A Review of Bluetooth: A platform for Intelligence Collection

OLE ANDRÉ HAUGE, Department of Information Security and Communication Technology (IIK), Norwegian University of Science and Technology (NTNU), Norway

A new intelligence surface is emerging with the increasing growth of IoT devices and the use of Bluetooth technology. We look at how Bluetooth can be used to collect data about the population or single persons and present scientific research discussing vulnerabilities that have been shown to allow remote code execution, man-in-the-middle attacks, information theft, and inherently leak behavioral data that can be used to profile phones and users. The research was found using literature reviews. We discovered that although many of the vulnerabilities demand advanced knowledge in reverse engineering, networking, and software development, a lot of data is still available by spoofing network traffic. Thus, making Bluetooth an accessible intelligence collection platform for anyone within a proximity of 100-360m depending on the Bluetooth technology.

CCS Concepts: • **Security and privacy → Mobile and wireless security**.

Additional Key Words and Phrases: Bluetooth, intelligence, security, vulnerabilities, exploitation

1 INTRODUCTION

"If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle." — Sun Tzu, Art of War [27]

Intelligence is important as it can provide advanced warnings of incidents and enables informed decision-making based on analyzed and well-presented data. The expression, "Intelligence", used in this paper will be based on the definition provided by the Norwegian Intelligence Service (NIS): "*intelligence* is defined as the result of the state-sanctioned collection, analysis, and assessment of data and information obtained overtly or covertly and compiled to provide an advantage in decision-making processes [24, p. 17]".

In this context, the Bluetooth platform can be advantageous in providing access to metrics surrounding the population's movements, emotional states, habits, and other private data to determine stronger intelligence decisions on how to engage or protect against terrorism, espionage, and other attacks. The same intelligence could also be used by smaller groups or businesses to improve their earnings and engagement with their client base.

We are accustomed to how big data analysis is used to tailor our online experience through helpful search suggestions and targeted advertising. On the other hand, we are also aware of how it can be used as a tool to manipulate the public. Several manipulation campaigns against elections have been discovered and attributed to both Russia and China [24, p. 6]. Coordinated inauthentic behavior is acknowledged by social media giants like Facebook [8], and is viewed as one of the largest problems on social media. The Norwegian Intelligence Service declared in their yearly threat and risk assessments report (2021) that one of the most pressing dangers in the next years is the manipulation of elections through social media [24, p. 6].

With the exponential development of IoT devices, a growing and less explored platform for data collection emerges, namely Bluetooth. The rapid development and deployment of billions of IoT devices, phones, computers, cars, and industrial devices, creates a need for unprecedented wireless communication. Bluetooth is one of if not the most used solutions for short-range wireless

Author's address: Ole André Hauge, oleahau@stud.ntnu.no, Department of Information Security and Communication Technology (IIK), Norwegian University of Science and Technology (NTNU), Teknologivegen 22, 2815, Gjøvik, Norway.

communication. Due to the personalized nature and use of IoT devices, they are inherently going to leak data about users, even without vulnerabilities in the design architecture. As is evident from the Apple continuity solution [17]. This makes Bluetooth a highly valuable platform to investigate for anyone who aims to collect data for intelligence, whether it is an intelligence service, domestic or foreign, or business companies aiming to improve their earnings.

In this paper, we aim to portray the potential of Bluetooth as a data collection platform for intelligence by discussing vulnerabilities and exploits in Bluetooth Classic (BC) and Bluetooth Low Energy (BLE), leading to data leakage. We will discuss the findings and resulting data that can be collected from (1) an RCE attack (BC), (2) an attack related to the Android platform (BLE), (3) an attack for the iPhone platform (BLE), and (4) an impersonation attack (both). We will not focus on information about other use-cases for the vulnerabilities or the implementation of countermeasures, nor will we discuss the topics related to ethics or legality.

The rest of this paper is structured as follows: Section two describes the Bluetooth architecture and important aspects that are needed to understand the following sections. Section three explains the methodology for the study. Section four presents the findings of the literature review. Section five discusses the findings further with a focus on relevance, capability, possibilities, and presents our assessment. Section six lists related work to aid the interested reader in finding relevant information and methods. Section seven concludes the paper followed by suggestions for future work.

2 BACKGROUND

The term Bluetooth refers to two technologies: Bluetooth Classic (BC) also known as Basic Rate/Enhanced Data Rate (BR/EDR), and Bluetooth Low Energy (BLE). Despite their typical use in connecting peripheral devices at close range, both Classic and BLE are capable of transmitting up to 100m in open areas, while the current BLE version, 5.0, is rated over 360m [6, p. 11].

In general, BC is preferred for applications using constant transmission of data between the paired devices, like streaming music to Bluetooth headphones connected to a mobile phone. BLE is often used to send short messages advertising a device's information to enable a seamless user experience. While both technologies operate in the 2.4 GHz unlicensed spectrum, each utilizes a different number and size of channels and incorporates channel hopping schemes to avoid interference with other ISM band traffic.

2.1 Bluetooth Stack / Architecture

The Bluetooth stack consists of three main components: (1) the host layer, (2) the controller layer, and (3) the application layer. The Bluetooth standard specifies the Host Controller Interface (HCI) as an interface between the host and the controller, which is used to send commands between them. The implementation of the host is done by the device's operating system, while the implementation of the controller is done by the firmware of the Bluetooth chip in the device. Vendors usually use their proprietary implementations of the host and controller as the standards do not provide a reference for this. There is a slight difference in the architecture of BC and BLE as seen in figure 1.

2.1.1 Bluetooth Controller. The Bluetooth controller consists of two layers. The first layer is the physical layer responsible for functionality like frequency hopping and time-division multiplexing using physical components like radio and baseband controllers.

The second layer depends on the Bluetooth technology, BC or BLE. For BC the Link Manager Protocol (LMP) is responsible for establishing connections and maintaining other functionalities of the device, like discoverability, authentication, and encryption. The LMP communication is unencrypted and unauthenticated, while BLE uses AES encryption for its link layer (LL). BLE has

