BISON: Attacking Bluetooth's Broadcast Isochronous Streams

Theo Gasteiger, Carlo Alberto Boano, and Kay Römer
Institute of Technical Informatics, Graz University of Technology, Austria
{gasteiger,cboano,roemer}@tugraz.at

Abstract

In this paper we present `BISON`, a novel attack on Bluetooth's broadcast isochronous streams (BISes), and demonstrate it on off-the-shelf hardware. `BISON` exploits the plaintext meta-data used for stream synchronization as well as the vague specification of the `Broadcast_Code` exchange to take over ongoing BISes and manipulate their content. With `BISON`, we are the first to raise awareness about the vulnerability of BISes, which are the stepping stone of several Bluetooth applications for audio diffusion at public locations. We further describe possible attack countermeasures and guidelines on how to design secure applications leveraging BISes.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]

General Terms

Security, Design.

Keywords

Bluetooth Low Energy, Isochronous, Audio, Security.

1 Introduction

In recent years, the Bluetooth Low Energy (BLE) specification has undergone extensive updates in order to improve performance and enable new use cases. These updates include, among others, the addition of physical layers enabling a higher data rate or longer communication range as well as the support for direction-finding, extended and periodic advertisements [6]. Another key feature recently introduced in the BLE specification are *isochronous channels* [26], which lay the foundation for the new LE Audio standard [20]. In fact, isochronous channels enable time-sensitive data transmission and synchronized data rendering across multiple receiver devices. This is a key feature for audio use cases such as "true wireless earbuds", where a simultaneous data reception is crucial: previously, Bluetooth audio data was transmitted to each earbud individually, and it was up to the manufacturer to ensure synchronization between both earbuds. Moreover, Isochronous channels support both connection-oriented and connection-less communication, which allows for bidirectional and unidirectional data transmission, respectively. Applications such as "true wireless earbuds" are examples of connection-oriented communication, in which data is disseminated in a bidirectional manner (e.g., to the speaker and from the microphone) and also referred to as a *connected isochronous stream* (CIS). In contrast, when using connection-less data transmission, a device can stream unidirectional audio data simultaneously to countless devices using a *broadcast isochronous stream* (BIS).

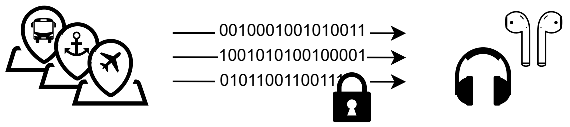

Figure 1. Exemplary BIS use cases. BISes enable the transmission of open broadcast audio streams as well as private (encrypted) broadcast audio streams in public spaces.

Broadcast audio in public spaces. BISes pave the way for a plethora of new use cases. For example, one can share audio data to small groups of devices, e.g., stream sound from a home TV to several earbuds worn by different family members. More importantly, one can broadcast audio data to large collections (potentially, an unlimited number) of devices in *public spaces*, enabling the creation of assisted hearing systems in public locations [35]. This realm of applications, including the replacement of telecoil for hearing-impaired people, will soon become reality, as the adoption of the LE Audio standard's broadcast audio feature (marketed as *Auracast*) increases [8]. The use of BISes for audio diffusion is indeed foreseen in several public locations, such as theaters, museums, bars, congress centers, and transport hubs [8, 15, 18, 20, 22, 29]. For example, navigation instructions and public transport disruption information will be broadcasted at airports, ferry terminals, railway stations, as well as bus stops [18, 20, 29]. Immersive audio guides for museums and exhibitions will be replaced by broadcast audio streams in which information is simultaneously transmitted in different languages [35]. Silent TV screens deployed at gyms, sport bars, hotels, and waiting rooms will broadcast sound to any surrounding headphones that are tuned to the correct audio stream [1, 35].

Vulnerability of the envisaged uses cases. Broadcast audio streams available in public areas may be open (unencrypted) or private (encrypted), as illustrated in Fig. 1. For example, an airport may provide gate announcements in the form of an unencrypted broadcast audio stream to any surrounding device. Alternatively, to avoid any manipulation of the audio stream by third parties (i.e., man-in-the-middle attacks changing the contents of the audio stream), one may encrypt the BIS using a `Broadcast_Code` (essentially, a passkey used to derive a symmetric session key). Such `Broadcast_Code` is necessary to decrypt the stream's content and needs to be shared with all intended recipients.

