SET YOUR TITLE USING \title

I. Background and Related Research

1.1 Overview

This chapter provides a technical summary of the Wi-Fi and Bluetooth Low Energy (BLE) protocols as they pertain to the work presented in this thesis. It also provides a brief overview of other comparable wireless protocols. It follows with an outline of the current state of Internet of Things (IoT) and smart home security, survey of open-source tools used in this work, and discussion of related research.

1.2 Wireless Protocols

1.2.1 Wi-Fi.

The 802.11 wireless specification defines the physical and link layers for communication in the 2.4 GHz radio band [1]. The 802.11 architecture contains four major physical components: (i) access points, (ii) wireless medium, (iii) stations (devices), and (iv) distribution systems (i.e., router) [17]. In a typical home Wi-Fi network, the access point (AP) and router are combined into one unit which is used to connect the network to the Internet. During setup of a secure network, such as the one analyzed in this work, the AP is assigned a Service Set Identifiers (SSID), channel number, and password. Each wireless station must prove knowledge of the password to associate with the AP and communicate within the network. Association to a secure network results in the encryption of wireless traffic sent within the network. Figure 1 depicts the frame format for the Media Access Control (MAC) Protocol

Data Unit (MPDU), the unit of data exchanged between entities using the physical layer (wireless medium). It also shows which portions are encrypted when connected to a secure wireless AP. Figure 2 provides the fields within the MAC Header which include addressing information for Wi-Fi packets at the link layer. Only the first three addresses pertain to this work and indicate the **BSSID!** (**BSSID!**), **SA!** (**SA!**), and **DA!** (**DA!**) MAC addresses [1]. The MAC Header is not encrypted.

Frame Control	Duration/ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	QoS Control	HT Control

Figure 1. MPDU format when using WPA-2 [1]

Frame Control	Duration/ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	QoS Control	HT Control

Figure 2. MAC Header Frame Format [1]

In normal operation, **WNIC!** (**WNIC!**)s only pass traffic destined to the client station dropping all other packets [17]. There are two other modes that select **WNIC!**s can operate in which include promiscuous and monitor mode [26]. First, promiscuous mode sets the **WNIC!** to pass any traffic with the station's associated AP as the **BSSID!** of the packets. The second mode, used throughout this work, is monitor mode which sets the **WNIC!** to pass any traffic regardless of the **DA!** or **BSSID!** to the client.

Two other values pertinent to this work include the time of packets and the Frame Size (FSize), or packet size. The packet timestamp is calculated by the host kernel, while the receiving application determines the packet size [2].

1.2.2 Bluetooth Low Energy (BLE).

The Bluetooth Special Interest Group (SIG) introduced BLE (Bluetooth Smart) in Bluetooth Core Specification v4.0 to complement Bluetooth Basic Rate/Enhanced Data Rate (BR/EDR) (Bluetooth Classic) [6]. Although these two implementations share some key attributes (e.g., both operate in the 2.4GHz band and use adaptive frequency hopping), they are different protocols with unique design goals [13]. While Bluetooth Classic is used in high-bandwidth applications, such as transferring files or streaming audio, BLE is designed to minimize power, cost, and data rate. These goals are accomplished by limiting overhead at every level of the architecture and using simple communication protocols. To this same end, BLE devices predominantly transmit state data, such as whether a light is on/off, in short, infrequent bursts. These characteristics make BLE ideal for IoT applications where battery life is a top priority.

Bluetooth Core Specification v5.0 was adopted in December 2016, however, the majority of commercially-available devices today still use v4.2; this work is focused on devices using v4.2 or older. The rest of this section discusses elements of the BLE architecture (shown in Figure 3) and protocol which are relevant to the focus of this research.