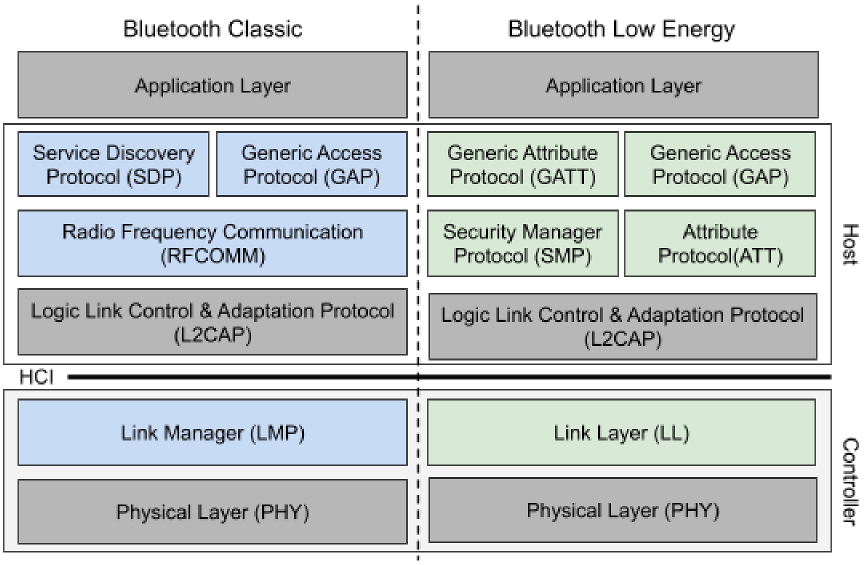
Bluetooth Classic
Bluetooth Low Energy
Application Layer
Application Layer
Service Discovery Protocol (SDP)
Generic Access Protocol (GAP)
Generic Attribute Protocol (GATT)
Generic Access Protocol (GAP)
Radio Frequency Communication (RFCOMM)
Security Manager Protocol (SMP)
Attribute Protocol(ATT)
Host
Logic Link Control & Adaptation Protocol (L2CAP)
Logic Link Control & Adaptation Protocol (L2CAP)
HCI
Link Manager (LMP)
Link Layer (LL)
Physical Layer (PHY)
Physical Layer (PHY)
Controller

Fig. 1. Bluetooth Classic and BLE Architecture.

the LL protocol that manages the advertising, scanning, standby, initiating, and connections of the device. In doing so it is also responsible for changing the role of the device corresponding to one of the aforementioned states.

2.1.2 Bluetooth Host. The Bluetooth host is the logical part of the Bluetooth architecture. It is responsible for connecting and exchanging data with the hosts of other peripheral devices. Below are brief explanations of protocols that will occur in the paper:

(1) L2CAP [both] enables end-to-end communication. It manages the connection between two Bluetooth devices, implementing features like QoS, flow-control, fragmentation, reassembly mechanisms, and offers data encapsulation services to upper layers.
(2) SMP [BLE] provides methods for device pairing and key distributions. It offers services to other protocol stack layers to securely connect and exchange data between BLE devices.
(3) GATT [BLE] is a service framework specifying sub-procedures to use ATT, where these sub-procedures handle data communications between two BLE devices. The applications and/or profiles use GATT directly.
(4) ATT [BLE] allows a BLE device to expose certain pieces of data or attributes.
(5) RFCOMM [BC] is used to generate the serial data stream, which can replace the transmission of data over serial ports.
(6) SDP [BC] broadcasts the services supported by the host device and the associated parameters to other devices, to establish a connection.
(7) ACL [both] is used to enable asynchronous data transfer at the link layer.
(8) SCO [both] is used to enable synchronous data transfer at the link layer, which is faster and more efficient than ACL, but ACL has to be used to set up the connection.

2.1.3 Host Controller Interface (HCI). HCI provides communication between the controller and the host through standard interface types. This HCI layer can be implemented either using API or by interfaces such as USB. Standard HCI commands and events are defined in the Bluetooth specifications [25].

2.1.4 *Application Layer.*

The application layer provides the functionalities offered to users. Application interoperability is accomplished by Bluetooth profiles. A device profile is defined by the general functionality of the device. Each profile contains settings to configure the communications, like the formats of the user interface and dependencies of other protocols. The host protocols decide how to interact with applications and profiles.

2.2 Bluetooth Network

Two or more Bluetooth devices can connect by a four-step process: (1) Link-Layer Connection (LLC), (2) pairing, (3) service discovery, and (4) profile/service connection, to form a Bluetooth network. LLC uses inquires to look for nearby devices and if the inquires are answered the devices start paging to connect on the link layer, either as ACL or SCO connections. The next step is pairing, where encryption and storing of previous connections are implemented. Then the devices will share device and service information about themselves, like their unique identities and capabilities. The connection is established, typically, at the service or profile level.

The network uses a dynamic piconet typology which is an ad-hoc network consisting of a master (central device) and up to seven active slaves (peripheral devices) with common clock synchronization and frequency hopping. While BC is limited to seven peripheral devices due to its 3-bit address space, BLE uses a 24-bit address limiting its number of possible peripheral devices only by the capacity of the network itself [9][12]. The clock timing and frequency hopping is used to ensure that the communication of one piconet is not overlapped with other nearby piconets.

The networks use time-division multiplexing to enable further scalability by having devices communicate with multiple devices in other piconets at the same time. These kinds of communications are referred to as scatternets.

The central device and peripheral devices can switch roles anytime after the establishment of an ACL physical link, which is what helps make the piconet dynamic.

3 METHOD

This paper follows the IMRAD [26] structure and is based on qualitative literature reviews in the fields of information technology and security, computer science, and information from the intelligence domain. We aim to follow the terminology belonging to the intelligence field, for which we consult the Norwegian Defence Intelligence Doctrine [23]. The foundation of discussion in this paper is based on relevant academic sources found using acknowledged online academic search engines and literary databases, like Google Scholar.

The selection of relevant publications regarding vulnerabilities, exploits, and data leakage was done to stay within the scope of this paper, as the length, technical level, and time given dictated that not all aspects of the topic could be covered. The most relevant of which are however mentioned in section six.

As this paper focuses on vulnerabilities and exploits leading to data leakage, it does not provide information about other use-cases for the vulnerabilities, nor the implementation of countermeasures.

Providing sufficient information and examples of novel vulnerabilities and exploitation has proven difficult due to the "cat and mouse" nature of information security and lack of practical testing. We have made our best effort to ensure the quality and novelty of the methods and papers presented.

To grasp the underlying aspects of this paper, please consult a more detailed overview of the Bluetooth architecture in chapter 2. It is viewed as a prerequisite to have some prior knowledge of the technology which is discussed.

4 RESULTS

After researching scientific papers and surveys we found that there are a plethora of opportunities for data collection, either due to vulnerabilities in the hardware, firmware, software, or the inherent nature of the devices themselves. Below is a selection of different vulnerabilities. We present an RCE attack on BC, an attack related to the Android platform, an attack for the iPhone platform, and a general vulnerability for impersonation attacks. The aim is to portray the attack surface by showing different vulnerabilities where we explain some of them, highlighting the exploitation possibilities, and in section five discussing the resulting data collected.

Actor and target models are used below to explain the vulnerabilities and exploitations. The actor is referred to as Charlie, while the targets are Alice and Bob. Each subsection title also includes a box e.g. "[BLE]" to indicate the Bluetooth technology in question.