Unfortunately, most of the envisaged applications broadcasting audio streams in public spaces favor an effortless joining procedure to the stream rather than ensuring authenticity, integrity, and confidentiality. On the one hand, for Auracast transmitters in public spaces, the use of *unencrypted* broadcast audio streams is actually recommended [24]. On the other hand, for encrypted audio streams, the distribution of the `Broadcast_Code` should be "easy to do" [6] and is envisaged with out-of-band methods such as QR codes displayed at prominent locations, or NFC tags mounted at dedicated areas [4, 20]. Concretely, this means that a user can simply scan a QR code at the gate (or tap his/her earbuds against a dedicated NFC terminal) to join a private BIS and receive the encrypted gate announcements [15]. However, *anyone* can do so, including a malicious actor, who can hence easily gain access to the key material of the encrypted BIS.

From a security perspective, this state-of-affairs is alarming, as there is a concrete risk of deploying several unprotected broadcast audio streams whose content can be easily manipulated by a malicious actor, as we discuss in § 3. This is very relevant in case the manipulation of audio information in a (private) BIS may result in injury or harm, e.g., in applications such as turn-by-turn navigation for visually-impaired people or wellness tips in a medical waiting room.

Contributions. In this paper, we present `BISON`, a practical attack to Bluetooth applications making use of BISes, and demonstrate it on off-the-shelf hardware. In `BISON`, we let a BLE device take over an ongoing BIS (i.e., impersonate the source of data transmissions) and manipulate its content (i.e., forge arbitrary payloads in both unencrypted and encrypted streams). We do so by exploiting the plaintext metadata used for BIS synchronization, the vague specification of the `Broadcast_Code` exchange, and BLE's channel map update procedure. Hence, with `BISON`, which we make available open-source, we are the first to raise awareness about the vulnerability of BISes, whose use is envisaged in a plethora of applications broadcasting audio streams in public spaces. We further describe possible attack countermeasures and guidelines on how to design secure applications leveraging BISes.

Paper outline. After providing background information on isochronous streams in § 2, we describe how a malicious actor could exploit the metadata necessary to synchronize to a BIS and the envisaged out-of-band distribution of a `Broadcast_Code` to perform an attack to an ongoing BIS in § 3. We then describe the `BISON` attack in § 4 and implement it on real hardware in § 5. After discussing possible countermeasures in § 6, we present related work in § 7 and conclude the paper in § 8, along with a discussion of future work.

2 Demystifying Isochronous Streams

We provide background information about BLE's new isochronous channel feature in § 2.1, and describe in detail the operations of broadcast isochronous streams in § 2.2.

2.1 Isochronous Channels and Streams

One of the key features introduced in the Bluetooth v5.2 specification is the ability to perform *isochronous communication*, i.e., to transmit time-bounded data between devices and enable synchronized data rendering across multiple receivers [31]. Isochronous communication uses the new *LE isochronous physical channel* (often referred to just as isochronous channel). When using the latter, data is only valid for a limited time, and is sent in *isochronous streams* (i.e., sequence of *events* repeated at regular time intervals). An isochronous channel specifies the exact timing at which events occur (and on which frequency, as adaptive frequency hopping is used). Each event serves as an *anchor point* for the timing of the subsequent event (which is spaced in time by a fixed `ISO` interval), and is used for data transmission.

Isochronous streams belong to *isochronous groups* in order to have a common timing reference across streams that need to be synchronized. For example, a group may consist of an audio stream destined to the left earbud and an audio stream destined to the right earbud; or it may consist of two audio streams containing the same info in two different languages.

Isochronous streams can be unicast (connection-oriented) or broadcast (connection-less). The former, also referred to as *connected isochronous streams* (CISes), are very similar to Bluetooth Classic due to the usage of acknowledgments and a point-to-point topology. The latter, also referred to as *broadcast isochronous streams* (BISes), are instead purely unidirectional and allow for simultaneous data transmission to multiple receivers. Both types of stream provide open (unencrypted) as well as private (encrypted) data transmission.

Fig. 2 (bottom) shows an exemplary isochronous communication, in which the events of a (broadcast) isochronous stream are separated in time by a fixed `ISO` interval, and are clustered into (broadcast) isochronous group events. In § 2.2 we describe how to establish BISes and detail their structure.

2.2 Broadcast Isochronous Streams (BISes)

BISes enable many attractive applications, as described in § 1, and benefit from the often asymmetric link budget of real-world applications [20]. This asymmetry is given by the fact that transmitter nodes are often mains-powered and, therefore, are not subject to stringent power requirements.

