



**SECURITY EVALUATION AND
EXPLOITATION OF
BLUETOOTH LOW ENERGY DEVICES**

THESIS

Anthony J. Rose, Captain, USAF
AFIT-ENG-MS-17-M-066

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-MS-17-M-066

SECURITY EVALUATION AND EXPLOITATION OF
BLUETOOTH LOW ENERGY DEVICES

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Electrical Engineering

Anthony J. Rose, B.S.E.E.

Captain, USAF

March 2017

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-17-M-066

SECURITY EVALUATION AND EXPLOITATION OF
BLUETOOTH LOW ENERGY DEVICES
THESIS

Anthony J. Rose, B.S.E.E.
Captain, USAF

Committee Membership:

Major Jason M. Bindewald, Ph.D.
Chair

Lieutenant Colonel Mason J. Rice, Ph.D.
Member

Barry E. Mullins, Ph.D., P.E.
Member

Abstract

The evaluation and defense of wireless systems is essential to meeting the United States' cyberspace mission, especially as the Department of Defense adopts a greater number of Internet of Things (IoT) devices. One of the most prevalent IoT technologies for short-range wireless communications is Bluetooth, and its widespread commercial use in access control makes it a primary target for cybersecurity experts. By evaluating Bluetooth wireless security and developing exploits and mitigation solutions, this thesis proves that most Bluetooth access control devices are vulnerable to exploitation, but can be protected. This is demonstrated through four contributions. First, a benchmark open-source range-finding tool determines the location of Bluetooth devices through a novel distance estimation method, increasing the state-of-the-art device location distance from 50 meters up to 1,000 meters. Second, evaluation of 17 Bluetooth Low Energy devices reveals 75% of the tested devices contain at least one vulnerability, resulting in unauthorized access. Thirdly, user behavior analytics demonstrate how malicious actors can exploit vulnerabilities for unauthorized device access and obtain sensitive information. Finally, five proposed mitigation solutions help eliminate many current wireless threats from nearly all Bluetooth devices, with specific solutions passed along to 10 vendors to address access control devices currently in commercial use. This research has shifted the perceived security of Bluetooth access control systems and has prompted additional research in the field, following publication at the International Conference for Critical Infrastructure Protection, International Conference for Cyber Warfare and Security, RSA Conference, and DEF CON.

Acknowledgements

I would first like to thank my family for all of their support during my academics. Next, I wish to thank my thesis advisor, Major Jason Bindewald, for your encouragement and guidance through the thesis process. I would also like to thank Major Benjamin Ramsey for his guidance through my research and for the time he spent as my thesis advisor. Finally, I would like to thank my thesis committee members, Lieutenant Colonel Mason Rice and Dr. Barry Mullins for their feedback and support.

Anthony J. Rose

Contents

	Page
Abstract	iv
Acknowledgements	v
List of Figures	ix
List of Tables	xi
List of Acronyms	xii
I. Introduction	1
1.1 Overview and Background	1
1.2 Problem Statement	3
1.3 Research Goals and Hypothesis	3
1.4 Approach	5
1.5 Assumptions and Limitations	6
1.6 Research Contributions	6
1.7 Thesis Overview	7
II. Background and Related Research	8
2.1 Overview	8
2.1.1 Bluetooth Classic	8
2.1.2 Bluetooth Low Energy	9
2.1.3 Current Bluetooth Tools	13
2.2 BLE Attack Vectors	17
2.2.1 Man-in-the-Middle Attacks	18
2.2.2 Encrypted Communication Attacks	18
2.2.3 Malware	19
2.2.4 Range-finding on other Protocols	19
2.2.5 Insteon	20
2.3 Conclusion	20
III. BlueFinder: A Range-finding Tool for Bluetooth Classic and Low Energy	21
3.1 Introduction	21
3.2 Objectives	22
3.3 Scenario	23
3.4 Assumptions	23
3.4.1 Response Variables	24
3.4.2 Controlled Variables	25

	Page
3.4.3 Constant Variables	26
3.4.4 Data Collection and Reduction	27
3.5 System Design	28
3.5.1 Hardware	28
3.5.2 Software	29
3.5.3 Python and Scapy	29
3.5.4 Packet Assembler	31
3.5.5 Packet Parser	31
3.6 Design Approach	31
3.6.1 Log-distance path loss model	32
3.6.2 Finding the reference RSS	32
3.6.3 Finding best-fit path loss exponent	33
3.7 Results and Analysis	35
3.7.1 Measured RSS vs. model-derived RSS	36
3.7.2 BlueFinder Tool	39
3.7.3 BlueFinder vs. Blue Hydra	40
3.8 Summary	42
IV. Securing BLE-enabled Locks against Unauthorized Access and Surveillance	44
4.1 Introduction	44
4.2 User Behavioral Analytics	44
4.3 Bluetooth Security Vulnerabilities	45
4.3.1 Plaintext Passwords	46
4.3.2 Password Obfuscation	47
4.3.3 Brute Forcing	48
4.3.4 Command Fuzzing	49
4.3.5 Hard-Coded Passwords	50
4.3.6 Man-in-the-Middle Attack	51
4.4 Scenario	55
4.5 Mitigation Techniques	58
4.5.1 Pairing and Bonding	58
4.5.2 Application Layer Encryption	63
4.5.3 Two-way Authentication	63
4.5.4 Geofencing	64
4.5.5 BLE-Guardian	64
4.6 Summary	66
V. Conclusion and Recommendations	67
5.1 Research Conclusion	67

	Page
5.1.1 Goal 1: Develop methods for detecting the location of Bluetooth Low Energy (BLE) and Bluetooth Classic (BTC) devices	67
5.1.2 Goal 2: Identify vulnerabilities in BLE devices and develop exploits that result in physical effects	67
5.1.3 Goal 3: Utilize vulnerabilities for unauthorized BLE device access and develop methods for extracting user data and generating User Behavior Analytics (UBA)	68
5.1.4 Goal 4: Propose mitigation solutions for the determined BLE vulnerabilities	68
5.2 Current State	68
5.3 Research Contributions	69
5.4 Recommendations for Future Work	69
5.5 Concluding Thoughts	71
Appendix A. Packet Assembler	72
Appendix B. BlueFinder	77
Bibliography	81

List of Figures

Figure	Page
1	Connection process between a master and slave device using BLE. 10
2	Established communication between client and server devices using BLE..... 10
3	BLE stack hierarchy for application, host, and controller. 11
4	BLE Advertising Report represented using the Host Controller Interface (HCI)..... 12
5	The Ubertooth One is a BLE and Bluetooth Classic (BTC) sniffer used throughout this thesis. 17
6	Hardware system diagram for Bluetooth range-finding using a laptop, Sena UD-100, and 15 dB antenna. 28
7	The Sena UD-100 Class 1 Bluetooth adapter with optional detachable antenna used for range-finding experiments and long-range Bluetooth communication. 29
8	Software design for parsing and assembling packets for determining distance estimation. 30
9	Target BLE and BTC devices used for Received Signal Strength (RSS) collection experiment. 34
10	Measured RSS at distances from 1 to up to 1,000 meters vs. model-derived RSS estimates. 38
11	List of currently-available BlueFinder options within the tool. 39
12	Using BlueFinder's distance estimation model to determine location of a BTC device..... 40
13	Comparison of Blue Hydra and BlueFinder distance estimation modeling using RSS. 41
14	Reverse-engineered Safetech command structure at the HCI level. 47

Figure		Page
15	Reverse-engineered Okidokeys command structure (in bytes) at HCI level and fuzzed packet.	50
16	A hard-coded password is found on the on Danalock by decompiling its Android Application Package (APK).	51
17	Sequence diagram of a rogue device attack on Mesh Motion Bitlock.	53
18	Sequence diagram of a relay attack on Mesh Motion Bitlock as described in Section 4.3.6.2.	54
19	Activity heat maps of (a) historic weekday behavior compared against time of day, (b) historic user behavior compared against weekday, and (c) historic user behavior compared against time of day.	57
20	The BLE version 4.1 long-term key generation as described in Section 6.1.1.	61
21	The BLE version 4.2 long-term key generation improves on version 4.1 shown in Figure 20.	62
22	Sequence diagram of BLE-Guardian defending a target from an attacking device [1].	65

List of Tables

Table		Page
2	Overview of Bluetooth Tools included in Kali Linux	14
3	Overview of Bluetooth Tools included in Kali Linux	15
4	Overview of Downloadable Bluetooth Tools	16
5	Response Variable Summary	25
6	Controlled Variable Summary	26
7	Constant Variable Summary	27
8	Mean Absolute Percentage Error (MAPE) for a range of P for LOS models.	36
9	MAPE for a range of P for BLOS models.	37
10	List of successfully exploited BLE devices by type.	46
11	Time expected to brute force a password by characters and length.	48
12	List of mitigation techniques and their employment difficulty used to solve BLE vulnerabilities.	60

List of Acronyms

AES	Advanced Encryption Standard.
APK	Android Application Package.
BD_ADDR	Bluetooth Device Address.
BLE	Bluetooth Low Energy.
BR	Basic Rate.
BTC	Bluetooth Classic.
DoD	Department of Defense.
DoS	Denial of Service.
ECDH	Elliptic Curve Diffie-Hellman.
EDR	Enhanced Data Rate.
EKG	electrocardiogram.
GATT	Generic Attribute Profile.
HCI	Host Controller Interface.
ICS	Industrial Control Systems.
IDS	Intrusion Detection Systems.
IoT	Internet of Things.
LTK	long-term key.
MAPE	Mean Absolute Percentage Error.
MitM	Man-in-the-Middle.
NFC	Near Field Communication.
NIST	National Institute of Technology.
PAN	Personal Area Network.
PKI	Public Key Infrastructure.
RFID	Radio Frequency Identification.

RSS	Received Signal Strength.
SDR	Software-Defined Radio.
SIG	Special Interest Group.
SSP	Secure Simple Pairing.
STK	short-term key.
TK	temporary key.
UBA	User Behavior Analytics.
UUID	Universally Unique Identifier.
WLAN	Wireless Local Area Network.

SECURITY EVALUATION AND EXPLOITATION OF BLUETOOTH LOW ENERGY DEVICES

I. Introduction

1.1 Overview and Background

Bluetooth Low Energy (BLE), also marketed as Bluetooth Smart, is a wireless protocol designed for interconnecting the Internet of Things (IoT). IoT appliances are an expanding market that includes linkages from home automation systems to Industrial Control Systems (ICS) and is expected to grow to more than \$19 trillion by 2020 [2]. BLE devices are expected to increase to more than one-third of all IoT devices with nearly 8.2 billion already in use worldwide [3]. IoT systems are becoming increasingly more intertwined with water, power, emergency services, health care, agriculture, transportation, and security systems [4]. Health care and security systems are two of the biggest adopters of BLE and are the focus of this thesis.

Manufacturer implementation of BLE focuses on usability over security, leaving many security concerns in a variety of commercial products. An example of a major security risk facing BLE devices is the use of signal strength to track wireless devices [5]. Tracking devices by signal strength has both a defensive purpose (e.g., detecting malicious devices) and offensive purpose (e.g., target tracking). The use of signal strength to track BLE devices can assist an adversary in implementing an attack. Vulnerabilities in BLE systems are widespread and allow for unauthorized access and exploitation against U.S. critical infrastructure.

Current estimates predict that a cyber attack on U.S. critical infrastructure could

cost more than \$700 billion [6]. Many facilities are implementing security systems and access control through IoT devices. Access control relies on physical security to manage admittance to sensitive locations. Typical implementations of access control include numerical access (personal identification number), Radio Frequency Identification (RFID), Public Key Infrastructure (PKI), and biometrics [7]. These solutions limit organizational ability to control credentials by not allowing access. In some cases, access revocation may be impossible or expensive without completely removing credentials for all users. An appeal of BLE or other wireless systems is that access can be granted at any time and centrally managed. This requires authentication between the user and organizational database by eliminating the need to manage devices independently. Implementation of new methods and technologies for controlling access to secure facilities is already occurring. New technologies that incorporate wireless security increase convenience, but may also open the door for cyber attacks. Companies are already releasing security systems using BLE locks for granting access to server rooms, power plants, water treatment facilities, manufacturing plants, and ATMs. For example, Onity [8] offers automation, manufacturing, and security products and has produced more than one million Bluetooth locking systems in 115 countries.

This research addresses the Department of Defense (DoD) Cyber Strategy goal [9] of establishing partnerships with private industry. The DoD has stated that it will work closely with private industry to validate and commercialize new innovations for cyber security, due to the DoD relying heavily on the expertise of the private sector. This research focuses on BLE access control systems, such as devices that are sold to DoD entities for securing facilities. Evaluating these systems is important for meeting United States strategic goals for the cyberspace mission, specifically, defending the DoD information network, securing DoD data, and mitigating risk to DoD missions

[9].

1.2 Problem Statement

This thesis addresses two problems. The first problem addressed is that a distance estimation model needs to be developed to accurately determine locations of Bluetooth devices. Locating Bluetooth devices is important in preventing attacks and proving potential cyber vulnerabilities. For example, rogue devices can easily eavesdrop conversations and steal important information without detection. Determining the location of malicious devices is critical in securing data.

The second problem addressed is determining if BLE devices are protecting users from threats. Threats are instances where malicious actors use vulnerabilities to gain access to devices and expose user information. Determining the usability of the user data is essential for evaluating adversary capabilities with behavioral analytics.

1.3 Research Goals and Hypothesis

The proliferation of BLE devices poses a major security risk to wireless systems. This thesis answers the following question. Can BLE access control devices be exploited and/or protected? The hypothesis is that BLE access control devices can be exploited and protected by evaluating their wireless security and developing exploits and mitigation solutions. In order to achieve this, two prerequisites and four goals need to be met to prove or disprove the hypothesis.

Prerequisites:

1. Gain fundamental knowledge on Bluetooth Low Energy (BLE) and Bluetooth Classic (BTC) wireless protocols.
2. Find and develop tools for sniffing and injecting BLE and BTC traffic.

Goals:

1. **Develop methods for detecting the location of BLE and BTC devices.**

The development of a range-finding tool is essential to locating Bluetooth devices by their Received Signal Strength (RSS).

2. **Identify vulnerabilities in BLE devices and develop exploits that result in physical effects.**

In addition, evaluate BLE devices and provide a comparison of their security implementations, specifically, this research focuses on assessing access control devices (e.g., deadbolts, padlocks, and safes) and evaluates numerous devices to determine the existence of security vulnerabilities and threats.

3. **Utilize vulnerabilities for unauthorized BLE device access and develop methods for extracting user data and generating User Behavior Analytics (UBA).**

The evaluation of potential for the extraction of data pertaining to users and its development of User Behavior Analytics (UBA). User behavior analytics is the ability to determine anomalous behavior through the evaluation of system logs containing user activity. The objective of using UBA is to identify if it has a potential offensive purpose.

4. **Propose mitigation solutions for the determined BLE vulnerabilities.**

The defense of BLE devices are ranked according to the exploits that the devices are protected against and the difficulty of implementation and maintenance.

Achieving these goals will validate BLE device security and their potential for exploitation and protection.

1.4 Approach

Prior research has been conducted extensively on Bluetooth with a majority of the focus on Bluetooth Basic Rate (BR) and Enhanced Data Rate (EDR) which are categorized as Bluetooth Classic (BTC) [10] [11]. This research focuses on the newest iteration of Bluetooth, also known as Bluetooth Low Energy (BLE), Bluetooth Smart, or Bluetooth 4.0. BLE diverges from BTC by implementing a unique method of communication not previously used. An survey of previous Bluetooth research determined the lack of research dedicated to finding BLE vulnerabilities and a shortage of open-source tools compatible with BLE. A major focus throughout this thesis is developing methods for building BLE exploits for plaintext passwords, obfuscated passwords, brute forcing, command fuzzing, hard-coded passwords, and Man-in-the-Middle (MitM) attacks. As well, the avoidance and prevention of attacks on BLE devices is proposed through mitigation techniques.

The experiments use a combination of hardware and software. The hardware used for monitoring wireless traffic in this thesis is the Ubertooth One, a specialized Software-Defined Radio (SDR) designed to collect Bluetooth traffic. In addition, an array of BLE devices (e.g., deadbolts, pad locks, keyboards, mice, medical devices, and pressure sensors) are required for an extensive security evaluation.

The results reveal that BTC and BLE devices can be successfully located from their RSS. The prediction accuracy, represented as Mean Absolute Percentage Error (MAPE), proves that the distance estimation model used to locate Bluetooth devices is successful. Additional evaluation proves that the tool developed in this thesis to locate Bluetooth devices, BlueFinder, outperformed an existing location detection platform, Blue Hydra [12]. Next, eight BLE vulnerabilities are identified through the evaluation of 17 BLE locks. This thesis successfully develops UBA through the use of discovered BLE vulnerabilities and proposes five mitigation techniques used to

eliminate wireless threats found in Chapter 4.

1.5 Assumptions and Limitations

The determination of the vulnerability of BLE devices to cyber attacks is the focus of this thesis. Extensive research has already been conducted for BTC vulnerabilities, prompting the focus to only be on BLE attacks and defenses. Other attack vectors (e.g. Wi-Fi) for exploiting Bluetooth devices are out of scope for this research.

1.6 Research Contributions

The contributions of this research leads directly to the understanding of vulnerabilities and their uses to exploit BLE access control devices by exposing potential cyber security threats. The first contribution is the development of a long-range Bluetooth range-finding tool, BlueFinder, published at the 12th International Conference on Cyber Warfare and Security Conference (ICCWS) [13]. This work develops the log-distance path loss model for BlueFinder, validates its accuracy against previous models, and proposes to improve other range-finding tools, such as the warwalking platform Blue Hydra [12].

The evaluation of 17 BLE access control devices led to the development of eight exploits. These exploits were developed to expose BLE vulnerabilities and increase manufacturer and consumer knowledge. Five defensive measures were proposed to prevent these exploits or mitigate their effects. The identified vulnerabilities and mitigation solutions were published in the International Journal of Critical Infrastructure Protection (submitted) [14], DEF CON 24 [15], IoT Village [16], and RSA Conference (to appear) [17]. The information presented to consumers and manufactures has led directly to more secure BLE devices.

1.7 Thesis Overview

The remainder of this thesis is organized as follows. Chapter 2 provides BLE and BTC background information and outlines fundamental protocol principles. In addition, this chapter discusses related research and current Bluetooth penetration testing tools. Chapter 3 outlines the experiment for developing a distance estimation tool using the log-distance path loss model. This chapter discusses the construction of the communication system that allows the user to interact with Bluetooth commands. Chapter 3 also constructs and demonstrates the range-finding tool, BlueFinder, and validates its accuracy against a similar tool, Blue Hydra. Chapter 4 describes vulnerabilities in BLE access control devices, such as plaintext passwords, password obfuscation, brute forcing, command fuzzing, hard-coded passwords, and MitM attacks. This chapter proposes mitigation techniques, such as pairing and bonding, application level encryption, geofencing, and a third party tool BLE-Guardian, to eliminate existing threats. Chapter 5 concludes the thesis with contributions, future work, and final thoughts.

II. Background and Related Research

This chapter addresses the first two prerequisites required to answer if BLE access control devices can be exploited and/or protected. The two requirements addressed are (1) gain fundamental knowledge on Bluetooth Low Energy (BLE) and Bluetooth Classic (BTC) wireless protocols and (2) find and develop tools for sniffing and injecting BLE and BTC traffic. To accomplish this, this chapter outlines differences between BLE and BTC wireless protocols, surveys open-source Bluetooth tools, and discusses related research.

2.1 Overview

Bluetooth builds upon the IEEE 802.15.1 wireless personal area network standard [18]. Currently, commercial Bluetooth devices implement two distinct classes of operation: Bluetooth Classic (BTC) and Bluetooth Low Energy (BLE). Although these protocols share some key implementation details (e.g., operate in the 2.4 GHz band and implement adaptive frequency hopping), they are designed for different types of applications, outlined below.

2.1.1 Bluetooth Classic.

The BTC moniker refers to any Bluetooth device that does not utilize the BLE link layer. This classification encompasses devices supporting Bluetooth versions 1.0 to 4.0, including Enhanced Data Rate (EDR) mode. EDR offers short-range wireless transmissions data rate up to 3 Mbps. At these rates, Bluetooth supports applications like audio streaming, computer networking, or large file transfers in a Personal Area Network (PAN) [19].