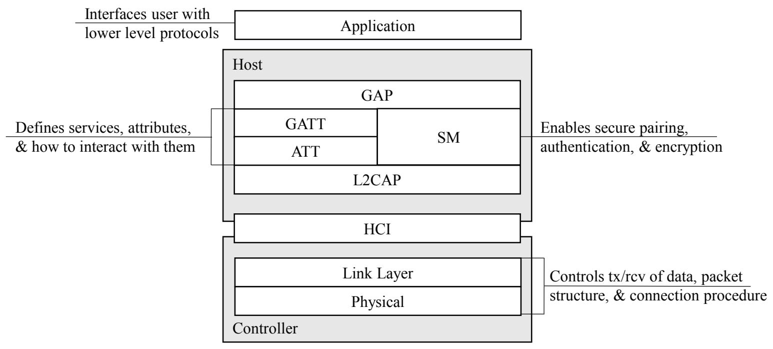
Interfaces user with lower level protocols
Application
Host
GAP
GATT
ATT
SM
L2CAP
Defines services, attributes, & how to interact with them
Enables secure pairing, authentication, & encryption
HCI
Link Layer
Physical
Controller
Controls tx/rcv of data, packet structure, & connection procedure

Figure 3. The Bluetooth Low Energy Architecture

1.2.2.1 Attribute Protocol.

Data is communicated between BLE devices in the form of "attributes" using a client-server architecture ruled by the Attribute Protocol (ATT). Each attribute contains state information addressed by a unique handle and type. These attributes are then grouped into characteristics based on discovery method and accessibility. The master device (e.g., smartphones, computers) typically acts as the client, periodically reading/writing information in the form of attributes from/to the server (e.g., locks, sensors) as required by a user. For example, a user (client) may request the status of a door lock using an ATT Read Request with the type "Lock Status" and handle 0x0019; the device (server) then responds with an attribute containing the value "Unlocked." When the user decides to lock the door, an ATT Write Request is sent with the value set as the user's password; if a valid value is provided, the lock changes state to "Locked." The attack presented in Section **??** of this work uses the ATT commands and Generic Attribute Profile (GATT) characteristics BLE devices use to communicate.

1.2.2.2 Security Manager.

The Security Manager (SM) defines a process to secure BLE connections called pairing and bonding [13]. When a device wants to create a new connection in which security parameters have not been previously exchanged or have been forgotten, the devices must first establish a trust relationship through the pairing process. While there are application specific ways to implement the SM, generally, pairing is accomplished by the devices exchanging pairing information, authenticating each other, encrypting the link, and then sharing keys. Three keys are exchanged during pairing: (i) a shared 128-bit Long Term Key (LTK) used to encrypt future connections with Advanced Encryption Standard (AES)-128 encryption; (ii) a 128-bit Connection

Signature Resolving Key (CSRK) for authentication; and (iii) a 128-bit Identity Resolving Key (IRK) for privacy. After pairing is accomplished, bonding is simply saving the keys for faster connection establishment in the future. If either device loses the encryption keys, the entire pairing and bonding sequence must be re-accomplished. The security of a connection is dependent on how the SM is implemented. Many of the devices examined in this work were vulnerable due to poor SM implementations that did not enforce encryption or authentication.

1.2.3 Physical and Link Layers.

The physical and link layers control finding devices, establishing connections, packet structure, and transmitting/receiving data. For a connection to occur, one device advertises its presence while another scans. When the scanning device sees the correct advertising device, a connection is created. These advertisements and the overall connection process are key to the work done in observing pattern of life information via BLE sniffers. Connections occur between one master and one slave and are broken up into a series of Connection Events (CE) with the master transmitting packets during a CE and each CE occurring on a different channel. The connection parameters are set by the master in the connection request packet and include the frequencies to be used and CEs interval. Each step of this process is shown in Figure 4 and described below.