Finding a BIS. Fig. 2 illustrates the BIS establishment procedure, which starts with the broadcaster transmitting advertisement packets on BLE's three primary advertisement channels (i.e., 37, 38, and 39). These advertisement packets indicate through the `AUX` pointer field that additional info about an ongoing BIS can be found in an extended advertisement packet, transmitted on one of BLE's general-purpose channels (i.e., 0, ..., 36). Given that the information required to describe the configuration and content of a BIS [5] may not fit into a single extended advertisement packet (i.e., it may be larger than 254 octets), the Bluetooth specification foresees the usage of a periodic advertisement train instead

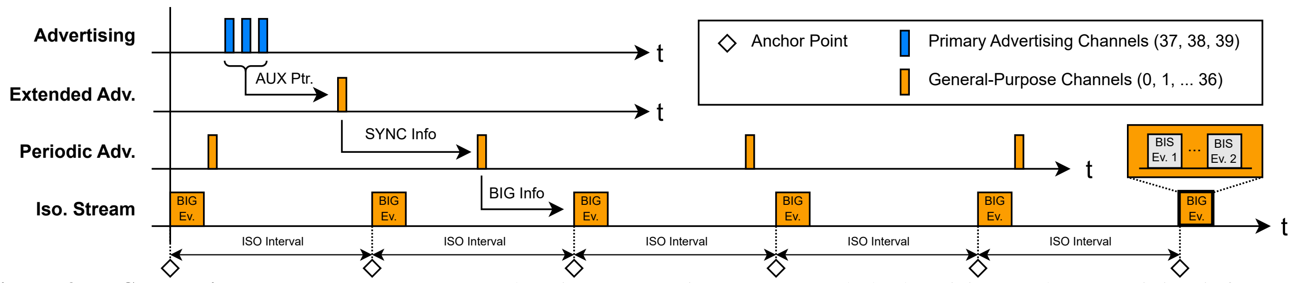

Figure 2. BIS establishment procedure. BLE advertisements point to an extended advertising packet containing information for synchronization to an ongoing periodic advertisement train. Each periodic advertisement contains information about the next anchor point, i.e., about the timing of the next BIG event (which encapsulates one or more BIS events, as shown in Fig. 3). BIG events (and, therefore, also BIS events) repeat over time at a fixed `ISO` interval.

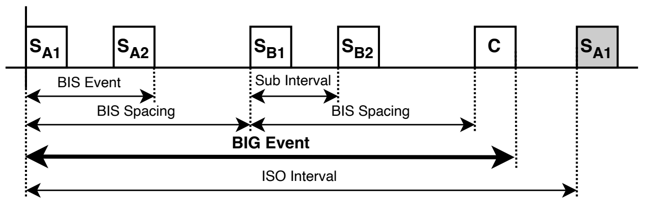

Figure 3. Sequential BIS structure. Every BIG event consists of one or more BISes (e.g., S_A, S_B), in turn consisting of one or more subevents (e.g., S_{A1}, S_{A2}). Optionally, every BIG event can contain a control subevent (C) at the end.

of an additional extended advertisement packet [6, 20]. Receiver nodes can hence use the `SYNC` information contained in the extended advertisement packet to synchronize to an ongoing periodic advertisement train, which contains all necessary information for synchronizing to an ongoing BIS [20].

Synchronizing to an ongoing BIS. Upon reception of a periodic advertisement, a receiver can extract the Additional Controller Advertising Data (ACAD) field. The latter contains the `BIG Info` field [10], which gives insights about the Broadcast Isochronous Group (BIG) as well as its timing. Such BIG can be thought of a container, encapsulating one or more BIS events, as well as optional control information [6], as shown in Fig. 3. The timing of the next BIG event is often called the next anchor point (shown as rhombus in Fig. 2).

BIS structure and subevents. A BIS can be described as a sequence of subevents with strict order and timing. Depending on the actual ordering of the subevents, one can distinguish between an interleaved and sequential BIS arrangement. Fig. 3 illustrates a sequential subevent ordering, where subevents belonging to one BIS (i.e., S_{A1} or S_{A2}) make up a BIS event. Every BIG event, in turn, consists of one or more BIS events, equally spaced apart (BIS spacing). Fig. 3 also illustrates an optional *control* subevent (*C*), which is transmitted after the last BIS event. Due to the unidirectional nature of a BIS, such control subevent is considered a crucial component, as it enables the transmission of control information (e.g., hopping sequence, updates to the channel map, and details about the stream termination). Finally, BIG events (and therefore BIS events) repeat at a fixed interval called `ISO` Interval, and represent the actual isochronous data stream.

