Security and Malware Vulnerabilities in Bluetooth Low Energy

CAP 6135 - Malware Software Vulnerability

Glaspey, Joshua
dept. of Electrical and Computer Engineering
University of Central Florida
Orlando, United Stated
jo248954@ucf.edu

Green, Alexander
dept. of Electrical and Computer Engineering
University of Central Florida
Orlando, United Stated
al101178@ucf.edu

***Abstract*— Bluetooth Low Energy (BLE) has become a foundational component of modern wireless communication, enabling a wide range of devices, particularly in the Internet of Things (IoT) and wearable technology sectors. Designed for low power consumption, BLE supports applications ranging from healthcare devices to smart home systems. However, the widespread adoption of BLE has also introduced significant security challenges. This report provides a comprehensive survey of the security vulnerabilities inherent in BLE, focusing on various attack methods, including eavesdropping, device cloning, denial-of-service (DoS) attacks, and cryptographic weaknesses, among many others. The report explores the impact of these vulnerabilities on real-world applications, particularly in sensitive sectors such as healthcare, personal security, and home automation. It also examines the mitigation strategies that can be employed to safeguard BLE-enabled devices against these threats. By drawing from the latest research and practical findings, the report offers a broad view of the evolving BLE security landscape. The contents of this report are organized into sections that cover specific attack types, their consequences, and proposed countermeasures.**

I. Introduction

Bluetooth Low Energy (BLE) has rapidly evolved into a foundational communication protocol, enabling a wide range of devices to connect wirelessly with minimal energy consumption. Initially introduced in 2010 as part of the Bluetooth 4.0 specification, BLE was specifically designed to address the growing need for low-power communication in devices that rely on battery life, such as health monitors, fitness trackers, smartwatches, home automation systems, and a variety of IoT devices [10][14].

The rapid increase in IoT devices has been a significant factor in BLE's success. By 2022, over 15 billion connected IoT devices were projected to be in use globally, many of which leverage BLE for their communication needs [14]. BLE offers several benefits, including its ability to transmit data over short distances, typically up to 100 meters, while maintaining low energy consumption through its time-sliced operation. This makes it ideal for applications where data needs to be exchanged infrequently but reliably, such as environmental monitoring systems, medical devices, and smart home technology [12].

Despite its numerous advantages, the widespread adoption of BLE has also introduced significant security challenges. BLE's inherent vulnerabilities stem from its design, which prioritizes ease of use and energy efficiency over robust security features. For example, BLE devices often rely on simple pairing methods that do not offer sufficient protection against eavesdropping or device impersonation attacks [13]. These security gaps, combined with the growing sophistication of attackers and the increasing integration of BLE into sensitive applications, have made BLE a prime target for cyber threats.

Eavesdropping, in particular, is one of the most common and dangerous types of attacks on BLE networks. Attackers can intercept and listen to unencrypted communications between BLE devices, potentially gaining access to sensitive data such as personal information, login credentials, or financial details [9]. Furthermore, the simplicity of the Bluetooth pairing process

leaves it vulnerable to attacks like PIN cracking, where an attacker can bypass weak authentication mechanisms to gain unauthorized access to a device [13]. These types of attacks can pose serious privacy and security risks, especially in environments like healthcare and home automation, where BLE devices are used to transmit sensitive data and control critical systems.

Another critical vulnerability in BLE is the risk of Denial-of-Service (DoS) attacks, including a relatively recent phenomenon known as Denial of Sleep (DoSL). This attack exploits the low-power nature of BLE by keeping devices in an active state, draining their battery and rendering them inoperable [11][12]. Such attacks can be damaging in large-scale IoT networks, where a well-coordinated DoSL attack could incapacitate a significant portion of a sensor network, leading to service disruptions and operational failures.

Moreover, as BLE-enabled devices become increasingly integrated into everyday life, the potential consequences of security breaches grow more severe. For example, BLE-based trackers such as Apple's AirTags and Samsung's SmartTags, which are designed to help users locate misplaced items, have raised privacy concerns due to their vulnerability to signal spoofing and unauthorized tracking [8]. In these devices, the communication and tracking data exchanged via BLE are not always adequately protected, making it easier for malicious actors to manipulate the system for purposes such as location spoofing or stalking.

This report seeks to provide a comprehensive survey of the security vulnerabilities inherent in Bluetooth Low Energy. It will explore various types of attacks targeting BLE devices, categorize them based on their impact, and examine mitigation strategies that can help protect users and systems from these threats. By drawing from recent research and practical findings, this survey aims to present a holistic understanding of BLE security risks.

The report is structured as follows:

I. **Introduction**

II. **Overview of Attacks and Vulnerabilities** will introduce each type of attack and security vulnerability, and categorize individual methods into categories. This will be complemented by a large visualization.

III. **Passive Eavesdropping Attacks** will define and explain this attack type, discuss its mitigation, and explore real-world applications.

IV. **Active Eavesdropping Attacks** will address the methods attackers use to intercept and manipulate BLE communications, along with countermeasures.

V. **Cryptographic Vulnerabilities** will examine weaknesses in BLE's cryptographic protocols and provide solutions to enhance encryption.

VI. **Device Cloning Methods** will focus on impersonation attacks, highlighting how attackers clone BLE devices and strategies for preventing this.

VII. **Distortion Methods** will investigate attacks that corrupt data and communication, along with strategies to safeguard data integrity.

VIII. **DoS Attacks** will cover the impact of Denial-of-Service attacks, including Denial of Sleep, and explore preventive techniques.

IX. **Surveillance Attacks** will discuss attacks aimed at tracking and spying on users through BLE devices, along with privacy-enhancing measures.

X. **Miscellaneous Attacks** will touch on less common but still critical BLE vulnerabilities.

XI. **Conclusion** will summarize the findings and propose directions for future improvements to BLE security.

II. Overview of Attacks and Vulnerabilites

BLE security encompasses a broad range of vulnerabilities and attack vectors that can compromise data integrity, confidentiality, and device availability. These vulnerabilities are categorized into eight primary groups: passive eavesdropping attacks, active eavesdropping attacks, device cloning methods, DoS attacks, cryptographic vulnerabilities, distortion methods, surveillance attacks, and miscellaneous attacks. Each category further branches into specific attack types that highlight different tactics and potential

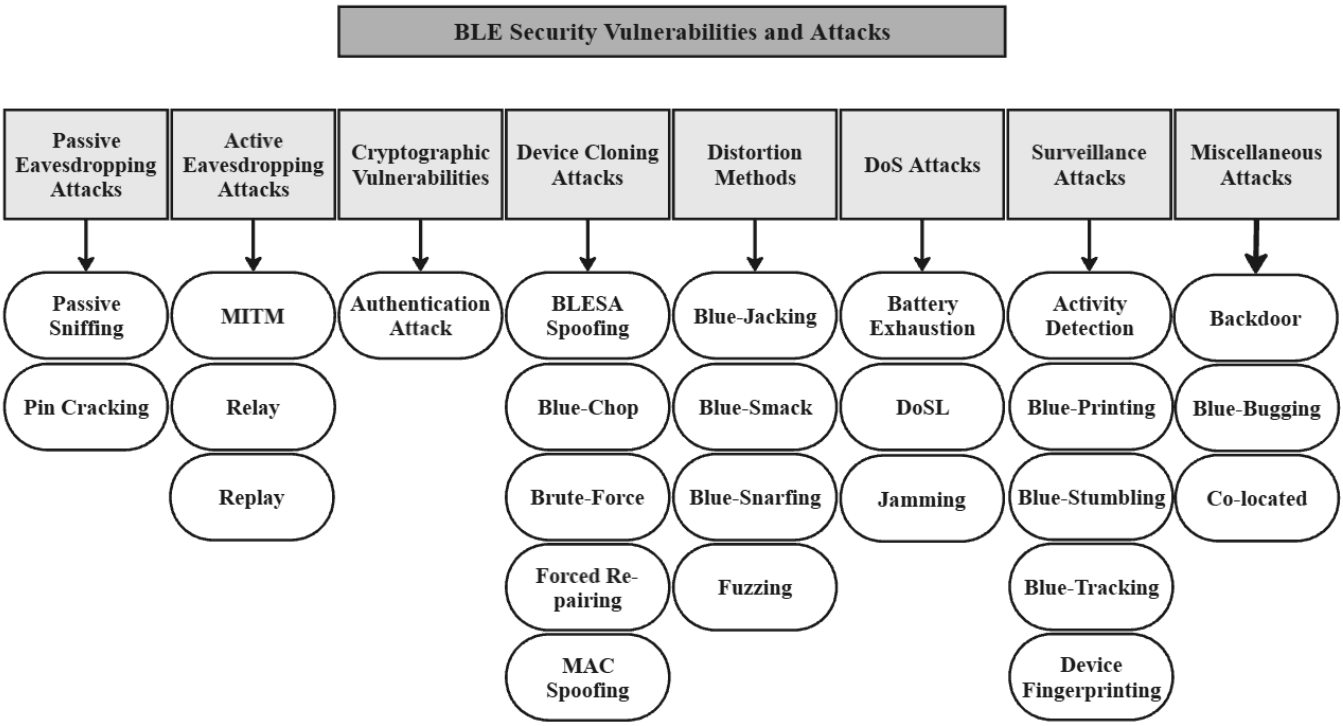

Figure 1: *BLE Security Vulnerabilities and Attacks diagram. Children nodes are below each parent node, and categorized through arrows.*

risks associated with BLE communication (Figure 1).

The following sections of this report will delve deeper into each category identified in the diagram, providing detailed definitions, explanations, mitigation techniques, and real-world examples to comprehensively address the BLE security landscape.

III. Passive Eavesdropping Attacks

A. *Definition*

Passive eavesdropping attacks represent a significant threat to BLE communication due to their stealthy nature, which enables attackers to intercept data without alerting the targeted devices. These attacks involve monitoring wireless communications without injecting or altering data, making them virtually undetectable. Because many BLE implementations prioritize power efficiency and simplicity over strong security, they often omit robust encryption or rely on outdated pairing procedures. As noted by Barua et al., even minor implementation oversights can lead to substantial leaks of private data such as personal identifiers, credentials, and health metrics [6].

By quietly observing and recording transmissions, especially during pairing or reconnection events—attackers can derive encryption keys, extract unencrypted data, or gather metadata used in surveillance or impersonation attacks. These risks are amplified in IoT and wearable systems where BLE is often used to transmit sensitive or continuous telemetry without adequate encryption [10].

This section will cover both passive sniffing attacks and PIN cracking.

B. *Passive Sniffing*

Passive sniffing is the primary form of passive eavesdropping against BLE devices. The attack targets data broadcast during advertisement, pairing, or connection phases of communication. BLE operates in the 2.4 GHz ISM band and uses adaptive frequency hopping across 40 channels. Although this hopping pattern is designed to improve reliability and reduce interference, tools such as Ubertooth One or software-defined radios (SDRs) are capable of tracking and recording these transmissions. Once synchronized to the hop pattern, the attacker can extract broadcasted metadata, service identifiers, and even raw user data. As explained by Cäsar et al., these sniffers can often capture sensitive communication

because BLE encryption is not enforced at the protocol level and must be explicitly implemented and configured by the device manufacturer [10].