BTC devices are identified by their 6-byte Bluetooth Device Address (BD_ADDR)

and perform up to 1,600 frequency hops every second across 79 channels. The hopping scheme and clock are negotiated during the connection process and controlled for the duration of the connection by the requesting device [19].

2.1.2 Bluetooth Low Energy.

While BTC focuses on sending maximum data without regard to power consumption, supporting devices such as music streaming and data storage. Conversely, power saving is a top priority for BLE devices. BLE offers an interface for low data rate Internet of Things (IoT) devices, such as temperature monitors and door locks [20]. BLE [21] is designed to provide a secure and robust wireless communication mechanism using minimal energy at data rates up to 1 Mbps. Its specifications are merged and published with existing communication modes in the Bluetooth 4.0 and above standard [22]. BLE employs a 40-channel frequency hopping scheme, which requires transceiver configurations that are not directly compatible with BTC modes. However, newer transceiver designs support both protocols.

The BLE connection process differs from BTC by minimizing expended energy. Devices advertise themselves on three specific channels that are dispersed across the 2.4 GHz band to avoid interference from IEEE 802.11 Wireless Local Area Networks (WLANs) [11]. A user connects to the device on the advertising channels to initiate the connection. BLE operates with a master and slave model where the master is typically the user (e.g. phone, tablet) and the slave is the device awaiting a connection (e.g., lock, heart rate monitor, thermostat). An example of the master and slave model for Bluetooth is seen in Figure 1. BLE devices are split into two categories depending on function: client and server. The client is the master in most cases, while the slave acts as the server. Common operations utilized in BLE communication include: read, write, notify, and indicate commands. Using these commands pushes or pulls data

between the client and server through the Generic Attribute Profile (GATT). Figure 2 illustrates a client and server interacting and using the read command.

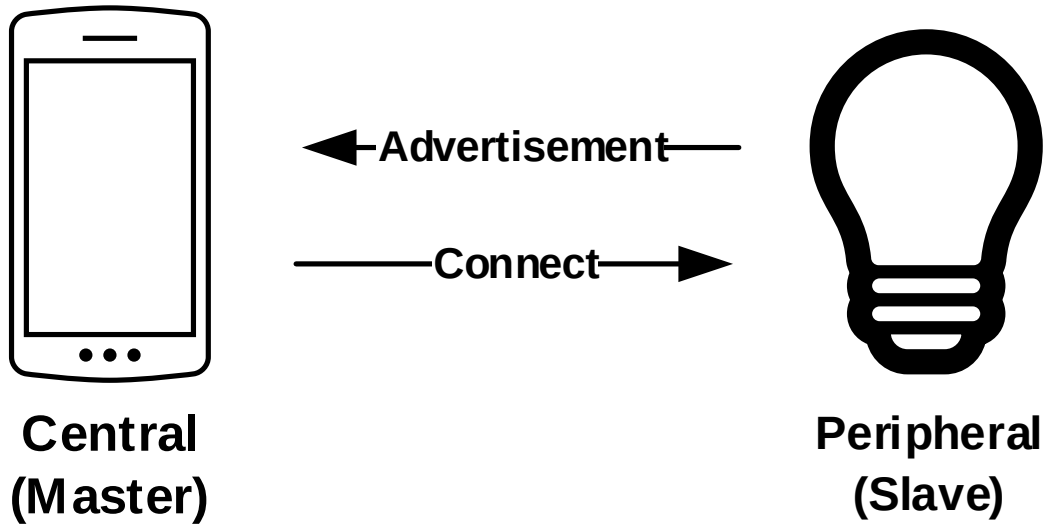


Figure 1. Connection process between a master and slave device using BLE.

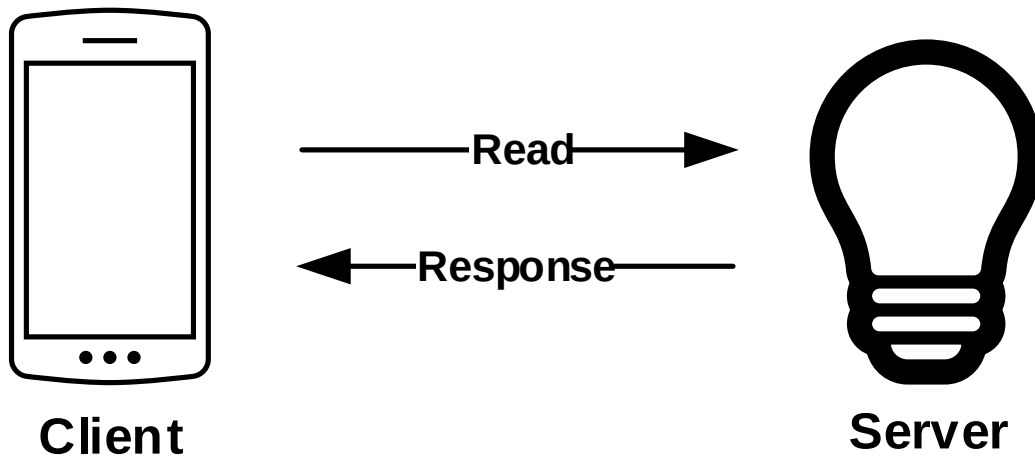


Figure 2. Established communication between client and server devices using BLE.

2.1.2.1 GATT.

The GATT, see Figure 3, is built as a hierarchy where the profile is at the top level and is composed of a series of services. Services are collections of characteristics that represent the behavior of a device. For example, a service could be listed as blood pressure monitor or heart rate monitor for medical devices or temperature readings for thermostats. Characteristics can fall into a few different categories underneath a service. They contain a Universally Unique Identifier (UUID), value, properties (read, write, notify, and indicate), and permissions. A UUID is a 16-bit or 128-bit identifier used by a manufacturer to specify custom services, however, some UUIDs are used universally across manufacturers. Finally, descriptors fall under characteristics and contain configuration flags and any metadata that a manufacturer may want to share.

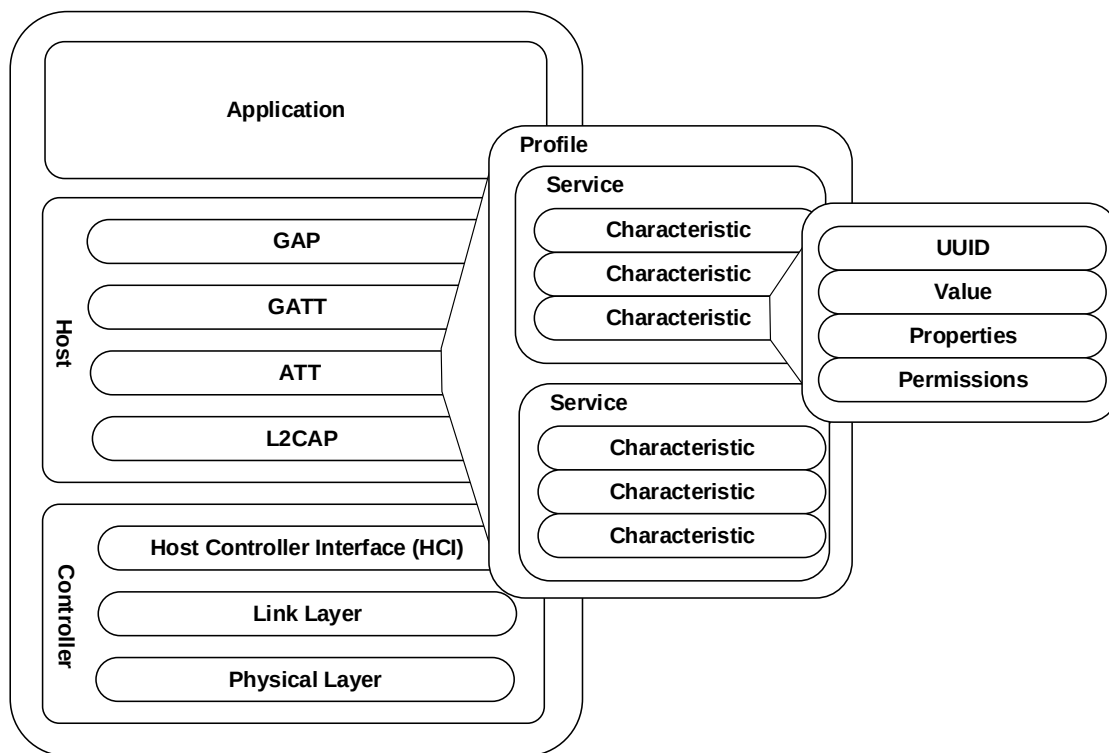


Figure 3. BLE stack hierarchy for application, host, and controller.

2.1.2.2 Host Controller Interface (HCI).

The Host Controller Interface (HCI) is the communication protocol used to communicate between the host and the Bluetooth controller. It is a uniform command interface to the controller. The HCI is responsible for sending commands and receiving data to/from the controller. This interface exists to allow the host to be easily changed without affecting controller operation. Figure 4 illustrates the structure of a typical command sent across the HCI. In this example, a user would be sending a command to listen for advertisement and the advertisement report (below) would be the response.

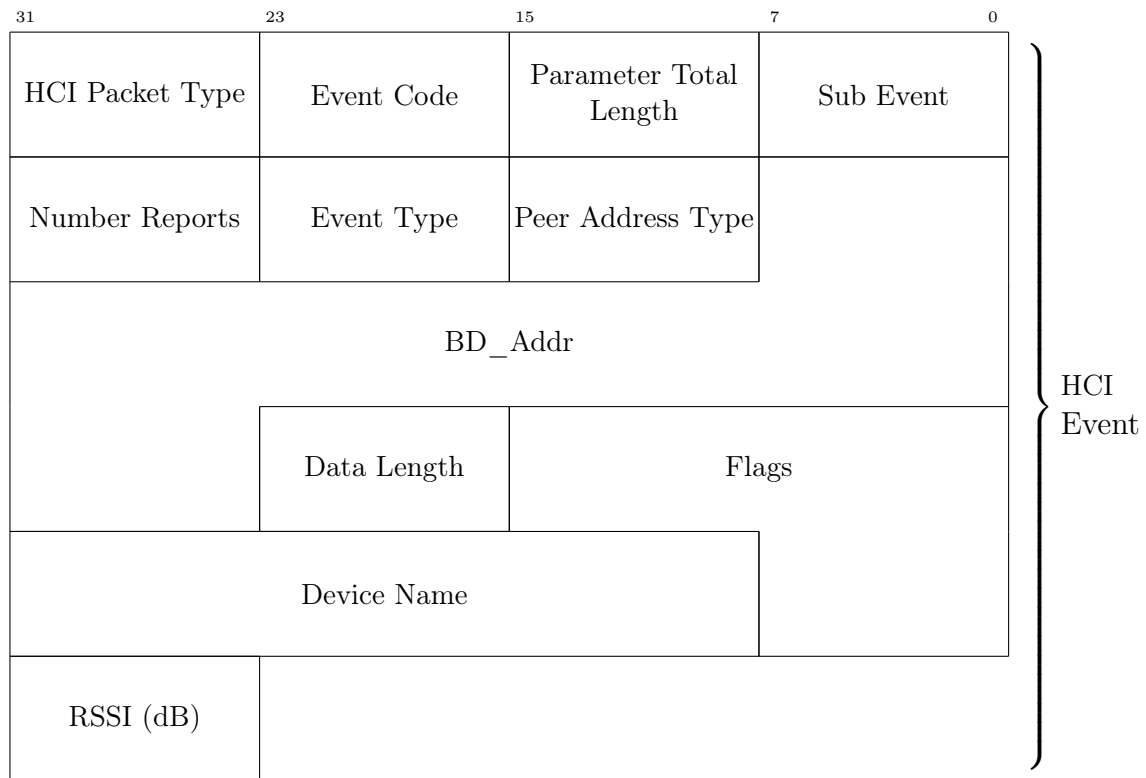


Figure 4. BLE Advertising Report represented using the HCI.

2.1.3 Current Bluetooth Tools.

This section presents a survey of currently available Bluetooth tools. Kali Linux is a popular open-source operating system for computer security professionals that is preloaded with a suite of over 600 penetration testing tools [23]. Of the evaluated Bluetooth tools, six are included in Kali 2016.2, two are included in older distributions of Kali, and five are available for download. Tables 1 and 2 provide an overview of the evaluated Bluetooth tools.

A summary of each tool is provided in the following subsections. No accurate range-finding tool currently exists in either Kali or open-source repositories. A ranging tool is a program that uses information from a transmitter (RSS, response time, etc.) to determine its location. A Bluetooth ranging tool present in Kali (Table 2 and 3) is BlueRanger; however, the tool does not provide quantified estimations of distance and it does not work for BLE devices. Of the non-Kali tools, Blue Hydra (Table 4) offers a distance estimation utilizing the RSS of a device and does not offer a long-range capability. Thus it would require a new distance estimation model and utilizes a reference RSS that the device provides. It was found that the advertised transmit power of a device is inaccurate with deviations of 5 dBm from the true reference RSS value. As a result, there is motivation to design and implement a tool that provides distance estimations of remote Bluetooth devices.

Table 2. Overview of Bluetooth Tools included in Kali Linux

Tool	Last Availability	BTC	BLE	Description
Bluelog	Kali 2016.2	Yes	No	Bluelog is a Bluetooth survey tool that transmits inquiries to discoverable devices to read their Bluetooth device address, device name, manufacturer, and device class. Bluelog is designed to be run at a static location for extended periods of time [23].
BTscanner	Kali 2016.1	Yes	No	BTscanner uses an inquiry request to determine information from the Host Controller Interface (HCI) and Service Discovery Protocol (SDP). A list of IEEE organizationally unique identifier tables aid in the assessment of device type. Scanning a device reveals its name, clock offset, class, any services running, and the manufacturer.
Redfang	Kali 2016.1	Yes	No	Redfang is a searching tool that can find hidden Bluetooth devices. This is accomplished by using a list of manufacturer codes for the first three bytes of the Bluetooth device address and an exhaustive iteration through the last three bytes of the address.
Spooftooph	Kali 2016.1	Yes	No	Spooftooph enables Bluetooth device spoofing, cloning, and can scan for devices with the potential to be cloned.

Table 3. Overview of Bluetooth Tools included in Kali Linux

Tool	Last Availability	BTC	BLE	Description
BlueRanger	Kali 2016.1	Yes	No	BlueRanger is a tool that estimates device movement by comparing its current link quality with a previously observed value and displays one of three conditions: hotter, colder, or neutral.
Bluesnarfer	Kali 2016.1	Yes	No	Bluesnarfer is a tool used to steal information from Bluetooth devices by exploiting security flaws in anonymous data requests. This tool is used to download mobile contact information and gain access to other stored data on mobile devices.
BlueMaho	Kali 2.0	Yes	No	BlueMaho is a tool for testing Bluetooth devices for security vulnerabilities. A database of exploits is stored within BlueMaho to gain access to Bluetooth devices.
Bluepot	Kali 1.1.0	Yes	No	Bluepot is a Bluetooth honeypot designed to accept and collect malware from devices. The most common attacks in this tool are for data exfiltration (e.g., BlueBugging and BlueSnarfing).

Table 4. Overview of Downloadable Bluetooth Tools

Tool	Last Availability	BTC	BLE	Description
Ubertooth	Download	Yes	No	Ubertooth is a wireless development platform capable of passively sniffing BLE and BTC connections. Ubertooth requires a specific hardware receiver, so it is not compatible with commodity USB adapters. Figure 5 illustrates an Ubertooth One produced by Great Scott Gadgets.
Crackle	Download	No	Yes	Crackle exploits a flaw in the BLE pairing process that allows the brute forcing of a temporary key. The attacker can break the long term key by using the temporary key and information from the pairing process. Decryption of all communication between devices is possible once the attacker has the long term key. This tool requires the collection of the pairing process and an Ubertooth.
BTLEJuice	Download	No	Yes	BTLEJuice is a man-in-the-middle attack framework that includes an interactive interface.
Gattacker	Download	No	Yes	Gattacker is an application used for a man-in-the-middle attack.
Blue Hydra	Download	Yes	Yes	Blue Hydra is a Bluetooth device discovery service built on top of the Bluez library [12]. Blue Hydra makes use of Ubertooth where available and attempts to track both BTC and BLE devices over time. It provides the user with device name, Bluetooth version, RSS, manufacturer, and estimated device distance.

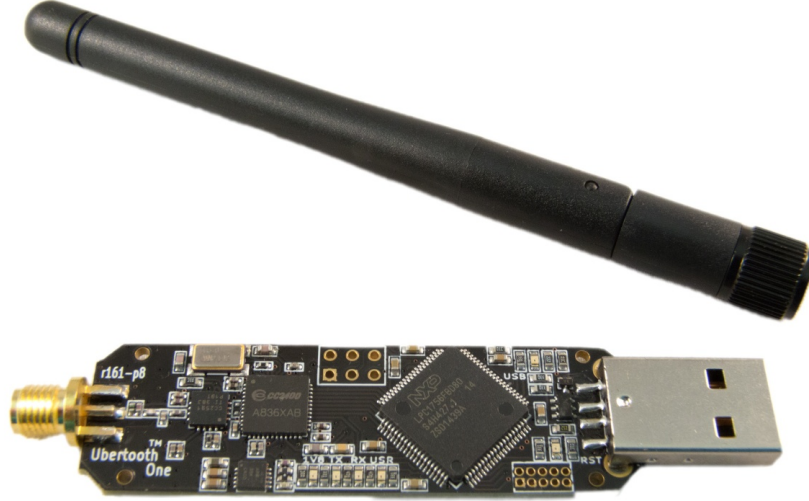


Figure 5. The Ubertooth One is a BLE and BTC sniffer used throughout this thesis.

2.2 BLE Attack Vectors

Currently few attacks have been published for BLE, with most wireless attacks directed towards BTC. However, wireless attacks developed for other protocols (e.g., Wi-Fi, Zigbee, Z-wave) can easily be adapted for BLE. Examples of attacks that are easily adapted for Bluetooth are replay attacks, device spoofing, and command fuzzing. A replay attack is when an attacker records a series of commands and later re-transmits them to a target device to gain unauthorized access [24]. Device spoofing is a serious problem that allows an attacker to hijack a connection. However, tools such as BlueID allow for Bluetooth devices to be fingerprinted by their clocks and

be used to identify the validity of a device [25]. Finally, an attacker uses command fuzzing to feed a device malformed data to find critical defects in its programming [26].

2.2.1 Man-in-the-Middle Attacks.

Haataja’s prior research explores BTC MitM attacks and countermeasures [27] [10]. His research focuses on two attacks named Bluetooth-No Input, No Output-Man-in-the-Middle Attack (BT-Nino-MITM) and Bluetooth-Secure Simple Pairing-Out-of-Band-Man-in-the-Middle Attack (BT-SSP-OOB-MITM) which uses BTC vulnerabilities associated with Bluetooth Secure Simple Pairing (SSP). In addition, two BLE MitM tools exist that provide a testing platform for penetration testing: BTLE-juice [28] and Gattacker [29]. These tools utilize a vulnerability on most BLE devices where they do not properly authenticate a conversation and use unencrypted communication.

2.2.2 Encrypted Communication Attacks.

Earlier versions of Bluetooth (4.1 and earlier) use a vulnerable encryption method which allows an attacker to gather all of the information required to generate the decryption key. Work by Ryan [11] establishes foundational BLE concepts for eavesdropping and decryption that are important for this thesis. Ryan’s work was the first to identify the weaknesses in the Bluetooth encryption process. He proposes solving eavesdropping encrypted conversation by improving the included Bluetooth encryption method by implementing an Elliptic Curve Diffie-Hellman (ECDH) key exchange. This suggestion led directly to improving the key exchange protocol in later Bluetooth releases. However, many manufacturers have yet to implement Bluetooth 4.2 into their devices which leaves them using an earlier, more vulnerable version.