BIS packet structure. Each packet sent over an isochronous channel follows the Protocol Data Unit (PDU) structure illustrated in Fig. 4. In such an ISO PDU, the header field

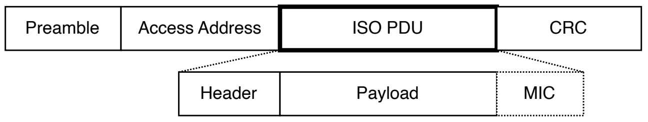

Figure 4. Isochronous physical channel PDU. Encapsulated in a BLE packet, the isochronous physical channel protocol data unit (ISO PDU) forms the LE air interface packet.

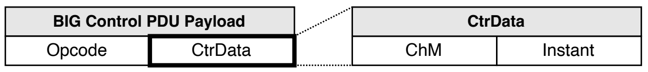

Figure 5. Channel map update info embedded in a BIG control PDU. Encapsulated in the BIG control PDU payload, the *CtrData* field specifies a new channel map (*ChM*) as well as the *instant* at which the change will be applied.

indicates whether it is a *BIS data PDU* or a *BIG control PDU*. The payload of the *BIS data PDU* contains the data to be transmitted (e.g., audio). The optional Message Integrity Check (MIC) field is used in case of an encrypted payload.

BIS robustness. In order to provide a reliable data stream without the usage of acknowledgment messages, a BIS can use strategies such as retransmissions or channel blacklisting. The former refer to the process of re-sending the same packet inside a BIS event, whereas the latter disables poor-performing channels inside the channel map. This procedure, also known as the broadcast isochronous *channel map update* procedure, allows a broadcasting device to specify a new channel map as well as the instant at which the specified channel map will be used. Fig. 5 illustrates the data structure for such update procedure, which is encapsulated in the payload of a *BIG control PDU*, and it is hence transmitted during the optional control subevent *C* shown in Fig. 3.

BIS security. A BIS can select one of three security modes. Depending on the selected mode, the stream can either be (i) unauthenticated and unencrypted[1], (ii) unauthenticated and encrypted or (iii) authenticated and encrypted. The so-called `Broadcast_Code`, a 16-octet parameter, can be used to derive a session key for encryption/decryption [6, 20][2]. The key derivation is based on the AES-CMAC algorithm,

[1]This mode is recommended for Auracast senders in public spaces [24].

[2]Receiver devices can check for the presence of a Group Initialisation Vector and Group Session Key Diversifier inside the `BIGInfo` and can, in combination with the `Broadcast_Code`, decrypt the data [6, 20].

and produces a symmetric session key that is identical on all receivers synchronized to a BIS [6]. As discussed in § 1, many applications envisage the acquisition of the `Broadcast_Code` through out-of-band methods such as QR codes or NFC tags displayed/placed at prominent locations. How to authenticate a BIS is left up to the implementer, and is not specified in the Bluetooth core specification.

3 Vulnerability of BISes

Even though BISes are designed to be secure, there are three aspects that can be exploited by a malicious actor to perform selective jamming of an ongoing stream, manipulate the (encrypted) payloads in BIS packets, or even fork the BIS, i.e., force a receiver to switch to a forged stream operating on a different channel map. We discuss these aspects next.

Plaintext metadata for BIS establishment. As discussed in § 2.2, any device that wants to synchronize to a BIS needs to follow a three-stage process: extracting the extended advertisement, synchronizing to the periodic advertisement train, and deriving the next anchor point (i.e., the timing and the frequency channel in which the next BIG event is transmitted). Although BIS payloads can be encrypted, the Bluetooth core specification does not foresee encryption of the metadata during the establishment process [6]. Therefore, any device within the communication range of the broadcaster can synchronize to the ongoing stream, and knows exactly at which time and on which frequency channel the next BIS event(s) will take place. A malicious actor can hence easily perform denial of service attacks, for example, by means of selective jamming, or by enforcing a stream termination with the transmission of a forged *BIG control PDU*. Moreover, in the case of unencrypted payloads in BIS packets, a malicious actor can also manipulate their content by sending forged packets at a higher transmission power (i.e., by overshadowing the original legitimate signal).