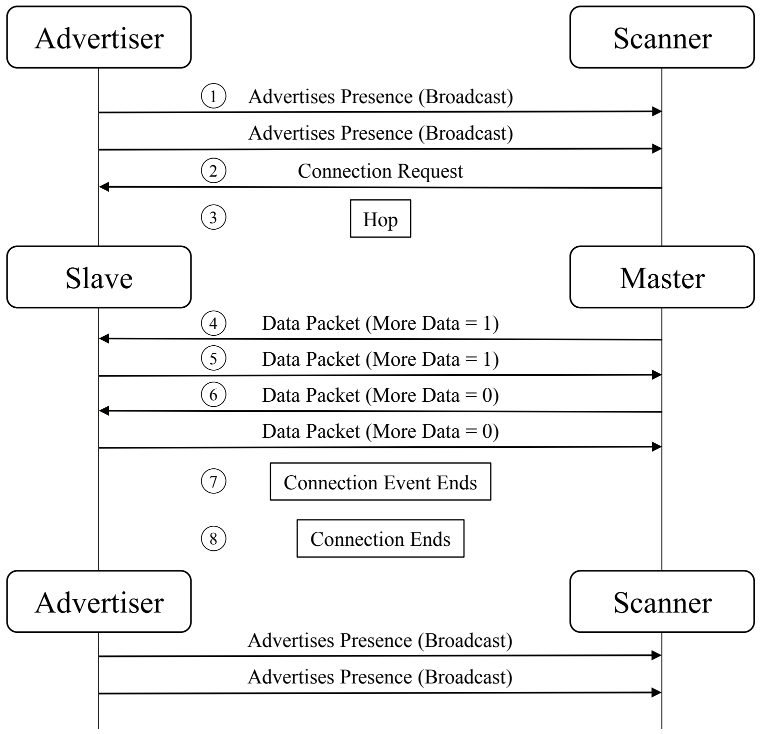
Advertiser
Scanner
1 Advertises Presence (Broadcast)
Advertises Presence (Broadcast)
2 Connection Request
3 Hop
Slave
Master
4 Data Packet (More Data = 1)
5 Data Packet (More Data = 1)
6 Data Packet (More Data = 0)
Data Packet (More Data = 0)
7 Connection Event Ends
8 Connection Ends
Advertiser
Scanner
Advertises Presence (Broadcast)
Advertises Presence (Broadcast)

Figure 4. The BLE Connection Process

1- Device Advertises Presence. A connection begins with a slave announcing its presence by broadcasting advertising packets on three advertisement channels (see Figure 6). Each advertising packet includes device information such as connectability, scannability, services provided, and name of the device. Advertising packets also include a "TxAdd" bit that indicates if the advertiser is using a public or random address. A master actively or passively scans the advertisement channels detecting connectible devices. Active scanning is a key concept for the security discussion in this work and the process is depicted in Figure 5. When actively scanning, a master observes an advertising packet and, if the device is scannable, sends a scan request to the device. The advertiser sends a scan response back with more information, typically expanding on the device name and possibly including broadcast data such as battery level. A master can only connect to a device that advertises its presence and is connectable.

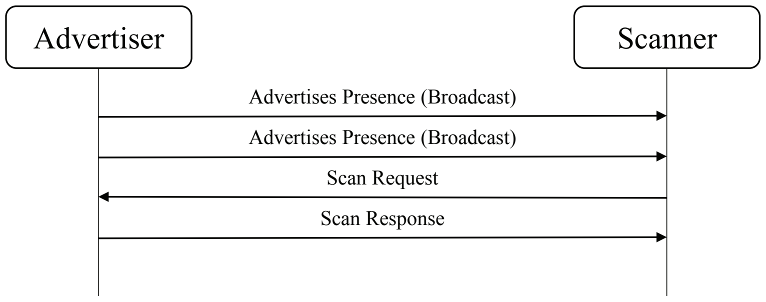
Advertiser
Scanner
Advertises Presence (Broadcast)
Advertises Presence (Broadcast)
Scan Request
Scan Response

Figure 5. Active Scanning Process

2- Initiator Sends Connection Request. Once a scanner observes a connectible device, a connection request packet is sent. This packet establishes all of the necessary parameters to start the connection to include the access address, connection interval, transmit window size and offset, hop interval, channel map, and sleep clock accuracy (SCA). The access address is a random value used to identify packets that are part of the connection. The transmit window, which is calculated using the transmit window offset and size, indicates when the first CE will occur. Likewise, the connection interval dictates when each subsequent CE will occur. The hop interval and channel map determine which frequencies will be used during the connection. Finally, the SCA is vital for synchronization and establishes the amount of time before and after a CE the slave must listen to account for any clock drift between the master and slave. The master may update a subset of these parameters at any time in a connection parameter update message. In this message, the master provides a future time at which the new parameters will take effect. To follow a connection, the sniffer used in this work must observe and implement all of the connection parameters and potential changes throughout the connection.