4.1 CVE-2020-0022 — BlueFrag [BC]

Ruge et al. [22] show in their paper that Bluetooth is vulnerable to RCE via the operating system, on-chip, and/or inter-chip. We include RCE because of the opportunities it presents in regards to data collection if one can remotely access someone's device. Here we present one of the methods that can be relevant to data collection. The effectiveness and usability of the method are further discussed in section five.

BlueFrag is a Bluetooth Classic (BC) zero-click RCE exploit discovered by the team working on the Frankenstein [22] by fuzzing the ACL used for data transfer with L2CAP [21]. To develop the RCE Ruge et al. [22] had to design the return-oriented program (ROP) chain and payload needed to execute arbitrary code. To do so they exploit how Broadcom implemented fragmentation of ACL packets.

4.1.1 Vulnerability Discovery. ACL packets can be used to send fragmented L2CAP packets (to maximize the ACL packet lengths) via the same wires as HCI, which is only allowed by the host's driver as long as the size limits for sending packets to the firmware are followed. Otherwise, the firmware will reject the L2CAP inputs. As the fragmentation and reassembling happens on the host the L2CAP might be vulnerable to heap exploitation.

The fragmentation happens when an L2CAP packet longer than the maximum buffer size is received. The fragmented packets are stored in a partial_packets table with the connection handle as the key. The final partial packet is copied to a buffer, where the end of the packet is stored in partial_packet->offset.

They used L2ping[1] and the fuzzer [22] against an Android device to flip bits randomly in the headers causing the Android Bluetooth daemon to crash. The crash was caused by how Broadcom implemented the fragmentation of L2CAP packets sent between the controller and the host.

The fragmentation used by Broadcom used word vice copying with memcpy to be more efficient, instead of copying individual bytes. The first 64 bytes of data were moved to registers before being written to the target location. Due to the complexity of the implementation, the edge cases like odd lengths and misaligned addresses have to be handled. However, the memcpy showed weird behavior regarding negative lengths resulting in it overwriting the last 66 bytes of the L2Ping request with what was stored previous to the last fragmented packet.

[1]L2ping sends an L2CAP echo request to a Bluetooth MAC address.

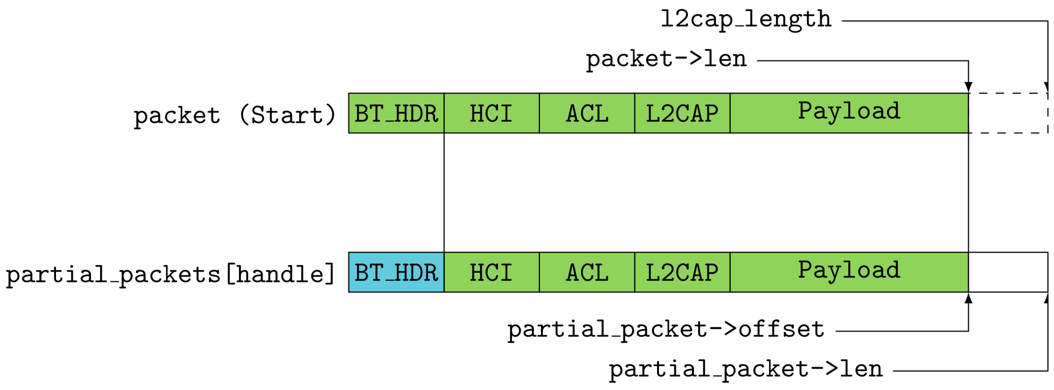

Fig. 2. ACL Packet Fragmentation [Public domain], via INSINUATOR. (https://insinuator.net/2020/04/cve-2020-0022-an-android-8-0-9-0-bluetooth-zero-click-rce-bluefrag/)

The crash happened when memcpy executed with a negative length inside its reassemble and dispatch function, causing it to endlessly copy memory, resulting in the crash when it started writing to unmapped memory.

4.1.2 ROP and Payload Development. This meant that they could use this vulnerability to overwrite the last 64 bytes of the packet with what was in front of their source address, which always were the BT_HDR, acl_hdr, and l2cap_hdr, thus leaking the connection handle of the remote device.

Using the connection handle they could control the last 44 bytes of an ACL packet, and by shortening the first fragmented packet they got the entire packet including headers, which enabled them to insert arbitrary data.

The arbitrary code used system functions found in libraries leaked on the heap to execute code on the remote device. To access them they had to defeat Address Space Layout Randomization (ASLR)[2], which they did by finding the base address for the libicuuc.so library using the offsets between functions on the leaked heap.

Libraries are usually protected by Clang's Call Flow Integrity (CFI) implementation[3] enforcing that only functions that belong to the affected library can be called. However, Ruge et al. [22] found that the ibchrome.so the library was not compiled with CFI enabled, allowing them to retrieve the address of the function as the fragmented packets were stored in a hash-map using the connection handle as a key.

With the known key, they were able to detect the library in the leak, and by using the maximum allowed packet size they had approximately 200 bytes to store their return-oriented program chain (ROP) and payload.

The aforementioned challenge with ASLR and CFI limited them to work with the machine instructions that were located in the libicuuc.so library to create the ROP. As they wanted to access more functions outside of the library they search for machine instructions that would enable this. They found dlsym, a function resolving and returning the address of supplied function names, to obtain the address to the system function outside of libicuuc.so. To return from the function call they used C++ object calls, namely a deconstructor called u_cleanup_60 responsible for iteration over a list of function pointers, and if a pointer is not NULL the address is called and cleared. Lastly,

[2]Commonly used to randomize the base addresses of executables, libraries, the heap, and the stack.
[3]Protects forward edges and prevents overwriting of functions on the heap

they needed to be able to execute code for which they found and used a function that allowed them to specify the next function using registers before performing a useful operation.

This enabled them to fill their payload buffer with different pointers pointing into their buffer, allowing them to get control of useful registers and execute arbitrary code.

Ruge et al. [22] showed that the attack worked in 30%-50% of the cases, and could be repeated endlessly to achieve RCE since the address spaces only got randomized on boot; crashing the Bluetooth daemon did not initiate new randomization of the address spaces.

4.2 Android — BadBluetooth [BLE]

The existing Bluetooth security mechanisms focus on the authentication of remote devices, the confidentiality of communication, and restricting the capabilities of untrusted apps on the host. This is achieved with pairing[4], encryption, and permissions respectively. The security mechanism has the same weak presumption as Wi-Fi, namely assuming that all remote devices are trustworthy. Xu et al. [31] reveal that an actor can manipulate the Bluetooth profiles on a remote device to attack paired Android phones.