Out-of-band key exchange. Broadcast audio streams may be private: in this case, the payload of the *BIS data PDU* is encrypted using a `Broadcast_Code`, which is then used to derive a symmetric session key between the broadcaster and all intended receivers. The encryption of BIS packets ensures that a malicious actor, not possessing the `Broadcast_Code`, is unable to send forged packets without being noticed. Unfortunately, as discussed in § 1, many of the envisioned applications broadcasting audio in public spaces envisage an effortless *out-of-band* exchange of the `Broadcast_Code`. For example, one can prominently display a QR code or mount an NFC tag in dedicated areas (e.g., at a bus stop, or at the doctor's waiting room). In fact, when it comes to the distribution of the `Broadcast_Code`, the Bluetooth specification is vague (e.g., "it should be easy to do" [6, 20]), and the Auracast documents [4] leave it up to the implementer, recommending either a static value that is printed on a label, or a *non-static* value that is displayed on a screen/TV. However, this means that a malicious actor also has an easy access to the `Broadcast_Code`, and can make use of it to replace the encrypted payload sent in a BIS packet with a forged one.

Channel map updates. To ensure the successful delivery of forged packets, an attacker needs to be in close proximity to the receiver (or have the ability to send at a very high transmission power), as a requirement for the overshadowing attack to work is that the received, forged signal is stronger than the legitimate one. In practice, an attacker can bypass this limitation by leveraging the Broadcast Isochronous Channel Map Update procedure specified in the Bluetooth core specification for channel blacklisting. In fact, an attacker can issue a forged *BIG control PDU* containing a channel map update: this updates the list of the future frequency channels being used by the receiver (e.g., to be different from that of the original stream), hence giving the attacker the chance to fork the BIS. Indeed, the receiver will no longer use the original channel map (from the legitimate broadcaster), but instead use the one of the attacker. As a result, the receiver will inadvertently listen to the forged data, and the attacker will take control of the receiver without the need of being in close proximity or of transmitting a signal that is stronger than that of the original broadcaster.

4 `BISON`: Attack Overview

Following our observations in § 3 about the vulnerability of BISes, we design and implement the corresponding attack: `BISON`. The `BISON` attack comprises three distinct phases: **I** synchronization, **II** overshadowing and **III** forking. Fig. 6 illustrates these three phases, where Alice represents an isochronous broadcaster, Bob represents an isochronous receiver and Mallory represents a malicious actor.

Phase I: Synchronization. Receiver nodes in communication range of Alice can synchronize to the ongoing BIS by executing the periodic advertising synchronization establishment procedure followed by the broadcast isochronous synchronization establishment procedure. The latter uses the information contained in the periodic advertisement packets (i.e., `BIGInfo`) to determine the BIS timing as well as the channel used for the reception of the next data packet. A malicious actor such as Mallory, can exploit this procedure and calculate the timing as well as the channel used in future BIS events. In other words, Mallory can synchronize to the BIS broadcasted by Alice, and calculate when and on which channel Bob will listen in the future.

Phase II: Overshadowing. Mallory, knowing the future moves of Bob, can now increase the transmission power (or get in close proximity of Bob) and, due to the capture effect [23], prevent Bob from receiving the broadcasts transmitted by Alice. If Mallory transmits invalid packets, Bob is considered a victim to a selective denial of service (DoS) attack, whereas if Mallory transmits forged packets, Bob is considered a victim to an impersonation attack. Mallory can also exploit the out-of-band key material exchange and forge the content of encrypted BIS packets. Additionally, a control subevent can be transmitted, indicating a channel map update, telling Bob that the ongoing BIS will use a different channel map starting at a given instant.

Phase III: Forking. As soon as the instant specified in the control subevent (*BIG control PDU*) used to indicate a channel map update is reached, Mallory and Bob will use the specified channel map. This update will ultimately allow Mallory to diverge from the BIS broadcasted by Alice, due to the influence of the channel map on the channel selection algorithm #2 [6], used for the channel calculation. At this

