wide range of commodity devices and demonstrate the effectiveness of the attack.

5.1 The Blacktooth Attack Scenario

Our attack scenario (Figure 7) includes a victim device–a OnePlus 7 Pro smartphone (Alice, Master in legitimate secure connections), an impersonated device, and a OnePlus Bullets Wireless 2 headset (Bob, Slave in legitimate secure connections). Mallory is a 5th-generation ThinkPad X1 Carbon laptop together with a CYW920819EVB-02 evaluation board. The ThinkPad X1 laptop is also used to brute force the encryption key K'_C. We present the relevant technical specifications of Alice and Bob in Table 2. As we mentioned in Section 4, Alice has already paired with Bob, and they are sharing a link key K_L which is unknown to Mallory.

5.2 The Blacktooth Attack Device

Mallory uses a CYW920819EVB-02 evaluation board [13] as the Bluetooth Controller connected to a ThinkPad X1 laptop running a Linux 4.14.111-based OS. Note that we have to modify both the Bluetooth Controller and Bluetooth Host according to our needs before we can use them to launch the attack.

We use several open-source tools to modify the Bluetooth Controller to patch the Bluetooth chip firmware to obtain the function

Table 2: Relevant Bluetooth attributes for Alice and Bob

	Alice	Bob
Bluetooth Name	OnePlus 7 Pro	OnePlus Bullets Wireless 2
Operating System	H_2OS 9.5.9 (Android 9.0)	-
Chipset	Snapdragon 855	QCC3034
Chip vendor	Qualcomm	Qualcomm
LMP Version	5.0	5.0
LMP Subversion	702	13942
Class of Device	`0x0c025a`	`0x180424`
Feature page 0	`0xfffe8ffed83f5b87`	`0xfffe8ffedbff5b87`
Feature page 1	`0x0700000000000000`	`0x0300000000000000`
Feature page 2	`0x5503000000000000`	`0x0f0300000000000`
AuthReq	`0x03`	`0x02`
IO capability	Display YesNo (`0x01`)	NoInputOnOutput (`0x03`)
Secure Connections	True	False

Table 3: Free tools used to implement the Blacktooth attack

Tool	Usage
InternalBlue	Patch CYW920819EVB-02 firmware in real-time
BIAS	Spoof link key authentication
KNOB	Downgrade encryption key to 1 byte of entropy
Wireshark	Monitor LMP and HCI
hcidump	Monitor HCI
hciconfig	Configure HCI interface
hcitool	Scan and enumerate Bluetooth devices
bluetoothctl	Manage and connect devices

we demand. For example, we use InternalBlue [27] to patch the CYW920819 SoC in real-time and read the RAM and the ROM of the firmware at runtime. We also use the BIAS toolkit [5] released by Antonioli *et al.* for impersonating purposes. To modify the Bluetooth Host [2], we amend the Linux kernel source code and recompile it to obtain the function we demand. For example, we modify hci_h4.c and some other files in the kernel to parse the H4 message from the Bluetooth Controller.

Our attack also uses free tools to facilitate information gathering and analysis. Table 3 lists all the relevant tools used in our implementation. The hardware/software cost of the attack is low.

5.3 Blacktooth Attack Implementation

The Blacktooth attack, described in Section 4, consists of the following steps: information gathering, identity forging and proactive connection request, authentication spoofing, encryption key negotiation and brute force, and profile change. Note that the MITM Blacktooth attack is an integration of two Blacktooth attacks targeting Alice and Bob, respectively. The attack against Bob has no essential difference from the attack against Alice. Without loss of generality, we present the implementation of our attack targeting Alice. In this subsection, we present the implementation based on our attack device and attack scenario described in Section 5.1.

Information gathering. To impersonate Bob, the first step for Mallory is to gather information about Bob, including Bluetooth address (BD_ADDR), device name, class of device (CoD), and other necessary atttributes. We use the CYW920819EVB-02 development board connected to a Linux-based laptop and start `hcidump` from the laptop. With this function, we are able to capture all Bluetooth HCI packets in a BTSnoop file, which contains all needed information. Then we use Wireshark to analyze those packets and select the packets that contain certain information.

Specifically, assuming we are around Alice at first, we use the CYW920819 board to scan and discover all Bluetooth devices nearby and identify Alice's Bluetooth address (BD_ADDR$_A$) according to its device name. Then we change the Bluetooth address of the board to BD_ADDR$_A$ and get close to Bob. Assuming Bob is powered on, we use the board to send a Connection Request to Bob or receive a Connection Request from Bob. Note that it is important to change the Bluetooth address to BD_ADDR$_A$ to make the information gathering process universal and stealthy.