2.2.3 Malware.

Malicious actors are using the lack of standard security on mobile devices to embed malware and access sensitive data and deny access to device features [30]. Intrusion Detection Systems (IDS) exist for mobile platforms, such as Android, with most being behavior-based. This allows for some attacks using malware to be detected. However, if the malware is embedded into another application the ability for the IDS to detect the compromised firmware would be decreased, especially if no signatures currently exist. An example of this type of attack is seen in Gutierrez’s work on compromising BLE temperature and pressure monitors by using over-the-air updates to upload malicious firmware [31] [32]. This work parallels the research in this thesis in developing new BLE exploits.

2.2.4 Range-finding on other Protocols.

Locating wireless devices is an important step for mitigating attacks. Research in the area of distance estimation for wireless devices has been conducted in numerous fields, with this thesis following the zbfnd model for Zigbee [5]. This work has shown a need for the development of a range-finding tool that can determine Bluetooth device location. Other distance estimation research has investigated the use of the log-normal shadowing propagation model to determine the location of a device [33] [34]. Early research into BTC device tracking uses the line-of-sight radio propagation model to determine devices up to 7 meters away with an error rate of nearly 1.2 meters [35]. This demonstrates that at this time there is no long-range model in place and motivates the development of a tool that can operate at longer distances.

Hall and Ramsey’s research in compromising Z-wave devices has laid the groundwork for many BLE attacks [36]. An important discovery from their work, demonstrates that accessing IoT devices that do not utilize proper authentication or encryp-

tion allows for an attacker to inflict physical damage, in this case destroying personal property.

2.2.5 Insteon.

Attacks on a similar wireless system used for home automation parallels attacks discussed later in Chapter 4 [37]. Insteon is a home automation protocol that uses lights, thermostats, and locks. Insteon utilizes unencrypted traffic with a central hub that is trusted by all devices within the network. An interesting feature of Insteon is that it uses other protocols in conjunction with their wireless protocol. Bluetooth and Wi-Fi allow communication between devices and adds functionality outside of their proprietary protocol. This adds additional functionality to the network, but also introduces more avenues for attack.

2.3 Conclusion

Extensive studies have been completed on Bluetooth and other wireless protocols (e.g., IEEE 802.11). However, little research has been published on exploiting BLE devices. This thesis expands on the current BLE research by investigating methods for detecting Bluetooth devices, discovering BLE vulnerabilities, enabling the development of User Behavior Analytics (UBA), and proposing countermeasures to defend BLE devices.

III. BlueFinder: A Range-finding Tool for Bluetooth Classic and Low Energy

3.1 Introduction

The development of a range-finding tool is essential to meeting the goals for this thesis. This chapter meets the first goal of (1) develop method for detecting the location of BLE and BTC devices. Meeting this goal is accomplished by adapting existing models (log-distance path loss model) to Bluetooth to calculate the distance of a device from its signal strength. This academic work is accepted to the International Conference on Cyber Warfare and Security [13] and the following sections are similar to the presented work and expands on it.

Bluetooth technology is designed for short-range wireless connectivity and is commonly found in devices such as smartphones, data loggers, and medical equipment. Over four billion Bluetooth enabled devices are expected to ship worldwide in 2016 [38]. Bluetooth-enabled medical devices are increasingly popular, particularly among electrocardiogram (EKG), pacemakers, and insulin pumps. Implantable medical devices are expected to grow at a rate of approximately 8% annually in the United States [39].

More than 2.5 million Americans rely on implantable medical devices to treat conditions such as cardiac arrhythmias and diabetes [40]. Many of these devices are low cost and use Bluetooth or other wireless protocols, leaving them susceptible to attacks that can cause serious harm. An example of the inherent vulnerability of low cost wireless devices is identified by the US Department of Homeland Security - Industrial Control Systems Cyber Emergency Response Team, which found that roughly 300 wireless medical devices have a hard-coded password vulnerability [41].

Currently available tools that perform penetration testing against Bluetooth de-

vices, such as those included in Kali Linux version 2016.2, lack long-range distance estimation capabilities for either BTC or BLE. Engadget published an article [42], that stated presently there is a serious need to discover Bluetooth devices and identify security vulnerabilities, which was the motivation behind the range-finding tool herein called BlueFinder. Additionally, another platform attempting long-range distance estimation, Blue Hydra, has limited capabilities due to relying on the transmit power advertised by the device and a low path loss exponent. Previous works leverage a distributed array of sensors to track Bluetooth transmitter movement [43] [44] [45] [46]. However, tools such as zbfnd, which provides ranging for ZigBee devices, do not exist for Bluetooth devices. This work adapts the log-distance path loss model used in zbfnd, requiring the fitting of several model parameters empirically, to Bluetooth [47]. The results show that BlueFinder is capable of estimating the distance of remote Bluetooth devices up to 1,000 meters with less than 20% error. This tool is developed for penetration testing or warwalking to rapidly locate BTC and BLE devices, which could enhance security by locating rogue devices.

Section 3.2 establishes the objectives for the range-finding experiment. Section 3.3 outlines the experiment in a scenario. Section 3.4 lists assumptions and describes the variables used. Section 3.5 outlines the system design in both hardware and software. Section 3.6 introduces and derives the log-distance path loss model and the RSS values for BTC and BLE devices. Performing the analysis of the results is in Section 3.7. In addition, this section introduces the BlueFinder tool and demonstrates its usage. Section 3.8 provides the conclusion and steps forward in securing Bluetooth devices.

3.2 Objectives

The purpose of this experiment is to meet goal (1) of this thesis. The goal was to develop methods for detecting the location of BLE and BTC devices. In this exper-

iment, the RSS of Bluetooth devices is recorded and used to calculate the distance between a transmitter and receiver. The hypothesis for this experiment is that a Bluetooth device can be successfully tracked through its RSS, while maintaining a relatively low sample size for the mean. This ties back to the original hypothesis that Bluetooth Low Energy (BLE) access control devices can be exploited and protected by evaluating their wireless security and developing exploits and mitigation solutions. The ability to track devices opens up the ability for bad actors to exploit devices through physical access. In addition, locating rogue devices on a network is a form of protection for security managers. This section determines what is the minimum sample size for the mean RSS that still produces a relatively accurate distance estimation. In this experiment, several different control factors are evaluated as enumerated in the following sections.

3.3 Scenario

The scenario defines the experiment of using a Bluetooth device at varying distances and tracking its location through RSS at a receiver. More specifically, a Bluetooth device generates packets at a set distance that the receiver measures and records. The receiver determines the signal strength of the transmitted packet (RSS). The receiver measures the signal strength internally, prior to the data being transmitted between the receiver and the computer through the HCI. A text file records the RSS data for post collection analysis in MATLAB. A detailed method for estimating device distance is outlined in Section 3.5.

3.4 Assumptions

Several assumptions are made since all possibilities cannot be accounted for. The following are the assumptions made for this experiment:

- Any single packet has a large range of RSS values
- Path-loss exponents does not equal those in a vacuum ($P = 2.0$)
- Communication channels contain noise from other electronics using the 2.4 GHz band (e.g., Wi-Fi and Zigbee)
- The environment adds Gaussian noise
- A device moves no faster than 20 meters/second
- The number of samples is less than 100

3.4.1 Response Variables.

Table 5 summarizes the experiment’s response variables.

- **Device Distance** (d) – Measured in meters, this is the distance between the transmitter and the receiver. Device distance is the final output of a location of a target device.
- **Mean Absolute Percentage Error** (M) – Measured in percent in form of Equation 1, this measures the prediction accuracy of a trending estimation. This equation uses n as the number of samples, D_t is the true distance, and d is the distance estimated by the model, with lower M values representing better performance.

$$M = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{D_t - d}{D_t} \right| \quad (1)$$

Table 5. Response Variable Summary

Response Variable	Normal operating level & range	Measurement precision & accuracy	Relation of response variable to objective
<i>Device Distance</i>	0 - 1000 m	20 m steps, measurement based on prior research	The distance a device is estimated to be from the receiver.
<i>Mean Absolute Percentage Error</i>	0 - 200 %	0.01, dependent on the significant figures of the rest of the data	Representation of the accuracy between the estimated distance and actual distance between the transmitter and receiver.

3.4.2 Controlled Variables.

Running this experiment in a combination of hardware and simulated environment allows many factors to be monitored, increasing the number of configurations that can be tested. Therefore, only a limited number of controlled variables are used to simplify the experiment. Table 6 summarizes the values used in the experiment for the following factors.

- **Reference Distance (d_0)** – Measured in meters, this is the initial distance used to initialize the log-distance path loss model. This value is chosen to be the distance that the reference RSS (RSS_0) will be measured from.
- **Received Signal Strength at Reference Distance (RSS_0)** – Measured in decibels-milliwatt (dBm).
- **Path loss exponent (P)** – The path loss exponent is used to calculate the signal strength degradation relative to the distance traveled. The path loss

exponent typically ranges from values of 1.2 to 6, depending on the environment [48].

- **Sample Size (n)** – Measured in number of packets, the sample size is used to determine the number of samples used to calculate the mean RSS.

Table 6. Controlled Variable Summary

Controlled Variable	Normal operating level and range	Proposed Settings
d_0	0 to 200 m	1, 10, 50, 100, 200
RSS_0	-40 to -10 dBm	-10, -15, -20, -25
P	1.5 to 3.5	2.0, 2.1, 2.3, 2.4, 2.5
n	10 to 100	10, 20, 30, 40, 50, 60, 70, 80, 90, 100

3.4.3 Constant Variables.

Several factors are held constant throughout the course of the experiment. This does not suggest that variations in these values do not impact the response variables, but are assumed constant to limit the scope of the experiment. Table 7 summarizes the experiment’s constant factors.

- **Packet Size** – This factor can affect the RSS if the packet sizes vary greatly. Using the same type of packet (advertisement) and same device ensures the same packet size throughout the experiment.
- **Signal Interference** – This factor effects the RSS and can be negated by running more test trials and by conducting tests back to back. Also, choosing an environment with minimal interference is preferable. The experiment mitigates signal interference by choosing an environment with minimal inference (e.g., outdoors).

- **Movement Space** – This value defines the space in which the experiment is conducted. The experiment is limited to a space of 50 x 1000 meters which allows for the receiver to be within the antenna’s main lobe.
- **Transmit Rate** – This is the data rate that BTC and BLE operates and is assumed to always be constant.

Table 7. Constant Variable Summary

Factor	Desired Experimental Level	Controlled How?	Anticipated Effects
<i>Packet Size</i>	20-30 bytes	Use same device and measure only advertisement packets (all have the same length for the same device)	Packet size will not change RSS
<i>Signal Interference</i>	minimal	Test conducted at roughly the same time	Signal interference will minimally change RSS between tests
<i>Movement Space</i>	50 x 1000 m	Test Configuration	None
<i>Transmit Rate</i>	1 Mbps	Test Configuration	None

3.4.4 Data Collection and Reduction.

Data collection is performed using an RSS logging function built using Python. Each measurement of RSS taken is stored for evaluation. Any calculations performed to summarize the data will be post-collection, so that tests for outliers and further analysis can be performed as needed. Summarizing the data prevents investigation into edge cases and outliers which would otherwise cause a greater amount of error in smaller data sets. Final design allows for data collection and processing to occur in real time to facilitate the production of a range-finding tool.

3.5 System Design

The experiment uses two frameworks simultaneously (hardware and software). The following subsections discuss the evaluation of both systems. Figure 6 illustrates the system outline for Bluetooth range-finding.

3.5.1 Hardware.

The simulation is running on a custom built workstation. Hardware is configured with a 64-bit IntelTM Haswell i7-4790k quad-core processor running at 4.0 GHz (auto overclock to 4.5 GHz) with 32 GB DDR3 RAM (PC-2400) on a ASUSTM ROG Maximus VII motherboard. The test system OS is Windows 10 with all unnecessary services and applications closed to prevent interference with the testing. The data is collected using a Sena UD-100 Class 1 Bluetooth adapter, see Figure 7, with an optional detachable antenna. Since this device allows for an external antenna, a 15 dBi Yagi antenna is equipped to greatly increase the range and sensitivity of the receiver compared to the 3 dBi omnidirectional antenna provided by the manufacturer.

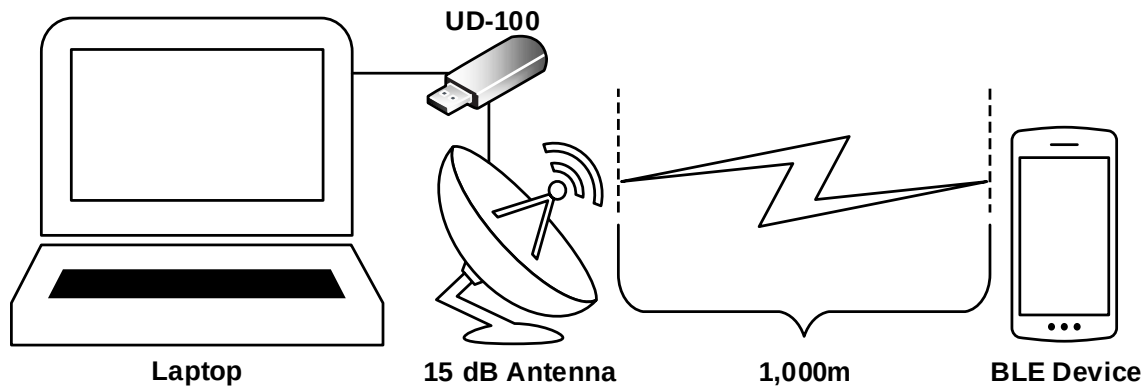


Figure 6. Hardware system diagram for Bluetooth range-finding using a laptop, Sena UD-100, and 15 dB antenna.



Figure 7. The Sena UD-100 Class 1 Bluetooth adapter with optional detachable antenna used for range-finding experiments and long-range Bluetooth communication.

3.5.2 Software.

The log-distance path loss model uses MATLAB to conduct the calculations. The generated model has minimal effect on the overall performance of the system since the only variables being varied are the reference distance (d_0), received signal strength at the reference distance (RSS_0), number of samples used for the mean (n), and the path loss exponent (P).

3.5.3 Python and Scapy.

Creating a communication method for Python to communicate with the Bluetooth transceiver is the first step. Python was chosen as the communication language used at the user level due to its easy implementation and the inclusion of the Bluetooth socket. The Bluetooth socket is a function built into Python that allows the program to bind to the HCI and send and receive commands. Figure 8 presents a flow diagram

of how the distance estimate is generated.

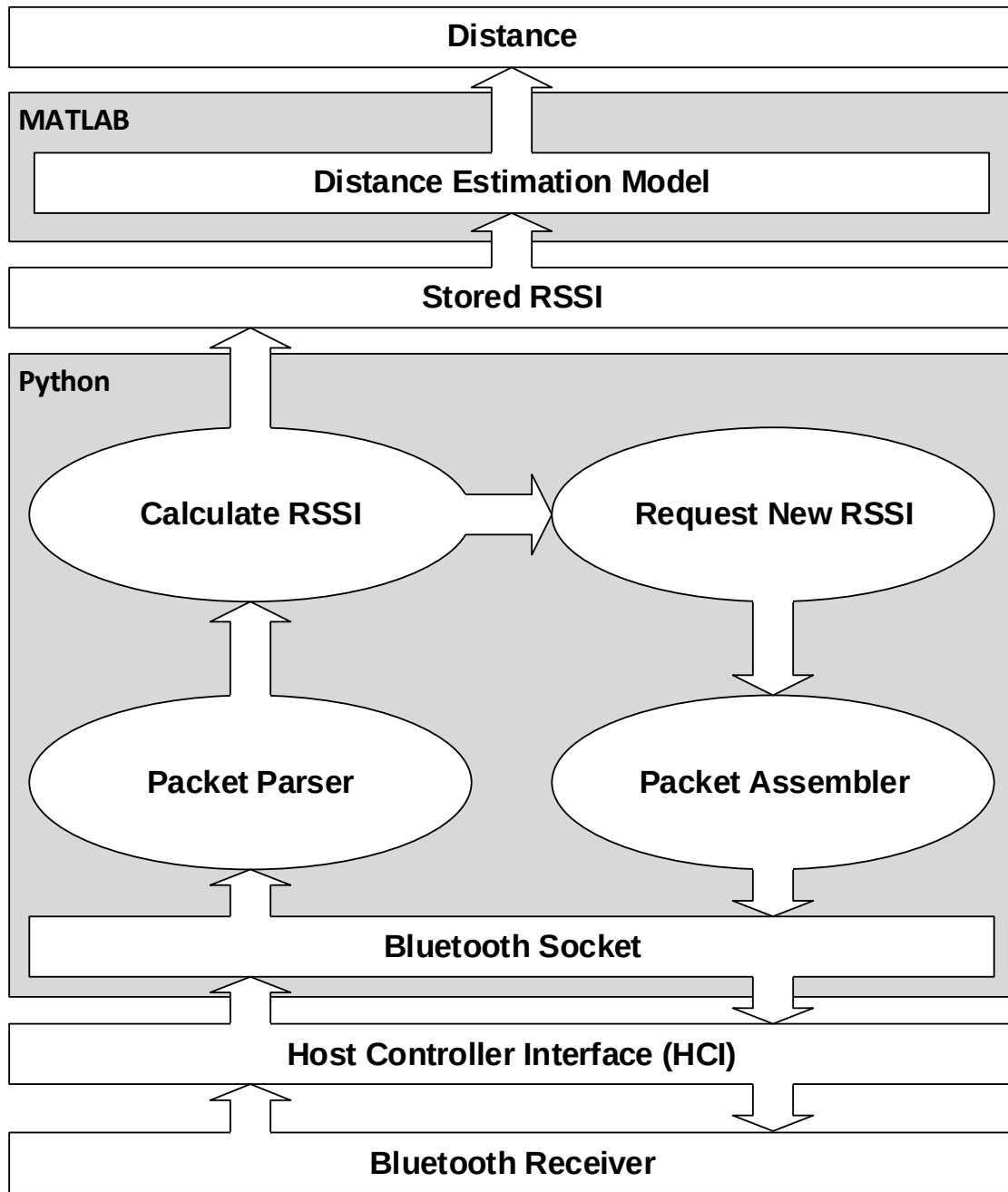


Figure 8. Software design for parsing and assembling packets for determining distance estimation.

3.5.4 Packet Assembler.

The packet assembler allows for the user to easily select Bluetooth commands which are sent via Python. The assembler generates the selected command inside a Python script. The purpose of the assembler is most packets being created and sent require HCI header information (e.g., data length, destination, and command type) which would otherwise require the user to manually create. The code in Python for the packet assembler can be found in Appendix A.

3.5.5 Packet Parser.

Packet parsing allows for HCI commands to be translated into readable ASCII text. Normally the HCI provides information in hexadecimal that would need to be decoded to be in a readable format. The next challenge is reading the data length field inside of a received packet to ensure only the data field is being read. Once the packet type is successfully identified, the parser can determine which byte is the data length.

Reading fields of data is extremely important for this research effort. For example, the RSS field is essential for the calculations in the distance estimation model. In addition, distinguishing devices can only be accomplished through parsing Bluetooth packets for BD_ADDR. This provides the motivation behind developing a Bluetooth packet parser within Python.

3.6 Design Approach

The design of BlueFinder follows the approach taken in [5] [47], where they develop a capability for ranging IEEE 802.15.4 devices. The mechanism employed for estimating distance is the log-distance path loss model, which models distance as a function of RSS. The utilization of this model requires determining the path loss ex-

ponent (P), and the reference RSS at one meter (A). Both are found empirically in the following sections for BLE and BTC devices.

3.6.1 Log-distance path loss model.

The log-distance path loss model is defined as

$$d = d_0 \cdot 10^{\frac{A-r}{10P}} \quad (2)$$

where d is the estimated distance to the transmitter in meters, P is the environmental path loss exponent, A is the reference RSS at $d_0 = 100$ meters, and r is the RSS in dBm as measured at an unknown distance from the transmitter [47]

The path loss exponent P ranges in value from 1.2 to 6, depending on the environment [48]. Propagation through free space is modeled by $P = 2$. In environments where signals propagate through waveguides, such as hallways, P is lower than the free space exponent. Alternatively, higher values of P up to 6.0 may result from indoor attenuation through walls or from interference in urban environments [49].

3.6.2 Finding the reference RSS.

The distance estimation model requires the RSS value when the target is at a known distance away as a reference point [50] [51] [52]. To determine a reasonable estimate for this parameter, RSS is recorded from a large sample of Bluetooth devices including heart rate monitors, smart phones, speakers, and computer peripherals. The pool of devices is organized into two classes, based on the device using either BTC or BLE. Consequently, the reference RSS is sampled from 15 BTC and 15 BLE devices.