BLE's pairing process is a common target for sniffing. In many legacy or low-cost devices, the Just Works pairing method is used. This method lacks authentication, meaning any nearby listener can intercept the key exchange process and, in some cases, derive encryption keys or continue monitoring the session. Farhan and Abosata [9] demonstrated this vulnerability by capturing data packets from BLE devices during pairing and extracting readable information from unprotected GATT operations. Their tests confirmed that even without disrupting the communication, a sniffer could capture complete message flows from poorly secured devices.

Another technical vulnerability stems from static MAC addresses. If a device does not implement address randomization, its identity can be persistently tracked across sessions and environments. Static identifiers, combined with plaintext metadata, allow adversaries to reconstruct usage patterns and user behaviors over time. This concern was also emphasized in Barua et al., who pointed out that even small implementation oversights—such as reusing addresses or using unencrypted reads—can lead to meaningful data leaks [6].

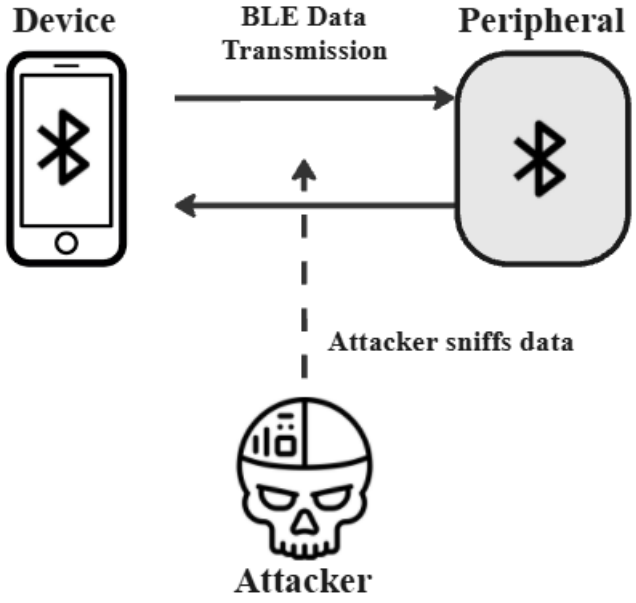

Figure 2: *Passive sniffing. The attacker captures BLE advertisement, connection, and GATT data packets without interference of the original signal.*

1) Mitigation

The most effective mitigation against passive sniffing is the use of BLE Secure Connections, introduced in version 4.2 of the specification. Secure Connections use Elliptic Curve Diffie-Hellman (ECDH) to establish authenticated and encrypted sessions, preventing attackers from capturing usable key material during pairing. Devices should avoid insecure pairing modes like Just Works unless absolutely necessary and instead employ authenticated modes such as Numeric Comparison or Passkey Entry. Beyond pairing, every data characteristic exposed over BLE should require encryption and authentication for access.

Cäsar et al. [10] emphasize that developers should ensure their devices use address randomization to prevent long-term tracking. Static MAC addresses must be avoided, and devices should rotate their public identifiers regularly. Proper firmware implementation is critical—supporting encryption in theory is not enough if keys are not securely stored or encryption is inconsistently applied. Regular key rotation and session expiration can further reduce the window of vulnerability. These mitigations, while well-supported in the BLE specification, are often poorly adopted in practice, making it essential for developers and manufacturers to prioritize their correct and consistent implementation [6].

C. PIN Cracking

PIN cracking attacks target the weak or predictable Personal Identification Numbers (PINs) used during the BLE pairing process. As shown by Shaked and Wool, early versions of Bluetooth allowed pairing using short numeric PINs, often 4 digits long, which could be brute forced in fractions of a second [13]. During pairing, devices combine this PIN with public information—such as random nonces and Bluetooth addresses—to derive the encryption key known as K_init using the E22 function of the SAFER+ cipher. Because the nonces and addresses are transmitted in the clear, an attacker who captures this data via passive sniffing needs only to guess the PIN.

The attack proceeds by iterating through all possible PIN values, computing the expected

K_init for each, and comparing it with the captured result. With only 10,000 potential 4-digit PINs, the key can be recovered in seconds. This not only reveals the original pairing secret but also enables attackers to decrypt ongoing or future communications, impersonate the device, or mount further attacks such as MITM [13]. These vulnerabilities are particularly dangerous in scenarios where devices reuse PINs or fail to upgrade to stronger authentication protocols.

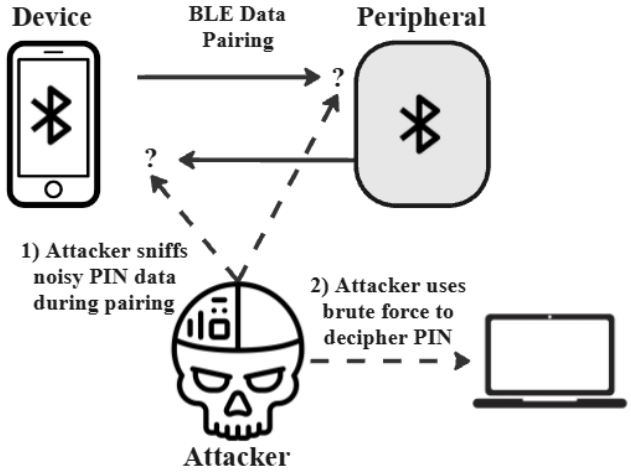

Figure 3: *PIN Cracking. The attacker passively listens during BLE pairing and then performs an offline brute-force attack on the intercepted pairing exchange to recover the short PIN and derive the session key.*

1) Mitigation

Mitigation strategies for PIN cracking begin with avoiding legacy pairing protocols altogether. Modern BLE specifications introduce Secure Simple Pairing (Bluetooth 2.1) and Secure Connections (Bluetooth 4.2), which use ECDH-based key exchanges that do not rely on low-entropy PINs. Device manufacturers should enforce these modern mechanisms and eliminate Just Works [6] or fixed-PIN options where possible. Additionally, implementing random PINs of greater length and requiring user confirmation through passkey entry or numeric comparison significantly reduces feasibility of brute-force attacks [4].

IV. Active Eavesdropping Attacks

A. Definition

Active eavesdropping attacks extend beyond simply listening to BLE communication—they involve intercepting, modifying, or injecting packets into the communication channel. These attacks aim to manipulate the behavior of devices, impersonate participants, or gain unauthorized access by tampering with data streams. Unlike passive attacks, which rely on the absence of encryption or weak pairing protocols, active eavesdropping actively violates the integrity and authenticity of a BLE connection. As explained by Cäsar et al., this type of attack is significantly more intrusive and can cause greater harm, especially if cryptographic protections are bypassed or downgraded [10].

BLE's focus on low power consumption and ease of connectivity, especially during its pairing and reconnection processes, often opens the door for these attacks. Adversaries can exploit insecure pairing methods or implementation flaws to conduct attacks such as Man-in-the-Middle (MITM), relay, or replay attacks, each of which can yield full control over the BLE session or leak sensitive information [2].

These attacks are especially potent in real-time applications such as smart locks, medical telemetry, and mobile payments, where authentication or command integrity is critical. Devices that do not validate the freshness, timing, or authenticity of packets beyond the initial pairing step are particularly vulnerable to active threats.

This section covers MITM attacks, relay attacks, and replay attacks.

B. Man-In-The-Middle (MITM)

A MITM attack in BLE involves an adversary positioning themselves between two devices during the pairing or communication process, acting as an invisible proxy that relays and potentially alters traffic between the legitimate parties. BLE is particularly susceptible to this type of intrusion when Just Works pairing is used, as it lacks authentication checks to verify identities during initial connection. Once a connection is hijacked, the attacker can intercept sensitive data, inject malicious commands, or impersonate a device to either endpoint, all without the knowledge of the original communicating parties. This is often facilitated by tools like Btlejuice, which function as a BLE man-in-the-middle framework, capable of simultaneously managing two BLE connections and forwarding messages

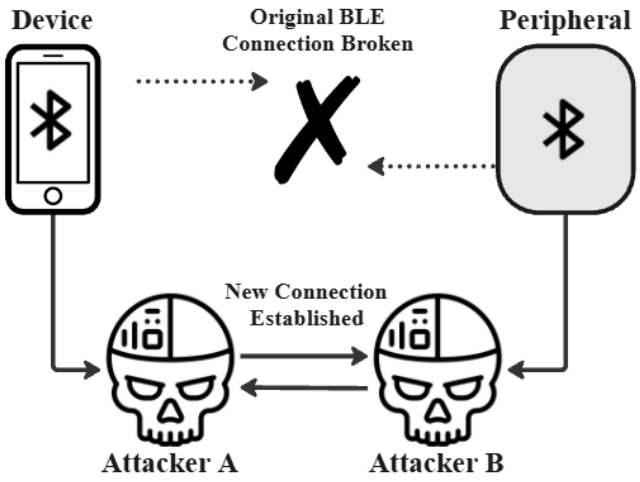

Figure 4: *Man-in-the-Middle attack. The attacker intercepts communication between the primary device and peripheral by spoofing both sides. The attacker performs two pairings - one for the device (Attacker A) and one for the peripheral (Attacker B) - while relaying data to avoid detection. This is enabled by weak or absent authentication during pairing.*

between them with full visibility and control over the traffic [14].

The technical procedure begins during the BLE pairing phase. The attacker monitors for nearby devices advertising connection availability. Once two devices initiate a pairing handshake, the attacker rapidly establishes connections with both—posing as the peripheral to the central device, and vice versa. This can be done using libraries such as pygatt or BlueZ and open-source tools like Btlejack or Btlejuice, which offer fine-grained packet-level access. The attacker relays L2CAP packets (Layer 2 BLE data) between devices and logs or modifies payloads on-the-fly. Vulnerabilities are particularly present in BLE versions prior to 4.2, where secure connections and MITM protection were optional or inconsistently implemented. Because Just Works pairing does not require numeric comparison or user authentication, an attacker faces no challenge in inserting themselves into the pairing process.

The threat of MITM attacks lies in their total compromise of confidentiality and integrity. Once in position, an attacker can steal keys during the initial pairing process, maintain long-term surveillance of sensitive health or authentication data, or alter packets to inject malicious instructions. This is especially dangerous in environments such as hospitals, where BLE is used for medical device communication, or smart homes where unauthorized access to locks or sensors could result in physical security breaches. Worse still, because attackers forward packets, the two legitimate devices may never detect that their communication has been compromised [6].

1) Mitigation

To mitigate MITM attacks in BLE environments, devices should employ secure pairing modes that include authentication verification, such as Numeric Comparison, Passkey Entry, or Out-of-Band (OOB) pairing. BLE Secure Connections, introduced in Bluetooth 4.2, leverage Elliptic Curve Diffie-Hellman (ECDH) key exchange to ensure that even if pairing messages are intercepted, the attacker cannot compute the final key. When Numeric Comparison is used, both devices display a six-digit code that the user must confirm visually, preventing MITM interference. Likewise, OOB pairing utilizes alternative channels like NFC to exchange public keys securely, removing the attacker's ability to intercept critical pairing data [10].