This attack exploits the vulnerability caused by the security model's focus on the device level and the issues with having the Bluetooth framework on the hosts. Below is a list of potential vulnerabilities discussed by the authors and their potential implications:

(1) Inconsistent authentication process on profiles — The Bluetooth core specifications do not cover how profiles should be verified.
(2) Overly openness to profile connection — The host proactively tries to connect to all profiles claimed by the remote device.
(3) Deceivable and vague UI — The name and ID of devices can be changed and there is no verification for the action of doing so. Further, little information is available to the user regarding Bluetooth, like profile information.
(4) Silent pairing with device — Just Works mode enables devices without display or input ability to initiate a connection without prompting the user. Manipulating the device configuration can exploit this feature.
(5) No permission management for profile — Bluetooth profiles do not regulate permissions in the Android. Third-party apps can exploit this to control the phone.

BadBluetooth exploits how peripheral devices can compromise the privacy of the user by getting access to higher privileges than intended. Below is a description of the attack scenario, but first, two assumptions have to be made:

(1) The target has installed a malicious app with Bluetooth permissions.
(2) The actor has compromised the firmware of a Bluetooth device (malicious device) to contain malicious code.

The attack is enabled by the ability to change profiles, change the icon, silent pairing, and connecting to hidden profiles on the target phone. The actor gets the malicious device to add the attack profile on the target's phone after the initial pairing to hide it from the user, by broadcasting the service record related to the profile using SDP. Charlie removes the attack profile after the attack. The target would struggle to see this as the phone does not show any information about the profiles that are in use. If the actor modifies the class of the device field [5] he can change the icon, further hiding his presence from the target. Silent pairing is achieved by configuring the remote device to use Just Works mode. The malicious app can then invoke the createBond() API with the MAC address of the device, thus avoiding the user confirmation for the pairing and actions. To get

[4]The four-step process covered in section 2.2

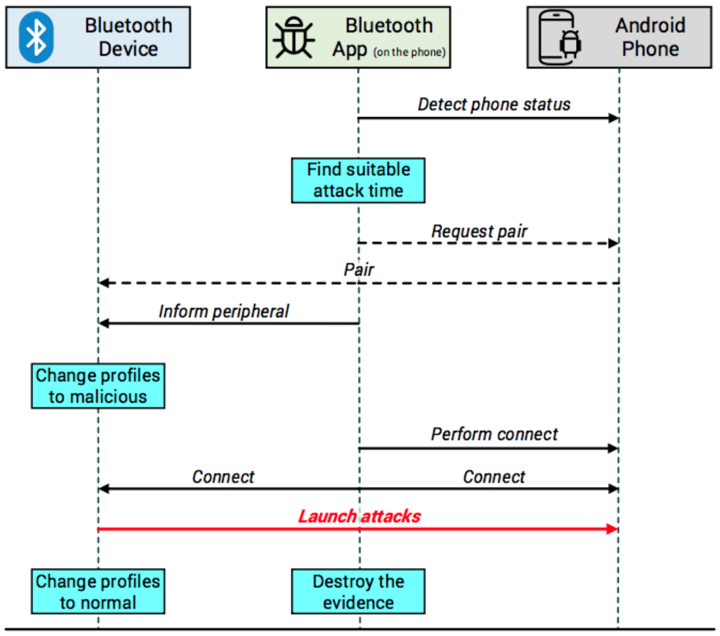

Fig. 3. BadBluetooth Attack Model [31]

access to the hidden profiles Charlie exploits Android's assumption that apps create proxy classes to operate profiles. By using a different framework than intended by Android, he gets direct access to the non-public classes and methods, including the proxies of the hidden profiles. These profiles enable him to establish a channel to the input profile of the remote device. At this point, the actor can covertly use the RFCOMM to communicate with the app.

Xu et al. [31] identified three profiles: Human Interface Device (HID), Personal Area Networking (PAN), and Hands-Free/Headset (HFP/HSP), that could be used to conduct the attack. We present one of the exploitations in the following subsection.

4.2.1 Exploiting Human Interface Device (HDI). Android enforces all runtime permissions by mandatory user involvement. However with the attack method described above the actor can use an HDI device profile to control the phone, emulating legitimate user actions like granting the malicious app runtime permissions, enabling it to be self-sustaining without the need of the remote device.

This enables Charlie to take screenshots, thus the extraction of sensitive data like emails and messages, screen layout, and installed apps by sending the screenshots to his own devices in a covert manner. This can be achieved by granting the app WRITE_EXTERNAL_STORAGE permission using the input ability to get the screenshot before transmitting them via the Internet. Afterward, the app can delete the screenshot. He can further open the camera and capture the surrounding environment, thus severely breaching the target's privacy.

As phones are used as personal vaults keeping a user's identity information and storing the passwords and tokens for many applications, the actor can access the corresponding websites or apps to which the tokens belong. To do so he may steal stored tokens like a verification code in a text message or log herself into a website through a remembered password.

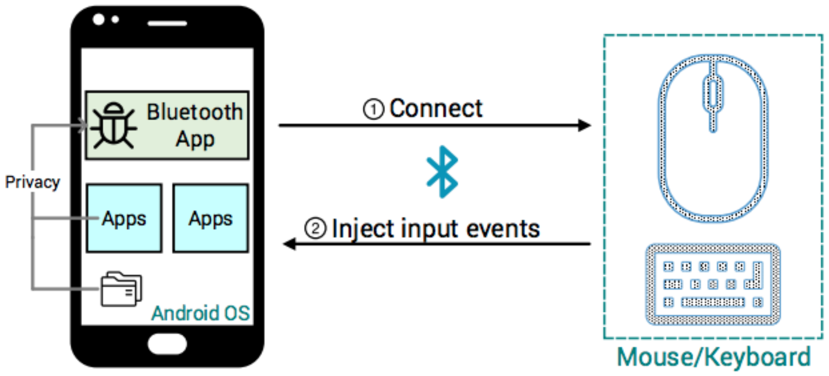

Fig. 4. BadBluetooth HDI Attack Model [31]

The attack is limited when the target phone is locked, as the actor cannot remotely unlock it. However, Charlie can still use some subset operations like rebooting the phone or turning on the camera.

4.3 Apple — Continuity [BLE]

Apple continuity is a term used by Apple to describe their connectivity features between their different Apple devices using BLE. For example, a user can copy the text on an iPhone and paste it on a MacBook as long as the devices are connected to the same iCloud account.

Specially designed BLE advertisement packets are shared between devices connected to the same iCloud account. Martin et al. [17] reverse engineered the proprietary Apple message formats and described how it could be exploited for fingerprinting of OS's and tracking. We further understand how the leaked data can be used to analyze behavioral characteristics.