There exists a type of Bluetooth device that does not respond to new inquiry packets after it is paired with a device (e.g., a phone), even if the paired device is not in the discoverable range. That means, if Bob belongs to this type, it cannot be discovered, and we are not able to gather information by sending a Connection Request to it using bluetoothctl. However, in order to establish a secure connection with the paired device, Bob has to intermittently send a Connection Request to the paired Bluetooth addresses. Since we have changed our Bluetooth address to BD_ADDR$_A$, we are able to receive the Connection Request from Bob and gather information from the packets in the feature exchange procedure.

In terms of attack stealthiness, since we only changed the Bluetooth address but did not have the link key, Mallory cannot get authenticated by Bob to establish a secure connection. That means we do not destroy the original bonding relationship between Alice and Bob. However, if we do not modify Mallory's Bluetooth address, there is a chance that it successfully establishes a secure connection with Bob, and Bob stores the bonding record, which may overwrite the original bonding relationship with Alice. The victim may notice that her smartphone fails to connect to Bob automatically, and may be aware of an attack.

The other method to gather information is to sniff the Bluetooth BR/EDR packets during the secure connection establishment phase. However, this method is less practical since Bluetooth BR/EDR sniffers are either extremely expensive (e.g., an Ellisys Bluetooth Explorer is priced at $17,500 [24]) or capacity-constrained (e.g., Ubertooth cannot capture all Bluetooth BR/EDR packets). Finally, it is possible to purchase a Bluetooth device that is identical to Bob, where Mallory can collect the Bluetooth attributes, so that it only needs to sniff Bob's Bluetooth device address and name.

Identity forging and proactive connection request. After gathering Bob's identity information, we modify the CYW920819 board firmware with the information to impersonate Bob. Here we use the BIAS toolkit to accomplish this task. We create the Impersonate File (IF) containing Bob's Bluetooth device information and generate the attack script. Finally, we use InternalBlue and the attack script to patch the board firmware, so that the CYW920819 board is transformed into the impersonated device Mallory.

Now we physically get close to the victim, scan for Alice, and send a Connection Request proactively using bluetoothctl. In this process, Mallory impersonates Bob and tries to reconnect to Alice like a benign Bluetooth device (headset). Since Alice has already bonded with Bob, no dialog box will pop up to ask for permission, and no notification will appear in the notification bar. The victim is totally unaware of the attack. Note that it is Mallory who triggered the connection; therefore, the victim's manual interaction is not needed in the connection. Mallory, rather than Alice, took the *Master* role, since Mallory initiated the Connection Request.

Authentication spoofing. After accepting the Connection Request from Mallory, Alice moves into the authentication procedure. From Alice's feature page 1 (see Table 2), we know that Alice (OnePlus 7 Pro) supports Secure Connections. That is to say that Mallory has to downgrade Secure Authentication to Legacy Authentication by informing Alice that Secure Connections are not supported during the connection establishment. Mallory also has to keep its role as the Master to avoid being authenticated by Alice.

To implement these capabilities on Mallory, we use the BIAS toolkit with InternalBlue. Since we have already configured Mallory to impersonate all Bob's attributes with the IF (as in the identity forging paragraph), we now employ the toolkit to downgrade Secure Connections and configure unilateral authentication for Mallory via the Attack File (AF). The Attack File contains the information about the attack devices, for example, the ROM and RAM addresses to be patched in the firmware of the Bluetooth chip. Related flags in the Bluetooth firmware should be modified to downgrade Secure Connections. For unilateral authentication, the Bluetooth firmware is modified to skip the process for Mallory to verify Alice's response.

Encryption key negotiation and brute force. After the authentication procedure, the next step is to start encryption. This process is completely handled between two Bluetooth Controllers, which means we have to patch the Bluetooth firmware to infect the encryption key negotiation procedure. Since we do not have the link key shared between Alice and Bob, we cannot compute the actual encryption key K'_C using encryption key generate functions. To get K'_C, Mallory has to convince Alice to accept the entropy of the encryption key to be 1 byte, which facilitates the K'_C brute force (see Section 4.4). Then we can take L2CAP headers from Alice as the oracle to brute force K'_C among all 256 candidates.

In the attack, we modify the CYW920819EVB-02 board firmware and set $L_{max} = 1$ and $L_{min} = 1$. In this way, Mallory always asks Alice to accept the encryption key of 1 byte. After the encryption key length negotiation is completed, Mallory has to compute all 256 possible K'_C and select the right one according to the oracles. Then Mallory has the ability to encrypt and decrypt messages by writing the K'_C to the place used to store the key in the RAM. It is realizable since Mallory can locate the address of the RAM used to store K'_C by reverse engineering. Taking advantage of the InternalBlue, Mallory can patch the RAM at runtime to write in the selected K'_C.