Each device, in turn, is placed 100 meters from a Bluetooth receiver that is connected to a laptop via USB. With the target device in place, the receiver collects 5,000 RSS observations. The observed data is added to one of the two RSS pools, depending

on the radio stack of the device. The reference RSS value for each class is calculated by averaging the observed RSS values for all devices of the class. After averaging each pool and quantizing the value to a RSS level available for the BCM20702A0, the reference RSS (A) for a BTC device is -10 dBm and BLE devices is -57 dBm.

3.6.3 Finding best-fit path loss exponent.

A series of experiments are conducted to determine the best-fit value for P in the path loss model by evaluating the measurement error of the model for a range of P values in different collection environments. Four distinct cases are considered by collecting against a BTC class device and a BLE class device, where each device is observed in a line-of-sight (LOS) and beyond line-of-sight (BLOS) scenario. For each of the four scenarios, RSS measurements are taken while varying the distance between a receiver and transmitter from 1 to 1,000 meters at 10 meter increments. For each measurement point, the mean RSS is calculated from 5,000 RSS observations.

The error in the range accuracy of the model is measured using MAPE [5] [47] [35]. MAPE (M) is calculated by

$$M = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{D_t - d}{D_t} \right| \quad (3)$$

where n is the number of samples, D_t is the true distance, and d is the distance estimated by the model, with lower values representing better performance. The path loss model estimates distance, which is compared with the true distance to calculate MAPE for all points. Parameter P is varied from 1.5 to 5.0 in increments of 0.1 to determine the path loss exponent with minimum MAPE for each of the four cases. Values for P outside this range are not considered because they incur MAPE values approaching 100%. The values derived for the reference RSS (A) in Section 4.2 are used in the path loss model as a function of the device being collected.

Thus, the best-fit value for the path loss exponent is the value for which incurs minimum MAPE. The goal was to minimize MAPE in all cases and produce exponents that are relevant to the tested scenarios. Since a user must possess additional knowledge about the remote device or environment to select the appropriate path loss exponent.

The selected BTC device for the experiment is the Cardioline Microtel, pictured in Figure 9a. The Cardioline Microtel is a Bluetooth EKG monitor that utilizes BTC to transmit EKG data to a remote machine. The BLE device under test is the Fitbit Surge. The Surge, shown in Figure 9b, is a dual-mode personal wireless heart rate monitor that continuously records and transmits biometric data using BLE. The Surge also communicates using the BTC stack to receive text, phone call notifications, and implements a music control interface.



(a) Cardioline Microtel EKG

(b) Fitbit Surge

Figure 9. Target BLE and BTC devices used for RSS collection experiment.

The receiver system, used to measure RSS samples from the two heart monitoring devices, is a USB Bluetooth transceiver connected to a laptop, running Kali Linux 2016.2. The BlueZ software package is loaded onto the laptop, which provides an implementation of the Bluetooth protocol stack for all Bluetooth versions. This package contains modules that implement functions not normally accessible to the user, such as providing access to the HCI and Service Discovery Protocol (SDP) libraries.

The receiver collects RSS values for the BTC device. Tools such as Bluelog or Redfang initiate the discovery and reconnaissance process by learning the BD ADDR of the targeted device. The BD ADDR is required to begin collecting RSS. A flood of ping messages is sent to the targeted device. The receiver uses the RSS function HCI tool built into Bluez to read the RSS values from the ping responses. The receiver uses the passive and active scan response messages, advertisements packets, to record the RSS values for BLE Devices. Moreover, BlueZ does not implement a direct method to read the RSS for devices using the BLE stack. Creating a script to extract the RSS values from raw HCI data alleviates the issue.

An open field in a park is used as the LOS test environment. The field contains low vegetation, no human activity, and few trees with a two-lane road running down the center. The BLOS environment contains densely packed trees, buildings, and cars that obstruct vision and radio frequency signals. The Bluetooth spectrum also incurs a higher amount of congestion in urban environments due to the increased density of other devices sharing the same spectrum.

3.7 Results and Analysis

Figures 10a - 10d illustrate that using raw RSS values to calculate a distance estimation model is not possible due to the range of values collected. It can be seen that a single RSS value can range nearly the entire spectrum of distances. However,

a mean RSS can be used which reduces the range of values significantly.

Table 8 - 9 shows the result of measuring distance error in terms of MAPE for an EKG and Fitbit in both LOS and BLOS environments over a range of distances from 1 up to 1,000 meters. Each row indicates the measured MAPE for each of the four cases for a particular path loss exponent, shown in the left-most column of the table. In all four cases, MAPE is minimized when $P = 3.2$ for LOS case and $P = 4.4$ in the BLOS scenario, shown in bold in Table 8 - 9. As a result, the path loss exponent value of $P = 3.2$ and 4.4 are chosen for the log-distance path loss model used in BlueFinder and the average MAPE incurred for all of the experimental cases is less than 20%.

3.7.1 Measured RSS vs. model-derived RSS.

The model predicts RSS for a device at distance (d) from the receiver. To illustrate the feasibility of this, the collected RSS values for each of the four cases are plotted as a function of distance. This is overlaid with the mean RSS at each distance with a sample size (N) equal to 100, the predicted value derived using $P = 3.2$, and either

Table 8. MAPE for a range of P for LOS models.

P	BTC (LOS)	BLE (LOS)
	Error (%)	Error (%)
	<i>RSSI₀</i> = -10 dBm	<i>RSSI₀</i> = -53 dBm
2.5	34	35
2.6	31	31
2.7	28	27
2.8	25	26
2.9	23	23
3.0	21	18
3.1	21	17
3.2	21	16
3.3	23	17
3.4	25	18
3.5	28	21

Table 9. MAPE for a range of P for BLOS models.

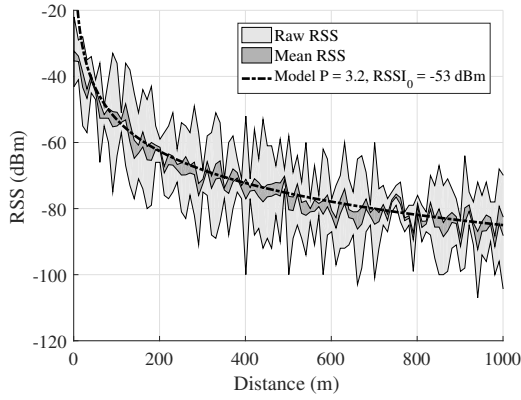
P	BTC (BLOS)	BLE (BLOS)
	Error (%) <i>RSSI</i> ₀ = -70 dBm	Error (%) <i>RSSI</i> ₀ = -30 dBm
4.0	18	26
4.1	17	25
4.2	16	25
4.3	17	24
4.4	17	23
4.5	18	23
4.6	19	23
4.7	20	24
4.8	21	26
4.9	22	27
5.0	25	30

$A = -10$ dBm or $A = -53$ dBm, depending on the device having BTC or BLE capabilities.

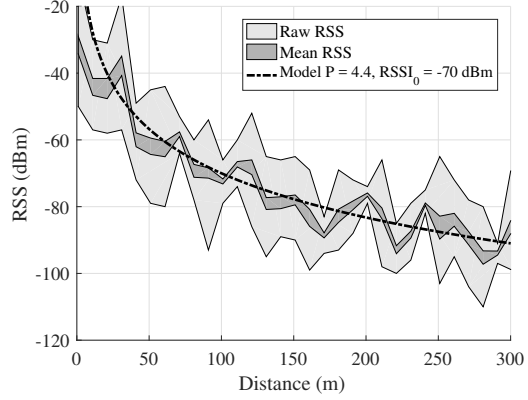
Figure 10a – 10d provides the results of the overlaid RSS predictions and measurements. The figures correspond to one of the four cases: BTC LOS, BTC BLOS, BLE LOS, and BLE BLOS. The x-axis corresponds to the distance between the receiver and target in meters, and the y-axis is the RSS in dBm. From the legend, the solid black line is the RSS as calculated using the log-distance path loss model using the provided parameters for P and A over many distance points. For each of the distances where RSS is measured in the experiment, all 5,000 of the collected samples are shown with their maximum and minimums in the lightly shaded region. Many RSS measurements overlap in the plot due to the quantization of RSS to the capabilities of the receiver. At every position containing RSS sample data, the sample mean is provided as the darker shaded region.

In nearly all cases, the predicted RSS value falls within the RSS values measured at the same distance. It can be seen that the range of the RSS used to generate the distance estimation is greatly reduced when comparing the mean RSS to the raw

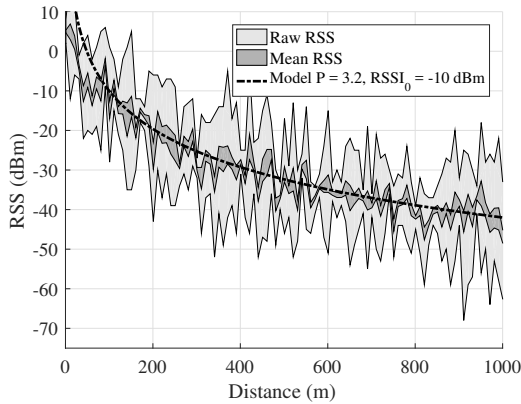
RSS. When the sample means are considered, the error of the model may be observed. Figure 10a - 10d show that the model deviates with a wider range of raw RSS values in the BLOS cases with values as high as 10 dBm from the sample mean.



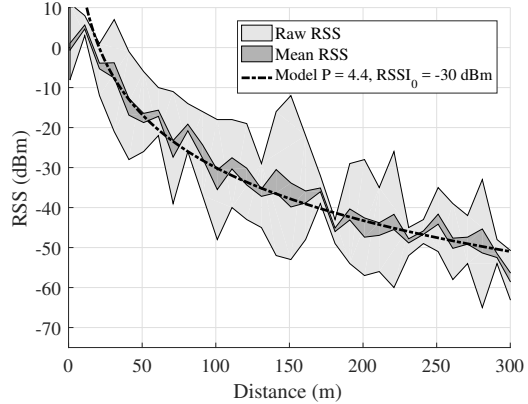
(a) Case 1: BLE LOS model compared against mean RSS and raw RSS using $P = 3.2$ and $RSSI_0 = -53$ dBm.



(b) Case 2: BLE BLOS model compared against mean RSS and raw RSS using $P = 4.4$ and $RSSI_0 = -70$ dBm.



(c) Case 3: BTC LOS model compared against mean RSS and raw RSS using $P = 3.2$ and $RSSI_0 = -10$ dBm.



(d) Case 4: BTC BLOS model compared against mean RSS and raw RSS using $P = 4.4$ and $RSSI_0 = -30$ dBm.

Figure 10. Measured RSS at distances from 1 to up to 1,000 meters vs. model-derived RSS estimates.

3.7.2 BlueFinder Tool.

BlueFinder is a new open-source tool developed by the authors that uses the distance estimation model developed in this thesis to locate Bluetooth devices. It is designed to run on a Linux machine with BlueZ Version 5.38 and a dual-mode Bluetooth adapter.

The tool requires the user to provide the BD ADDR and to specify the operating mode of the target device (e.g., either BTC or BLE). Other Bluetooth sniffing tools available in Kali provide this information. Figure 11 demonstrates both of the parameters needed for operating BlueFinder. The tool estimates distance every second using the best-fit model and a mean RSS value, where 100 samples per second are used to generate the mean RSS. A continuous display of estimated distances to the targeted device is presented to the user as illustrated in Figure 12.

```
root@kali:~/BlueFinder v1.1# ./BlueFinder.sh

      BlueFinder v1.1
./BlueFinder.sh <sel> <BT_Addr>

      DESCRIPTION
<sel>          C - Classic, B - Low-Energy
<BT_Addr>      Bluetooth Address
```

Figure 11. List of currently-available BlueFinder options within the tool.

```

root@kali:~/BlueFinder v1.1# ./BlueFinder.sh C 20 40:33:1A:EC:34:8A

Scanning...
3.0 m
2.4 m
1.4 m
1.0 m
1.0 m

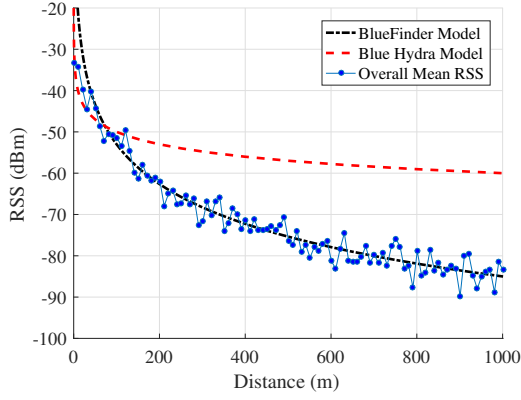
```

Figure 12. Using BlueFinder’s distance estimation model to determine location of a BTC device.

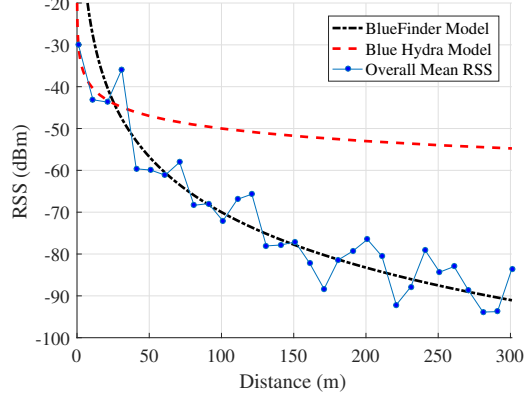
3.7.3 BlueFinder vs. Blue Hydra.

A major difference between distance estimation models in Blue Hydra and BlueFinder is the method used to generate the reference RSS and the path loss exponent. Blue Hydra determines its reference RSS by reading the transmit power from an advertisement packet. An issue with using the advertised transmit power is that only around 10% of devices provide this information. In addition, this research found that the reliability of the transmit power to be low due to most devices using a value of $A = 0$ or 3 dBm for $d_0 = 1$ meters. In addition, the path loss exponent for their model was considerably lower than the expected value for free space. The estimated distance was not very accurate since their model uses $P = 1$. Figures 13a - 13d demonstrate the performance of BlueFinder and Blue Hydra and how their models compare against the mean RSS. It is important to note that Blue Hydra does not utilize a mean RSS value, but instead uses the raw RSS data, which was demonstrated in Figure 10a - 10d to be inconsistent.

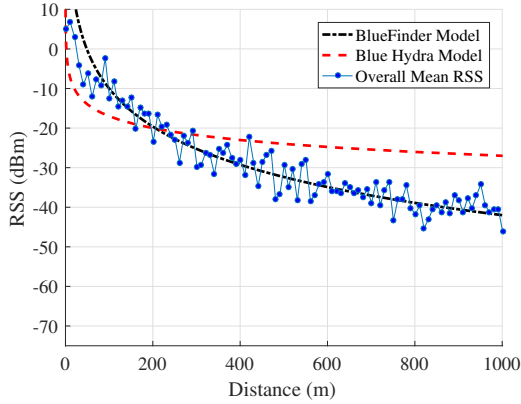
A limitation of BlueFinder is that it sacrifices performance at closer distances to increase its overall accuracy for the model. This is accomplished by decreasing the $RSSI_0$ at 100 meters slightly to compensate for the large drop in RSS, this causes the model at closer distances to not be able to distinguish between 1 and 20 meters.



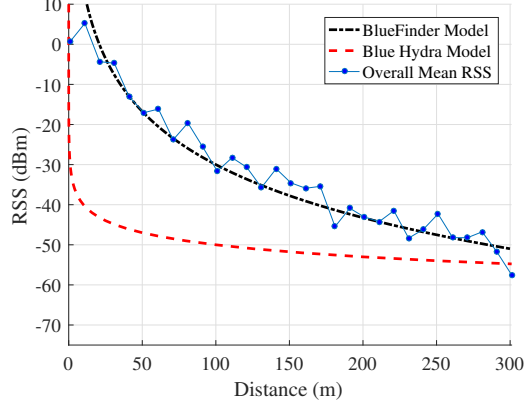
(a) Case 1: Mean BLE LOS RSS at various distances vs. BlueFinder and Blue Hydra models, $P = 3.3$.



(b) Case 2: Mean BLE BLOS RSS at various distances vs. BlueFinder and Blue Hydra models, $P = 4.4$.



(c) Case 3: Mean BTC LOS RSS at various distances vs. BlueFinder and Blue Hydra models, $P = 3.3$.



(d) Case 4: Mean BTC BLOS RSS at various distances vs. BlueFinder and Blue Hydra models, $P = 4.4$.

Figure 13. Comparison of Blue Hydra and BlueFinder distance estimation modeling using RSS.

However, using a two-model approach can correct this when the $RSSI_0$ is below a specific threshold. Using a model designed for closer distance in this type of scenario increases the overall performance of the tool.

Blue Hydra outperforms BlueFinder in scenarios when the device is relatively near the receiver. It is difficult for the receiver to distinguish small changes at relatively close distances due to antenna saturation. The RSS precision of a receiver is limited to whole numbers, which leaves a high degree of error between two values.

3.8 Summary

BlueFinder measures the effective range to both BTC and BLE devices using Kali Linux and a Bluetooth transceiver at long ranges. At ranges from 1 to 1,000 meters, the MAPE is less than 20%. In this thesis, the log-distance path loss model is adapted to estimate distances of BTC and BLE devices in an office hallway and an open field setting. The findings demonstrate that the distance estimation model gives the best performing tool at distances greater than 50 meters with the MAPE value for the Blue Hydra model being 95% compared to the BlueFinder model operating below 20%. This model can be adapted for Blue Hydra to greatly increase the reliability of its distance estimation. The code in Python for BlueFinder can be found in Appendix B.

Exploration for future work includes use of this tool in different radio frequency (RF) environments. Specific targets include high clutter environments such as urban areas or more complex office geometries. Previous work has shown that filtering, such as applying a Kalman filter, can reduce the error in raw RSS data [52][53]. Investigating the improvement of BlueFinder using filtering methods prior to calculating distance should be considered. In addition, evaluating more Bluetooth devices to compute the reference RSS value increases the accuracy of BlueFinder. Adapting BlueFinder to the Ubertooth One would allow the RSS to be determined passively and enable stealthy tracking of Bluetooth devices, similar to the method that Blue Hydra uses to determine RSS, however some minor coding improvements can increase their compatibility of 10% to nearly 100% that BlueFinder experiences due to the parsing method.

Building BlueFinder accomplishes the goal (1) develop methods for detecting the location of BLE and BTC devices. The ability to track Bluetooth devices is the first step in identifying potential security vulnerabilities. The next chapter explores

BLE vulnerabilities that are exploited once a device has been located. In addition, BlueFinder can be used to track the location of malicious devices to find them and disarm them.

IV. Securing BLE-enabled Locks against Unauthorized Access and Surveillance

4.1 Introduction

The discovery of Bluetooth Low Energy (BLE) vulnerabilities is a core requirement for meeting the purpose of this thesis. This chapter achieves the remaining goals that were established. The goals met are (2) Identify vulnerabilities in BLE devices and develop exploits that result in physical effects, (3) Utilize vulnerabilities for unauthorized BLE device access and develop methods for extracting user data and generating User Behavior Analytics (UBA), and (4) Propose mitigation solutions for the determined BLE vulnerabilities. Meeting goal (2) is accomplished by revealing and developing vulnerabilities in 13 BLE access control devices. Next, goal (3) is achieved by demonstrating the ability to extract sensitive personal information from a BLE device and conduct analysis within a scenario. Finally, goal (4) is met by proposing five mitigation techniques to counter the developed exploits. This academic work is submitted to the International Journal of Critical Infrastructure Protection and the following sections are similar to the presented work.

4.2 User Behavioral Analytics

User Behavior Analytics (UBA) detects anomalies that indicate potential for insider threats or targeted attacks by tracking and analyzing user behaviors. Defensive mechanisms use UBA to help prevent attacks. This research proposes an offensive UBA approach. One of the most important stages for penetration testers is reconnaissance due to the importance of gaining detailed knowledge of a target prior to an attack [54]. A target is continuously monitored for all activity from a facility during this phase.