Apple uses resolvable private random addresses (RPRA) for their BLE-enabled devices to ensure privacy and prevent leakage of public MAC addresses. With RPRA devices authenticate each other with local 128-bit keys, known as Identity Resolving Keys (IRK), and pseudo-random 22-bit sequences (prand), which are used as inputs to a one-way hash function producing a 24-bit hash value. The RPRA is then used as the source MAC address for a set time, usually 15 minutes as the standard defines this as the minimum amount of time.

RPRA has the advantage of allowing potential peers to determine if they already know a device. If the potential peer has the IRK of the remote device it tries to connect to, it can determine whether an advertised random address belongs to that device or not through the following process:

(1) The device computes the 24-bit hash value given the prand value from the resolvable private address and the pre-shared IRK.
(2) It confirms that it associates the identity with the IRK if the values computed match the lower 24 bits of the resolvable address.

Apple takes advantage of the GATT feature in BLE, which is a framework that lets devices discover, read, and write information to and from other BLE-enabled devices. This enforces GATT profiles on the devices which defines the services provided by the device. The services are comprised of characteristics. Both the services and characteristics are identified by Universally Unique Identifiers (UUIDs), to standardize their meaning.

The researchers discovered that GATT inherently leaked model data like the device information and model number when it was requested using GATT queries [3]. An identifier for an iPhone 7 could look like: “iPhone9,1” [17].

An actor can use this to collect the UUIDs for devices by sending GATT query requests. Although he must continuously transmit data he can do so covertly due to two factors: (1) he can use a random source MAC address, and (2) the GATT queries do not prompt the users for acknowledgment. To detect this attack, the network would have to be monitored with the intent of preventing this activity. The authors go on to discuss how this can be used for tracking in combination with the device information, however that is not in the scope of this paper. Interested readers can find more information in their paper [17].

Furthermore, the authors discover that the BLE traffic caused by the continuity features discloses several attributes that can be used to collect behavioral data about the users. Examples of this are (1) the sequence number measurement of the user’s interactions that iterates persistently over time and are regularly broadcasted, and (2) the Nearby Action Code feature that broadcasts information about how the device is used at any given time to the other Apple devices.

4.4 Bluetooth Impersonation Attacks [Classic]

The Bluetooth Impersonation Attacks (BIAS) exploits a design flaw during the secure connection establishment when two devices exchange their capabilities, authenticate the link key (LK), compute the session key, and activate the encryption. The secure connection on Bluetooth is established either with Legacy Secure Connection (LSC) or Secure Connections (SC), depending on the two device’s capabilities. The goal of the attack is to establish a secure Bluetooth connection with one of the two devices by impersonating the other [2].

The target model could be as follows: Consider two target users, Alice and Bob. Both using a secure Bluetooth link to communicate. The actor, Charlie, assumes that Alice and Bob already share a long-term key, known as the link key. The key has been agreed by both by using LSC. Charlie also assumes that Bob is the master and Alice is the slave, and that Bob wants to establish a secure connection with Alice using the link key.

The actor model could be as follows: Charlie does not have the link key of Bob and Alice, and he does not observe them while they securely pair. Charlie can eavesdrop, decode and manipulate unencrypted packets, and jam the Bluetooth spectrum. He also knows the public data about Alice and Bob, which he captures during their secure connection establishment as it is not encrypted. When the secure connection is established, Charlie can jam the Bluetooth spectrum forcing Alice and Bob to disconnect and re-establish a secure connection.

The attack is made possible because of how the LK is authenticated between Alice and Bob. When they establish a connection (LSC, SC) both have to authenticate that they have the shared LK. The secure bond between Alice and Bob is identified by three properties: the LK, the Bluetooth address of Alice (BTADDA), and the Bluetooth address of Bob (BTADDB). Charlie has to change his Bluetooth address to be that of Alice or Bob, depending on whom he is impersonating. However, he will not be able to prove the ownership of LK as he does not have it. This is how Bluetooth authentication is assumed to protect against impersonation attacks.

But, Antonioli et al. [2] show that the BIAS exploits (1) the unencrypted secure connection establishment, (2) lack of mutual authentication requirement for LSC, (3) a Bluetooth device can switch role at any time after the establishment of the baseband connection, and (4) SC paired devices can use LSC during secure connection establishment.

To conduct the BIAS, Charlie targets LSC and SC authentication procedures during secure connection establishment. Both procedures authenticate LK using a challenge-response protocol, and the procedure selection depends on Alice’s and Bob’s supported features.

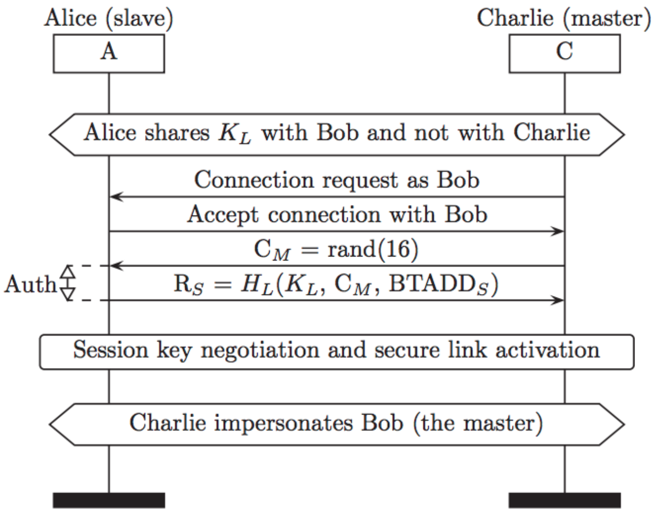
Alice (slave)
A
Charlie (master)
C
Alice shares K_L with Bob and not with Charlie
Connection request as Bob
Accept connection with Bob
$C_M = rand(16)$
Auth
$R_S = H_L(K_L, C_M, BTADD_S)$
Session key negotiation and secure link activation
Charlie impersonates Bob (the master)

Fig. 5. BIAS Attack Model [2].

For an LSC connection, Alice and Bob use the legacy authentication procedure with the LK [25] to establish a secure connection as seen in figure 5 based on a challenge/response mechanism. The master computes and sends a challenge value (CM) to the slave. The slave computes the response (RS) using a one-way hash function and transmits it to the master. The master then computes the same hash function with the same inputs and compares the result to the response from Alice. The master agrees that they share the same key if the resulting values are equal.

The main issue in the standard is that it does not require the LSC to be done mutually by both Alice and Bob [2]. This means that only the master would have to authenticate the slave, thus if Charlie impersonates the master he can complete the secure connection without having to authenticate having the LK to the slave.

This could be achieved with Charlie sending a connection request to Alice, impersonating Bob. Charlie has already captured Bob's address and capabilities and uses them to change his address and capabilities to Bob's. Charlie then sends the challenge to Alice, and Alice computes the response based on his LK, CM, and BTADDA, and transmits it back to Charlie. Charlie then completes the session key negotiation and secure link establishment as Bob, without having to prove he has the LK to Alice.