3- Hop. As shown in Figure 6, the BLE frequency band is divided into forty channels separated by 2-MHz. These frequencies are distributed into three advertising channels and thirty-seven data channels. When in a connection, a master and slave communicate on one channel per CE. After each CE, both the master and slave hop to a new frequency per the Channel Map, Hop Increment, and hopping algorithm; these parameters are established by the master at the beginning of a connection or in a parameter update and are non-negotiable.

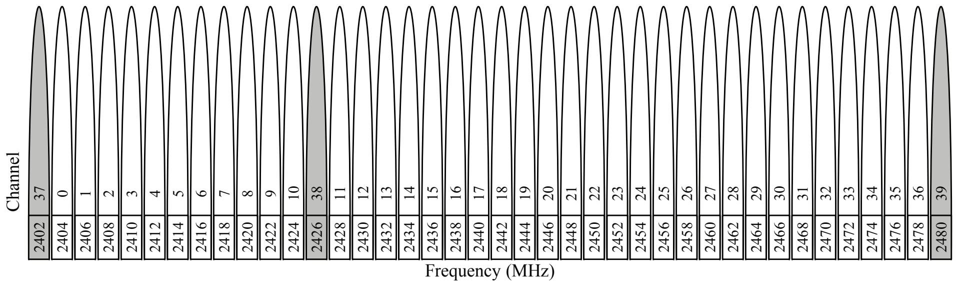

Figure 6. Bluetooth Low Energy channel mapping; darker channels represent advertisement channels

4- Master Sends Data Packet. The first data packet sent in a CE is called an anchor which establishes the timing for all future CEs. A master can send an empty packet to maintain the connection.

5- Slave Responds. The slave must always respond to a received data packet from the master unless two consecutive packets are received with an invalid Cyclic Redundancy Check (CRC). However, to conserve energy, the slave does not have to listen to a predetermined number of CEs.

6/7- Packets are Sent until Connection Event Ends. The length of a CE is at most the predetermined connection interval, but may be shorter depending on how much data needs to be transmitted. If the slave is listening to the CE and responds to the anchor packet, the master and slave exchange packets until a CE end condition is met. There are four ways to close a CE– if neither device has more data to send (indicated by the more data bit in the packet); if the more data bit is set and either slave or master do not receive a subsequent packet within 150 μs; if two consecutive packets are received with an invalid CRC; or if the connection interval is reached.

There may be multiple CEs in one connection; the end of a CE does not mean the connection will end. This allows the slave to conserve energy and not listen when no information is sent from the master and the master to send packets to a new device while still maintaining a connection with the initial device. CEs can be likened to bursts of data and each CE ends at the end of the burst while the overall connection is still maintained.

8- Connection Ends. A connection continues until either device sends a terminate indication packet, if no packets are received within the supervision timeout, or if the message integrity check fails. After a connection ends, the slave resumes advertising its presence.

1.2.4 Other Wireless Protocols.

Other protocols used in IoT applications include those based on IEEE 802.15.4 (e.g., ZigBee) and on the ITU-T G.9959 recommendation (e.g., Z-Wave) [24]. These wireless protocols have many of the same privacy leakage issues found in BLE and Wi-Fi. Similar to BLE, these protocols have proper encryption, but do not encrypt the physical layer [24]. This creates unique security challenges for wireless broadcast networks in which anyone with a properly-tuned receiver can see these data packets. Also, the physical and link layers for each of these protocols inherently advertise legitimate information before and after a connection is established that can be used by an attacker.

Zigbee advertises MAC addresses as well which has been used to infer whether a person is in a room or not [16]. Z-Wave provides source, destination, and home identification information that can be used in reconnaissance and device tracking [5].