Time is critical when it comes to gaining information on a target and acting on it. The ability to minimize time spent between reconnaissance and infiltrating a target would greatly enhance an attack’s chances of success [47]. In the past, an attacker would physically monitor a facility to gain information, however, utilizing information already present on BLE devices would give a significant advantage to the attacker. A number of devices store system logs that contain useful UBA information (e.g., user names and time stamps). Applying statistical analysis to a large number of system logs would generate meaningful information for an attacker. Patterns of behavior could be inferred through UBA which would indicate ideal times to inject malicious behavior and avoid detection. The first step to obtaining system logs is to use exploits developed for BLE locks (described in the following section).

4.3 Bluetooth Security Vulnerabilities

The analysis of 17 BLE locks reveals 13 devices were vulnerable to eight exploits (documented in Table 10) [15]. A wide variety of attacks against BLE devices exist which include vulnerabilities mistakenly implemented by developers due to poor design decisions and others that are inherent in the protocol. This section discusses vulnerabilities present in BLE devices and how an adversary may use these exploits. Note that (A) denotes plaintext passwords, (B) password obfuscation, (C) brute forcing passwords, (D) command fuzzing, (E) hard-coded passwords, and (F) MitM attacks.

The hardware required for BLE eavesdropping is affordable. Higher-end devices such as the HackRF One [55] or Ubertooth One [56] are alternatives which contain higher power amplifiers and detachable antennas. Replacing the antenna can increase the operational range of these devices. Increased sniffer range eliminates the need to be near a target to obtain credentials. Pairing a long-range sniffer with a high-power

Table 10. List of successfully exploited BLE devices by type.

Device Name	Type	A	B	C	D	E	F
Safetech Quicklock Doorlock	Deadbolt	✓		✓			✓
Vians Doorlock	Deadbolt		✓				✓
Lagute Sciener Doorlock	Deadbolt		✓				✓
Okidokeys	Deadbolt				✓		✓
Poly-control Danalock	Deadbolt					✓	✓
Ceomate Doorlock	Deadbolt	✓	✓				✓
Safetech Quicklock Padlock	Padlock	✓		✓			✓
Elecycle EL797	Padlock		✓				✓
Elecycle EL797G	Padlock		✓				✓
Mesh Motion Bitlock	Padlock						✓
iBlulock	Padlock	✓		✓			✓
Safetech Gunbox 2.0	Gun Safe	✓		✓			✓
Plantraco Phantomlock	Cabinet Lock	✓		✓			✓

Bluetooth adapter (e.g., Sena UD-100) allows an individual to send commands across distances up to a half mile.

The National Institute of Technology (NIST) [19] reported in June 2012 that improper Bluetooth security implementation is highly susceptible to wireless attacks (e.g., Denial of Service (DoS), eavesdropping, MitM attacks, message modification, and resource misappropriation). Using these attacks on Bluetooth systems can provide attackers with unauthorized access to sensitive information.

4.3.1 Plaintext Passwords.

A plaintext password exists when an unencrypted password traverses an unencrypted channel. The password is in a readable format and offers no protection to the user or device. This vulnerability is very common among devices [42] [57] and allows an adversary to eavesdrop the conversation through specialized hardware or embed software that monitors HCI traffic. Plaintext passwords can be used to gain access to secure facilities, change administrative privileges, or obtain system logs.



Figure 14. Reverse-engineered Safetech command structure at the HCI level.

Figure 14 illustrates the simplicity of using a plaintext password in an attack. The Safetech [58] command structure is unpublished and was discovered through reverse-engineering commands, a process that malicious actors could implement. This command structure is used in all Safetech BLE devices (e.g., door locks, padlocks, and safes). The first byte represents an opcode used to distinguish if the device should read the password (00) or change a user password (01). Someone with malicious intent can change a user password by changing this first byte and inputting the current user password into the next 4 bytes. The final 4 bytes are then used to set the new user password. Note that the password for this device is limited to only numbers.

4.3.2 Password Obfuscation.

Password obfuscation offers improved security over plaintext passwords, however, it still leaves a major security risk to the user. An obfuscated password uses hashing or encryption to reduce the risk of exposure [59]. The issue with using an obfuscated password is that it can be recorded and replayed to the BLE lock.

Replaying the password allows an adversary to gain access to the lock, even though the password is unknown. An adversary can gain access whenever they want with the sniffed password, if the device uses the same hashing algorithm for the password every time. However, some higher level functions may not be accessible, depending on security implementation. This could deter an attack that requires the password to gain access to higher functions.

An example of password obfuscation is found on the Ceomate door lock. This

product was reverse-engineered to determine the method used to send the password. The lock uses a proprietary hashing process which was unable to be determined. An attacker can still gain access to the device, even though the original password and hashing algorithm are unknown by replaying the recorded credentials.

4.3.3 Brute Forcing.

Brute forcing is a type of attack where numerous iterations of a password or hash are sent to a system with intentions of gaining access [60]. This attack on a system requires using a plaintext password or an obfuscated password. However, if credentials are inconsistent then this type of attack is not practical.

Eavesdropping is not required to observe the conversation containing the password when using a brute force attack. This allows an adversary to attack a device with no user interaction. Brute forcing can be extremely slow depending on the complexity of the password. For example, a device using a 6-digit pin code could be brute forced in hours, while an 8-character password would require nearly a month. Table 11 illustrates the amount of time an attacker would need to dedicate to brute force a password based on the number of characters used and password length. It is important that password length is emphasized over the number of characters available. Doubling the password length exponentially increases the number of password combinations,

Table 11. Time expected to brute force a password by characters and length.

Available Characters	Password Length	Password Combinations (millions)	Expected Completion Time (years)
10	8	10^2	0.03
10	16	10^{10}	3.17
128	8	7.2×10^{10}	22.83
128	16	5.1×10^{27}	1.61×10^{18}
256	8	1.8×10^{13}	5,475
256	16	3.4×10^{32}	1.06×10^{23}

while doubling the available characters has a much smaller affect on the number of combinations. Timeouts between connections will increase the time between sending passwords and ultimately reduce the speed of the overall attack.

Manufacturers using pin codes to protect their system need to require long password lengths. A pin code using only numbers and a maximum length of eight would require at most 12 days to brute force. The example attack in Table 11 limits the speed to 6,000 passwords/minute. Factors that limit the example attack are how quickly the transmitting device sends packets and the speed the receiver translates the data. However, speed of the brute force can be greatly increased if a web server is used to store the credentials.

4.3.4 Command Fuzzing.

Command fuzzing is when an application accepts an invalid command that has been modified to mimic a valid command in hopes that the device will enter a new state [61]. This modification consists of changing individual bytes of a packet until the application accepts the invalid command. The intention of fuzzing a device is to force a transition into an unstable state where it will perform in a way that was not intended. For example, a lock may go into an error state from a fuzzed command and open without proper authentication. Opening while in an error state is usually a design decision for fire safety, since locking would be dangerous. A device may default to a locked state when entering an error state, but these scenarios are typically found in prison lock systems where locking as default would be more practical. Fuzzing becomes an issue when designers implement proprietary encryption. Well established encryption methods already exist (e.g., Advanced Encryption Standard (AES)) and have been proven to offer a secure channel for communication [62].

Figure 15 is the reversed-engineered Okidokeys command structure for authen-

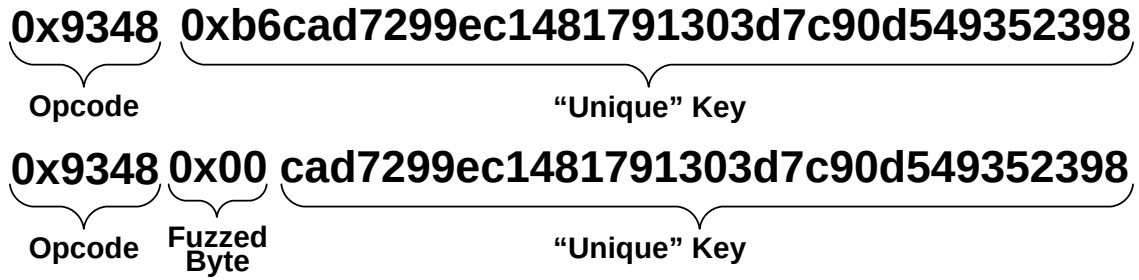


Figure 15. Reverse-engineered Okidokeys command structure (in bytes) at HCI level and fuzzed packet.

tication. The implemented security was cited as using a highly secure encryption method that offers the same protection as AES 256-bit. Okidokey advertises that this security is a patented cryptographic solution. Evaluation revealed that the generated keys were not unique as of this writing. These keys displayed patterns that were not typical in AES or other encryption algorithms and prompted fuzzing of a previously valid command which forced the lock to open in an error state.

4.3.5 Hard-Coded Passwords.

Hard-coded passwords occur from poor programming practices, where designers leave passwords in applications. This vulnerability affects more than 40% of Android applications [63]. These passwords are not easily found and require decompiling the application into readable code. Another attack vector for acquiring the administrative password is embedding a keystroke logger on the device through malware. Hard-coded passwords offer an attacker the ability to gain access to developer options inside of an application or bypass the built-in security.

The easiest method for detecting and obtaining hard-coded passwords on BLE devices is decompiling an APK. An attacker decompiles the application once the APK has been removed from the device. Programs such as Bytecode Viewer offer a user friendly environment to reverse-engineer APKs into readable Java code. This readable

```

public String getPassword(){
    Cursor localCursor = getReadableDatabase().query("USER_TABLE",
        DatabaseContract.UserTableColumns, null, null, null, null, null);
    if (localCursor == null) {
        return "";
    }
    if (localCursor.moveToFirst())
    {
        byte[] arrayOfByte = xor(new String(Base64.decode(
            localCursor.getString(local.getColumnIndex("password"))
                .getBytes(),1).getBytes(),"thisisthesecret".getBytes()));
        localCursor.close();
        return new String(arrayOfByte);
    }
    return "";
}

```

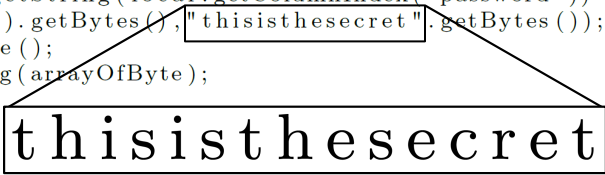


Figure 16. A hard-coded password is found on the on Danalock by decompiling its APK.

code is parsed for keywords, revealing hard-coded passwords, developer comments or other information.

Figure 16 illustrates a hard-coded password found by decompiling the Danalock application. Decompiling reveals the method of encryption and other information that should be unknown. The plaintext password is stored in a table with the passphrase "thisisthesecret". An adversary can discover user passwords and gain access to the lock by extracting the application. This opens the door for other opportunities for an attacker to gain access to information on a target.

4.3.6 Man-in-the-Middle Attack.

A Man-in-the-Middle (MitM) attack occurs when two devices are unknowingly connected to a third device that relays information between the two communicating devices [18]. This is due to a majority of devices using an unauthenticated connection, allowing an attacker to eavesdrop the user and the device. An attacker does not require passive eavesdropping when using a MitM attack, giving attackers the ability to modify commands in real-time. Allowing an attacker to access the conversation

and implement malicious commands without user knowledge. Many tools have been developed that implement MitM attacks, two open-source tools are GATTacker [28] and BTLEjuice [29]. Two attacks that are similar to or use the MitM architecture are explored in the following subsections.

4.3.6.1 Rogue Device Attack.

A rogue device attack is based off of a MitM attack, where the attacker impersonates the target device with the intention of convincing the user that the attacker is the target device. A majority of applications do not properly authenticate with a device before sending commands, allowing an attacker to clone the target device and send advertisements. The user application initiates a connection once it receives the cloned device advertisement. The user application sends various commands to the cloned device as if it was their actual device. These commands include passwords or nonces which are used later to gain access to the target device. Nonces are random numbers that are only used once and protect a connection from a replay attack.

The advantage of this attack is that it can be used to exploit a web server storing credentials. The user application will not be able to distinguish between the true device and cloned device, allowing an attacker to steal credentials from the web server. Accomplishing this attack only requires the cloned device to interact with the user application.

Figure 17 outlines a rogue device attack on Bitlock. This product does not use a plaintext password, however, it has a predictable nonce, allowing an adversary to collect numerous credentials and use them later to control the lock. The user in this case must have an Internet connection to receive credentials from the web server. An attacker connects to the lock and sends an invalid credential with the intention of receiving the current nonce value. The value is sent with the initial connection and

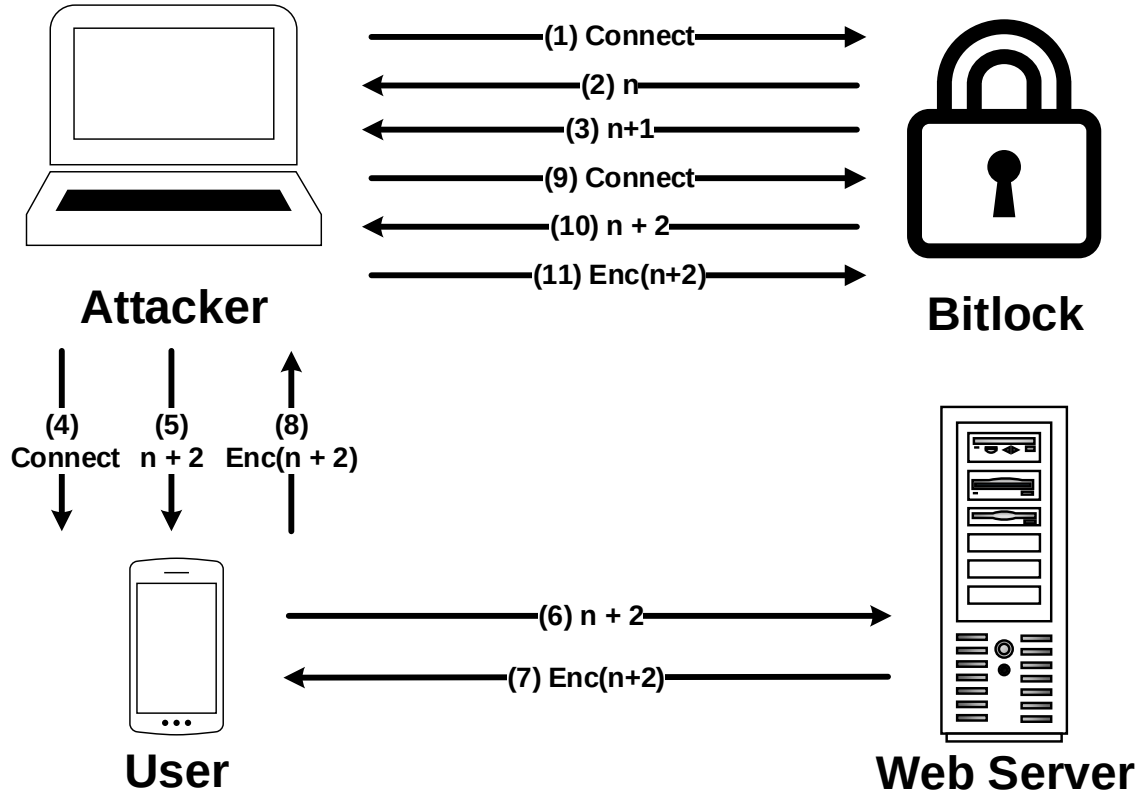


Figure 17. Sequence diagram of a rogue device attack on Mesh Motion Bitlock.

is incremented by one when receiving an invalid credential. The user is unaware that their device is connected to a spoofed lock while receiving the next nonce. During the connection, the user forwards the nonce to the web server and receives back the credentials. Finally, the credentials are sent from the user to the spoofed lock.

The attacker is not limited to receiving only one set of credentials, since the web server trusts the user application. An attacker can flood the user application with nonces in the hope of building a table to permanently have access to the device. Using the table of credentials as a look-up table, where an attacker can open the lock at any time.

4.3.6.2 Relay Attack.

A relay attack is similar to a rogue device attack, but is designed specifically for scenarios when the nonces are truly random and a rogue device attack is not possible. The attacker impersonates the target device and forces the user to communicate through a bridge to their own device. Using a bridge allows the user and device to communicate even though they are not near one another. This forces the attacker to impersonate the target device and user into communicating. The structure depicted in Figure 18 demonstrates using two rogue devices to create a relay attack: (i) Rogue Device 1 connects to the user; and (ii) Rogue Device 2 connects to the target device.

The target device generates a nonce and sends it to the cloned user (Rogue Device 1). Rogue Device 2 connects to the user and impersonates the target device.

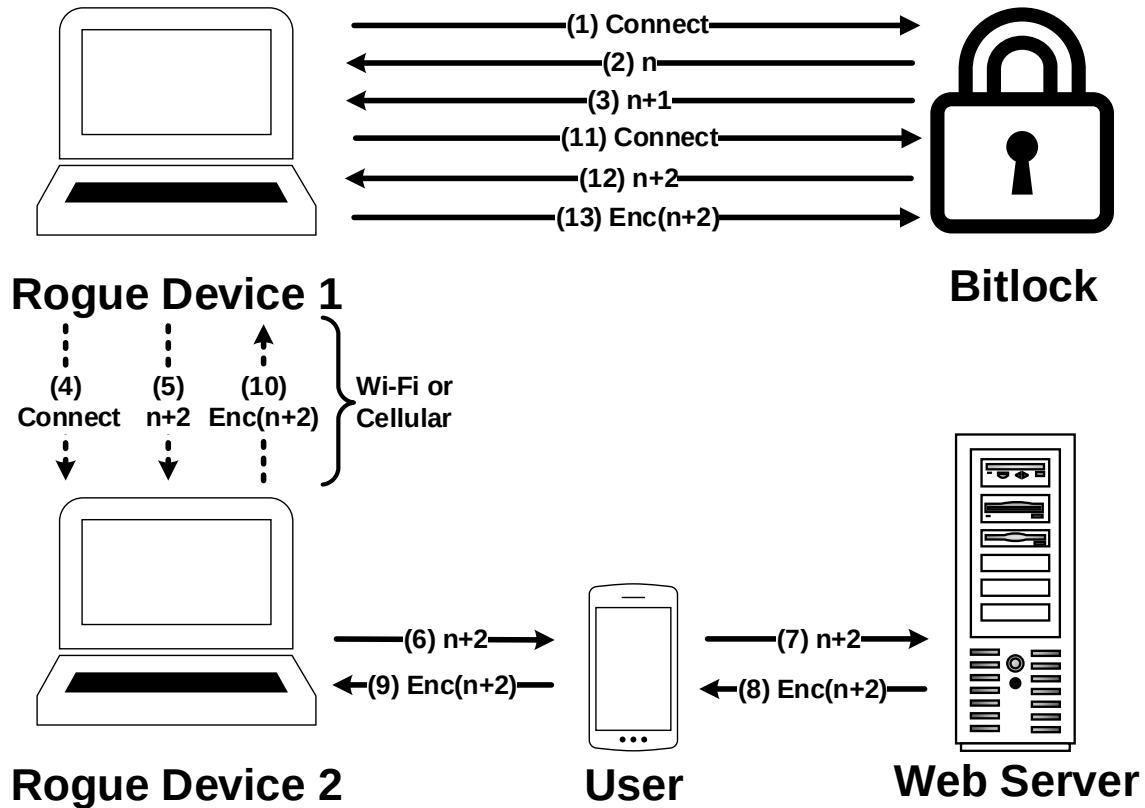


Figure 18. Sequence diagram of a relay attack on Mesh Motion Bitlock as described in Section 4.3.6.2.

Establishing a bridge (e.g., Wi-Fi, cellular, or other methods) once both rogue devices are in place. This allows communication between the rogue devices and facilitates the hand-off of the nonce during the attack.

Rogue Device 2 sends the nonce to the user, who then forwards the nonce to the web server to generate credentials. This phase of the attack mirrors the rogue device attack discussed in Section 4.3.6.1. Finally, the credentials that the user unexpectedly generated are passed from Rogue Device 2 back to Rogue Device 1. These credentials are used by the attacker to gain access to the device. The danger of a relay attack is that a user can be anywhere as long as a rogue device is nearby to impersonate the target device. This type of attack is usable on many devices, since large organizations that require many access points rely on a central server to handle all credentials. Bitlock is used as an example of the relay attack. The setup is exactly the same, except a second device is used to extend the range of the original attack and relay the communication.