What makes this attack even more powerful is Charlie's ability to impersonate the slave by exploiting the Bluetooth's role switch procedure. Bluetooth uses a master-slave medium access protocol to keep the master and slave synchronized. According to the standard the master and slave roles can be switched at any time after baseband paging[5] is completed, enabling Charlie to impersonate the slave device by initiating a role switch and become master before the authentication of his is started. He can then complete the secure connection establishment without having to authenticate as mentioned above.

The method for SC is further described by Antonioli et al. [2] in their paper.

[5]see section 2.2

5 DISCUSSION

In this section, we discuss how the aforementioned vulnerabilities and exploits present a wide surface for the data collection on the Bluetooth platform. We further discuss the probability and the capabilities that are needed to take advantage of them.

As presented in the previous chapter there are several vulnerabilities in the Bluetooth architecture at the hardware, firmware, and software level, as well as in the standardization document that all contribute to data leakage or collection in some way.

The Bluetooth standard does not cover all elements of the implementation of the Bluetooth architecture and is flawed in some aspects, like how a user can request a role switch during an authentication procedure, allowing him to avoid proving that he is in possession of the previously shared key. This leads to vulnerabilities that can be exploited by an actor as we have seen in the previous sections. Furthermore, this lack of documentation might also be the reason why some of the vulnerabilities exist in the other architecture levels.

Continuous patches and updates are released for the software and firmware, while the adversaries find new vulnerabilities and ways of exploiting them. However, it does not help to release a patch if it is not updated by its users as people often suppress or delay installing such updates. They, therefore, become easy targets for an actor who wishes to collect data on them. If the vulnerabilities were patched and fixed it is still possible for vulnerabilities of the same nature to reoccur in new IoT technologies or due to added functionality in Bluetooth or the corresponding firmware and hardware.

Apart from patching, we also see the poor implementation of security mechanisms meant to employ mutual authentication as a common denominator. Improving the standards to include best practices for the implementation of the security solutions for authentication would mitigate attacks targeting the pairing of devices.

Furthermore, there is a high probability of the existence of zero-day exploits that are yet to emerge or be documented. Due to the critical nature of zero-day exploits, it is unlikely that an adversary will disclose it if the potential adversary advantage for data leakage is great enough.

5.1 CVE-2020-0022 — BlueFrag [BC]

As seen in section 4.1, an actor could use RCE to execute arbitrary code and get access to one or more targeted phones. There are limitations in what could be achieved, in terms of how much code can be executed on the target, but an actor with sufficient capabilities would be able to plan and execute an attack aimed to extract or install code that will help to collect data about the targets.

This type of attack requires substantial research as there is little publicly available work done on reverse engineer Bluetooth available[16], [22], which means that an actor has to spend time and resources to do so themselves. It is more likely that an actor will choose to use Wi-Fi and its corresponding vulnerabilities as it is more researched and can reach an even wider demographic. However, an actor might want to exploit Bluetooth in special cases where other tools are failing, and in that regard, Bluetooth RCE remains a likely tool for targeted intelligence collection.

5.2 Android — BadBluetooth [BLE]

Android is one of the most used operating systems for phones, and thus presents a huge potential for intelligence collection. The amount of data and information that is stored on and processed by phones today is wast, taking into account that people often use them as digital data vaults for important passwords. Again we see that, partly due to lacking standardization, the Android phones were vulnerable to a combination of malicious apps and compromised Bluetooth firmware.

As mentioned in section 4.2, this attack is based on the user downloading a malicious app, which is a likely scenario. Although Google is doing its best to ensure the safety of the apps available via Google Play, malicious apps are still downloaded [10][28]. These malicious apps are becoming more and more useful as people have started to realize the potential the phone presents. Apps acting like key loggers have been used to steal credentials and have for example been a problem for Tesla as attackers have tricked users to download a key logger app before entering the code to open and start the car in their app, thus giving the attacker access to the car.

In the cases where Wi-Fi proves unreliable, the use of Bluetooth as an intelligence resource would be fairly available to an actor. The actor in this context probably has a suite of pre-made malicious apps, and if they do not, they are not that hard to manufacture or buy. The next step is to modify the firmware of a device that is to be used as a control unit to communicate with the app, which according to Xu et al. [31] does not need to be too hard to accomplish if proper research is available. We, therefore, see this as a likely tool for an actor that aims to collect data on a targeted person.

5.3 Apple — Continuity [BLE]

The actor can also collect data by simply spoofing the network, as Bluetooth devices, especially iPhone with the Continuous solution, constantly transmit BLE messages about the status of the device and user. If one for example utilizes analysis methods found in biometrics this data can be used to characterize users and provide information about their behavior, age, gender, emotional states, etc., even if the data was collected from anonymous users.

There are several methods an actor can use to spoof or collect data from many targets at a time, like installing spoofing devices in public areas. A more novel approach could be to, by the use of RCE or social engineering, download malicious apps on target's phones that turn them into packet sniffers without their knowledge. Thus, the actor would not need to have sniffers installed in all areas of operation, and the reach could be potentially greater. For instance, an infected device could be brought into an enclosed guarded area that the actor does not have access to, and actively sniff the Bluetooth traffic of the other devices. Furthermore, it could enable sniffing in areas without Wi-Fi or mobile coverage e.g. inside a restricted facility. The device would capture the data inside and once it is connected to the internet or have cellular coverage again, it can forward the collected data covertly to the actors. This could be exacerbated by implementing the vulnerabilities in Bluetooth malware. This demands even higher capabilities of the actors in question. We found some information about Bluetooth malware [14], but the discussion of this is reserved for future work.

This data leakage is likely to be used by actors seeking to collect intelligence on single targets or larger masses, as it is accessible and cost-effective. The actor would need to be capable of conducting meaningful analysis of the collected data, but other than that it is low-hanging fruit, for intelligence agencies, businesses, or hacktivists.

5.4 Bluetooth Impersonation Attacks [Classic]

With BIAS, the actor jams the Bluetooth signals after spoofing the network to get the device information needed to impersonate the targets. The targets might discover that they are no longer communicating with the other party, but the actor can simply use the attack to act as a man-in-the-middle, or he might attack a connection between the user and one of his peripheral devices, like a Bluetooth keyboard, and imitate that device instead.