Efforts are being made to provide techniques to limit the amount of data leakage by these protocols. Some examples include periodically changing MAC addresses, encrypting lower-layer data packets, and not setting devices in active service discovery mode SSIDs [12]. This work seeks to observe and prevent privacy leakage in BLE and Wi-Fi through the understanding of IoT leakage and design of mitigation tools.

1.3 Smart Home Technologies

A list of smart home technologies relevant to the work in this paper is provided:

- **Devices**: BLE or Wi-Fi devices such as switches, smart outlets, cameras, or sensors. Devices can be connected to and controlled by controllers.
- **Controllers**: A master device such as an iPhone or Android phone that connects to a device within the smart home to get status updates or change states.
- **Hub**: A system that sits on the home network, connects to different devices via the manufacturer **API!** (**API!**), and exposes control of the devices via a centralized application on the `controller`. Hubs often provide access to the devices while a user is away from the smart home. Examples of hubs include Apple's HomeKit and the open-source server Homebridge.
- **Apple's HomeKit**: A hub that provides a controller with voice control and automation capabilities for devices.
- **Homebridge**: An open-source server that emulates the iOS HomeKit **API!** to expose supported devices to Apple's HomeKit. Added as a hub in the HomeKit, it allows a user to use Siri voice commands to control devices that are not typically supported within HomeKit.
- **Applications**: Many smart home devices require proprietary applications to interact with the device's full range of capabilities. A controller must use these applications to control the device.

1.4 Tools

Table 1 provides a list of open-source tools used throughout this work.

Table 1. Wi-Fi and BLE tools used throughout this work.

Tool Name	Version	Description
Ubertooth One	Firmware: 2017-03R2	Bluetooth sniffer with open-source firmware and hardware [20]
BlueZ	5.43	Linux Bluetooth stack with utilities to scan for BLE devices and transmit packets [21]
Plugable USB	2.0	Commercial Broadcom BCM20702-based Bluetooth adapter to communicate with Bluetooth Devices
Alfa Card	AWUS036ACH	802.11ac Wireless Adapter
Airodump-ng	Aircrack-ng 1.2	Wi-Fi network security tool to capture raw 802.11 frames
Python	2.7.10	Programming language used in scripting
Pyshark	0.3.7.8	Python wrapper allowing python packet parsing with wireshark dissectors [15]
Scapy	2.3.3	Interactive packet manipulation tool used to send or receive 802.11 packets [28]

1.5 Related Research

Although Wi-fi and BLE smart home devices are becoming commonplace, the privacy leakage and security vulnerabilities of these devices is largely unexplored. In 2016, Ed Skoudis presented a voice controlled and automated IoT smart office architecture, J.A.R.V.I.S. [25]. J.A.R.V.I.S. represents the way forward for smart homes by integrating devices, Apple's Homekit, and automation, but Skoudis admits that security was an afterthought in developing the architecture. At the end of his presentation, Skoudis challenges developers to explore the security implications of the growing IoT field. The Smart Home Automation Architecture (SHAA) developed in this work is influenced by Skoudis' work, but extends on it by expanding on the number of devices, including BLE devices, and integrating a privacy leakage mitigation method (Mitigation of IoT Leakage (MIoTL)). This work also explores the privacy consequences of a smart home architecture such as J.A.R.V.I.S. by analyzing privacy leakage in BLE and Wi-Fi devices.

While the BLE specification defines security procedures to encrypt the payload, generate private addresses, and provide authentication [7], implementation of the SM is left up to the developer; each additional security measure contributes to increased energy consumption [13]. Limiting power, developing devices quickly, and other design constraints drive developers toward poor implementation of the SM, leaving devices with essentially no Link Layer authentication or encryption.

Recently, oversight in Link Layer security has enabled researchers to crack twelve BLE locks from up to a quarter mile away [23]. Two man-in-the-middle frameworks were developed due to the lack of Link Layer security that allow home automation denial of service, data manipulation, and command injection [14][27]. The lack of lower-layer security employment also creates vulnerabilities in firmware update procedures; a team of researchers were able to upload customized firmware onto a BLE industrial monitor that then provided false sensor readings or locked out legitimate users [3]. Similarly, a lack of encryption results in unintended privacy leakage. In a few recent studies focused on BLE wearable fitness trackers, one group of researchers observed device address and connection information in the clear that enabled them to identify users based off of activity level and gait [9], while another group used device addresses and Received Signal Strength Indicator (RSSI) to track a user wearing a Fitbit Surge up to 1,000 meters away with greater than eighty percent accuracy [22]. Privacy data has also been used to create pattern mining models to track tourist attraction visits in Belgium to help determine the best locations to put hotels [29].