4.4 Scenario

Numerous devices store system logs of user activity with information (e.g., user names, permissions and time stamps). User information can be obtained once an attacker has gained access to a device through the attacks listed in Section 4.3. An attacker can extract system logs from locks and analyze this information to construct UBA, a profile of activity on a facility with obtainable information. UBA would greatly increase the evaluation an adversary can conduct by allowing for assessments of facilities and user activity. Building patterns of behavior has an intelligence value that needs to be taken into account, since bad actors can use this information to gain insight into an organization's inner workings.

This section presents a scenario giving insight into how organizations may imple-

ment security using BLE locks. In this scenario, a manufacturing facility uses a series of BLE locks for security. The BLE system is designed to have a central server manage credentials for all employees, requiring employees to authenticate with the server through an application on their mobile device. The simulated data in the scenario mimics real data found on these devices. A proven method of extracting real data is developed, but simulated data is still required to meet the goals of the scenario. Activity for this scenario are given as the chance users were present within a facility using historical data. This information does not predict future habits, however, it can be used to infer future behavior.

1. The attacker connects to security door lock and scans for all services, characteristics, and descriptors.
2. The attacker uses the scanned information to construct an identical BLE device. The clone device is used to impersonate the lock and convince the user application to mistakenly transmit credentials.
3. Concurrently, the attacker uses a second device near the lock to impersonate the user. This setup mirrors the relay attack discussed in Section 4.3.6.2.
4. The attacker relays information, such as nonces and credentials, from the user to the lock through the relay attack. The relay attack allows for access to the BLE lock for exploitation.
5. The attacker accesses developer and administrator privileges to create additional accounts and download system logs.
6. Using UBA (via system logs) reveals patterns of behavior, examples seen in Figures 19a - 19c. Proper analysis of the logs exposes a detailed picture of

facility operations by highlighting user activity within a facility depending on the time or day of the week.

7. The attacker determines an appropriate time to infiltrate the facility through the information gathered. Obtaining physical access to the facility allows for malicious acts against manufacturing, power generation, or water treatment.

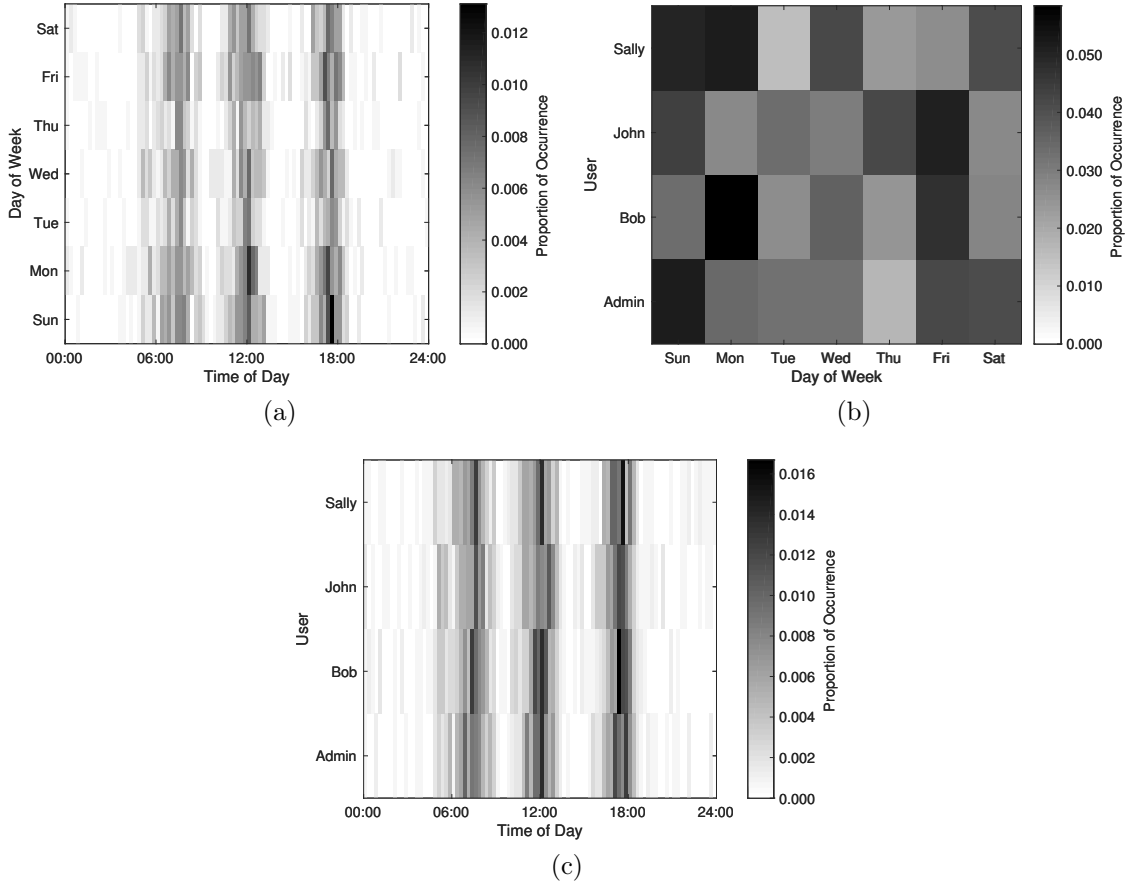


Figure 19. Activity heat maps of (a) historic weekday behavior compared against time of day, (b) historic user behavior compared against weekday, and (c) historic user behavior compared against time of day.

The analyzed data can provide important information when cross referenced to public records of employees. User activity is analyzed in three formats for the scenario: (i) individual user activity by day of the week; (ii) individual user activity by time of day; and (iii) all user activity by day of week and time of day. Figure 19a demonstrates that overall user activity can be determined by comparing user activity to the time of day that they are active. The user activity of entering or leaving a facility is indicated by a heat map, where darker shades represent higher activity. Figure 19b highlights the user activity of historical days worked. Finally, Figure 19c breaks down user activity within any given day by the time of day. The attacker can determine the ideal day and time to access the facility. Further information can be inferred, such as odd activity at specific times of day for specific users which may indicate specific jobs (e.g., maintenance workers or administrators).

4.5 Mitigation Techniques

Many mitigation techniques exist that can avert attacks against Bluetooth. Table 12 lists types of mitigation techniques and the vulnerabilities they protect against. The last two columns of Table 12 rank the implementation and maintenance difficulty of each proposed solution.

4.5.1 Pairing and Bonding.

Pairing and bonding protect against malicious eavesdroppers. Two processes occur with an initial connection. The first step is pairing which is an exchange of security features and capabilities. This begins with the client and establishes which type of input and output mechanisms exist on the device and dictates the type of bonding. Bonding exists after pairing and the keys have already been generated and exchanged. Bonding is a more permanent encryption method used to save the key that will be

used in future connections [64]. When devices are bonded, they can encrypt the connection without the need to exchange keys. BLE utilizes AES-CCM encryption once the key exchange process has been completed.

BLE uses a Secure Simple Pairing model, where devices use one of four pairing modes: (i) Just Works, (ii) Passkey Entry, (iii) Numeric Comparison and (iv) Out-of-band Communication. The pairing mode decides what type of temporary key (TK) will be used for encrypting communication using a 128-bit AES key.

(i) **Just Works** is the simplest method and offers little protection to the device.

This method sets the TK to all zeros, allowing any eavesdropper the ability to immediately guess the TK. The Bluetooth Special Interest Group (SIG) states in their documentation that Just Works provides no protection against eavesdropping and MitM attacks [18].

(ii) **Passkey Entry** requires the user and device to use the same 6-digit pin code as the TK, while the rest of the 128-bit AES key is zero padded. The Passkey Entry provides only slightly more protection against eavesdropping than Just Works and MitM attacks which has been shown in previous work to be brute force-able [11].

(iii) **Numeric Comparison** is similar to Passkey Entry, but both devices input a 6-digit pin independently. This greatly reduces the probability of brute forcing both pin codes.

(iv) **Out-of-band Communication** uses the full 128-bit TK and is communicated across a non-BLE channel, usually using Near Field Communication (NFC) technology. Another method for using this approach is sending the TK across BTC, since most devices are already equipped to handle both BLE and BTC. However, this method is not typical and was only present in 1 of 17 tested

Table 12. List of mitigation techniques and their employment difficulty used to solve BLE vulnerabilities.

Mitigation Techniques	A	B	C	D	E	F	Employment Difficulty	Maintenance Difficulty
Pairing and Bonding	✓	✓	✓	✓			Low	Low
App Layer Encryption	✓	✓	✓	✓			Medium	Medium
Two-way Authentication	✓	✓	✓	✓			Medium	Low
Geofencing						✓	Medium	High
BLE-Guardian						✓	High	High

devices. The need for using an out-of-band channel is extremely important and is the best option when using secure simple pairing [27].

Using the Numeric Comparison technique is a simple solution for manufacturer implementation, since Bluetooth protocol already supports this type of authentication and would not require additional development. This method is only practical if the developers use the key exchange improvements found in Bluetooth version 4.2. No developers studied have used the newest version and are still operating on version 4.0 and 4.1. A new key generation process is implemented in Bluetooth version 4.2 which uses an ECDH key generation and implements new procedures for the key generation. Pairing and bonding protects against plaintext password, password obfuscation, brute forcing, and fuzzing.

4.5.1.1 Bluetooth Version 4.1 Link Layer Encryption.

An added feature when using pairing and bonding is the ability to establish link layer encryption. The encryption method used in version 4.0 and 4.1 is derived from the devices being paired initially and uses the long-term key (LTK) seen in Figure 20. The process for generating the LTK begins with the TK determined through the pairing modes mentioned earlier (i.e., Just Works, Numeric Comparison, Passkey Entry, and Out-of-band Communication). The TK is used to encrypt the short-term

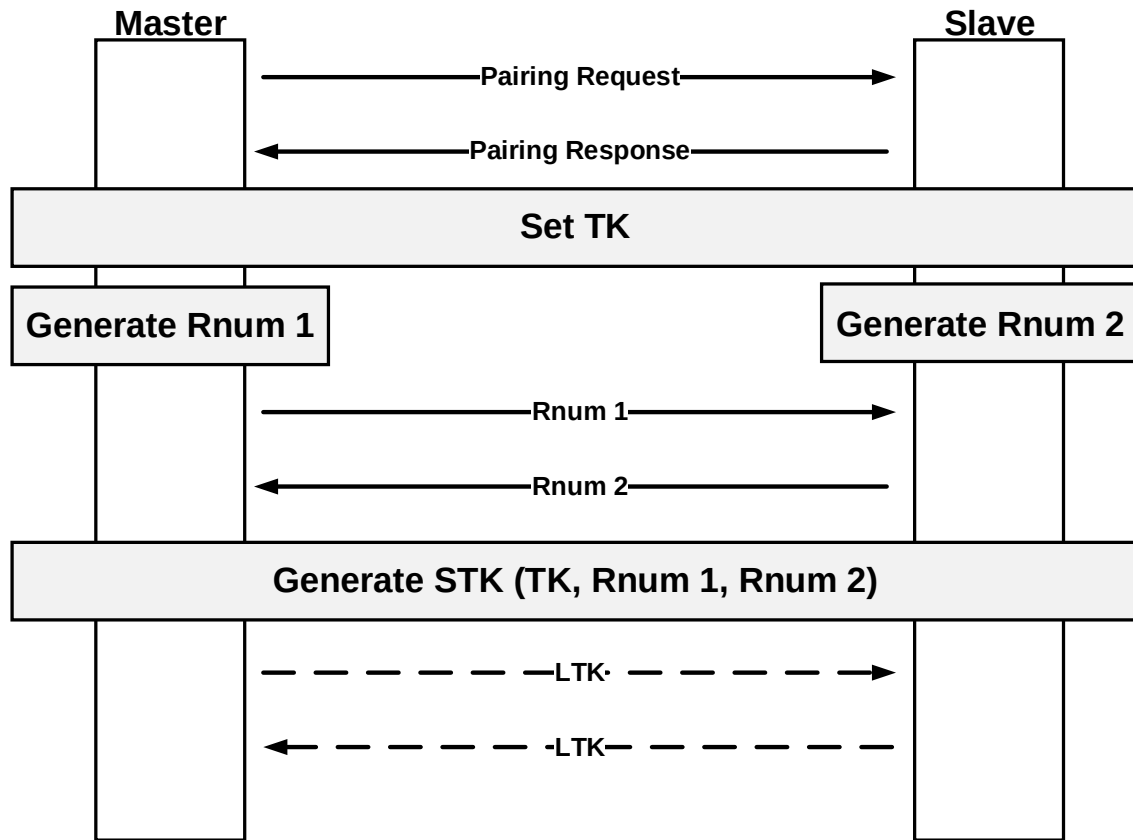


Figure 20. The BLE version 4.1 long-term key generation as described in Section 6.1.1.

key (STK) which is generated using the TK and two random numbers from the master and slave. Finally, the STK is used to encrypt the LTK and will be saved and used for all other communication.

4.5.1.2 Bluetooth Version 4.2 Link Layer Encryption.

BLE version 4.2 no longer uses a STK, but instead uses a key derived from an ECDH key. The new method is illustrated in Figure 21. A pairing request occurs which establishes the method of generating keys. A public key is exchanged to initiate the LTK generation process. Once the public key is exchanged, each device independently computes an ECDH key with the public key of the other device. Next,

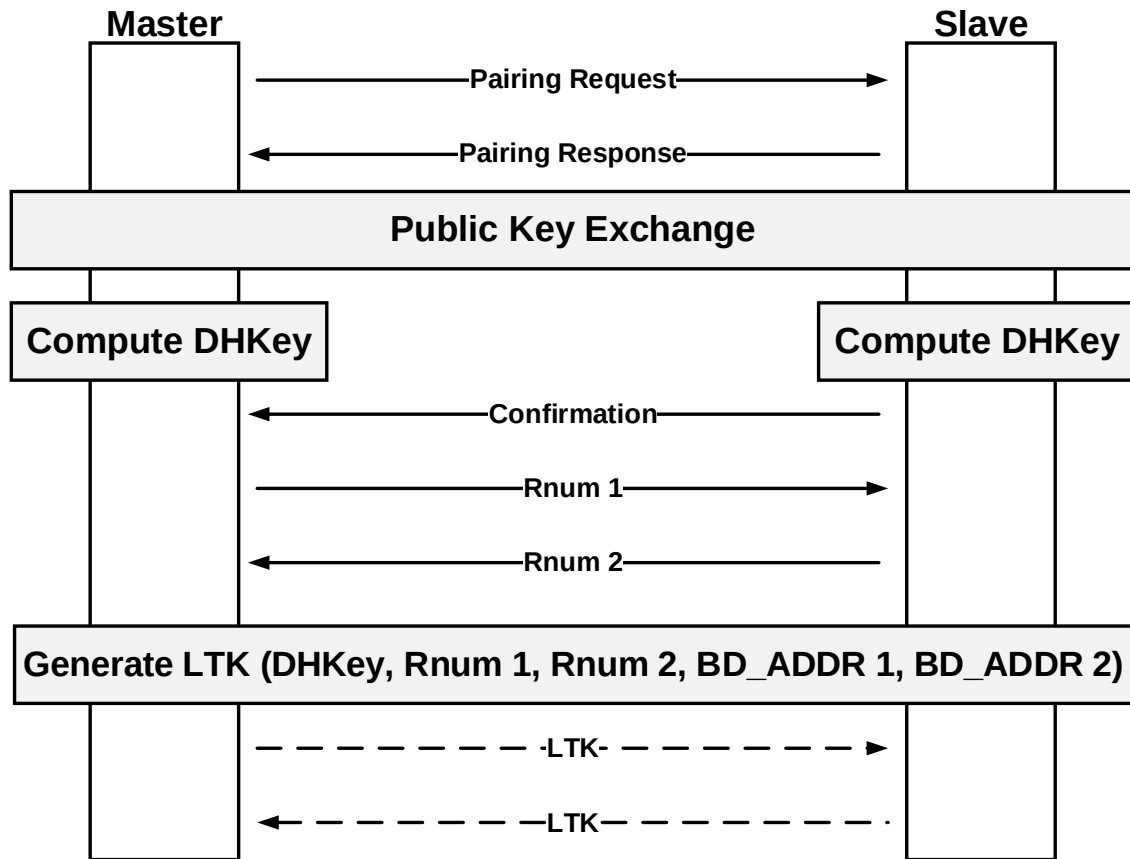


Figure 21. The BLE version 4.2 long-term key generation improves on version 4.1 shown in Figure 20.

the slave computes a confirmation message that the master uses to check against its own key. The master sends a random number to the slave if the confirmation passes. The slave responds with a random number allowing the LTK generation to begin.

The LTK requires five parameters: (i) ECDH key; (ii) random number 1 from the master; (iii) random number 2 from the slave; (iv) Bluetooth device address of the master; and (v) Bluetooth device address of the slave. The important change in the protocol is that the ECDH is never transmitted and is computed independently to protect against eavesdroppers. Passive eavesdropping is no longer possible in version 4.2 due to the difficulty of guessing the private key [10]. Compared to the legacy method which uses the TK to compute the STK to send the LTK. The rest of the

information needed to generate the LTK in version 4.2 are the random numbers sent by each device and their BD_ADDR. The connection is encrypted and authenticated once the LTK is established with the key being used for all future connections.

4.5.2 Application Layer Encryption.

Application layer encryption is one of the most popular methods for securing BLE devices. The reason behind using application layer encryption is that it does not require new devices to be paired, but instead relies on the user and device to establish keys that will be used to encrypt and decrypt credentials. Application layer encryption can be more complicated than using the standard pairing that is offered by BLE, due to the difficulty of managing keys at the application layer compared to the link layer [65]. However, the additional complexity of application layer encryption adds an additional layer of security if combined with link layer encryption. Good cryptographic practices, such as a true random number generator and non-proprietary encryption algorithms, are essential for manufacturer implementation. Application layer encryption protects against plaintext password, password obfuscation, brute forcing and fuzzing.

4.5.3 Two-way Authentication.

Two-way authentication protects against the rogue device attack by forcing the user and device to not immediately trust the connection. This method does not use the included link layer encryption. Instead, a public/private key model is used between the devices to authenticate. For example, the user's public key is used by the lock to encrypt a nonce ($N1$) and sends it to the user. At the same time, the lock sends a plaintext nonce ($N2$) to the user. The user decrypts $N1$ with their private key and encrypts $N2$ with the locks public key. Next, the user replies to the

lock with the decrypted $N1$ and encrypted $N2$. Using an asynchronous encryption method ensures that both user and lock have a public and private key and prevents a rogue device from impersonating a device. A rogue device would not successfully attack a connection without the private key of one of the devices and the public key of the other. Two-way authentication protects against plaintext password, password obfuscation, brute forcing, fuzzing, and MitM attacks.

4.5.4 Geofencing.

Geofencing protects users from unauthorized access by requiring a user to be within a specific distance of a GPS coordinate to request credentials from a web server. A virtual fence is created around a device, where a user must be within a set distance (usually a few feet) to gain access. Geofencing prevents cloned devices from exploiting users into providing credentials. Geofencing is best combined with other mitigation techniques due to offering no additional protection against eavesdroppers while the user is within the geofence perimeter.

An example of this can be found on the consumer BLE lock, August Lock. This device implements a geofence around the device and prevents unauthorized access to occur unless the user is within this perimeter. Some vulnerabilities do still exist, such as replacing the firmware with malicious code or gaining access to developer-only features that were supposed to be removed before the application was released [66]. Geofencing only protects against rogue device and relay attacks.

4.5.5 BLE-Guardian.

BLE-Guardian is a system that protects the user privacy by controlling who can discover, scan and connect to a device through an administrator [1]. The biggest avenue for attack that BLE-Guardian defends against is advertisements. Most MitM

attacks take advantage of advertisement packets that do not provide privacy and security protection. However, BLE-Guardian controls a device's advertisement packets through reactive jamming and manages the connection request approval process. These two processes protect the system because the true advertisement packets are shielded from passive eavesdropping and is demonstrated in Figure 22.