Profile change. After the previous attack phase, Mallory has the ability to impersonate Bob and establish a secure connection with Alice without the link key K_L. In a normal connection establishment procedure, only the device that has the link key K_L is able to pass the authentication procedure and establish a secure connection with Alice. In other words, if Mallory can successfully establish a secure connection with Alice, Alice thinks Mallory has the link key K_L. Since the bonding procedure is to store the link key created while pairing for use in subsequent secure connections, Alice and Bob store the link key K_L after the pairing procedure. From the aspect of Alice, if a device has the link key K_L corresponded to Bob's, Alice will regard the device as Bob because K_L should only be shared between Alice and Bob. It is equivalent to Mallory "inheriting" Bob's binding relationship with Alice or, in other words, Mallory bonding with Alice. Therefore, it is reasonable to make Mallory has a real bonding relationship with Alice.

Table 4: Blacktooth attack evaluation result

Manufacturer	Device Model	Operating System	Chip Model	Producer	Bluetooth Version	Blacktooth Attack	MITM Attack
Apple	iPhone 12	iOS 15.4.1	339S00761	USI	5.0	✓	✓
Apple	iPhone 12	iOS 15.0.1	339S00761	USI	5.0	✓	✓
Apple	iPad Air 3	iPadOS 15.2	39S00551	USI	5.0	✓	✓
Apple	Macbook Pro 2018	macOS Monterey 12.3.1	BCM4364B0	Broadcom	5.0	✓	✓
Google	Pixel 5	Android 12.0	Snapdragon 765G	Qualcomm	5.0	✓	✓
HONOR	V30 Pro	Harmony 2.0.0	Hi1103	HiSilicon	5.1	✓	✓
HONOR	V20	Harmony 2.0.0	Hi1103	HiSilicon	5.0	✓	✓
HONOR	V8	EMUI 8.0.0 (Android 8.0)	Kirin 955	HiSilicon	4.2	✓	✓
HUAWEI	Mate 30	Harmony 2.0.0	Hi1103	HiSilicon	5.1	✓	✓
HUAWEI	Mate 9	EMUI 9.1.0 (Android 9.0)	BCM43455XKUBG	Broadcom	4.2	✓	✓
HUAWEI	P10	EMUI 9.1.0 (Android 9.0)	BCM43455XKUBG	Broadcom	4.2	✓	✓
OnePlus	9R	ColorOS 11.2 (Android 11.0)	Snapdragon 870	Qualcomm	5.1	✓	✓
OnePlus	7 Pro	Hydrogen OS 9.5.9 (Android 9.0)	WCN3998	Qualcomm	5.0	✓	✓
OPPO	Find X2 Pro	ColorOS 12.1 (Android 12.0)	Snapdragon 865	Qualcomm	5.1	✓	✓
OPPO	A72n 5G	ColorOS 7.2 (Android 10.0)	MT6853V	MediaTek	5.1	✓	✓
realme	X50 5G	realme UI 2.0 (Android 11.0)	WCN3998	Qualcomm	5.0	✓	✓
Redmi	K30 Pro	MIUI 12.5.4 (Android 11.0)	WCN3998	Qualcomm	5.1	✓	✓
Samsung	Galaxy S10	One UI 3.1 (Android 11.0)	KM8D03042	Murata	5.0	✓	✓
Xiaomi	Mi 10 Pro	MIUI 12.5.12 (Android 11.0)	QCA6391	Qualcomm	5.1	✓	✓
Xiaomi	Mi 10	MIUI 13.0.4 (Android 12.0)	QCA6391	Qualcomm	5.1	✓	✓
Lenovo	ThinkPad X1 Carbon (5th)	Windows 10 Pro 1909	8265	Intel	4.1	⋆	⋆
Lenovo	ThinkPad X1 Carbon (5th)	Linux 4.14.111	8265	Intel	4.1	⋆	⋆

✓ Confirmed to be vulnerable to the Blacktooth attack.
⋆ Cannot complete a secure connection establishment after changing profiles

In order to implement this attack phase, we use Mallory's bonding record to take the place of Bob's once. We add A2DP and HFP records to Mallory's SDP record list with open-source code to implement the headset functions. With the features set in the previous steps, Mallory is almost a clone of Bob. When Mallory pairs Alice for the first time, it broadcasts its A2DP and HFP services using SDP as Bob. Then Alice stores the service records of Phone calls and Media audio, which is the same as Bob's service records.