Privacy leakage in Wi-Fi has likewise been exploited in recent research. Researchers have used Wi-Fi MAC addresses sent in the clear and RSSI values to create location tracking systems on campuses, crowd tracking tools at mass events, and in Customer Relationship Management (CRM) allowing commercial businesses to track customer interactions and data [18][8][4]. A group from the United Kingdom were able to use raw Wi-fi signals to create fingerprinting techniques able to identify applications used on a mobile phone [4]. One researcher was able to use raw Wi-Fi signals to activate alerts when a security camera observes motion [19]. This research, however, did not look at other types of smart home devices or provide methods of mitigation.

In response to these vulnerabilities, a number of different efforts have been made to increase Wi-Fi and BLE security and privacy. For Wi-Fi, this includes periodically changing MAC addresses, randomizing FSize, and encrypting lower-layer data packets. M. Gruteser and D. Grunwald provide a framework to change MAC addresses frequently while still maintaining wireless connectivity [18]. The technique of chaffing and winnowing, as introduced by R. Rivest, can be adapted in smart home technologies to send secure packets intertwined with fake packets of random size to make events impossible to identify [19]. In BLE, devices need to make their advertisements private. Fawaz et al. designed an authentication system, BLE-Guardian, that restricts who can discover, scan, and connect to BLE devices [11]. BLE-Guardian uses jamming techniques to hide advertisements from unauthorized users. It is limited, however, to protecting devices prior to a connection and does not hide packets that are transmitted after a connection is created. As privacy data is still leaked during a connection, much of the reconnaissance information mentioned above can still be collected by an attacker. With BLE-Multi, Gutierrez et al. developed an enhancement to the Ubertooth One BLE scanner that enables sniffing of multiple connection simultaneously [10]. However, the scanner is limited to tracking three connections simultaneously and only saw an eighty-five percent probability of successful packet capture.

1.6 Conclusion

This chapter presents a brief technical summary of the Wi-Fi and BLE protocols and how their security features relate to those of other comparable wireless protocols. It provides background on key smart home technologies and open-source tools as they pertain to this work. It observes related research into the development of automated smart home architecture, how BLE and Wi-Fi properties leave them open to privacy leakage, and current efforts in securing IoT. While research has been done in the realm of Wi-Fi and BLE privacy leakage, little work has provided a broad review of privacy leakage from smart home devices in the wild or methods to secure smart homes from data leakage. This thesis contributes to the field of IoT security, specifically privacy within a smart home, by illustrating how devices leak data and demonstrating how users can prevent leakage through mitigation techniques.

Bibliography

1. *Wireless LAN Medium Access Control, MAC, and Physical Layer, PHY, Specification.*

2. Tcpdump manual page, 2017. [Last accessed on December 8, 2017], Available at *tcpdump.org/tcpdump_man.html.*

3. J. Guiterrez Del Arroyo and B. Ramsey. Securing bluetooth low energy enabled inudstrial monitors. *Proceedings of the Twelfth International Conference on Cyber Warfare and Security*, 2017.

4. J.S Atkinson et al. Your wifi is leaking: What do your mobile apps gossip about you? *Future Generation Computer Systems*, 2016.

5. C. Badenhop, J. Fuller, J. Hall, B. Ramsey, and M. Rice. Evaluating itu-t g.9959 based wireless systems used in critical infrastructure assets. In *International Conference on Critical Infrastructure Protection.* Springer International Publishing, 2015.