Beyond secure pairing, systems should employ regular session key updates and encrypted link-layer communication with authenticated encryption such as AES-CCM. BLE implementations should ensure that reconnections are not automatically accepted unless authenticated keys match. Monitoring and logging tools can also be used to detect suspicious pairing events or simultaneous dual-role device behavior (as in Btlejuice usage). Overall, security-conscious implementation of BLE pairing profiles and continuous authentication checks are essential to minimize the feasibility of MITM intrusions in sensitive environments.

C. Relay

Relay attacks manipulate proximity assumptions in BLE communication by introducing intermediary devices that transmit messages between the central and peripheral devices across larger-than-expected distances. These intermediary devices, sometimes called "ghost devices," allow an attacker to appear as if they are within acceptable physical proximity to a trusted device, enabling actions like unlocking

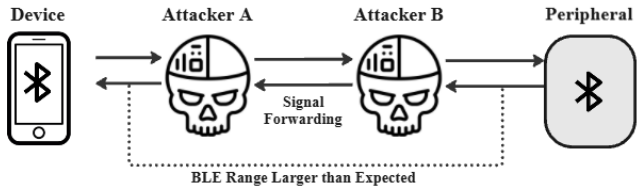

Figure 5: *Relay attack. Two attackers act as radio relays, extending BLE range by forwarding messages between a distant legitimate central and peripheral device. Since BLE pairing and authentication are not aware of physical distance or latency, the devices proceed as if they're directly connected.*

BLE-secured doors or confirming payments. BLE systems that rely solely on signal presence and strength to confirm proximity are highly vulnerable to such deception, as they do not distinguish whether the signal has traveled indirectly through a relay. This flaw has been demonstrated in BLE-based access systems, including automotive and building entry devices [10].

At a technical level, relay attacks are implemented using two cooperating devices: one near the target device (usually a peripheral) and another near the legitimate user device (usually a smartphone). These relay devices are programmed to transmit all received BLE packets in real-time over a separate communication channel (e.g., Wi-Fi or a long-range radio link), thereby allowing communication to occur over distances far exceeding BLE's intended operating range. Tools such as Relay Attack Toolkit or custom Python scripts using the BlueZ stack and HCI interface allow attackers to transparently forward advertisement, pairing, and data packets without modifying their contents. Because BLE lacks timing-based proximity verification or distance bounding protocols, it cannot detect the latency introduced by the relay, enabling the attacker to bypass physical access restrictions.

Relay attacks are dangerous because they exploit the foundational BLE design assumption that two devices within communication range are physically near one another. BLE is widely used in keyless entry systems for cars, smart locks in homes, and secure authentication in mobile apps. A successful relay attack in any of these scenarios could result in theft, unauthorized access, or identity compromise. Unlike other BLE attacks, relay attacks do not require decrypting traffic or spoofing device credentials, making them exceptionally accessible to attackers with minimal expertise and equipment [6].

1) Mitigation

To mitigate relay attacks, BLE systems must move beyond simple signal detection and incorporate proximity verification strategies. One approach is the use of Received Signal Strength Indicator (RSSI) filtering, but this is often unreliable due to environmental variability. More advanced defenses involve round-trip time measurements to estimate distance between communicating devices. By precisely measuring the time it takes for a message to be sent and acknowledged, the system can infer physical distance and reject connections that exceed acceptable bounds.

In addition, BLE devices should enforce strict connection whitelisting and pairing authorization, such that devices must be authenticated prior to enabling sensitive operations like unlocking or payment confirmation. Pairing data should be stored securely and periodically invalidated, ensuring that malicious relays cannot exploit stale credentials. Future implementations may also consider ultra-wideband (UWB) support alongside BLE, which provides centimeter-level distance estimation, thereby eliminating reliance on BLE alone for proximity-sensitive functions.

D. Replay

Replay attacks involve capturing legitimate BLE communication packets and retransmitting them at a later time without modification. In replay attacks, attackers exploit the trust that devices have in previously authenticated data, reusing old messages to gain unauthorized access, trigger certain actions, or disrupt normal operations. For instance, replaying captured unlock commands could permit unauthorized access to physical locations or sensitive data resources repeatedly [10].

Replay attacks technically rely on capturing valid BLE messages, often through passive or active eavesdropping. Once these messages are captured, attackers store them for future retransmission. The vulnerability exploited here is the lack of freshness checks or timestamps within the BLE communication protocol, which

fails to distinguish between new and old messages. As a result, BLE devices accept replayed packets as legitimate, thereby enabling unauthorized or malicious activities.

The threat from replay attacks stems from their simplicity and repeatability. Attackers do not need to understand, modify, or even decrypt the packets they capture. If BLE devices accept repeated messages without verifying their freshness or uniqueness, replay attacks become a powerful method to force device behavior remotely. For instance, if an attacker replays a "door unlock" command, the door may open even though the original sender is absent. In healthcare systems, replaying telemetry packets could pollute patient data logs with misleading values, leading to dangerous outcomes [1].

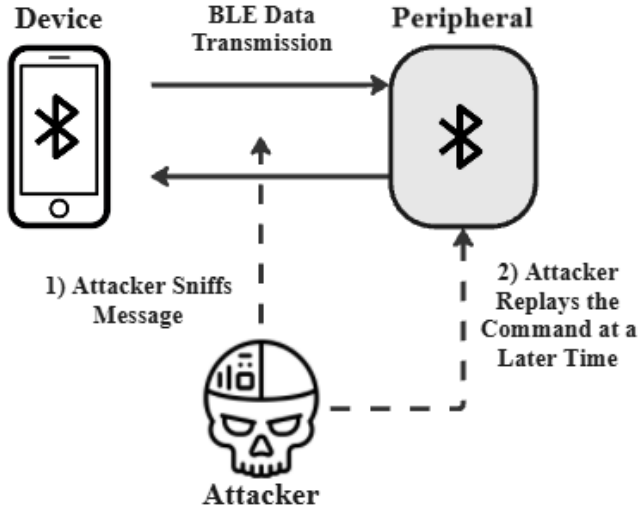

Figure 6: *Replay attack. The attacker records a legitimate BLE packet during a valid session, then replays it later to trigger the same action. Without freshness checks like timestamps or nonces, the device accepts the replayed message as valid.*

1) Mitigation

Mitigating replay attacks requires BLE systems to implement strong message freshness guarantees. This can be done by including nonces—unique random values appended to every message—that the receiving device checks for duplication. If the nonce has been seen before, the message is discarded. Likewise, timestamps can be used to enforce time-based validity windows, though this approach requires loosely synchronized clocks between devices and may not be suitable for all ultra-low-power systems.

A complementary mitigation approach is the use of session-based tokens or message counters. In this strategy, each session is initialized with a unique key or counter value, and each subsequent message must increment the counter. If a message with an old counter is received, it is rejected as a replay. Pairing this with AES-CCM authenticated encryption ensures not only confidentiality but also integrity and freshness of the communication. Developers should be encouraged to use BLE 5.x features that support such mechanisms by default, and vendors should rigorously test reconnection and session-handling logic to ensure that replay windows are closed effectively [10].

V. Cryptographic Vulnerabilities

A. Definition

Cryptographic Vulnerabilities in BLE are generated primarily from weak key management, insecure pairing modes, or poorly implemented encryption schemes. Older BLE pairing methods often fail to provide authentication, like "Just Works" pairing. This can leave devices exposed to passive eavesdropping or active MITM attacks. Secure Connections, which was introduced in BLE v4.2, often fail to legacy pairing due to backward compatibility, which continues to expose them to cryptographic risks [6].

This section will cover authentication attacks as a cryptographic vulnerability.

B. Authentication Attack

Authentication attacks exploit insecure or missing verification mechanisms during BLE's pairing process. For example, devices that use "Just Works" pairing do not validate the identity of their communication partner. This is a weakness that attackers can exploit by impersonating legitimate devices to initiate unauthorized data exchange or session hijacking. Since BLE devices are typically constrained and lack user interfaces, enforcing more secure pairing is often skipped, making these attacks easier to launch in practice [6].

Additionally, BLE legacy connections exchange important cryptographic material in plaintext, such as temporary keys (TK) and random values used to compute Short Term Keys (STKs). An attacker passively sniffing this

information can easily brute-force the TK and derive the STK, giving them access to the encrypted communication channel. Currently, real-world demonstrations of Authentication Attacks are demonstrated on wearable devices, such as smart bands, to illustrate the risk posed by insecure or lack of authentication methods [14].

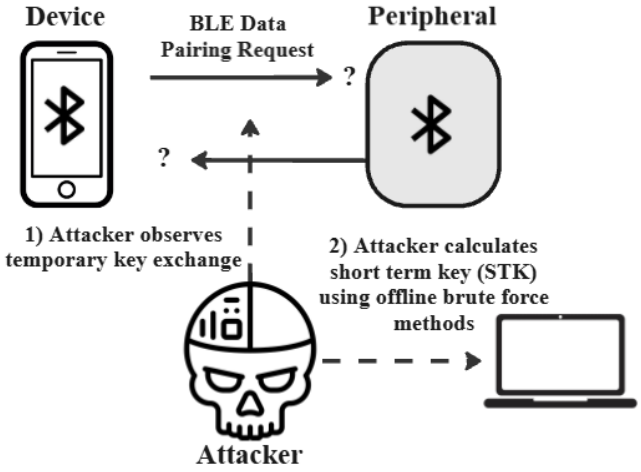

Figure 7: *Authentication Attack. The attacker passively listens to BLE pairing between a central and peripheral device. If weak pairing (like "Just Works") is used, the attacker can derive the shared encryption key offline and later decrypt and/or interfere with the session.*

1) Mitigation

Devices should enforce Secure Connections by default and avoid legacy pairing modes whenever possible. Pairing methods that require user interaction, like Numeric Comparison or Passkey Entry, should be used on devices with suitable interfaces. Even on constrained devices, ephemeral keys and MAC address randomization help protect against long-term tracking and authentication bypass [6][14].

VI. Device Cloning Attacks

A. Definition

Device cloning methods involve the malicious replication of a legitimate BLE device's identity or behavior in order to deceive other devices, users, or systems into accepting a counterfeit device as genuine. These attacks are typically used to bypass authentication, extract private data, or carry out unauthorized actions within trusted BLE ecosystems. BLE is especially vulnerable to cloning attacks because many of its default behaviors—such as broadcasting static MAC addresses, automatically reconnecting to known devices, and limited cryptographic re-verification—do not enforce strong identity validation [8][10].

This section will cover BLESA, blue-chop attacks, brute force attacks, forced re-pairing attacks, and MAC spoofing attacks.

B. BLESA

The BLE Spoofing Attack (BLESA) specifically exploits vulnerabilities in the reconnection procedures of Bluetooth Low Energy devices. Unlike initial pairing, BLE reconnections frequently occur automatically, without rigorous authentication checks, especially in poorly implemented BLE stacks. BLESA targets this implicit trust during reconnection phases, enabling attackers to impersonate previously authenticated devices without providing proper cryptographic credentials. Consequently, attackers exploit devices that inadequately re-authenticate paired devices, causing legitimate devices to reconnect to malicious impersonators unintentionally.