For the arbitrary-command-injection attack, we utilize open-source code to implement the HID application as a keyboard and a mouse and add the HID record to the SDP service list. Then we run the application in the Host, the Linux-based laptop, and proactively send a connection request to Alice. Since Alice has already paired with Mallory, it will accept the request and establish a secure connection with Mallory (as described above). Due to vulnerabilities #4 and #5, Alice accepts the HID connection request and grants the input permission to Mallory. In this way, Mallory can operate Alice using a keyboard and mouse, for example, opening applications, sending messages, making calls, or turning off Alice.

For information-stealing attacks, we implement the PBAP application and add the PBAP record to the SDP service list. Then we run this application and connect with Alice. Since Alice has already granted Bob permission to access the phone book, Mallory can get all the contact information stored in Alice's phone book.

It is worth emphasizing that all injection and stealing attacks do not need any pre-installed (benign/malicious) applications or user interaction. Only the vulnerabilities in Bluetooth Specification are utilized by Mallory. The whole attack process is completely silent.

Attack efficiency. The Blacktooth attack described above can be completed in a few seconds. The time to collect device information is approximately 3 seconds and this process can be finished ahead of the actual attack process. It takes less than 1 second to crack the encryption. Except for those two processes, the rest of the Blacktooth attack uses the same time as the normal connection establishment. Overall, the attack is very efficient and highly practical.

5.4 Blacktooth Attack Evaluation Setup

With our Blacktooth attack implementation (presented in Section 5.3), we are able to conduct the Blacktooth attack against different Bluetooth devices. We consider an attack scenario with two victims, Alice and Bob, and an attacker Mallory. Alice is any Bluetooth device at our disposal, Bob is a OnePlus Bullets Wireless 2 headset (Table 2), and Mallory is a CYW920819EVB-02 development board connected to a Linux-OS-based ThinkPad X1 Carbon laptop. Alice is paired with Bob, and the link key K_L is unknown to Mallory. No additional (benign or malicious) application needs to be installed in Alice or Bob. Mallory impersonates Bob and tries to gain access to control Alice. Bob should be present when Mallory gathers the information of Bob but is not required to be present when Mallory launches the Blacktooth attack against Alice. For the *MITM Blacktooth* attack, Mallory impersonates Alice and Bob simultaneously and tries to establish secure connections with both Alice and Bob.

5.5 Blacktooth Attack Evaluation Result

Our evaluation results are shown in Table 4. Overall, we evaluate our Blacktooth attack on 21 different devices (ThinkPad X1 Carbon 5th is evaluated with Windows and Linux respectively). The first 6 columns indicate the device manufacturer, device model, OS, chip model, chip producer, and the supported Bluetooth version.

The last 2 columns evaluate the Blacktooth attack and the MITM Blacktooth attack. A checkmark (✓) indicates a successful attack (silently connected, changed profile, retrieved data, and operated the device), while a star (★) indicates that a device cannot complete a secure connection establishment after changing profiles.

In Table 4, we demonstrate that *all the devices* with different Android (from Android 8.0 to the latest Android 12.0), iOS, iPadOS, macOS, or HarmonyOS operating systems are victims of the original and MITM Blacktooth attacks. In Windows and Linux, Mallory cannot complete the connection procedure after it changes the profiles. However, all the devices are vulnerable to the Blacktooth attack if Mallory uses Bob's original profiles to launch the attack (e.g., Bob is a headset and Mallory can inject voice commands).

It is also demonstrated in Table 4 that the original and MITM Blacktooth attacks are effective against different device models, operating systems, Bluetooth chip models, and Bluetooth versions. The experiments confirm that our Blacktooth attacks are entirely silent, non-interactive, standard-compliant, and practical. With the original Blacktooth attack, Mallory can successfully escalate its privileges through profile change on all Android and iOS devices. With the MITM Blacktooth attack, Mallory can further eavesdrop on all the real-time communications between the Android/iOS device and the Bluetooth peripheral/wearable device. We discover that the firmware of the Bluetooth chips used by the commodity devices on the market are not easy to patch for compatibility considerations. The experiment results confirm that the proposed Blacktooth attack is a serious threat to billions of Bluetooth BR/EDR device users.

6 DISCUSSION

In this section, we discuss several practical issues with the Blacktooth attack: the discoverable and connectable state assumptions, the failed profile changes in Linux and Windows, and the security of the E_0 stream cipher. Finally, we discuss potential countermeasures to defend the original and MITM Blacktooth attacks.