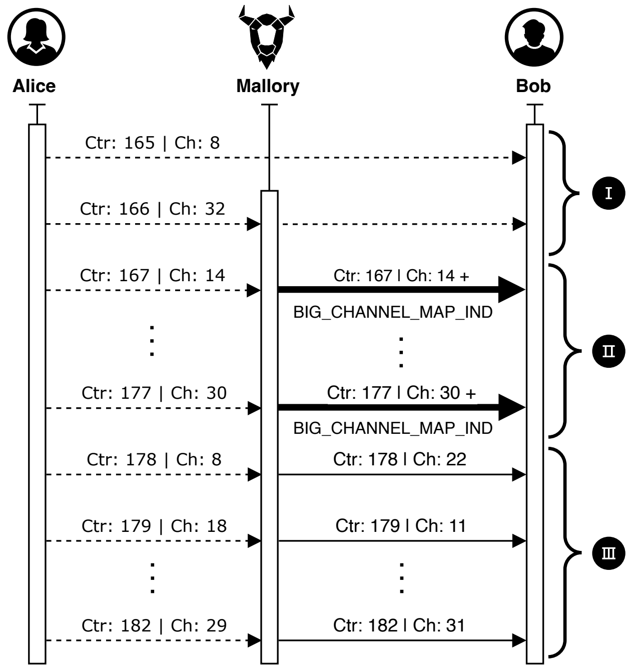

Figure 6. BISON architecture. In attack phase I, Mallory synchronizes to the ongoing BIS, broadcasted by Alice. In phase II, Mallory overpowers the BIS and adds a channel map update subevent. As soon as the specified instant is reached (phase III), Bob uses the updated channel map and will no longer listen to the BIS broadcasted by Alice.

point, Mallory does not need to overpower Alice anymore, resulting in Bob undoubtedly following Mallory, under the impression that Mallory is Alice.

5 BISON in Action

We demonstrate the effectiveness of BISON on two Nordic Semiconductor nRF5340 audio development kits, representing Alice and Bob, and one nRF52840 development kit representing Mallory, as illustrated in Fig. 6. All three development kits utilize the Zephyr real-time operating system [34], where simplified versions of the *broadcast audio souce* and *broadcast audio sink* samples demonstrate Alice transmitting a broadcast audio stream and Bob synchronizing to it. Mallory, which we release as open-source[3], uses a modified implementation of the *synchronized receiver* sample, including Zephyr's BLE controller adaptations, executing all phases of the BISON attack, as described in § 4. To highlight the potential real-world impact of BISON, we reconstruct an envisaged Auracast application, and showcase the attack illustrated in Fig. 7. Specifically, we consider a scenario in which Mallory passes by an ongoing broadcast audio stream, overtakes it with BISON using the same transmission power of Alice, and – after BISON's execution – is able to walk away from Bob) while still making sure that Bob receives its forged packets and not those from Alice. Note that this is possible thanks to the devices' inherent clock inaccuracies and the fact that isochronous receivers adjust their clock drift on the basis of the *ongoing* stream: even if Alice would

[3]https://iti.tugraz.at/bison

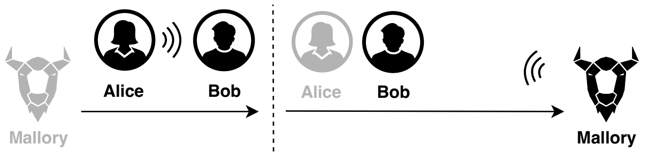

Figure 7. Setup of the video demonstrating BISON in action. Left: Mallory starts the attack while moving towards Bob. Right: after BISON forked the channel map, Mallory can still impersonate Bob despite physically moving away.

transmit a packet in the same channel used by Mallory, Bob would remain synchronized to Mallory. We include a video showing the attack with the BISON code and documentation[3].

6 Countermeasures & Discussion

Although it is well-known that a symmetric key used by multiple devices facilitates impersonation attacks [25], BISON shows that even new standards, such as Bluetooth v5.2, suffer from this vulnerability. As a result, all users of broadcast audio applications in public spaces relying on the authenticity of a BIS may be affected by the suboptimal definition of the `Broadcast_Code` exchange procedure, as discussed in § 3. This, in combination with the fact that the metadata needed for the broadcast isochronous synchronization establishment procedure as well as the fact that *BIG control PDUs* are transmitted in plaintext, allows for selective DoS attacks, assuming the attacker can overshadow the original BIS.

Nevertheless, BISes allow for a multitude of novel use cases and should be considered a *revolutionary and highly valuable technology*, especially when deployed in a secure fashion. Therefore, this section lists several countermeasures to limit the BISON attack surface.

Improved `Broadcast_Code` exchange. Similar to a CIS, a BIS could employ a separate BLE connection for the exchange of key material, and terminate the connection as soon as the exchange is finished. Whilst the use of a separate connection would affect the unidirectional nature of a BIS, we believe that this process should be enforced in all Auracast applications [4, 7, 9].