Implementation of BLE-Guardian requires an Ubertooth One in addition to the device it is protecting. An advantage of this approach is that it actively prevents attacks on a device compared to the standard encryption approach. This implementation would not require any additional work from the manufacturers of the devices

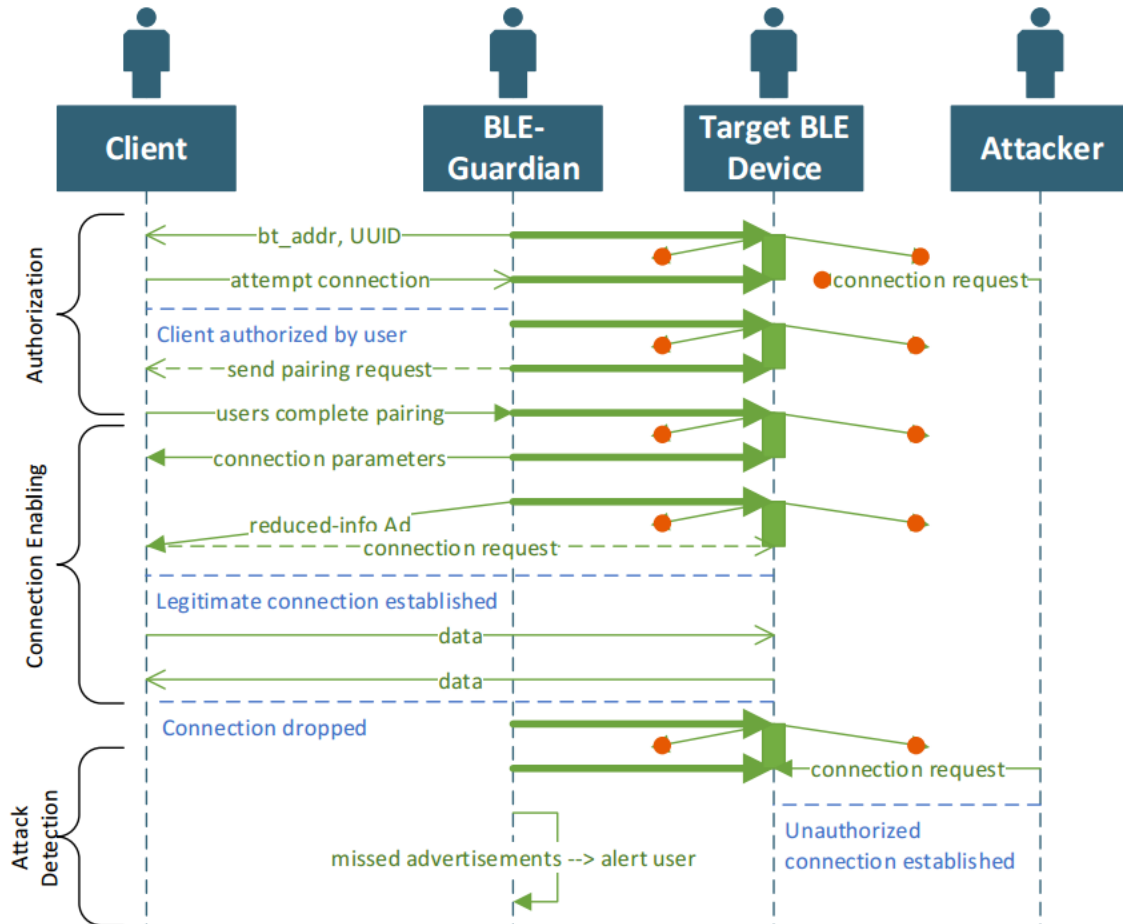


Figure 22. Sequence diagram of BLE-Guardian defending a target from an attacking device [1].

and can be used in conjunction with other techniques to increase security. The downside of this approach is that it requires additional hardware from the user which may not be practical in its current state. BLE-Guardian protects against MitM, rogue device, and relay attack vulnerabilities.

4.6 Summary

The purpose of this research is improve security of BLE security systems in critical infrastructure. Findings revealed 13 of 17 security devices contained one or more vulnerabilities which can be mitigated using various solutions described herein. Current countermeasures to BLE attacks already exist and require minimum implementation and development by manufacturers. The existing pairing and bonding feature that BLE has implemented provides enough security to eliminate the basic attacks discussed herein. While some sophisticated mitigation techniques are more intensive to implement, they offer additional protection against advanced attacks. An alternative method for protecting assets by shielding device activity exists with BLE-Guardian, but requires a high amount of user implementation and maintenance.

This chapter accomplishes the remaining goals set for this thesis. Goal (2) is met by identifying 8 vulnerabilities across 13 BLE devices. Goal (3) is met by demonstrating that compromising a BLE devices yields information used to develop pattern of human behavior through UBA. Finally, the last goal (4) is met by proposing five mitigation techniques to eliminate the potential threats proposed in this thesis.

V. Conclusion and Recommendations

5.1 Research Conclusion

The research question that this thesis answers is, can Bluetooth Low Energy (BLE) access control devices be exploited and/or protected? The hypothesis stated that Bluetooth Low Energy (BLE) access control devices can be exploited and protected by evaluating their wireless security and developing exploits and mitigation solutions, which was proven throughout this thesis. This research reveals that BLE security can be compromised through the completion of the following four goals established to validate the hypothesis.

5.1.1 Goal 1: Develop methods for detecting the location of BLE and Bluetooth Classic (BTC) devices.

Chapter 3 adapts the log-distance path loss models to successfully determine the range of both BLE and BTC devices. A Bluetooth packet parser and assembler is constructed to accomplish this goal. These tools paved the way for the development of BlueFinder which can determine the location of a Bluetooth device at ranges from 1 to 1,000 meters with less than 20% error. BlueFinder outperformed another tool, Blue Hydra, with error rates being 75% less.

5.1.2 Goal 2: Identify vulnerabilities in BLE devices and develop exploits that result in physical effects.

Chapter 4 evaluates 17 BLE devices and reveals eight distinct vulnerabilities across 13 different devices. This research accomplishes this goal by developing exploits for the discovered vulnerabilities on BLE access control devices. The physical effect that is accomplished is rendering security through BLE locks void.

5.1.3 Goal 3: Utilize vulnerabilities for unauthorized BLE device access and develop methods for extracting user data and generating User Behavior Analytics (UBA).

Chapter 4 expands on the previous goal by utilizing comprised BLE devices. BLE data collected from devices is successfully extracted from compromised devices. UBA extracts patterns of human behavior that reveals potential security gaps within an organization. This type of analysis demonstrates the effectiveness of offensive UBA.

5.1.4 Goal 4: Propose mitigation solutions for the determined BLE vulnerabilities.

Chapter 4 proposes five mitigation techniques that manufacturers and consumers can implement to counter existing exploits. These solutions already exist and require increasing manufacturer and user knowledge to motivate change. This thesis provides the required knowledge to drive this change.

5.2 Current State

Previously no other research has identified vulnerabilities facing Bluetooth access control devices. This thesis successfully develops exploits for commercial Bluetooth products and mitigation solutions for BLE vulnerabilities. In addition, this research successfully develops a tool for determining the location of BTC and BLE devices. During the release of the tool at DEF CON 24 [15], another war-walking tool, Blue Hydra [12], was released. The assessment of the tool, BlueFinder, against Blue Hydra determined that BlueFinder utilizes an overall more accurate model.

5.3 Research Contributions

The contributions of this thesis are numerous to the Bluetooth community. First, the development of the first long-range Bluetooth range-finding tool, BlueFinder, and proposed improvements to allow for Blue Hydra to increase its accuracy through an updated model [13]. Second, this work develops eight distinct exploits for 13 BLE access control devices. These exploits were promptly disclosed to the manufacturers and consumers. This information was presented to a crowd of more than 2,000 people and appeared on more than 100 news articles to ensure the public properly understood the risk [15]. Most if not all of the proposed exploits from this research are preventable through the suggested mitigation techniques.

5.4 Recommendations for Future Work

Many research opportunities exist within BLE that can still be investigated. Future work that researchers can complete and expand upon is outlined in the list below.

1. The latest version of BLE addresses major issues identified in earlier versions of the protocol. Most devices have yet to implement the newest version, Bluetooth version 4.2, which allowed the research to focus on version 4.1 devices. Ideally, future research would evaluate BLE devices implementing the newest version and determine if the updated security measures prevent previously identified vulnerabilities.
2. The first step of follow-on research is evaluating newly released BLE devices. The investigation into commercially-available devices such as Lockitron, Schlage NDE Wireless Lock, Haven Connect, Toor, and Sesame would increase understanding of existing BLE vulnerabilities significantly. In addition, Hilton hotels

have recently purchased Onity DirectKey to create a digital key and check-in system. The evaluation of this system allows for determining its usability and security.

3. Compromising firmware is another good follow-on to this research. BLE devices have the ability to utilize over-the-air updates which can be captured. Due to the availability of sniffers, the replication of an update would be an easy next step. Allowing for a device to be compromised with malicious firmware.
4. Early on, this research explored the concept of building a warflying platform using a commercial drone and tie into its on-board GPS to plot locations of BLE devices. An early concept for this research was considered, but was not pursued, to build a warflying platform for determining the location of BLE devices. This platform would use a device, such as Ubertooth, to locate BLE devices and tie into the on-board GPS to plot their locations. Similar projects have been completed for WiFi network detection, but this would likely be the first Bluetooth warflying project.
5. Two Man-in-the-Middle (MitM) tools were released during this thesis: BTLE-juice and Gattacker. Both implement numerous functions, however, they both left out a key feature that would allow for cloning devices. This ability can be used to save a device's characteristics for later spoofing. This allows a rogue device attack to use any profile at any time and have the option of saving multiple profiles for later use. The future work that can be completed is implementing a profile system for saving device settings. This can be implemented within existing tools as added functionality or as an entirely new platform.

5.5 Concluding Thoughts

The development of the range-finding tool, BlueFinder, was successful in this research effort. In addition, this research discovers multiple security vulnerabilities for Bluetooth devices. The four goals that were met during this thesis are:

1. Develop methods for detecting the location of BLE and BTC devices.
2. Identify vulnerabilities in BLE devices and develop exploits that result in physical effects.
3. Utilize vulnerabilities for unauthorized BLE device access and develop methods for extracting user data and generating UBA.
4. Propose mitigation solutions for the determined BLE vulnerabilities.

This thesis accomplishes goal (1) by creating a distance estimation model and integrating the model into BlueFinder for determining Bluetooth device location. Evaluating numerous devices and discovering 13 vulnerable devices met the requirements for goal (2). Goal (3) is accomplished by extracting system logs from Bluetooth locks and conducting analysis on the information. User behavior analytics was adapted to discover peak times of user behavior and assist in exposing security risks associated with storing user data. Finally, goal (4) is met by identifying five mitigation techniques and proposing them to counter existing vulnerabilities.

Appendix A. Packet Assembler

```
#!/bin/python

import time
from scapy.all import *
import os
import sys
import code
import argparse
from threading import Thread
import gevent
from binascii import unhexlify
import subprocess

ACL_packet = "4c00"

# Create Connection with default parameters
def Connect(BT_conn, addr, random):
    HCI_packet_type = "01"
    createleconn = "0D20"
    param_length = "19"
    scan_interval = "6000"
    scan_window = "3000"
    init_filter = "00"
    peer_addr = random
    BD_addr = addr
    own_addr = "00"
    conn_interval_min = "2800"
    conn_interval_max = "3800"
    conn_latency = "0000"
    supv_timeout = "2A00"
    min_CE = "0000"
    max_CE = "0000"

    raw = HCI_packet_type + createleconn + param_length +
    scan_interval + scan_window + init_filter + peer_addr +
    BD_addr + own_addr + conn_interval_min + conn_interval_max +
    conn_latency + supv_timeout + min_CE + max_CE

    print("Connected")
    raw = bytearray.fromhex(raw)
    BT_conn.send(raw)
    time.sleep(.5)
    return BT_conn

# Create Connection with specific parameters
def Cust_Connect(BT_conn, addr, random, scan_int, scan_win, own_address, conn_int_min,
conn_int_max, supv_time):
    HCI_packet_type = "01"
    createleconn = "0D20"
    param_length = "19"
    scan_interval = scan_int
    scan_window = scan_win
    init_filter = "00"
    peer_addr = random
    BD_addr = addr
    own_addr = own_address
    conn_interval_min = conn_int_min
    conn_interval_max = conn_int_max
    conn_latency = "0000"
    supv_timeout = supv_time
    min_CE = "0000"
    max_CE = "0000"
```

```

    raw = HCI_packet_type + createleconn + param_length + scan_interval +
    scan_window + init_filter + peer_addr + BD_addr + own_addr +
    conn_interval_min + conn_interval_max + conn_latency + supv_timeout +
    min_CE + max_CE

    print("Connected")

    raw = bytearray.fromhex(raw)
    BT_conn.send(raw)
    time.sleep(.3)
    return BT_conn

# Write Request
def writereq(s, handle, value):
    # ACL Packet header
    HCI_packet_type = "02"
    data_length = ""
    L2CAP_length = ""
    L2CAP_prot = "0400"
    opcode = "12"

    # Set L2CAP and Data lengths
    L2CAP_l = (int(len(value)) + 6)/2

    if len(value) < 18:
        data_length = "0" + str2hex((len(value) + 14)/2) + "00"
        L2CAP_length = "0" + str2hex(L2CAP_l) + "00"

    elif 18 <= len(value) < 50:
        data_length = str2hex((len(value) + 14)/2) + "00"
        L2CAP_length = "0" + str2hex(L2CAP_l) + "00"
        if len(value) > 26:
            L2CAP_length = str2hex(L2CAP_l) + "00"

    elif 50 < len(value):
        data_length = str2hex((len(value) + 14)/2) + "0"
        L2CAP_length = str2hex(L2CAP_l) + "00"
        if 58 < len(value):
            L2CAP_length = str2hex(L2CAP_l) + "0"

    else:
        data_length = "0" + str2hex((len(value) + 14)/2) + "00"
        L2CAP_length = "0" + str2hex(L2CAP_l) + "00"

    raw = HCI_packet_type + ACL_packet + data_length + L2CAP_length +
    L2CAP_prot + opcode + handle + value

    print("Writing_" + value + "_to_handle:_ " + handle)
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

# convert integer to string(hex) in correct format (0x0001 -> 01)
def str2hex(value):
    hexvalue = str('{0:x}'.format(int(value)))
    return hexvalue

# Disconnect
def disconnect(s):
    raw = "010604034C0013"
    raw = bytearray.fromhex(raw)
    s.send(raw)
    print("Disconnected")

def prepwrite(s, offset, handle, value):

```

```

# ACL Packet header
HCI_packet_type = "02"
data_length = ""
L2CAP_length = ""
L2CAP_prot = "0400"
opcode = "16"

# Set L2CAP and Data lengths (needs implementation like write req)
L2CAP_l = (int(len(value)) + 10)/2

if len(value) > 14:
    data_length = str2hex((len(value) + 18)/2) + "00"
    if len(value) > 22:
        L2CAP_length = str2hex(L2CAP_l) + "00"
else:
    data_length = "0" + str2hex((len(value) + 18)/2) + "00"
    L2CAP_length = "0" + str2hex(L2CAP_l) + "00"

raw = HCI_packet_type + ACL_packet + data_length + L2CAP_length + L2CAP_prot +
opcode + handle + offset + value
print(raw)
raw = bytearray.fromhex(raw)
s.send(raw)
time.sleep(.3)

def execwrite(s):
    # ACL Packet header
    HCI_packet_type = "02"
    data_length = "0600"
    L2CAP_length = "0200"
    L2CAP_prot = "0400"
    opcode = "18"
    value = "01"

    raw = HCI_packet_type + ACL_packet + data_length + L2CAP_length + L2CAP_prot +
opcode + value
    print(raw)
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

def bindsock():
    output = subprocess.check_output(['bash', '-c', "hciconfig hci0 down"])
    BT_conn = BluetoothUserSocket()
    return BT_conn
    time.sleep(.5)

def spoof(BT_conn, BD_addr):
    raw = bytearray.fromhex("01011000")
    BT_conn.send(raw)

    change_addr = "0101fc06" + BD_addr
    raw = bytearray.fromhex(change_addr)
    BT_conn.send(raw)

    raw = bytearray.fromhex("01030c00")
    BT_conn.send(raw)
    time.sleep(.3)

def switchrole(BT_conn, BD_addr):
    role = "01"

    change_role = "010b0807" + BD_addr + role
    raw = bytearray.fromhex(change_role)
    BT_conn.send(raw)
    time.sleep(.3)

```

```

def exMTUreq(BT_conn, size):
    # change MTU
    MIU = "024C0007000300040002" + size
    raw = bytearray.fromhex(MIU)
    BT_conn.send(raw)
    time.sleep(.3)

def conn_oriented_chan(BT_conn, data):
    HCI_packet_type = "02"
    data_length = "1b00"
    L2CAP_length = "1700"
    L2CAP_prot = "0401"

    raw = HCI_packet_type + ACL_packet + data_length + L2CAP_length + L2CAP_prot + data
    print(raw)
    raw = bytearray.fromhex(raw)
    BT_conn.send(raw)
    time.sleep(.3)

def readreq(BT_conn, handle):
    read = "024C000700030004000a" + handle
    raw = bytearray.fromhex(read)
    BT_conn.send(raw)
    time.sleep(.3)

    # Write Request
def writecmd(s, handle, value):
    # ACL Packet header
    HCI_packet_type = "02"
    data_length = ""
    L2CAP_length = ""
    L2CAP_prot = "0400"
    opcode = "52"

    # Set L2CAP and Data lengths
    L2CAP_l = (int(len(value)) + 6)/2

    if len(value) < 18:
        data_length = "0" + str2hex((len(value) + 14)/2) + "00"
        L2CAP_length = "0" + str2hex(L2CAP_l) + "00"

    elif 18 <= len(value) < 50:
        data_length = str2hex((len(value) + 14)/2) + "00"
        L2CAP_length = "0" + str2hex(L2CAP_l) + "00"
        if len(value) > 26:
            L2CAP_length = str2hex(L2CAP_l) + "00"

    elif 50 < len(value):
        data_length = str2hex((len(value) + 14)/2) + "0"
        L2CAP_length = str2hex(L2CAP_l) + "00"
        if 58 < len(value):
            L2CAP_length = str2hex(L2CAP_l) + "0"

    else:
        data_length = "0" + str2hex((len(value) + 14)/2) + "00"
        L2CAP_length = "0" + str2hex(L2CAP_l) + "00"

    raw = HCI_packet_type + ACL_packet + data_length + L2CAP_length + L2CAP_prot +
opcode + handle + value
    print("Writing_" + value + "_to_handle:" + handle)
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

def remote_features(s):

```

```

raw = "011b04020b00"
raw = bytearray.fromhex(raw)
s.send(raw)
time.sleep(.1)

def ext_features(s):
    raw = "011c04030b0001"
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.1)

def remote_name(s):
    raw = "0119040a8a34ec1a334002000000"
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.1)

def info_ext_feature(s):
    raw = "020b000a00060001000a0102000200"
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.1)

def info_fixed_chan(s):
    raw = "020b000a00060001000a0202000300"
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.1)

def LEsetscan(s, scan_enable, filter_dub):
    HCI_packet_type = "01"
    set_scan = "0c20"
    param_length = "02"
    scan_enable = scan_enable
    filter_dub = filter_dub

    raw = HCI_packet_type + set_scan + param_length + scan_enable + filter_dub
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

def LEsetscanparam(s, scan_type):
    HCI_packet_type = "01"
    scan_param = "0b20"
    param_length = "07"
    scan_type = scan_type
    scan_interval = "1000"
    scan_window = "1000"
    own_addr = "00"
    scan_filter = "00"

    raw = HCI_packet_type + scan_param + param_length + scan_type + scan_interval +
scan_window + own_addr + scan_filter
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