6.1 The Discoverable and Connectable State

As mentioned in Section 4.1, Mallory can proactively send a Connection Request to Alice to trigger a stealthy connection establishment process and claim the Master role by utilizing Vulnerability #1. The precondition that Mallory can discover and send a Connection Request to Alice is that Alice must be in the *discoverable and connectable state*. Theoretically, the Bluetooth Specification does not require a device to be in a discoverable state when Bluetooth is turned on. However, we find that none of the mainstream operating systems (Android, iOS, Windows, and macOS) provide the users with an option to make their devices discoverable/undiscoverable, which means that the victim devices (Alice) are always in the discoverable and connectable state when Bluetooth is turned on. Moreover, we observe that the majority of the users, especially those who use Bluetooth headsets, rarely turn off Bluetooth. Hence, their smartphones (Alice) are always in the discoverable/connectable state, and they are vulnerable to the Blacktooth attack.

6.2 Profile Change in Linux and Windows

As shown in Table 4, the Blacktooth attack fails to change profiles for privilege escalation on Linux and Windows operating systems. In the experiments, the connections are dropped once the attacker attempt to switch profiles. For Linux, we examine its source code and find that the connection is blocked because the request to connect to a new profile triggers an L2CAP_CR_SEC_BLOCK flag. Consequently, the Linux system sends a "Security Block" packet to reject the new profile connection. However, the profile change can be completed if the connection is established by Alice, who takes the Master role in connection initialization. We consider that such a mechanism is also vulnerable since the Linux system does not inform the user of the profile changes. In Windows, a "PSM not supported" packet is sent to reject the profile change, when Mallory attempts to switch to a new profile. Unlike the Linux system, the profile change is not permitted even if Alice takes the Master role to initiate the connection with Mallory. Since Windows is not open source, we do not know its internal mechanism.

6.3 E_0 Stream Cipher Security

Bluetooth devices use the E_0 stream cipher to encrypt data if Secure Connections are not supported. However, E_0 is considered a weak stream cipher in cryptography and many published attacks [15, 16, 25, 26, 37, 38] can significantly decrease the security of this cipher. Zhang *et al.* [38] proposed a practical attack against the E_0 stream cipher with complexity under 2^{25}, and this attack only takes a few seconds to restore the original encryption key. All previous research indicates that the E_0 cipher is not practically secure even using encryption keys with larger entropy.

6.4 Defense against Blacktooth

The Blacktooth attack evaluated in Section 5 is enabled by five vulnerabilities in the Bluetooth Specification (introduced in Section 3.3). The combination of those vulnerabilities allows an attacker to silently connect to and escalate privileges on the victim device to inject arbitrary commands and steal data. We now briefly discuss a list of countermeasures to patch those vulnerabilities.

Enforcing confirmation dialog (Vulnerability #1). The stealthiness of the Blacktooth attack largely relies on the fact that the malicious Bluetooth device can trigger secure connection establishment and complete the whole process without the victim's confirmation. This ensures the complete stealthiness of the Blacktooth. To fix this vulnerability, a Bluetooth device should ask the user for confirmation on connection requests even if the requester is bonded. This defense may be deployed at a cost of usability since it forces users to confirm each connection manually. However, we believe it is necessary for the security of Bluetooth devices. We would expect more research efforts on this issue to balance usability and security, e.g., to evaluate the risk of each connection and only require user intervention for risky connections. This countermeasure can be implemented in the Bluetooth Host (OS).

Enforcing mutual authentication (Vulnerability #2). The unilateral Legacy Authentication enables the malicious Bluetooth device to bypass the victim's Bluetooth device's authentication. To mitigate this vulnerability, if the peer device does not support Secure Authentication, a Bluetooth device should mandate mutual authentication using Legacy Authentication. In case the peripheral device and the user's device have already paired with Secure Connections, the user's device should enforce Secure Authentication

for all subsequent connections, or notify the user in case Secure Authentication is rejected. This countermeasure can be easily implemented in the Bluetooth Host (OS).

Enforcing long encryption key (Vulnerability #3). The low entropy of the encryption key makes it vulnerable to brute-force attacks. To patch this vulnerability, a Bluetooth device should always use a large entropy, e.g., 16 bytes. To achieve the desired level of security and protect itself, the Master device may enforce $1 << L_{min} \leq 16$. The implementation of this countermeasure requires the modification of the Bluetooth Controller (firmware).

Enforcing profile authentication and authorization (Vulnerability #4, #5). The excessive flexibility to profile connection and excessive trust on paired devices facilitate an attacker to add sensitive profiles and get the privilege to control the victim's Bluetooth device without getting the user's confirmation. To fix these vulnerabilities, the master device (e.g., smartphone) should monitor the profile lists, turn off the permissions to newly added profiles by default, and notify the user in case of a profile list modification. With a prompt from the mobile OS, the user should decide whether to accept or reject the profile change. This countermeasure can be implemented in the Bluetooth Host (OS).