During our work, we realized that more vulnerabilities can be used to covertly achieve a man-in-the-middle state since as we have chosen a broader focus for this paper we are not able to discuss them all. However, the KNOB attack discovered by Antonioli et al. [1] shows how severe some of the vulnerabilities in BC can be. KNOB enables an actor to covertly break the BC security exploiting a vulnerability in the negotiation protocol used for the encryption keys and establish a man-in-the-middle attack. Similar to BIAS, the actor does not need any knowledge of the link or encryption keys, but he has to be capable of modifying the LMP packets that are used by the encryption key negotiation protocol of devices wanting to agree upon the entropy of the key. This negotiation is transparent on the network and not authenticated or encrypted, which enables him to force a lower entropy for the negotiated key between e.g. Alice and Bob, which he can easily brute force using the InternalBlue [16] toolkit combined with Ubertooth One [19] as shown by Antonioli et al. [1].

We can further imagine that an actor in this context has the capability of using these vulnerabilities to enable mass man-in-the-middle attacks where an actor can exploit several users of Bluetooth in a short period over a short distance e.g. if the actor is on a plane or an airport, where almost everyone is using some kind of Bluetooth technology. One way to do this could be by an automated version of the previously discussed BIAS, where the adversary would be able to quickly jam the people's connections before impersonating more or less everyone, thus getting access to the data of all the users.

There are multiple vulnerabilities, and it seems to be fairly simple to exploit in the context of a capable intelligence agency. We also see this as a viable tool for hacktivists that want to collect data about their targets.

5.5 Worthy Mentions

In our research, we discovered that there is inherent data leakage that comes from just using the technology as it produces data about the users. Below is a summary of two interesting ways IoT devices in the future can leak data about the owners.

5.5.1 Light Ears. Smart lights are IoT devices that let the user change colors, brightness, and more. Some of the newer features are audio-visualizing and video-visualizing. Maiti and Jadliwala [15] look at how an adversary can deduce what a targeted user watches or listens to by visually eavesdropping on the emitted light from the smart bulb, without attacking the user's wireless network. The authors also go into further details on how the actor can use the IR-light functionality of the smart lights to exfiltrate data [15].

5.5.2 CovertBand. Nandakumar et al. [18] presents what they call a "[...] novel method for low-cost, covert physical sensing and, by doing so, surfaces new privacy threats." They demonstrate how a smartphone and portable speakers can play music with embedded, inaudible signals that can be used to track people's locations and activities in a room or through obstacles like a door or a wall. The authors go through how this is done in great detail in their paper [18].

5.6 Limitations

This paper was limited by the time, technical level, and length that were given in the scope, which meant that we had to neglect a lot of good sources as we wanted to portray a wider range of opportunities. However, we have dedicated chapter six below to related work we found that can further help the interested reader to explore the field and opportunities.

The scope also meant that we did not discuss the legal or ethical aspects of the vulnerabilities and exploits that are covered in this paper. We are aware that intelligence services are restricted by strict laws and regulations that are meant to limit their actions but enable the population to trust the intentions of their national agencies [23]. However, the covered material might as well be used by agencies or people that are not enforced by the same ethics, laws, and regulations.

6 RELATED WORK

Garbelini et al. [7] and Ruge et al. [22] presents two fuzzing frameworks, SweynTooth and Frankenstein, which targets Bluetooth. The first of which focuses on BLE specifically, while Frankenstein uses advanced firmware emulation to be able to avoid the lower speed, limited repeatability, and debugging resulting from over-the-air fuzzing. InternalBlue is a Bluetooth binary patching and experimentation framework that was developed by the authors of InternalBlue [16]. In their paper, they demonstrate its powers to discover security issues. All three papers are of interest to the reader who seeks a greater understanding of the Bluetooth framework.

As we were not able to cover tracking issues with Bluetooth, be aware that some of the studies that have been presented in this paper also mention tracking as an issue Martin et al. [17]. Other researchers have covered different aspects of the problem. Becker et al. [4] discuss how the BLE device's use of public advertising channels enables adversaries to track the devices. Further problems with Bluetooth finders, which are used to find objects, are discussed by Weller et al. [29] as they discover vulnerabilities in the technology that leaks private data about the owners.

Malware is one thing we wished we had more time to cover as it presents a huge potential to spread and exploit the vulnerabilities presented in this paper. Mahboubi et al. [14] covers a general look and the impact of how malware on mobile platforms with a chapter about the use of Bluetooth. It can be a good read for the interested reader that wants to get an understanding of the possibilities for mobile malware.

In this paper, we showcased how vulnerabilities in BC enable BIAS and other man-in-the-middle attacks. However, this is not unanimous for BC. In their BLESA paper Wu et al. [30] discuss several security issues stemming from how the pairing of devices with BLE can be exploited as they often require little to no user interaction.

At last, three papers covering the security threats of Bluetooth; from practical analysis to literature reviews, were used to map out this paper [13],[20],[11]. They are all of interest for anyone who wants to get a handle on the vulnerabilities that exist, in addition to those that leak data, which has been the focus of this paper.

7 CONCLUSION

From an intelligence perspective, Bluetooth appears as a good resource for data collection, both for targeting single persons or masses. We have seen that an actor can collect behavioral data that, if analyzed properly, can be used to profile the users and deduce their decisions and habits, both for marketing but also as a tool for mass manipulation. Big data analysis has proven to be one of the most powerful tools of the digital age, similarly to how social networking sites have been used as a data collection source, Bluetooth vulnerabilities can be utilized to collect data. We also find that the vulnerabilities could be used to track, extract data, listen to calls, and even take control of targeted devices through RCE. Some of the methods discussed are complex and require preparation and sufficient capabilities, and might be less accessible for hacktivists or smaller groups, but for a capable intelligence agency, all of the techniques should be accessible and are likely to be used to collect data in the future. However, the monitoring of network information is a simple task, and might in most cases yield the best results vs. cost. In conclusion, although many of the vulnerabilities demand advanced knowledge in reverse engineering, networking, and software development, a lot of the data is still available by spoofing network traffic, making Bluetooth a data collection platform for anyone within a proximity of 100-360m depending on the Bluetooth technology.

8 FUTURE WORK

Future work should look at practical applications of the vulnerabilities and methods discussed in this paper to better present the potential risk and impact. For example by investigating the implementation of the vulnerabilities in Bluetooth or mobile malware for increased malware propagating. Other work should be aimed towards mitigating the vulnerabilities on the different levels of Bluetooth architecture that enable data leakage and extraction. Further research can be conducted on how to limit the inherent data leakage from Bluetooth technology.