6. Bluetooth SIG. *Specification of the Bluetooth System Core Version 4.0*, 2010. *https://www.bluetooth.com/specifications/adopted-specifications/legacy-specifications.*

7. Bluetooth SIG. *Specification of the Bluetooth System Core Version 4.2*, 2010. [Last accessed on August 30, 2017], *www.bluetooth.com/specifications/bluetooth-core-specification.*

8. B. Bonne, A. Barzan, P. Quax, and W. Lamotte. Wifipi: Involuntary tracking of visitors at mass events. *2013 IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks*, 2013.

9. A. Das, P. Pathak, C. Chuah, and P. Mohapatra. Uncovering privacy leakage in ble network traffic of wearable fitness trackers.

10. J. Gutierrez del Arroyo, J. Bindewald, S. Graham, , and M. Rice. Enabling bluetooth low energy auditing through synchronized tracking of multiple connections. *Proceedings of the Eleventh Annual IFIP WG 11.10 Conference on Critical Infrastructure Protection*, 2017.

11. K. Fawaz and K. Kim. Protecting privacy of ble device users this paper is included in the proceedings of the protecting privacy of ble device users. *25th USENIX Security Symposium*, 2016.

12. B. Greenstein, R. Gummadi, J. Pang, M. Chen, T. Kohno, S. Seshan, and D. Wetherall. Can ferris bueller still have his day off? protecting privacy in the wireless era. *Proceedings of 11th USENIX workshop on Hot Topics in Operating Systems*, pages 1–6, 2007.

13. R. Heydon. *Bluetooth Low Energy: The Developer's Handbook.* Pearson Education, Inc., Crawfordsville, Indiana, 2013.

14. S. Jourdois. Btlejuice: Bluetooth smart (le) man-in-themiddle framework, 2016. Available at *//github.com/DigitalSecurity/btlejuice.*

15. KimiNewt. pyshark, 2017. Available at *https://github.com/KimiNewt/pyshark.*

16. D. Konings, A. Budel, F. Alam, and F. Noble. Entity tracking within a zigbee based smart home. *Proceedings of 23rd International Conference on Mechatronics and Machine Vision in Practice*, 2016.

17. J. Kurose and K. Ross. *Computer Networking A Top-Down Approach Featuring the Internet.* Pearson Education, Inc., Crawfordsville, Indiana, 2017.

18. K. Xu X. Yu X. Hong M. Zhou, Z. Tian and H. Wu. Scanme: Location tracking system in large-scale campus wi-fi environment using unlabeled mobility map. *Expert Systems with Applications*, 41(7):3429–3443, 2014.

19. C. Madrigal.

20. M. Ossman and D. Spill. Project ubertooth: an open source 2.4 ghz wireless development platform suitable for bluetooth experimentation, 2014. Available at *ubertooth.sourceforge.net.*

21. BlueZ Project. Bluez: official linux bluetooth protocol stack, 2016. Available at *bluez.org.*

22. A. Rose, J. Gutierrez del Arroyo, and B. Ramsey. Bluefinder: A range-finding tool for bluetooth classic and low energy. *Proceedings of the Twelfth International Conference on Cyber Warfare and Security*, 2017.

23. A. Rose and B. Ramsey. Picking bluetooth low energy locks from a quarter mile away. 2016. presented at DEF CON 24 *(media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/).*

24. H. Sharma and S. Sharma. A review of sensor networks: Technologies and applications. *Recent Advances in Engineering and Computational Sciences*, pages 6–8, 2014.

25. E. Skoudis.

26. E. Skoudis and T. Liston. *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses.* Pearson Education, Inc., Upper Saddle River, NJ, 2006.

27. J. Slawomir. A node.js package for ble using man-in-the-middle and other attacks, 2016. Available at *//github.com/securing/gattacker.*

28. G. Valadon and P. Lalet. Scapy: the python-based interactive packet manipulation program and library, 2016. Available at *www.secdev.org/projects/scapy*.

29. M. Versichele, L. de Groote, M. Claeys Bouuaert, T. Neutens, I. Moerman, and N. Van de Weghe. Pattern mining in tourist attraction visits through association rule learning on bluetooth tracking data: A case study of ghent, belgium. *Tourism Management*, 44:67–81, 2014.