We disclosed the Blacktooth attack with the Bluetooth SIG and the manufacturers of all the tested devices. As of the submission of this paper, we received positive responses from several vendors. We discussed the defense mechanisms with them and agreed that some vulnerabilities could be promptly patched at the OS level.

7 RELATED WORK

A recent guide from NIST to Bluetooth security was provided in 2017 [28]. The guide presents some Bluetooth vulnerabilities associated with different versions that can be utilized to launch attacks, some of which are utilized in our Blacktooth attack. The guide mentions that "discoverable and/or connectable devices are prone to attack", but it doesn't explain specifically how the vulnerability may be taken advantage of. Previous surveys even did not consider the potential threat of connectable devices [14, 19]. In fact, prior works neglect the vulnerability of unfixed roles and arbitrary connection requesters, which is significant to launching a practical attack.

Some attacks focus on a specific aspect of Bluetooth BR/EDR. For instance, the KNOB attack [4] utilizes Vulnerability #3 to force two unaware victims to negotiate a 1-byte entropy encryption key, while a MITM attacker can decrypt all traffic between them by brute-forcing the encryption key. However, the KNOB attack is a passive attack, which makes it hard to inject arbitrary commands to take control of the victim devices. The BadBluetooth attack [36] utilizes Vulnerability #4 in Android devices to make profile changes and to escalate the privileges. However, the BadBluetooth attack requires a pre-installed malicious application in the Android smartphone to initiate the connection establishment, which is a strong and impractical assumption in the real world. We also confirm that Vulnerability #4 does not only exist in the latest Android but also in iOS, macOS, iPadOS, and HarmonyOS. The BlueMirror attack [12] implements reflection attacks on the Bluetooth pairing protocol. However, according to [12], the reflection attackers must take the Slave role during the paring process, which means the attacker cannot launch an attack proactively, therefore, its effect is limited.

Some attacks are implemented to take control of smartphones and laptops via Bluetooth BR/EDR without user interactions. For example, the BlueBorne attack [7] exploits some flaws on Windows, iOS, Linux, and Android to invade target devices while the victims are unconscious. The Bluebugging attack [18] allows issuing AT commands to the vulnerable devices without prompting the owner to download SMS messages and make phone calls. However, the vulnerabilities these prior attacks utilized are all fixed in the new version of the OSes, and they are not effective anymore. Our Blacktooth attack is effective for the devices with the newest OSes.

We also notice some attacks on Bluetooth Low Energy (BLE) devices. Zhang *et al.* [39] discovered that a device in SCO mode might suffer a downgrade attack due to the inappropriate handling of error codes. Jasek [20] provides new tools to launch attacks via GATT of BLE. Note that Bluetooth BR/EDR and BLE are two separate protocol stacks, which means the vulnerabilities may not exist in both. Moreover, the roles of BLE devices are fixed, which means an attacker is not able to establish a connection proactively.

In order to efficiently identify vulnerabilities in the implementation of Bluetooth, researchers create various automatic fuzzing tools to uncover flaws both in firmware [29] and applications [40]. It is obvious that these automatic tools facilitate vulnerability discovery. There is still a way to go to transform the vulnerabilities into practical attacks in real scenarios. Our Blacktooth attack is a practical attack that can cause severe consequences for security.

Last, attempts are made to migrate the impacts of arbitrary profile change [35, 36]. Unfortunately, the vulnerabilities still exist in various latest versions of OSes. As for now, no effective countermeasure has been adopted to prevent our Blacktooth attack.

8 CONCLUSION

We present the Blacktooth attack against Bluetooth BR/EDR. Our attack exploits a series of vulnerabilities of the Bluetooth Specification to impersonate a Bluetooth peripheral device, establish a connection with the victim device (e.g., a smartphone), and escalate privileges by changing profiles. As a result, the attacker can inject arbitrary commands and steal sensitive data from the victim device, without alerting or interacting with the user during the entire process, without any pre-installed malicious agent on the victim device, and without having to know or authenticate the link key.

The Blacktooth attack is practical since it removes the main obstacle in the initial stage of Bluetooth connection establishment, where user interaction is needed for the Master to connect to the malicious peripheral device. Moreover, through successful attacks against 21 different Bluetooth devices from various manufacturers and different configurations, we demonstrate that the Blacktooth attack is practical, stealthy, and powerful.

ACKNOWLEDGMENTS

We thank all anonymous reviewers and our shepherd for their valuable comments and suggestions. This work is supported in part by the National Natural Science Foundation of China under Grant No. 61972371 and No. U19B2023, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under Grant No. Y202093.

REFERENCES

[1] 2021. AOSP Bluetooth Services. https://source.android.com/devices/bluetooth/services. (2021). accessed: Oct., 2021.