Asymmetric encryption. When leveraging a separate connection, one could replace the symmetric key derived with the `Broadcast_Code` with an asymmetric key in order to protect from impersonation attacks. However, one needs to take into account that the usage of asymmetric cryptography (i.e., digital signatures) introduces additional computational complexity, which may not be practical or feasible on resource-constrained low-power embedded devices.

Disabling channel map updates. A receiver could disable channel map updates altogether. However, doing this would preclude compliance to the Bluetooth core specification.

Monitoring physical features. Similar to BlueShield [33], a system utilizing physical features to detect spoofing attacks, a receiver could monitor the received signal strength (RSS) of packets to detect overshadowing attempts. Whilst it is possible to detect abnormal changes in the RSS caused by a device suddenly sending packets at a much higher transmission power, this may not be trivial in mobile settings, due to the vagaries of signal propagation in complex environments.

7 Related Work

Several BLE vulnerabilities have been disclosed over the years, leading to a series of possible attacks. Projects such as Ubertooth [27], BtleJack [12] and SPADE [28] highlight the impact of sniffing attacks, and exploit the plaintext transmission of important metadata such as the preamble, access address, or extended advertisement SYNC information. BtleJack and SPADE exploit this metadata to jam certain packets, and in the case of BtleJack, even highjack the BLE connection between two devices. Building on sniffing attacks, GATTacker [21] or BTLEjuice [11] go one step further by executing a man-in-the-middle (MITM) attack, impersonating the target peripheral, allowing not only for data sniffing, but also for data manipulation. Moreover, InjectaBLE [13] exploits the window widening, used for sleep clock inaccuracy compensation, to inject packets at the beginning of the window widening event to highjack even ongoing connections.

In contrast to sniffing and impersonation attacks, device-tracking attacks allow for location-based user tracking, invading the user privacy [16]. Although Bluetooth implements the so called privacy feature, allowing for frequent device address changes, manufacturers often unintentionally use data in advertisement packets that can be leveraged to uniquely identify users [16]. Even when implemented correctly, the BLE traffic transmitted by wearable fitness trackers may represent the gait of a person, allowing for unique user identification [17].

Furthermore, multiple attacks on the pairing procedure as well as key establishment procedure were disclosed. Attacks such as KNOB [2] or BlueMirror [14] are used to attack the pairing process, while BLURtooth [3] or BLESA [32] exploit already paired devices. Moreover, attacks such as BlueDoor [30] or Sweyntooth [19] even combine multiple vulnerabilities, allowing not only for privacy invasion, but also for remote code execution in certain scenarios.

Although a multitude of attacks on BLE have been disclosed, no attack on isochronous channels has been published yet. In general, research on isochronous channels is at its infancy and currently only reviews how isochronous channels work. BISON *is the first study on the vulnerability on BISes and provides detailed insights about the attack surface, exemplary attack scenarios, as well as possible countermeasures.*

8 Conclusions and Future Work

In this paper, we have presented BISON, a novel attack on BISes showing that broadcasting applications envisaged to be implemented in a multitude of public settings fail to provide confidentiality, integrity and authenticity. After raising awareness about the vulnerability of BISes, we provide an open-source implementation of the attack and describe possible countermeasures. We hope that our work serves the community as a basis to revise the BIS specification and Auracast recommended best practices, towards the creation of secure applications broadcasting audio in public spaces.

Future work includes the extension of BISON to exploit audio data, demonstrating the impact on LE Audio certified devices, as well as an evaluation of different mitigation strategies to minimize the BISON attack surface.

9 References

[1] M. Afaneh. The Ultimate Guide to What's New in Bluetooth version 5.2. https://tinyurl.com/2wennsyk. Last access: Mar 2023.

[2] D. Antonioli et al. The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR. In *Proc. of the 28th USENIX Security Symposium*, 2019.

[3] D. Antonioli and other. BLURtooth: Exploiting Cross-Transport Key Derivation in Bluetooth Classic and Bluetooth Low Energy. In *Proc. of the ASIA CCS Conf.*, 2022.

[4] Bluetooth SIG. Auracast Simple Transmitter Best Practices Guide, v1. Oct 2022.

[5] Bluetooth SIG. Basic Audio Profile, v1.0.1. Jun 2022.