REFERENCES

[1] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. 2019. *The KNOB is Broken: Exploiting Low Entropy in the Encryption Key Negotiation Of Bluetooth BR/EDR.* https://www.usenix.org/system/files/sec19-antonioli.pdf

[2] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. 2020. *BIAS: Bluetooth Impersonation AttackS, Proceedings of the IEEE Symposium on Security and Privacy (S&P).* https://francozappa.github.io/about-bias/publication/antonioli-20-bias/antonioli-20-bias.pdf

[3] Apple. 2021. *Identify your MacBook Pro model.* https://support.apple.com/en-us/HT201300

[4] Johannes K Becker, David Li, and David Starobinski. 2019. *Tracking Anonymized Bluetooth Devices.* https://www.researchgate.net/publication/334590931_Tracking_Anonymized_Bluetooth_Devices/fulltext/5d3308db92851cd04675a469/Tracking-Anonymized-Bluetooth-Devices.pdf

[5] Bluetooth. 2021. *Assigned Numbers for Baseband.* https://www.bluetooth.com/specifications/assigned-numbers/baseband/

[6] Bluetooth. 2021. *Bluetooth Core Specification Version 5.0 Feature Overview.* https://www.bluetooth.com/bluetooth-resources/bluetooth-5-go-faster-go-further/

[7] Matheus E. Garbelini, Chundong Wang, Sudipta Chattopadhyay, Sun Sumei, and Ernest Kurniawan. 2020. *SweynTooth: Unleashing Mayhem over Bluetooth Low Energy.* https://www.usenix.org/conference/atc20/presentation/garbelini

[8] Nathaniel Gleicher. 2021. *Removing Coordinated Inauthentic Behavior.* https://about.fb.com/news/2020/10/removing-coordinated-inauthentic-behavior-september-report/

[9] Carles Gomez, Joaquim Oller, and Josep Paradells. 2012. *Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology.* file:///Users/oleandrehauge/Desktop/sensors-12-11734.pdf

[10] Google. 2021. *Help protect against harmful apps with Google Play Protect.* https://support.google.com/googleplay/answer/2812853?hl=en

[11] Shaikh Shahriar Hassan, Soumik Das Bibon, Md Shohrab Hossain, and Mohammed Atiquzzaman. 2018. *Security threats in Bluetooth technology.* https://www.sciencedirect.com/science/article/pii/S0167404817300615

[12] Laird. 2021. *Does Bluetooth low energy feature a piconet as in classic BT?* https://www.lairdconnect.com/support/faqs/does-bluetooth-low-energy-feature-piconet-classic-bt

[13] Santiago Figueroa Lorenzo, Javier Añorga Benito, Pablo García Cardarelli, and Jon Alberdi Garaia. 2017. *A Comprehensive Review of RFID and Bluetooth Security: Practical Analysis.* https://www.researchgate.net/publication/330603078_A_Comprehensive_Review_of_RFID_and_Bluetooth_Security_Practical_Analysis

[14] Arash Mahboubi, Seyit Camtepe, and Hasmukh Morarji. 2017. *A Study on Formal Methods to Generalize Heterogeneous Mobile Malware Propagation and Their Impacts.* https://ieeexplore.ieee.org/document/8105807

[15] Anindya Maiti and Murtuza Jadliwala. 2018. *Light Ears: Information Leakage via Smart Lights.* https://arxiv.org/pdf/1808.07814.pdf

[16] Dennis Mantz, Matthias Schulz, Jiska Classen, and Matthias Hollick. 2019. *InternalBlue – Bluetooth Binary Patching and Experimentation Framework.* https://arxiv.org/pdf/1905.00631.pdf

[17] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik Rye, Brandon Sipes, and Sam Teplov. 2019. *Handoff All Your Privacy – A Review of Apple's Bluetooth Low Energy Continuity Protocol.* https://arxiv.org/pdf/1904.10600.pdf

[18] Rajalakshmi Nandakumar, Alex Takakuwa, Tadayoshi Kohno, Shyamnath, and Gollakota. 2017. *CovertBand: Activity Information Leakage using Music.* https://musicattacks.cs.washington.edu/activity-information-leakage.pdf

[19] Michael Ossmann. [n.d.]. *Ubertooth.* https://github.com/greatscottgadgets/ubertooth

[20] Mookyu Park, Haengrok Oh, and Kyungho Lee. 2019. *Security Risk Measurement for Information Leakage in IoT-Based Smart Homes from a Situational Awareness Perspective.* https://pubmed.ncbi.nlm.nih.gov/31075883/

[21] Jan Ruge. 2020. *CVE-2020-0022 an Android 8.0-9.0 Bluetooth Zero-Click RCE – BlueFrag.* https://insinuator.net/2020/04/cve-2020-0022-an-android-8-0-9-0-bluetooth-zero-click-rce-bluefrag/

[22] Jan Ruge, Jiska Classen, Francesco Gringoli, and Matthias Hollick. 2020. *Frankenstein: Advanced Wireless Fuzzing to Exploit New Bluetooth Escalation Targets.* https://www.usenix.org/system/files/sec20-ruge.pdf

[23] Norwegian Intelligence Service. 2021. *FORSVARETS ETTERRETNINGSDOKTRINE (2021).* https://www.forsvaret.no/om-forsvaret/organisasjon/etterretningstjenesten/Etterretningsdoktrine_2021.pdf/_/attachment/inline/55bb2c9b-7d43-4e7d-b7fa-c44991654a40:45144a0f0efb5dfe84d424ff06d61c0d80dc111a/Etterretningsdoktrine_2021.pdf
[24] The Norwegian Intelligence Service. 2021. *FOCUS.* https://www.forsvaret.no/aktuelt-og-presse/publikasjoner/fokus/rapporter/Focus2021-english.pdf/_/attachment/inline/450b1ed0-1983-4e6b-bc65-4aa7631aa36f:21c5241a06c489fa1608472c3c8ab855c0ac3511/Focus2021-english.pdf
[25] Bluetooth SIG. 2021. *Bluetooth Core Specification 5.2.* https://www.bluetooth.com/specifications/bluetooth-core-specification
[26] Springer. 2020. *Overview of IMRaD structure.* https://www.springer.com/gp/authors-editors/journal-author/overview-of-imrad-structure/1408
[27] Sun-tzu and Samuel B. Griffith. 1964. *The art of war.* Oxford: Clarendon Press.
[28] Haoyu Wang, Hao Li, and Yao Guo. 2019. Understanding the Evolution of Mobile App Ecosystems: A Longitudinal Measurement Study of Google Play. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1988–1999. https://doi.org/10.1145/3308558.3313611
[29] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. 2020. *Lost and Found: Stopping Bluetooth Finders from Leaking Private Information.* https://arxiv.org/abs/2005.08208
[30] Jianliang Wu, Yuhong Nan, Vireshwar Kumar, and Dave (Jing) Tian. 2020. *BLESA: Spoofing Attacks against Reconnections in Bluetooth Low Energy.* https://www.usenix.org/system/files/woot20-paper-wu-updated.pdf
[31] Fenghao Xu, Wenrui Diao, Zhou Li, Jiongyi Chen, and Kehuan Zhang. 2019. *BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals.* https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_06B-4_Xu_paper.pdf