[2] 2021. Bluez - Official Linux Bluetooth Protocol Stack. http://www.bluez.org. (2021). accessed: Oct, 2021.

[3] Albazrqaoe, Wahhab and Huang, Jun and Xing, Guoliang. 2016. Practical bluetooth traffic sniffing: Systems and privacy implications. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. 333–345.

[4] Antonioli, Daniele and Tippenhauer, Nils Ole and Rasmussen, Kasper. 2019. The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR. In *USENIX Security*. USENIX Association, 1047–1061.

[5] Antonioli, Daniele and Tippenhauer, Nils Ole and Rasmussen, Kasper. 2020. BIAS: bluetooth impersonation attacks. In *IEEE Symposium on Security and Privacy*.

[6] Antonioli, Daniele and Tippenhauer, Nils Ole and Rasmussen, Kasper. 2020. Key negotiation downgrade attacks on bluetooth and bluetooth low energy. *ACM Transactions on Privacy and Security (TOPS)* 23, 3 (2020), 1–28.

[7] Armis. 2017. The Attack Vector BlueBorne Exposes Almost Every Connected Device. https://armis.com/research/blueborne/. (2017). accessed: Oct., 2021.

[8] Biham, Eli and Neumann, Lior. 2019. Breaking the bluetooth pairing–the fixed coordinate invalid curve attack. In *Proceedings of International Conference on Selected Areas in Cryptography (SAC)*. Springer, 250–273.

[9] Bluetooth SIG. 2015. Human Interface Device Profile 1.1.1. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=309012. (2015). accessed: Oct., 2021.

[10] Bluetooth SIG. 2019. Bluetooth Core Specification v5.2. https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726. (2019). accessed: Oct., 2021.

[11] Bluetooth SIG. 2021. 2021 Bluetooth Market Update. https://www.bluetooth.com/wp-content/uploads/2021/01/2021-Bluetooth_Market_Update.pdf. (2021). accessed: Oct., 2021.

[12] Tristan Claverie and José Lopes Esteves. 2021. Bluemirror: reflections on bluetooth pairing and provisioning protocols. In *Proceedings of 2021 IEEE Security and Privacy Workshops (SPW 2021)*. IEEE, 339–351.

[13] Cypress. 2021. CYW920819EVB-02 Evaluation Kit. https://www.cypress.com/documentation/development-kitsboards/cyw920819evb-02-evaluation-kit. (2021). accessed: Oct., 2021.

[14] Dunning, John. 2010. Taming the blue beast: A survey of bluetooth based threats. *IEEE Security & Privacy* 8, 2 (2010), 20–27.

[15] Scott Fluhrer and Stefan Lucks. 2001. Analysis of the E0 encryption system. In *Proceedings of the 8th Annual International Workshop on Selected Areas in Cryptography (SAC 2001)*. Springer, 38–48.

[16] Jovan Dj Golić, Vittorio Bagini, and Guglielmo Morgari. 2002. Linear cryptanalysis of Bluetooth stream cipher. In *Proceedings of the 21st Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2002)*. Springer, 238–255.

[17] Haataja, Keijo and Toivanen, Pekka. 2008. Practical man-in-the-middle attacks against bluetooth secure simple pairing. In *Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*. IEEE, 1–5.

[18] Herfurt, Martin. 2004. Bluebugging. https://trifinite.org/trifinite_stuff_bluebug.html. (2004). accessed: Oct., 2021.

[19] Jakobsson, Markus and Wetzel, Susanne. 2001. Security weaknesses in Bluetooth. In *Proceedings of the 2001 Cryptographers' Track at the RSA Conference*. Springer, 176–191.

[20] Jasek, Sławomir. 2016. Gattacking Bluetooth smart devices. In *Proceedings of the Black Hat USA Conference*.

[21] Levi, Albert and Çetintaú, Erhan and Aydos, Murat. 2004. Relay Attacks on Bluetooth Authentication and Solutions. In *Proceedings of the 19th International Symposium on Computer and Information Sciences (ISCIS)*, Vol. 19. Springer, 278–288.

[22] Lindell, Andrew Y. 2008. Attacks on the pairing protocol of bluetooth v2. 1. *Black Hat USA, Las Vegas, Nevada* (2008).

[23] Lonzetta, Angela M and Cope, Peter and Campbell, Joseph and Mohd, Bassam J and Hayajneh, Thaier. 2018. Security vulnerabilities in Bluetooth technology as used in IoT. *Journal of Sensor and Actuator Networks* 7, 3 (2018), 28.

[24] Loveless, Mark. 2017. BLUETOOTH HACKING TOOLS COMPARISON. https://trifinite.org/trifinite_stuff_bluebug.html. (2017). accessed: Oct., 2021.