Technically, BLESA attacks capitalize on subtle implementation flaws in the BLE protocol stack. BLE devices typically store cryptographic session keys after initial pairing, implicitly trusting that previously authenticated devices will consistently present these keys during subsequent reconnections. Attackers use packet sniffers like Ubertooth to capture legitimate BLE traffic between paired devices, then use malicious BLE devices configured to broadcast identical identifying information (such as MAC addresses and service UUIDs) during reconnection phases. Vulnerable target devices, failing to validate cryptographic keys adequately during these phases, connect to the attacker's device, mistaking it for a legitimate peer due to reliance on previously stored session information rather than active verification [10].

BLESA is particularly dangerous because it exploits a behavior common in many commercial BLE stacks: automatic reconnection without re-authentication. In mass-market IoT devices and wearables, reconnections are often designed for speed and simplicity, meaning authentication overhead is skipped. An attacker could use BLESA to push malicious firmware updates, read private data, or disrupt services in

scenarios such as remote healthcare or smart surveillance.

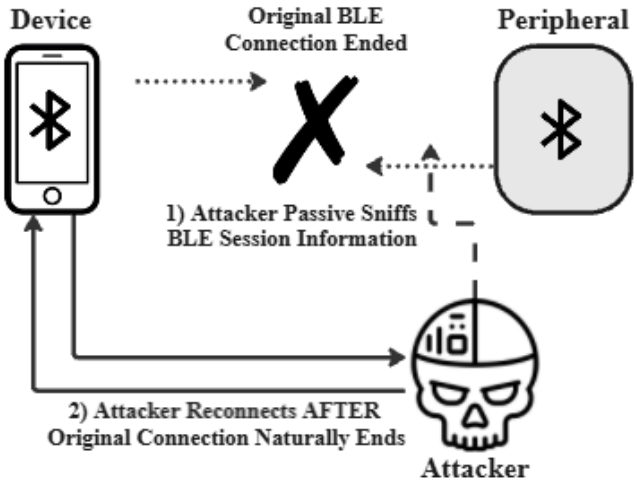

Figure 8: *BLESA Spoofing Attack. After a temporary disconnection, the attacker impersonates a previously paired device using the same MAC address and GATT profile. If the central device does not re-authenticate on reconnect, it resumes communication with the attacker believing it is the original peripheral.*

1) Mitigation

Mitigation of BLESA requires validating keys or session tokens on every reconnection. BLE Secure Connections provides tools to verify peer identity using long-term keys, which should be checked against stored credentials during any connection re-initiation. Devices must avoid accepting reconnections purely based on known MAC addresses or cached session data.

Moreover, firmware developers should ensure that reconnection triggers do not reset security states or default to lower authentication levels. Session keys should be invalidated after disconnection, requiring full re-validation before resuming communication. In high-risk applications, reconnection events should be logged and reviewed, and access to sensitive functions should require re-authentication at the application level, such as user presence or biometric input.

C. Blue-Chop

The Blue-Chop attack targets the Bluetooth piconet—a network topology where one master device connects to multiple slave devices. Rather than attempting to steal data or compromise encryption, the Blue-Chop attack aims to create communication disruption by overloading the

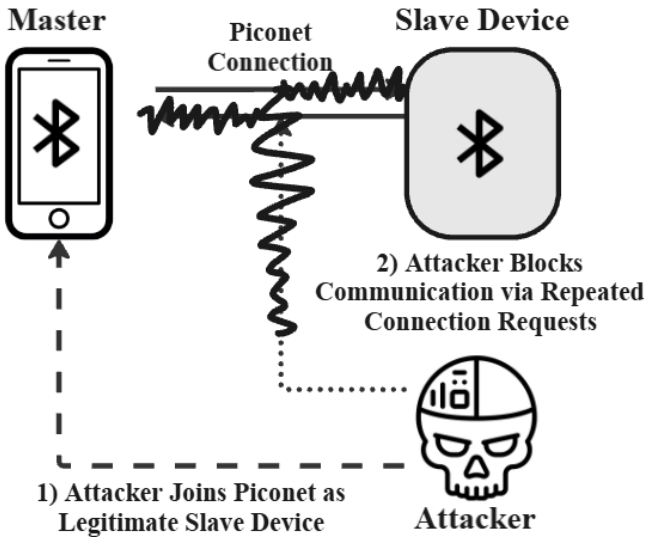

Figure 9: *Blue-Chop Attack. The attacker impersonates slave devices and floods the piconet with spoofed joined requests, overwhelming the master's scheduling system and preventing normal devices from communicating.*

piconet's capacity. It achieves this by injecting a rogue device that impersonates a legitimate participant and sends continuous connection requests, thereby jamming the network and preventing normal devices from communicating effectively. This attack leverages the piconet's resource limitations and the master's responsibility for scheduling communications, effectively flooding its bandwidth allocation mechanisms and paralyzing its ability to service legitimate slaves [2].

Technically, the attacker spoofs the Bluetooth Device Address (BD_ADDR) of an active participant in the piconet and attempts to join the network as a new slave. Once admitted by the master device, the attacker continuously sends repeated connection or service requests to all devices in the piconet. This can be executed using programmable radios or software-defined devices that allow manipulation of Bluetooth protocol-level behaviors. If the master supports multiple simultaneous connections, the attacker's rogue device rapidly rotates through available device slots, initiating and dropping connections, or creating enough noise to interfere with the Time-Division Duplexing (TDD) scheduler responsible for allocating communication slots in the piconet.

The Blue-Chop attack is particularly dangerous in time-sensitive or critical BLE applications, such as medical telemetry, industrial IoT controls, or

automotive systems, where the loss of communication can result in degraded system reliability or even physical harm. Because it exploits standard behavior in Bluetooth network formation, including legitimate connection sequences, it is difficult to detect without protocol-level monitoring.

1) Mitigation

Mitigating Blue-Chop attacks requires limiting the number of simultaneous connections a master can manage and enforcing strict timeout and rejection policies for devices that initiate excessive connection attempts. Firmware on BLE master devices should monitor for connection abuse patterns—such as repeated disconnection/reconnection attempts or behavior from cloned BD_ADDRs—and blacklist suspicious devices automatically. Piconets should be designed to accept only whitelisted or cryptographically verified devices during the connection phase, thereby reducing the likelihood of rogue admission.

Moreover, devices in a piconet should implement behavior-based intrusion detection systems capable of identifying anomalous scheduling requests or bandwidth abuse. Incorporating resilience strategies, such as dynamic reallocation of master-slave roles, packet buffering under attack conditions, and periodic validation of device authenticity, can make BLE networks significantly more robust to resource exhaustion attacks like Blue-Chop.

D. Brute-Force

A brute force attack against BLE targets the pairing process, specifically when devices use short, predictable PINs or keys. This form of attack involves systematically guessing pairing credentials until the correct one is found. BLE devices often rely on numeric PINs or static passkeys—especially in legacy or cost-sensitive devices—and these values can be exhaustively searched in real-time if the keyspace is small enough. Brute force attacks typically aim to gain unauthorized pairing with a target device, after which the attacker may access encrypted communication, manipulate services, or clone device behavior [4].

To implement this attack, the attacker uses a BLE development kit or SDR-based tool to simulate repeated pairing attempts with a target device. Using tools such as crackle or custom scripts with libraries like pygatt or GATTacker, the attacker cycles through all possible PIN values—often only 4 digits—during the key exchange phase. The attacker monitors the device's response to determine when a correct PIN results in successful pairing. In some cases, this process is enhanced by intercepting pairing messages during a legitimate exchange and replaying them with substituted values to isolate unknown components. Because some devices do not throttle or block excessive pairing attempts, the attacker may brute-force all 10,000 possible PINs in a matter of seconds to minutes depending on device behavior and timing constraints.

The primary danger of brute force attacks lies in their low barrier to entry. No advanced cryptographic knowledge is needed, only a script and a compliant target. Once the correct PIN is discovered, the attacker can pair with the device at will, gaining access to all unprotected services and data characteristics. In a health monitoring device, this could include medical records or control commands; in a smart home system, it could grant control of security sensors, lights, or locks. Because brute force attacks do not leave obvious signs of intrusion, compromised devices may continue operating as if they are secure, even while being actively monitored or manipulated.

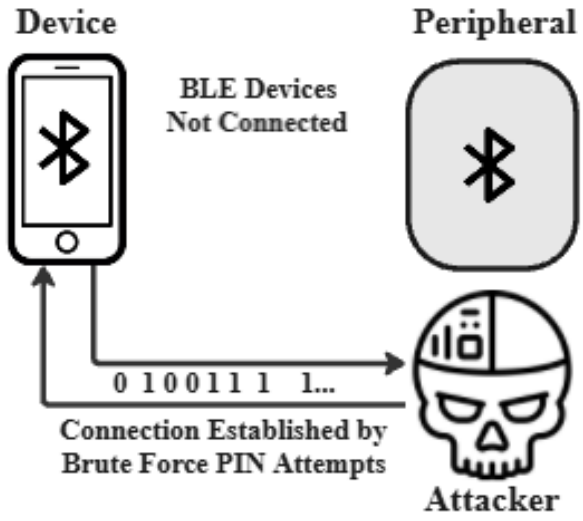

Figure 10: *Brute Force Attack. The attacker repeatedly attempts to pair with the device using all possible PIN combinations. If the device allows unlimited attempts and uses a short static PIN, the attacker can discover it and gain full access.*

1) Mitigation

To defend against brute force attacks, developers should implement pairing rate limits and lockout mechanisms. After a certain number of failed pairing attempts, devices should delay further attempts or require user intervention to reset the pairing process. Devices should also log pairing failures and make this information accessible for administrators or users reviewing security logs.

More fundamentally, devices must move away from PIN-based pairing and instead use BLE Secure Connections with Elliptic Curve Diffie-Hellman key exchange and authenticated pairing methods. These protocols make it computationally infeasible to brute-force a pairing key and remove reliance on low-entropy secrets. Even in constrained environments, default PINs should be randomized per device, and any pairing configuration that uses static keys or hardcoded PINs should be treated as a critical vulnerability.

E. Forced Repairing

A forced re-pairing attack involves deliberately interrupting a previously established BLE connection, compelling the victim devices to initiate a new pairing process. Attackers aim to exploit vulnerabilities exposed during the re-pairing process, capturing cryptographic keys or forcing devices into insecure pairing methods. This attack leverages BLE's standard reconnection mechanisms where paired devices reestablish connections automatically. By triggering intentional disruptions—often through selective packet jamming or signal interference—the attacker induces repeated re-pairings, creating opportunities to intercept and compromise critical pairing exchanges.

The attacker first identifies an active BLE session using passive sniffing, then terminates it using techniques such as L2CAP message flooding, HCI-level disconnection commands, or selective jamming on the BLE frequency. As the devices attempt to re-pair, the attacker positions a rogue device to respond faster than the legitimate device or injects pairing packets that cause fallback to insecure pairing modes like Just Works. The attacker can then complete the pairing process with one or both devices, gaining access to session keys and enabling future communication interception or injection [6].

This attack is dangerous because it bypasses existing trust relationships and security protections. Even if the original pairing used BLE Secure Connections, a forced re-pairing can revert the session to insecure protocols if the devices are not configured to reject downgrade attempts. In environments with unattended or autonomous BLE devices—such as industrial sensors, fitness trackers, or door locks—attackers can persistently exploit this window without alerting users or administrators.