[6] Bluetooth SIG. Bluetooth Core Specification, v5.4. Jan 2023.

[7] Bluetooth SIG. Broadcast Audio Scan Service, v1.0. Sep 2021.

[8] Bluetooth SIG. Key Use Cases in Public Locations. https://www.bluetooth.com/auracast/public-locations. Last access: Mar 2023.

[9] Bluetooth SIG. Public Broadcast Profile, v1.0. Jul 2022.

[10] Bluetooth SIG. Supplement to the Bluetooth Core Spec., v11. 2023.

[11] D. Cauquil. BtleJuice Framework. https://github.com/DigitalSecurity/btlejuice. Last access: Mar 2023.

[12] D. Cauquil. You'd better secure your BLE devices or we'll kick your butts! https://bit.ly/3nwHZOr. Last access: Mar 2023.

[13] R. Cayre et al. InjectaBLE: Injecting Malicious Traffic into Established BLE Connections. In *Proc. of the 51st DSN Conf.*, 2021.

[14] T. Claverie and J. L. Esteves. BlueMirror: Reflections on Bluetooth Pairing and Provisioning Protocols. In *Proc. of the IEEE SPW*, 2021.

[15] S. Cohen. Auracast looks to radically change how you use Bluetooth. https://bit.ly/3ziihQ3. Last access: Mar 2023.

[16] M. Cäsar, T. Pawelke, J. Steffan, and G. Terhorst. A survey on Bluetooth Low Energy security and privacy. *Comput. Netw.*, 205(C), 2022.

[17] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra. Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers. In *Proc. of the 17th HotMobile Worksh.*, 2016.

[18] F. Dugand. Unmute the World With Auracast Broadcast Audio. https://bit.ly/3JRquzI. Last access: Mar 2023.

[19] M. E. Garbelini et al. SweynTooth: Unleashing Mayhem over Bluetooth Low Energy. In *Proc. of the USENIX ATC*, 2020.

[20] N. Hunn. *Introducing Bluetooth LE Audio*. 2022.

[21] S. Jasek. GATTacking Bluetooth Smart Devices. https://gattack.io/whitepaper.pdf. Last access: Mar 2023.

[22] G. Kimathi. Auracast Broadcast Audio: The New Way To Share Audio. https://tinyurl.com/yc23uuch. Last access: Mar 2023.

[23] K. Leentvaar and J. Flint. The Capture Effect in FM Receivers. *IEEE Transactions on Communications*, 24(5), 1976.

[24] J. Marcel. Answers to Commonly Asked Questions About Auracast Broadcast Audio. https://bit.ly/42TtSCJ. Last access: Mar 2023.

[25] J. Padgette, J. Bahr, M. Batra, R. Smithbey, L. Chen, and K. Scarfone. Guide to Bluetooth Security. NIST 800-121 Rev. 2. Jan 2022.

[26] K. Ren. 10 Frequently Asked Questions on LE Isochronous Channels. https://bit.ly/40kDDbz, 2020.

[27] M. Ryan. Bluetooth: With Low Energy Comes Low Security. In *Proc. of the 7th USENIX WOOT*, 2013.

[28] D. Ryoo et al. SPADE: Secure Periodic Advertising using Coded Time-Channel Rendezvous for BLE Audio. In *Proc. of the 19th DCOSS-IoT Conf.*, 2023.

[29] The OnQ Team. LE Audio: A new age of Bluetooth audio sharing. https://bit.ly/3lRcYnP. Last access: Mar 2023.

[30] J. Wang et al. BlueDoor: Breaking the Secure Information Flow via BLE Vulnerability. In *Proc. of the 18th MobiSys Conf.*, 2020.

[31] M. Woolley. Bluetooth Core Specification Version 5.2 Feature Overview, v1.0.1. Dec 2020.

[32] J. Wu et al. BLESA: Spoofing Attacks against Reconnections in Bluetooth Low Energy. In *Proc. of the 14th USENIX WOOT*, 2020.

[33] J. Wu et al. BlueShield: Detecting Spoofing Attacks in Bluetooth Low Energy Networks. In *Proc. of the 23rd RAID Symp.*, 2020.

[34] Zephyr Project. About the Zephyr Project. https://zephyrproject.org/learn-about. Last access: Mar 2023.

[35] A. Zignani. LE Audio, Auracast Broadcast Audio, and the Future of Bluetooth Audio. https://bit.ly/3lSxIvg. Last access: Mar 2023.