[25] Yi Lu, Willi Meier, and Serge Vaudenay. 2005. The conditional correlation attack: A practical attack on bluetooth encryption. In *Proceedings of the 25th Annual International Cryptology Conference (CRYPTO 2005)*. Springer, 97–117.

[26] Yi Lu and Serge Vaudenay. 2004. Cryptanalysis of Bluetooth keystream generator two-level E0. In *Proceedings of the 10th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2004)*. Springer, 483–499.

[27] Mantz, Dennis and Classen, Jiska and Schulz, Matthias and Hollick, Matthias. 2019. InternalBlue-Bluetooth binary patching and experimentation framework. In *Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM, 79–90.

[28] Padgette, John. 2017. Guide to Bluetooth Security. *NIST Special Publication* 800:121 (2017).

[29] Ruge, Jan and Classen, Jiska and Gringoli, Francesco and Hollick, Matthias. 2020. Frankenstein: Advanced wireless fuzzing to exploit new bluetooth escalation targets. In *USENIX Security*. 19–36.

[30] Shaked, Yaniv and Wool, Avishai. 2005. Cracking the bluetooth pin. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services (MobiSys)*. 39–50.

[31] Spill, Dominic and Bittau, Andrea. 2007. BlueSniff: Eve Meets Alice and Bluetooth. *USENIX Workshop on Offensive Technologies* 7 (2007), 1–10.

[32] Sun, Da-Zhi and Mu, Yi and Susilo, Willy. 2018. Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5. 0 and its countermeasure. *Personal and Ubiquitous Computing* 22, 1 (2018), 55–67.

[33] von Tschirschnitz, Maximilian and Peuckert, Ludwig and Franzen, Fabian and Grossklags, Jens. 2021. Method confusion attack on bluetooth pairing. In *Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1332–1347.

[34] Wong, Ford-Long and Stajano, Frank and Clulow, Jolyon. 2005. Repairing the bluetooth pairing protocol. In *Proceedings of the 13th International Conference on Security Protocols*. 31–45.

[35] Wu, Jianliang and Wu, Ruoyu and Antonioli, Daniele and Payer, Mathias and Tippenhauer, Nils Ole and Xu, Dongyan and Tian, Dave Jing and Bianchi, Antonio. 2021. LIGHTBLUE: Automatic Profile-Aware Debloating of Bluetooth Stacks. In *USENIX Security 21*. 339–356.

[36] Xu, Fenghao and Diao, Wenrui and Li, Zhou and Chen, Jiongyi and Zhang, Kehuan. 2019. BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals. In *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. NDSS Symposium.

[37] Bin Zhang, Chao Xu, and Dengguo Feng. 2013. Real time cryptanalysis of Bluetooth encryption with condition masking. In *Proceedings of the 33rd Annual Cryptology Conference (CRYPTO 2013)*. Springer, 165–182.

[38] Bin Zhang, Chao Xu, and Dengguo Feng. 2018. Practical cryptanalysis of Bluetooth encryption with condition masking. *Journal of Cryptology* 31, 2 (2018), 394–433.

[39] Zhang, Yue and Weng, Jian and Dey, Rajib and Jin, Yier and Lin, Zhiqiang and Fu, Xinwen. 2020. Breaking secure pairing of bluetooth low energy using downgrade attacks. In *Proceedings of the 29th USENIX Security Symposium (USENIX Security)*. USENIX Association, 37–54.

[40] Zuo, Chaoshun and Wen, Haohuang and Lin, Zhiqiang and Zhang, Yinqian. 2019. Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 1469–1483.

A LEGACY AUTHENTICATION

Figure 8 describes the Legacy Authentication process.

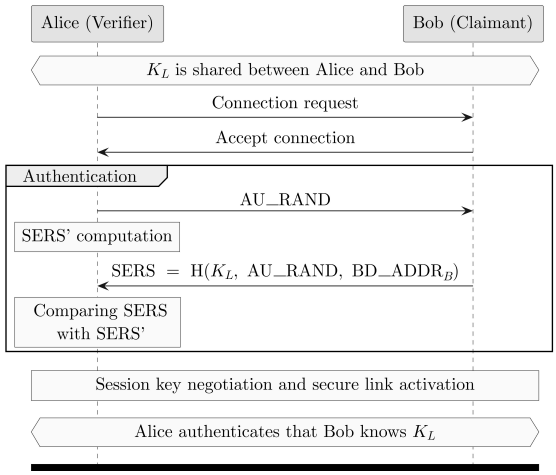

Figure 8: Legacy Authentication process. The legacy Authentication process authenticates devices unilaterally. To achieve mutual authentication, the authentication process should be completed by Alice and Bob respectively.