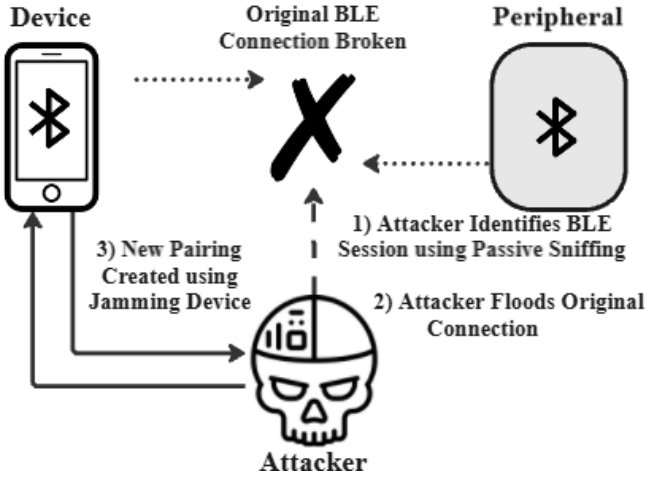

Figure 11: *Forced Re-pairing: The attacker triggers a disconnect between the two devices, then injects themselves into the repairing process. If fallback pairing is allowed (e.g. Just Works), they can complete pairing and impersonate one of the devices.*

1) Mitigation

Mitigating this attack requires strict enforcement of authenticated pairing during every pairing attempt. Devices should be configured to refuse pairing unless authenticated mechanisms like Numeric Comparison, Passkey Entry, or Out-of-Band (OOB) pairing are used. Just Works pairing should be explicitly disabled in production environments where trust and data security are essential.

Additionally, devices should implement rate-limiting for pairing attempts and monitor for repeated disconnections followed by re-pairing. Secure pairing should be remembered and reused unless explicitly invalidated, and any change in key exchange method should trigger a security alert or require user confirmation. These practices

collectively reduce the feasibility of exploiting the pairing process to introduce a rogue device or intercept new session credentials [10].

F. MAC Spoofing

Media Access Control (MAC) spoofing involves an attacker impersonating a legitimate BLE device by replicating its MAC address. In BLE communications, devices broadcast unique identifiers, primarily their MAC addresses, allowing easy recognition and pairing among devices. Attackers exploit this broadcast nature by passively scanning advertising channels, capturing these addresses, and subsequently configuring their malicious devices to advertise the same addresses. This effectively deceives other BLE-enabled devices or applications into establishing connections or communications with the attacker-controlled device instead of the legitimate one [4][14].

Technically, an attacker conducting a MAC spoofing attack first uses a device such as Ubertooth or a similar Bluetooth packet sniffer to passively capture BLE advertising packets transmitted openly on advertising channels - recall that Bluetooth communicates with 40 available public channels, three of which are advertising (channels 37, 38, and 39). After collecting legitimate MAC addresses, attackers program their BLE hardware (often easily accessible hardware platforms or software-defined radios) to broadcast these addresses. This spoofing operation does not require breaking encryption or sophisticated cryptographic skills; the complexity resides primarily in passive interception and simple hardware reconfiguration. Hence, MAC spoofing remains accessible even to moderately equipped attackers, making it a prevalent attack vector.

The danger of MAC spoofing stems from the ease with which trust can be subverted. Once accepted, the spoofed device can receive user commands, send falsified data, or intercept sensitive exchanges. In BLE-enabled home automation systems, this could mean unlocking smart doors; in healthcare, it could involve spoofing readings from a medical sensor. Since spoofing does not require cracking encryption or modifying firmware, even relatively unskilled attackers with basic hardware can execute it effectively, and detection is difficult without additional monitoring layers.

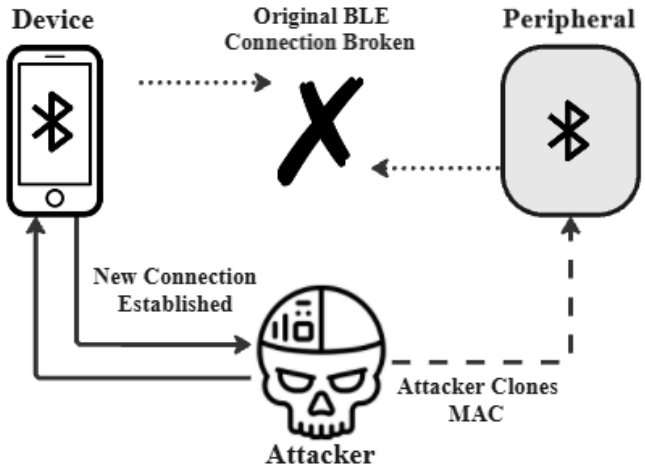

Figure 12: *MAC Spoofing. The attacker clones the MAC address of a previously paired device. When the original device is out of range, the central device connects to the spoofed one, mistaking it for the legitimate peripheral.*

1) Mitigation

To mitigate MAC spoofing, devices should use Resolvable Private Addresses (RPAs), which are dynamically generated and change periodically. These addresses can only be resolved by trusted devices with the correct Identity Resolving Key (IRK), ensuring that only legitimate peers can interpret a changing address as the same entity. BLE 4.2 and newer support this feature natively, and developers should ensure that their devices rotate addresses consistently, especially in public or sensitive environments.

Further protection requires shifting away from MAC-based trust entirely. Devices should require cryptographic validation upon connection, ideally using BLE Secure Connections with mutual authentication. Application-layer logic can further enhance this by implementing session-specific credentials, digital signatures, or encryption tokens to validate identity. Whitelisting should be based on cryptographically verified identities rather than static identifiers, ensuring that even a perfectly cloned MAC address is not sufficient to impersonate a trusted peer [14].

VII. Distortion Methods

A. Definition

Distortion methods include attacks that exploit BLE communication for disruption, misinformation, or denial-of-service (DoS). These attacks, often rooted in the protocol's openness and weak protection of public channels, include variants like blue-jacking, blue-snarfing, blue-smaking, and protocol fuzzing. Many BLE devices still fail to validate incoming messages or enforce strict state transitions, leaving them vulnerable to such manipulations [2]. According to Patel et al., these attacks are particularly concerning as they can lead to significant loss of data through denial of service attacks, eavesdropping, resource misappropriation, and message modification [1].

This section will cover blue-jacking, blue-snarfing, blue-smacking, and fuzzing from the perspective of distortion method attacks.

B. Blue-Jacking

Blue-Jacking is a benign-looking but potentially disruptive attack where unsolicited messages are sent to BLE-enabled devices via advertising or pairing requests. Although originally more common in Classic Bluetooth, BLE implementations that allow automatic pairing or process received messages without user verification are still susceptible. It can be a steppingstone to more serious social engineering or phishing attacks [2].

Attackers may use tools such as modified BLE sniffers to flood nearby devices with messages mimicking legitimate advertisement payloads. In public settings or environments with lots of smart devices, such as an airport or hospital, Blue-Jacking can be used to trick users into installing malicious applications or accepting fake connections [14]. Patel et al. demonstrated how this attack can be executed on Android devices (Nexus 7) running older Bluetooth systems using the Bluetooth send command to transmit unwanted files. Their research showed that devices running Bluetooth version 3 were particularly vulnerable to this attack [1].

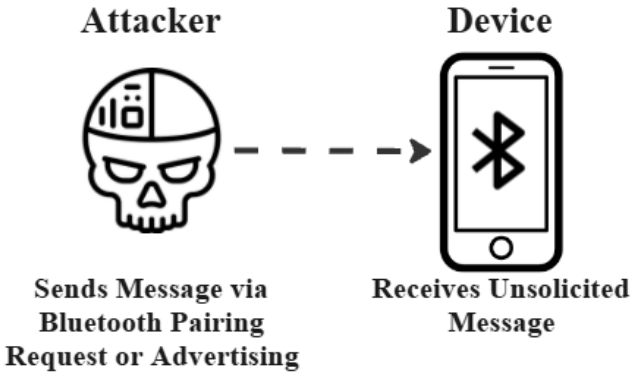

Figure 13: *Blue-Jacking. The attacker sends unsolicited messages (e.g. notes, media) using the OBEX protocol to nearby Bluetooth devices. Though it doesn't grant control, it can confuse or socially engineer users.*

1) Mitigation

User interaction should be mandatory for all pairing attempts, and devices should default to non-discoverable modes. Unused BLE services should be disabled, and access control policies should be enforced for GATT services. BLE firmware should sanitize and validate all received messages to prevent spoofed advertisements or unsolicited requests [2][14]. Additionally, users should ensure they're using newer Bluetooth versions (beyond version 3.0), which have improved security against Blue-Jacking attacks [1].

C. Blue-Smack

Blue-Smack is a DoS attack based on L2CAP echo request flooding. Due to its primary function of flooding BLE stacks to disrupt communication, we classify it under distortion methods. BLE stacks with poor buffer management or no flood protection may crash or become unresponsive when repeatedly hit with malformed or oversized packets. This form of attack, although originally targeting Classic Bluetooth, remains feasible against vulnerable BLE stacks [2].

BLE wearables and IoT hubs are particularly vulnerable due to their limited processing resources. Repeated packet injections may deplete memory or battery life, forcing system reboots or degrading performance over time. In some cases, persistent DoS conditions can be created remotely using low-cost hardware like Ubertooth or Btlejack [14]. Patel et al. demonstrated this vulnerability by using the l2ping command to send

oversized packets to Bluetooth devices. Their experiment successfully executed a denial-of-service attack on an HMDX JAM speaker by setting the packet size to 600 and thread count to 512, rendering the speaker inaccessible to legitimate connection attempts [1].

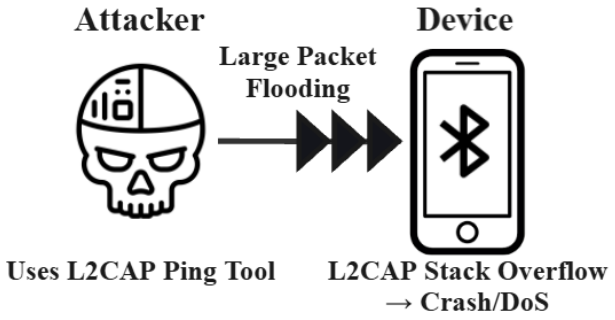

Figure 14: *Blue-Smack. Oversized L2CAP echo requests are sent to the target device to overwhelm the input buffer, leading to denial of service (DoS) or system crash.*

1) Mitigation

BLE stacks must enforce packet rate-limiting, buffer size checks, and malformed packet detection. Operating systems or firmware should implement timeout and watchdog routines to recover from flooding conditions automatically [2][14]. Users should ensure their devices remain within secure ranges and turn off Bluetooth functionality when not in use, especially in public spaces, to minimize exposure to such attacks [1].