```

Appendix B. BlueFinder

```
#!/bin/python

import time
import binascii
import os
import threading
import signal
from scapy.all import *

def main():
    user_input = [None]
    s = bindsocket()
    LEsetscan(s, "00", "00")

    mythread = threading.Thread(target=get_user_input, args=(user_input,))
    mythread.daemon = True
    mythread.start()

    if len(sys.argv) == 1:
        print '\n' + ".....BlueFinder_v1.3a"
        print ".....python_BlueFinder.py<sel><-l><Ant><BD_Addr>"
        print '\n\n' + ".....DESCRIPTION"
        print ".....<sel>.....c-Classic, -b-Low-Energy"
        print ".....<-l>.....Long-range_mode_for_use_with_UD-100(optional)"
        print ".....<Ant>.....Antenna_Gain_[3,-9,-15].....(optional)"
        print ".....<BD_Addr>.....Bluetooth_Device_Address"

    elif len(sys.argv) == 3:
        if sys.argv[1] == "-b":
            btle(s, sys.argv[2], user_input, -60)
        elif sys.argv[1] == "-c":
            classic(s, user_input)
        else:
            print("Error:_invalid_selection")

    elif len(sys.argv) == 5:
        if sys.argv[3] == "3":
            btle(s, sys.argv[4], user_input, -26)
        elif sys.argv[3] == "9":
            btle(s, sys.argv[4], user_input, -24)
        elif sys.argv[3] == "15":
            btle(s, sys.argv[4], user_input, -22)
        else:
            print("Error:_invalid_gain")

    else:
        print("Error:_invalid_input")

    disconnect(s)
    sys.exit()
    return(s)

def get_user_input(user_input_ref):
    user_input_ref[0] = raw_input()

def classic(s, user_input):
    BD_Addr = sys.argv[2]
    BD_Addr = BD_Addr.replace(":", "")
    BD_Addr = bytearray.fromhex(BD_Addr)
    BD_Addr.reverse()
    BD_Addr = binascii.hexlify(BD_Addr)
```

```

clas_connect(s, BD_Addr)
time.sleep(2)

while True:
    data = None
    data = s.recv()
    data = str(data).encode("hex")
    if data[0:4] == "0403":
        code = data[8:12]
        break

while not keystop():
    try:
        RSSI = []
        while len(RSSI) < 100:
            data = s.recv()
            data = repr(data)

            raw = bytearray.fromhex("02" + code + "34003000010008d62c0041"
42434445464748494a4b4c4d4e4f505152535455565758595a5b5c5d5e5f60616263646566676841424344")
            s.send(raw)
            time.sleep(.05)
            rssi(s)

            for line in data.splitlines():
                for part in line.split():
                    if "load=\\x0b" in part:
                        new = part[10:18]
                        new = new.replace("\\x", "")
                        RSSI.append(s16(int(("0x"+new),16)))

        dist(RSSI, 0)
    except KeyboardInterrupt:
        print "\nExiting_BlueFinder"
        LEsetscan(s, "00", "00")
        disconnect(s)
        sys.exit()
        return()

def s16(value):
    return -(value & 0x80) | (value & 0x7f)

# btle
def btle(s, input_addr, user_input, int_RSSI):
    LEsetscanparam(s, "01")
    LEsetscan(s, "01", "00")

    while not keystop():
        try:
            RSSI = []
            BD_Addr = []
            while len(RSSI) < 5:
                data = s.recv()
                data = repr(data)

                for line in data.splitlines():
                    for part in line.split():
                        if "addr=" in part:
                            BD_Addr.append(part[5:22])
                        if "load=" in part:
                            RSSI.append(s16(int(part[8:10],16)))
                for i, item in enumerate(BD_Addr):
                    if item.lower() != input_addr.lower():
                        del RSSI[i]
                        del BD_Addr[i]

```

```

        dist(RSSI, int_RSSI)

    except KeyboardInterrupt:
        print "\nExiting_BlueFinder"
        LEsetscan(s, "00", "00")
        disconnect(s)
        sys.exit()
        return()

def dist(RSSI, int_RSSI):
    distance=[]
    RSSI=map(float, RSSI)
    for i in RSSI[0:]:
        #printtofile(i) # enable if recording RSSI
        distance.append(round(10*((int_RSSI - i)/20),1))
    avgdistance = round(sum(distance) / float(len(distance)),1)
    print "%r_m" % (avgdistance)

def clas_connect(BT_conn, addr):
    HCI_packet_type = "01"
    opcode = "0504"
    param_length = "0d"
    addr = addr
    packet_type = "18cc"
    page_scan_rep = "02"
    page_scan = "00"
    clock_offset = "0000"
    role = "01"

    raw = HCI_packet_type + opcode + param_length + addr + packet_type +
    page_scan_rep + page_scan + clock_offset + role
    #print raw
    raw = bytearray.fromhex(raw)
    BT_conn.send(raw)
    time.sleep(.5)

def LEsetscan(s, scan_enable, filter_dub):
    HCI_packet_type = "01"
    set_scan = "0c20"
    param_length = "02"
    scan_enable = scan_enable
    filter_dub = filter_dub

    raw = HCI_packet_type + set_scan + param_length + scan_enable + filter_dub
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

def LEsetscanparam(s, scan_type):
    HCI_packet_type = "01"
    scan_param = "0b20"
    param_length = "07"
    scan_type = scan_type
    scan_interval = "1000"
    scan_window = "1000"
    own_addr = "00"
    scan_filter = "00"

    raw = HCI_packet_type + scan_param + param_length + scan_type + scan_interval +
    scan_window + own_addr + scan_filter
    raw = bytearray.fromhex(raw)
    s.send(raw)
    time.sleep(.3)

def bindsock():

```

```

        output = subprocess.check_output(['bash','-c', "hciconfig_hci0_down"])
        BT_conn = BluetoothUserSocket()
        return BT_conn
        time.sleep(.5)

def rssi(s):
    ran = bytearray.fromhex("010514020b00")
    s.send(ran)

def disconnect(s):
    raw = "01060403400013"
    raw = bytearray.fromhex(raw)
    s.send(raw)

def printtofile(RSSI):
    f = open('lm.txt','a')
    f.write(str(RSSI) + '_')
    f.close()

def keystop(delay = 0):
    return len(select.select([sys.stdin], [], [], delay)[0])

if __name__ == "__main__":
    while not keystop():
        try:
            main()
        except KeyboardInterrupt:
            print "\n_Exiting_BlueFinder"
        except IndexError:
            print "Error:_Index"
        finally:
            sys.exit()

```

Bibliography

1. K. Fawaz, K.-H. Kim, and K. G. Shin, "Protecting privacy of ble device users," *Proceedings of the Twenty-Fifth USENIX Security Symposium*, pp. 1205–1221, 2016.
2. G. Press. (2014) Internet of Things By The Numbers: Market Estimates And Forecasts. Forbes. [Online]. Available: <http://www.forbes.com/sites/gilpress/2014/08/22/internet-of-things-by-the-numbers-market-estimates-and-forecasts/>
3. Bluetooth Special Interest Group. (2016) Bluetooth 5 Quadruples Range, Doubles Speed, Increases Data Broadcasting Capacity by 800%. [Online]. Available: <https://www.bluetooth.com/news/pressreleases/2016/06/16/-bluetooth5-quadruples-rangedo-bles-speedincreases-data-broadcasting-capacity>
4. National Security Telecommunications Advisory Committee, "NSTAC Report to the President on Secure Government Communications," 2013.
5. B. Ramsey, B. Mullins, and E. White, "Improved Tools for Indoor ZigBee Warwalking," *Proceedings - Conference on Local Computer Networks (LCN)*, pp. 921–924, 2012.
6. J. Meserve. (2007) US Sources: Staged cyber attack reveals vulnerability in power grid. CNN. [Online]. Available: <http://www.cnn.com/2007/US/09/26/power.at.risk/index.html>
7. Corporation Reliability and Critical Infrastructure, "Advanced solutions for critical infrastructure protection," pp. 1–18, 2012.
8. Onity. (2016) Electronic Locks. [Online]. Available: <http://en.onity.com/products/Pages/Electronic-Locks.aspx>
9. US Department of Defense, "The DoD Cyber Strategy," 2015. [Online]. Available: https://www.defense.gov/Portals/1/features/2015/0415_cyber-strategy/Final_2015_DoD_CYBER_STRATEGY_for_web.pdf
10. K. Haataja, K. Hyppnen, S. Pasanen, and P. Toivanen, *Bluetooth Security Attacks: Comparative Analysis, Attacks, and Countermeasures*. Berlin Heidelberg, Germany: Springer-Verlag, 2013.
11. M. Ryan, "Bluetooth: With Low Energy Comes Low Security," *Proceedings of the 7th USENIX Conference on Offensive Technologies*, pp. 1–7, 2013.
12. Pwnie Express. (2016) Wired and Wireless Device Threat Detection. [Online]. Available: <https://www.pwnieexpress.com/>

13. A. Rose, J. G. D. Arroyo, J. Bindewald, and B. Ramsey, "Bluefinder: A rangefinding tool for bluetooth classic and low energy." International Conference on Cyber Warfare and Security, 2017, (to appear).
14. A. Rose, J. Bindewald, B. Ramsey, R. Rice, and B. Mullins, "Securing ble-enabled locks against unauthorized access and surveillance," *International Journal for Critical Infrastructure Protection*, 2017, (submitted).
15. A. Rose and B. Ramsey, "Picking bluetooth low energy locks from a quarter mile away." presented at DEF CON 24, 2016.
16. A. Rose and B. Ramsey, "Picking bluetooth low energy locks from a quarter mile away." presented at IoT Village, 2016.
17. A. Rose, "All your lock are BLEong to us," presented at RSA Conference, 2017, (to appear).
18. Bluetooth Special Interest Group. (2014) Specification of the bluetooth system covered core package version 4.2. [Online]. Available: www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=286439&_ga=1.67631506.289442471.1481896385
19. K. Scarfone, J. Padgett, and L. Chen, "Guide to Bluetooth Security, NIST Special Publication 800-121 (Revision 1)," National Institute of Standards and Technology, Gaithersburg, Maryland, 2012.
20. S. Raza, P. Misra, Z. He, and T. Voigt, "Bluetooth smart: An enabling technology for the internet of things." 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2015, pp. 155–162.
21. R. Heydon, *Bluetooth Low Energy: The Developer's Handbook*. Upper Saddle River, New Jersey: Prentice Hall, 2013.
22. R. Heydon and N. Hunn, "Bluetooth Low Energy," *Bluetooth Special Interest Group*, 2012. [Online]. Available: http://www.cl.cam.ac.uk/teaching/1314/TopIssues/Files/L4_BLE.pdf
23. Kali Linux. (2016) Kali Linux Tools Listing. [Online]. Available: <https://www.kali.org/>
24. X. Pan, Z. Ling, A. Pingley, W. Yu, N. Zhang, K. Ren, and X. Fu, "Password Extraction via Reconstructed Wireless Mouse Trajectory," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 4, pp. 461–473, 2015.
25. J. Huang, W. Albazrqaoe, and G. Xing, "BlueID: A practical system for Bluetooth device identification." Proceedings of the 33rd IEEE International Conference on Computer Communications, 2014, pp. 2849–2857.

26. A. Takanen, "Fuzzing: The Past, the Present and the Future," *Symposium on Information and Communications Technology Security*, pp. 201–211, 2009.
27. K. Haataja and P. Toivanen, "Two Practical Man-in-the-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasures," *IEEE Transactions on Wireless Communications*, vol. 9, no. 1, pp. 384–392, 2010.
28. D. Cauquil, "BTLEJuice Bluetooth Smart (LE) Man-in-the-Middle Framework," 2016. [Online]. Available: <https://github.com/DigitalSecurity/btlejuice>
29. S. Jasek, "Gattacking Bluetooth Smart Devices." BlackHat, 2016, pp. 1–41.
30. Z. Aung and W. Zaw, "Permission-Based Android Malware Detection," *International Journal of Scientific & Technology Research*, vol. 2, no. 3, pp. 228–234, 2013. [Online]. Available: www.ijstr.org
31. J. Guitierrez Del Arroyo and B. Ramsey, "Securing bluetooth low energy enabled industrial monitors," 2017, (to appear).
32. J. Guitierrez Del Arroyo and B. Ramsey, "How Do I BLE Hacking?" presented at Wireless Village - DEF CON 24, 2016.
33. M. Botta and M. Simek, "Adaptive Distance Estimation Based on RSSI in 802.15.4 Network," *Radioengineering*, vol. 22, no. 4, pp. 1162–1168, 2013.
34. G. Ionescu, C. de la Osa, and M. Deriaz, "Improving Distance Estimation in Object Localization with Bluetooth Low Energy," *SENSORCOMM 2014: The Eighth International Conference of Sensor Technologies and Applications*, pp. 45–49, 2014.
35. S. Zhou and J. K. Pollard, "Position Measurement using Bluetooth," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 2, pp. 555–558, 2006.
36. J. Hall and B. Ramsey, "Breaking Bulbs Briskly by Bogus Broadcasts," presented at ShmooCon, 2015.
37. M. Caleb and B. Ramsey, "InsteON, InsteOFF, InsteOpen." presented at Wireless Village - DEF CON 24, 2016.
38. Consumer Technology Association. (2016) IoT is Getting Much Smarter. [Online]. Available: <http://www.cta.tech/i3/Features/2016/March-April/IoT-is-Getting-Much-Smarter>
39. The Freedonia Group. (2012) Implantable Medical Devices. [Online]. Available: <http://www.freedoniagroup.com/industry-study/2852/implantable-medical-devices.htm>

40. M. Goodman, "Who Does the Autopsy? Criminal Implications of Implantable Medical Devices," *Security*, pp. 1–4, 2015.
41. US Department of Homeland Security Industrial Control Systems Cyber Emergency Response Team. (2013) Medical Devices Hard-Coded Passwords. [Online]. Available: <http://www.sans.edu/research/security-laboratory/article/bluetooth>
42. R. Baldwin. (2016) Researcher finds huge security flaws in Bluetooth locks. Engadget. San Francisco, California. [Online]. Available: <https://www.engadget.com/2016/08/10/researcher-finds-huge-security-flaws-in-bluetooth-locks/>
43. L. Pei, R. Chen, J. Liu, H. Kuusniemi, T. Tenhunen, and Y. Chen, "Using Inquiry-based Bluetooth RSSI Probability Distributions for Indoor Positioning," *Journal of Global Positioning Systems*, vol. 9, no. 2, pp. 122–130, 2010.
44. L. Pei, R. Chen, J. Liu, T. Tenhunen, H. Kuusniemi, and Y. Chen, "Inquiry-based Bluetooth Indoor Positioning via RSSI Probability Distributions," *2nd International Conference on Advances in Satellite and Space Communications (SPACOMM)*, pp. 151–156, 2010.
45. M. Altini, D. Brunelli, E. Farella, and L. Benini, "Bluetooth Indoor Localization with Multiple Neural Networks," *IEEE 5th International Symposium on Wireless Pervasive Computing*, pp. 295–300, 2010.
46. R. Faragher and R. Harle, "An Analysis of the Accuracy of Bluetooth Low Energy for Indoor Positioning Applications," *Proceedings of the 27th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+)*, pp. 201–210, 2014.
47. B. Ramsey, B. Mullins, W. Lowder, and R. Speers, "Sharpening the Stinger: Tuning KillerBee for Critical Infrastructure Warwalking," *Proceedings - IEEE Military Communications Conference (MILCOM)*, pp. 104–109, 2014.
48. A. Neskovic, N. Neskovic, and G. Paunovic, "Modern Approaches in Modeling of Mobile Radio Systems Propagation Environment," *IEEE Communications Surveys & Tutorials*, vol. 3, no. 3, pp. 2–12, 2000.
49. S. Liu and A. Striegel, "Accurate Extraction of Face-to-Face Proximity using Smartphones and Bluetooth," *Proceedings - International Conference on Computer Communications and Networks*, pp. 1–5, 2011.
50. M. Petrova, J. Riihijarvi, P. Mahonen, and S. Laellla, "Performance Study of IEEE 802.15.4 Using Measurements and Simulations," *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 1, pp. 487–492, 2006.

51. A. Ghose, C. Bhaumik, and T. Chakravarty, "BlueEye," *1st International Workshop on Pervasive Urban Crowdsensing Architecture and Applications*, pp. 1135–1142, 2013.
52. J. Jung, D. Kang, and C. Bae, "Distance Estimation of Smart Device using Bluetooth," in *Proceedings of the 8th International Conference on Systems and Networks Communications (ICSNC)*, pp. 13–18, 2013.
53. J. Jung, D. Kang, J. Choi, and C. Bae, "D2D Distance Measurement using Kalman Filter Algorithm for Distance-based Service in an Office Environment," *International Conference on Advanced Communications Technology*, pp. 6–9, 2015.
54. T. O'Connor, *Violent Python: A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers*, 1st ed. Syngress Publishing, 2013.
55. Great Scott Gadgets. (2016) HackRF One. [Online]. Available: <https://greatscottgadgets.com/hackrf/>
56. M. Ossmann, "Project ubertooth: Building a better bluetooth adapter." presented at ShmooCon, 2011.
57. B. Ramsey, B. Mullins, R. Speers, and K. Batterton, "Watching for weakness in wild WPANs," *Proceedings - IEEE Military Communications Conference (MILCOM)*, pp. 1404–1409, 2013.
58. Safetech. (2016) SafeTechProducts.com - Products. [Online]. Available: <http://safetechproducts.com/products.php>
59. S. Boonkrong and C. Somboonpattanakit, "Dynamic Salt Generation and Placement for Secure Password Storing," *IAENG International Journal of Computer Science*, vol. 23, no. February, 2016.
60. N. Flor and H. Shannon, "Technology Corner: Brute Force Password Generation—Basic Iterative and Recursive Algorithms," vol. 6, no. 3. *Journal of Digital Forensics, Security, and Law*, 2011, pp. 79–86.
61. P. Oehlert, "Violating assumptions with fuzzing," vol. 3, no. 2. *IEEE Security and Privacy*, 2005, pp. 58–62.
62. C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World, Second Edition*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2002.
63. S. Gold, "Android a secure future at last," vol. 7, no. 2. *Engineering & Technology*, 2012, pp. 50–54.

64. K. Townsend, R. Davidson, and C. Cufi, *Getting Started with Bluetooth Low Energy*. Sebastopol, California: O'Reilly Media, 2014.
65. D. Herrmann, *Complete Guide to Security and Privacy Metrics: Measuring Regulatory Compliance, Operational Resilience, and ROI*, 1st ed. Boston, Massachusetts: Auerbach Publications, 2007.
66. Jmaxxz, "Backdooring the Front Door: Hacking a "Perfectly Secure" Smart Lock." presented at DEF CON 24, 2016.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 23-03-2017		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From — To) Sept 2015 — Mar 2017	
4. TITLE AND SUBTITLE Security Evaluation and Exploitation of Bluetooth Low Energy Devices				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Rose, Anthony J., Capt, USAF				5d. PROJECT NUMBER 17G310	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENG-MS-17-M-066	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Department of Homeland Security ICS-CERT POC: Neil Hershfield, DHS ICS-CERT Technical Lead ATTN: NPPD/CSC/NCSD/US-CERT Mailstop: 0635, 245 Murray Lane, SW, Bldg 410, Washington, DC 20528 Email: ics-cert@dhs.gov phone: 1-877-776-7585				10. SPONSOR/MONITOR'S ACRONYM(S) DHS ICS-CERT	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.					
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT Bluetooth is one of the most prevalent technologies for short-range wireless communications, and its widespread commercial use in security and medical devices makes it a prime target for penetration testers and cyber security experts. This thesis evaluates currently available Bluetooth penetration testing tools. The survey demonstrates that these tools have a limited range of capabilities and most tools are only compatible with a single protocol. This discovery motivates the development of BlueFinder, an open-source range-finding tool compatible with both BTC and BLE devices. Further evaluation reveals 75% of the tested security and access control systems had one or more vulnerabilities which resulted in unauthorized access. Malicious actors use these vulnerabilities to gain access to a facility and extract sensitive information, developing patterns of life on an organization from this information and putting a high amount of risk onto the consumer. This work presents existing solutions to avoid BLE security breaches with most fixes requiring minimal implementation overhead for developers.					
15. SUBJECT TERMS Bluetooth Low Energy, Access Control, User Behavior Analytics, Range-finding					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Barry E. Mullins, Ph.D., AFIT/ENG
U	U	U	U	101	19b. TELEPHONE NUMBER (include area code) (937) 255-3636, 7979; Barry.Mullins@afit.edu