D. Blue-Snarfing

Despite being more prevalent in Classic Bluetooth, BLE versions that maintain backward compatibility or use insecure profiles are vulnerable. For example, certain fitness trackers/smart bands expose heart rate data even before secure pairing is completed, allowing attackers to scrape personal health data silently [6]. Patel et al. used the "bluesnarfer" tool built into Kali Linux to attempt accessing phonebook entries on both Android and iOS devices. Their tests revealed that modern devices running Bluetooth version 3 and above were largely protected against this type of attack [1].

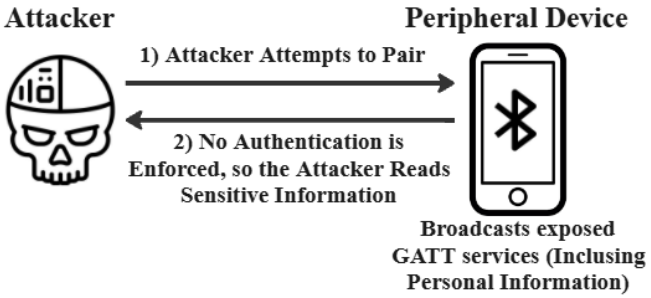

Figure 15: *Blue-Snarfing. Although originally a classic Bluetooth attack, Blue-Snarfing can affect BLE devices that expose sensitive GATT services without requiring pairing. Personal data may be silently scraped using a blue-snarfer tool, especially in devices with insecure default profiles.*

1) Mitigation

BLE devices should enforce attribute-level access control using authenticated reads and writes. Sensitive services should not be available until after secure pairing. Developers must ensure devices do not default to permissive GATT permissions in production firmware [6]. Users should regularly update their device firmware and use devices with newer Bluetooth versions, which implement improved security protocols against Blue-Snarfing attacks [1].

E. Fuzzing

Fuzzing involves sending malformed, semi-random packets to test the resilience of the BLE stack. Unlike blue-smacking which aims to overwhelm the system with volume, fuzzing specifically targets protocol implementation flaws by introducing unexpected variations in packet structure and content. While often used for security testing, malicious actors can exploit this technique to discover unknown vulnerabilities and trigger faults or bypass authentication [6].

BLE devices lacking strict protocol conformance may crash, leak memory, or enter inconsistent states under fuzzing. Several smart bands and wearable devices have been shown to exhibit unpredictable behavior when exposed to fuzzing tools like BtleJuice or Adafruit BLE sniffers [14].

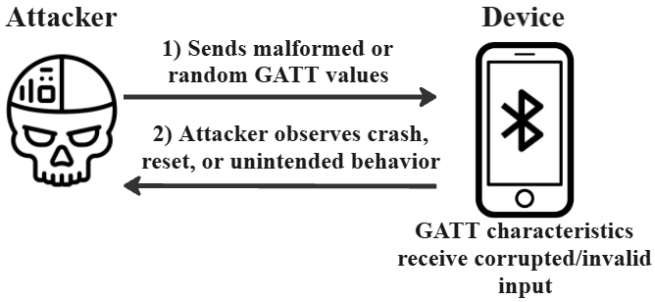

Figure 16: *Fuzzing. The attacker sends invalid or randomized data to GATT characteristics. If input validation is weak, the device may crash, become unresponsive, or misbehave.*

1) Mitigation

Device manufacturers should apply fuzz testing during development and conduct conformance testing against Bluetooth SIG standards. Runtime checks, protocol sanitization, and input validation are critical to defend against runtime exploitation [6][14].

VIII. DoS Attacks

A. Definition

Denial of Service (DoS) attacks in BLE aim to render a device or network unavailable by exhausting resources, overwhelming bandwidth, or disrupting normal operation. While BLE is designed for power efficiency and intermittent transmission, it also sacrifices persistent defensive mechanisms such as strict channel isolation or bandwidth policing. As a result, BLE devices are often vulnerable to service disruption even by attackers with minimal resources or technical sophistication.

Unlike data theft or spoofing attacks, DoS attacks target system reliability and availability—often a critical security goal in safety-sensitive applications like health monitoring, building automation, or industrial control. This section will cover Battery Exhaustion Attack, DoSL Attack, and Jamming Attack.

B. Battery Exhaustion

Battery exhaustion attacks target BLE's vulnerability to energy depletion by maintaining prolonged communication sessions or creating high-frequency interactions that force the target device to transmit or process more than its intended capacity. Unlike DOSL, which keeps a device awake, battery exhaustion maximizes energy draw by exploiting high-power BLE features—such as data transfer, characteristic writes, or continuous advertisement response handling. These attacks are often effective on wearable devices, IoT modules, or any platform where power conservation is critical [11].

The technical procedure for a battery exhaustion attack typically begins with scanning for BLE-enabled devices and identifying their available services using the Generic Attribute Profile (GATT). Once a device is found, the attacker establishes a connection and initiates repeated service discovery, read, or write operations to any publicly available characteristics. If the attacker controls a central device (e.g., a smartphone or Raspberry Pi), it can write to writable characteristics as rapidly as the BLE connection allows. If the device allows read operations without authentication, the attacker can continuously poll sensor values, forcing the peripheral to process and transmit data repeatedly. These actions may not violate BLE protocol rules, but they drastically increase CPU and radio usage, depleting battery within hours or days depending on the attack intensity.

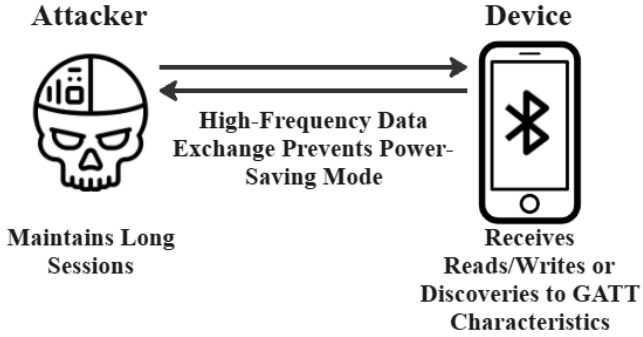

Figure 17: *Battery Exhaustion Attack. The attacker maintains a connection and continuously reads/writes to GATT characteristics, forcing the device to remain active and consume power rapidly. Over time, this significantly reduces battery life.*

1) Mitigation

To mitigate battery exhaustion, devices must enforce authentication for all nontrivial operations. Read, write, or notify permissions should require an encrypted connection or verified pairing, and unauthenticated connections should be restricted to minimal functionality (e.g., limited advertisements or generic information only). BLE's attribute-level

access controls can be used to prevent repeated access to high-cost services unless the connection is authenticated.

In addition, BLE firmware should monitor connection behavior and detect anomalies such as excessive reads, writes, or session durations from untrusted peers. If abnormal traffic is observed, the device should throttle responses, issue disconnection commands, or escalate to radio suspension. Combining access control with behavior-based thresholds offers a balance between usability and security without requiring continuous human oversight.

C. DoSL

A Denial of Sleep (DOSL) attack exploits the energy-conservation mechanisms of BLE-enabled wireless sensor networks by repeatedly triggering events or messages that prevent a device from entering low-power sleep mode. BLE devices are specifically optimized for minimal energy usage and rely heavily on aggressive sleep scheduling to conserve battery. In a DOSL attack, the adversary transmits malicious requests, advertisement probes, or spurious connection events to force the target to remain awake, drastically increasing energy consumption and shortening device lifespan [12].

To implement this attack, the adversary uses a BLE sniffer or custom device to scan for nearby advertising peripherals. Once discovered, the attacker initiates repeated connection attempts or sends LL_CONTROL messages that elicit a response or state transition. The attacker may not complete the connection handshake; instead, the goal is to wake the BLE chip from sleep, force it into an active state, and keep it transmitting or listening. In many BLE stacks, this prevents the device from returning to deep sleep until timeout intervals expire. More aggressive variants may maintain short-lived GATT sessions and then abruptly disconnect, repeating the pattern indefinitely. Because the BLE protocol allows devices to respond automatically to connection requests without user interaction, the victim continues to consume energy in responding—even if authentication fails.

The danger of DOSL attacks lies in their impact on device availability, battery life, and operational integrity. For resource-constrained sensors, such attacks can significantly degrade battery performance, requiring premature device replacement in remote deployments. In medical or industrial contexts, device failure from energy exhaustion can result in delayed alerts, data gaps, or total service loss. Furthermore, these attacks are silent; the device appears to be functioning normally until energy reserves are depleted, making them particularly insidious in long-term use environments.

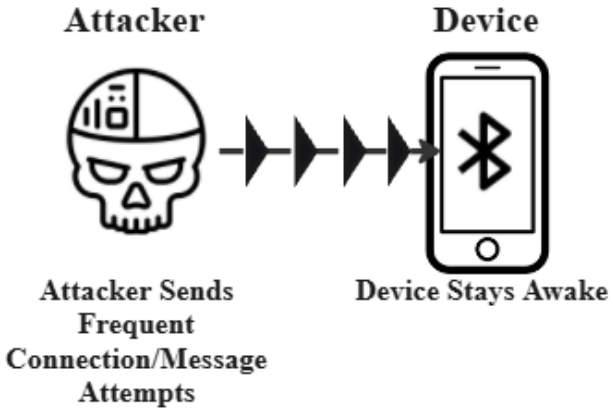

Figure 18: *Denial of Sleep Attack. The attacker repeatedly triggers BLE connection attempts or transmits packets to keep the device in an active state, preventing it from entering low-power sleep mode and draining energy over time.*

1) Mitigation

Mitigation of DOSL attacks begins with connection and advertising control policies. Devices should limit the frequency at which they accept new connection attempts, particularly if repeated from the same source address. Firmware should also enforce backoff timers and reject repeated connection attempts that occur within abnormal timing windows. BLE 4.2 and later allow configurable connection intervals and filtering policies that can be tuned to reject devices not in a whitelist or without a stored pairing key.

At the application level, developers should configure sensors and BLE peripherals to enter sleep modes aggressively and remain there unless authenticated activity is detected. Advertising intervals can be randomized or lengthened, and devices should automatically shift to private resolvable addresses to prevent persistent tracking and targeted wakeups. Logging mechanisms may also be used to detect repeated wake events and raise alerts or trigger defensive behaviors, such as

temporary radio shutdown or adaptive throttling of responses.

D. Jamming

Jamming attacks represent a physical-layer denial-of-service tactic in which the attacker emits continuous or selectively timed radio noise to prevent BLE devices from communicating. Because BLE operates in the 2.4 GHz ISM band and uses adaptive frequency hopping (AFH) across 40 channels, jamming must be either broad-spectrum (affecting the entire band) or synchronized with BLE's channel usage patterns. The attacker's goal is to prevent devices from exchanging pairing requests, advertisement packets, or data transmissions, thereby making communication unreliable or impossible [7].

The implementation of a jamming attack involves the use of a software-defined radio (SDR), such as HackRF One or BladeRF, which is programmed to emit a continuous wave signal or pulse-based interference across selected BLE frequencies. A more sophisticated version of this attack—known as reactive jamming—relies on detecting BLE packet headers in real time and injecting interference precisely when those packets are transmitted, which disrupts transmission without requiring full-band noise. By focusing on advertisement channels (37, 38, and 39), the attacker can prevent devices from discovering each other, while targeting data channels can interfere with active sessions. Because BLE's AFH relies on detecting and avoiding noisy channels, a smart jammer can manipulate which frequencies are avoided, forcing the BLE system into suboptimal or degraded performance modes.

Jamming is dangerous because it breaks the fundamental assumptions of BLE availability and reliability. In high-density environments like hospitals, smart factories, or transportation hubs, even temporary jamming can cause widespread service degradation, device desynchronization, and data loss. Since BLE devices lack built-in jamming detection and do not typically operate on alternative frequencies, users have little visibility into whether a device is failing due to signal collision, environmental interference, or malicious jamming.

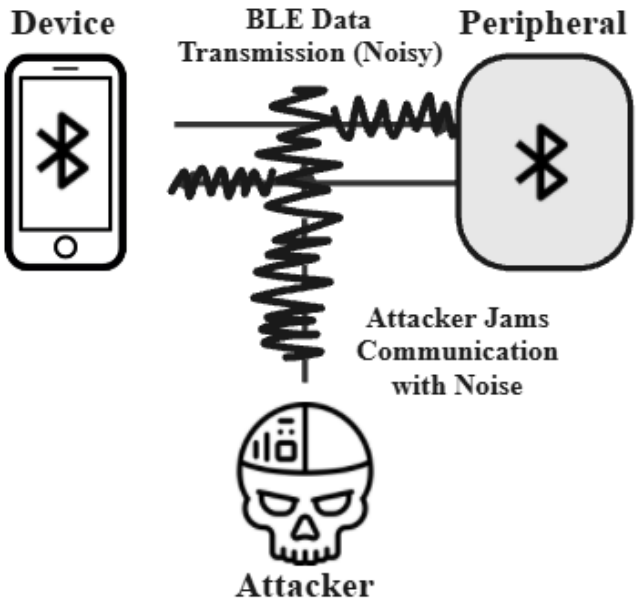

Figure 19: *Jamming Attack. An attacker sends radio noise or timed interference, disrupting BLE communication between central and peripheral devices and causing packet loss, failure to pair, or denial of service.*

1) Mitigation

Mitigation of jamming attacks begins with leveraging BLE's AFH capabilities, which allow devices to detect jammed channels and adapt their frequency hopping pattern accordingly. However, AFH is not always aggressive enough, especially in devices with constrained BLE stacks. Developers should ensure that BLE implementations support full AFH compliance and avoid pinning communication to specific frequency subsets.

More advanced mitigation strategies include spread spectrum techniques or the use of signal fingerprinting to detect anomalous interference patterns. Application-layer systems can log transmission failures and alert administrators when devices fail repeatedly in an otherwise healthy environment. In highly critical deployments, BLE can be paired with secondary wireless channels—such as Wi-Fi or sub-GHz radios—to maintain redundancy in the presence of interference. While BLE jamming cannot always be prevented, its impact can be reduced through resilience engineering and early detection mechanisms.

IX. Surveillance Attacks

A. Definition

Surveillance attacks in BLE exploit the broadcast nature of advertising packets and weaknesses in device anonymity to monitor, track, or profile users. Static MAC addresses, unprotected UUIDs, and predictable data patterns allow adversaries to map user activity, location, and device types without needing to pair [6].

This section will cover activity detection, blue-printing, blue-stumbling, blue-tracking, and device fingerprinting attacks.

B. Activity Detection

This attack relies on inferring user behavior based on BLE transmission intervals and payload sizes. For instance, wearable devices transmitting more frequently during physical activity expose patterns that correlate to user movement or health status [6].

Attackers use passive sniffing to collect this data, then run pattern analysis to derive states like sleeping, walking, or exercising. BLE's lack of traffic padding or randomized scheduling makes this attack surprisingly effective [14].

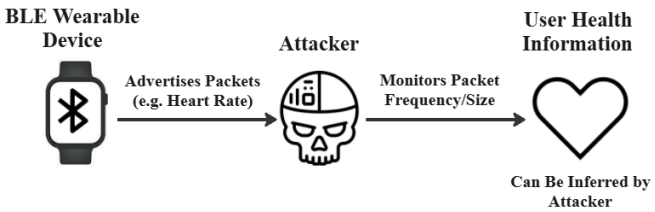

Figure 20: *Activity Detection Attack. By analyzing the frequency and size of BLE advertisements from BLE peripherals such as fitness bands, the attacker can infer physical activity or habits of the user without needing access to data payloads.*

1) Mitigation

To counter this, devices can transmit on randomized schedules or insert obfuscating traffic. Compression and encryption of application-layer payloads help further mask user activity patterns [14].

C. Blue-Printing

Blue-Printing identifies a device's make, model, and supported services based on its advertisements and GATT characteristics. Even without connecting, attackers can fingerprint devices like smartwatches or health monitors and target them in future attacks [2].

This kind of profiling is often the first step in a more complex attack. Once a target is identified, the adversary can develop specific exploits tailored to firmware or known bugs in that device family, enhancing their success rate in penetrating BLE-based systems [6].

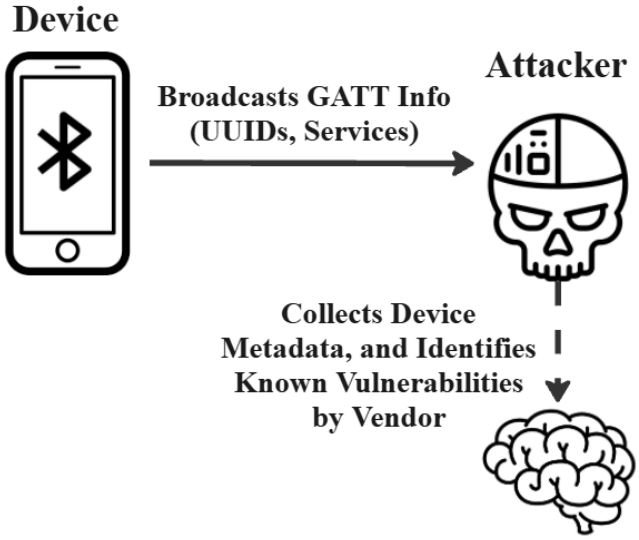

Figure 21: *Blue-Printing. The attacker uses tools like nRF Connect to collect GATT metadata publicly broadcasted by BLE peripherals to identity make, model, and software version. This data can be used to plan targeted future exploits.*

1) Mitigation

Use of generic advertisement payloads, randomized MAC addresses, and concealment of UUIDs until after secure pairing significantly limits blue-printing. Manufacturers should also avoid hardcoded device names in advertisement fields [2][6].

D. Blue-Stumbling

This is a passive scanning attack that involves collecting BLE advertisement data (e.g., MACs, RSSI) over time to map device locations and behaviors. Tools like Wireshark and Btlejack allow attackers to log these broadcasts and associate them with individuals in specific environments like hospitals or workplaces [6].

Long-term logging may reveal user movement patterns, routines, or asset inventory, especially when BLE beacons are deployed for tracking. When combined with fingerprinting or ID

correlation, Blue-Stumbling becomes a severe privacy threat [14].

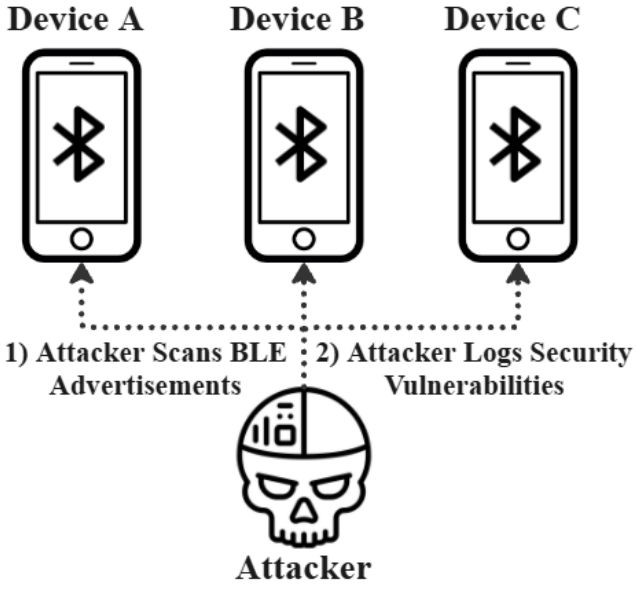

Figure 22: *Blue-Stumbling. This attack involves passive scanning of BLE advertisements in public spaces to log devices with outdated firmware or insecure services. It is typically a precursor to larger-scale surveillance or exploit campaigns.*

1) Mitigation

Devices should implement privacy-preserving features such as Resolvable Private Addresses (RPAs) and periodically rotate advertising identifiers. BLE 4.2 and newer support these capabilities, which must be enabled and configured correctly [6][14].

E. Blue-Tracking

Blue-Tracking targets devices that fail to rotate identifiers, allowing attackers to monitor user movement across locations. Many wearables and medical devices maintain static IDs for extended periods, compromising user privacy in public areas [6].

Even if encryption is used after pairing, the initial advertisement phase leaks metadata. For example, during a fitness session, the transmission frequency of some devices can reveal if a user is active or stationary, facilitating behavior profiling [14].

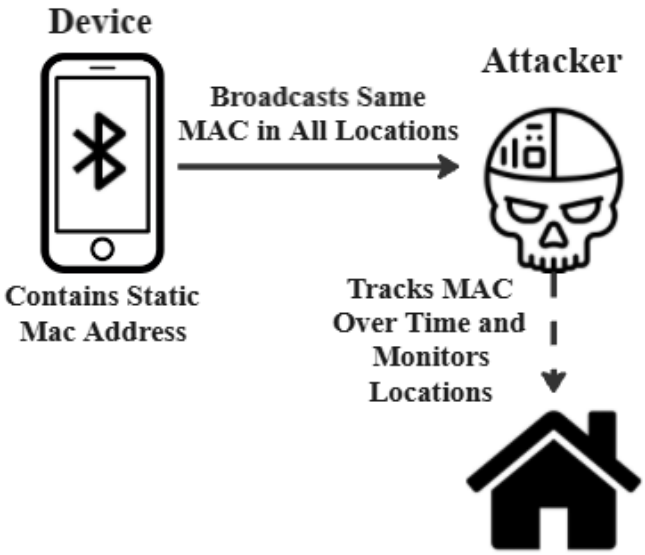

Figure 23: *Blue-Tracking. When a BLE device uses a fixed MAC address or UUID, an attacker can track its movement across multiple environments. This allows for location profiling and stalking.*

1) Mitigation

BLE devices must rotate MAC addresses frequently using IRKs. Additionally, traffic randomization—through dummy payloads or obfuscation—can confuse simple behavioral inference models [6][14].

F. Device Fingerprinting

BLE devices can be uniquely identified based on unique patterns in timing, advertisement structure, supported features, or connection behavior. These fingerprints remain stable even when MAC addresses are randomized, enabling persistent tracking [6].

Such fingerprinting can be used to track individuals across sessions or environments. It also enables attackers to bypass privacy controls by focusing on non-changing protocol features [2].

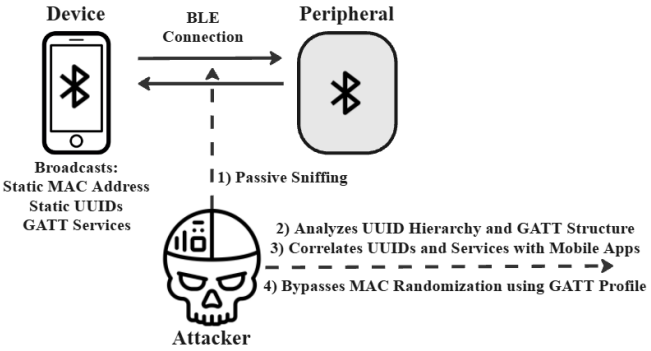

Figure 24: *Device Fingerprinting Attack. The attacker passively sniffs BLE advertisement packets and GATT service broadcasts to extract static MAC addresses, UUIDs, and profile structures. By correlating these with IoT mobile applications, they uniquely identify the device even if MAC randomization is enabled.*

1) Mitigation

Developers should ensure their BLE stacks conform strictly to Bluetooth specifications and minimize deviations that enable fingerprinting. Uniform timing behavior and randomized advertisement structures reduce fingerprint stability [6].

X. Miscellaneous Attacks

A. Defintion

Miscellaneous attacks exploit implementation-specific bugs, hidden developer interfaces, or improper use of BLE protocol features. These often don't fit into traditional categories but are equally dangerous, especially in consumer devices where security may be sacrificed for usability [2].

This section will cover backdoor, blue-bugging, and finally co-located attacks.

B. Backdoor

Some BLE devices include undocumented GATT services or debug commands for manufacturing and testing. If not removed, these can be activated remotely by attackers who know the access UUIDs or formats [14].

In real-world analyses, wearable devices like smart bands have been found to respond to undocumented write commands. These commands can trigger firmware changes, sensor reads, or even factory resets [14].

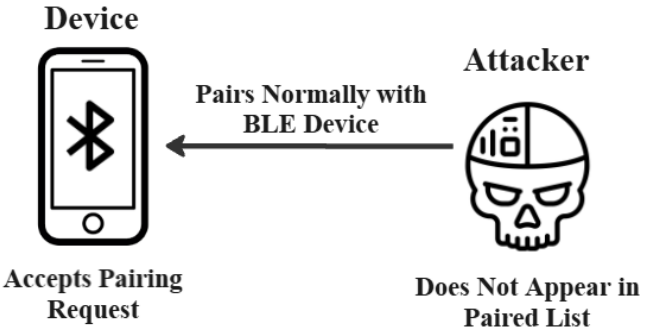

Figure 25: *Backdoor Attack. The attacker establishes a stealth pairing session that is not visible in the device's trusted device list, allowing persistent unauthorized access even after reboot or device inspection.*

1) Mitigation

Manufacturers must audit firmware for all undocumented features and lock down all BLE service access behind authenticated layers. Firmware signing and validation at runtime help ensure integrity [14].

C. Blue-Bugging

Originally found in Classic Bluetooth, Blue-Bugging enables attackers to gain unauthorized control over a device's messaging and calling features. BLE implementations that include legacy support stacks or improperly isolated system services may still be exposed [2].

For example, debug commands or exposed UART interfaces accessible over BLE may allow attackers to issue control commands. This is particularly dangerous in smart locks or medical devices, where system functions are mapped to BLE commands [14].

1) Mitigation

All developer and debug interfaces should be removed or disabled before production deployment. BLE services should require strict authentication and privilege separation [14].

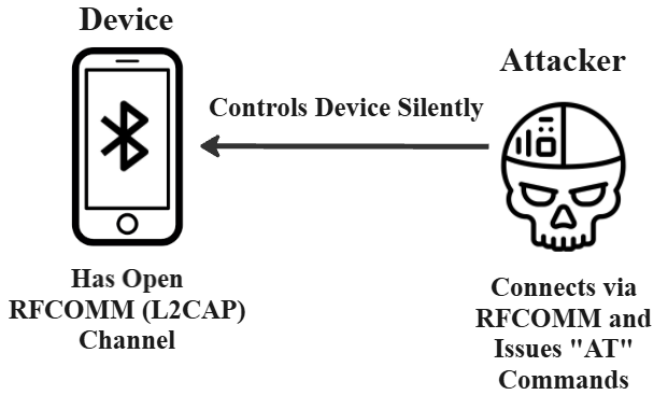

Figure 26: *Blue-Bugging. The attacker establishes an unauthorized RFCOMM session with the target, issues attention (AT) commands, and takes control of the device. This allows the attacker to initiate calls, steal messages, or alter settings.*

D. Co-located

These attacks exploit physical proximity, allowing adversaries to intercept pairing attempts or relay signals to impersonate nearby devices. BLE's relatively short range (~10m) makes it seem secure, but co-located relay attacks have been demonstrated in labs and public environments [14].

For example, a smart lock may pair with a spoofed phone that is actually relaying data from an attacker device several meters away. Without user authentication, such attacks go unnoticed [6].

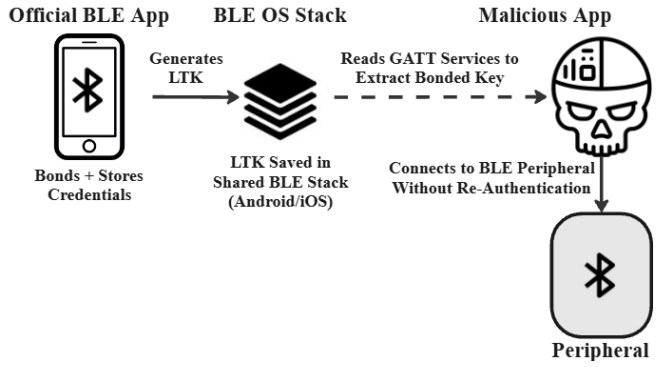

Figure 27: *Co-located Attacks. A malicious app installed on the same mobile device can exploit shared BLE pairing credentials to access a bonded BLE peripheral without user consent. This occurs due to OS-level key storage shared between all apps.*

1) Mitigation

BLE devices should implement user presence confirmation, secure pairing methods, and proximity checks [14]. This could include on-screen confirmation, RSSI threshold and other techniques to defend against co-located spoofing [14].

XI. CONCLUSION

Our survey reveals how Bluetooth Low Energy faces numerous security challenges that threaten its expanding ecosystem. These vulnerabilities—from eavesdropping to device cloning—create particularly serious risks in healthcare and home automation, where sensitive data flows through BLE channels daily. While BLE's designers prioritized energy efficiency and user convenience, these choices have unintentionally created security gaps that attackers continue to exploit, much like leaving windows unlocked in favor of easy access.

Addressing these vulnerabilities demands a multi-layered approach. Hardware manufacturers need to integrate tamper-resistant components, while software developers must implement stronger encryption and authentication protocols. This balancing act resembles retrofitting an older home with modern security—maintaining compatibility while significantly improving protection where it matters most.

Looking ahead, researchers should develop security protocols that work within BLE's power constraints without compromising protection. We need standardized testing frameworks that can identify vulnerabilities before devices reach consumers' hands. As BLE continues powering everything from medical monitors to smart locks, these security improvements won't just be technical necessities—they'll be essential safeguards for the privacy and safety of millions of users worldwide.

REFERENCES

[1] N. Patel, H. Wimmer, and C. M. Rebman, "Investigating Bluetooth Vulnerabilities to Defend from Attacks," IEEE Xplore, Oct. 01, 2021. https://ieeexplore.ieee.org/document/9604655 (accessed Jan. 20, 2022).

[2] S. S. Hassan, S. D. Bibon, M. S. Hossain, and M. Atiquzzaman, "Security threats in Bluetooth technology," Computers & Security, vol. 74, no. 0167-4048, pp. 308–322, May 2018, doi: https://doi.org/10.1016/j.cose.2017.03.008.

[3] P. Cope, J. Campbell, and T. Hayajneh, "An investigation of Bluetooth security vulnerabilities," 2017 IEEE 7th Annual Computing and Communication

Workshop and Conference (CCWC), Jan. 2017, doi: https://doi.org/10.1109/ccwc.2017.7868416.

[4] A. Lonzetta, P. Cope, J. Campbell, B. Mohd, and T. Hayajneh, "Security Vulnerabilities in Bluetooth Technology as Used in IoT," Journal of Sensor and Actuator Networks, vol. 7, no. 3, p. 28, Jul. 2018, doi: https://doi.org/10.3390/jsan7030028.

[5] E. Blancaflor, P. M. G. Purificacion, R. B. Atienza, J. J. M. Yao, and D. A. C. Alvarez, "Exploring the Depths of Bluetooth Attacks: A Critical Analysis of Bluetooth Exploitation and Awareness of Users," 2023 6th International Conference on Computing and Big Data (ICCBD), pp. 52–59, Oct. 2023, doi: https://doi.org/10.1109/iccbd59843.2023.10607255.

[6] A. Barua, M. A. Al Alamin, Md. S. Hossain, and E. Hossain, "Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey," IEEE Open Journal of the Communications Society, vol. 3, pp. 251–281, 2022, doi: https://doi.org/10.1109/ojcoms.2022.3149732.

[7] H. Pirayesh and H. Zeng, "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey," IEEE Communications Surveys & Tutorials, pp. 1–1, 2022, doi: https://doi.org/10.1109/comst.2022.3159185.

[8] H. Alamleh, M. Gogarty, D. Ruddell, and A. A. S. AlQahtani, "Securing the Invisible Thread: A Comprehensive Analysis of BLE Tracker Security in Apple AirTags and Samsung SmartTags," arXiv.org, Jan. 24, 2024. https://arxiv.org/abs/2401.13584

[9] M. Farhan and Nasr Abosata, "Eavesdropping in Bluetooth networks," International Journal of Current Engineering and Technology, vol. 13, no. 2, pp. 579–585, Mar. 2023, Available: https://www.researchgate.net/publication/369663505_Eavesdropping_in_Bluetooth_networks

[10] M. Cäsar, T. Pawelke, J. Steffan, and G. Terhorst, "A survey on Bluetooth Low Energy security and privacy," Computer Networks, vol. 205, p. 108712, Mar. 2022, doi: https://doi.org/10.1016/j.comnet.2021.108712.

[11] B. R. Moyers, J. H. Dunning, R. C. Marchany, and J. G. Tront, "Effects of Wi-Fi and Bluetooth Battery Exhaustion Attacks on Mobile Devices," Hawaii International Conference on System Sciences, Jan. 2010, doi: https://doi.org/10.1109/hicss.2010.170.

[12] J. Uher, R. G. Mennecke, and B. S. Farroha, "Denial of Sleep attacks in Bluetooth Low Energy wireless sensor networks," Zenodo (CERN European Organization for Nuclear Research), Nov. 2016, doi: https://doi.org/10.1109/milcom.2016.7795499.

[13] Y. Shaked and A. Wool, "Cracking the Bluetooth PIN," Proceedings of the 3rd international conference on Mobile systems, applications, and services - MobiSys '05, 2005, doi: https://doi.org/10.1145/1067170.1067176.

[14] T. Nagrare, P. Sindhwad, and F. Kazi, "BLE Protocol in IoT Devices and Smart Wearable Devices: Security and Privacy Threats," arXiv.org, May 05, 2023. https://arxiv.org/abs/2301.03852