SENSING VULNERABLE BLE DEVICES VIA CHIP MODEL FINGERPRINTING

by

KADRIYE TURKYILMAZ

(Under the Direction of Le Guan)

ABSTRACT

Bluetooth Low Energy (BLE) is a popular wireless communication technology introduced in Bluetooth 4.0. Devices that leverage this technology include home security systems, medical devices, and similar apparatuses whose integrity is crucial for people's safety. A vulnerable BLE device can cause user privacy leaks and be a steppingstone toward more severe threats in enterprise environments. This study proposes a BLE sniffer for the enterprise environment to raise an alarm whenever a nearby vulnerable device is detected. The system senses vulnerable devices by sniffing advertising packets and fingerprinting device chip models. Six fitness-tracker device data were used to train two machine learning algorithms targeted to predict chip models based on advertising packet timing and size information inherent to the chip hardware. The results show that detecting vulnerable devices and identifying their specific vulnerabilities based on CVE records is possible, according to the 87.5% accuracy score.

INDEX WORDS: Bluetooth Low Energy, Chip model fingerprinting, Packet sniffing, Machine learning

SENSING VULNERABLE BLE DEVICES VIA CHIP MODEL FINGERPRINTING

by

KADRIYE TURKYILMAZ

B.S., Bilecik Seyh Edebali University, Turkey, 2016

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2022

SENSING VULNERABLE BLE DEVICES VIA CHIP MODEL FINGERPRINTING

by

KADRIYE TURKYILMAZ

Major Professor:	Le Guan
Committee:	Sheng Li
	Kyu Hyung Lee

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
May 2022

ACKNOWLEDGEMENTS

I would like to thank everyone who supported me academically during my master's program. Firstly, I would like to thank my Major Professor Le Guan who accepted me as a graduate student and guided me to complete my thesis. I would like to thank Sheng Li and Kyu Hyung Lee to be my committee members and support this study.

I would like to thank the Republic of Turkey to give me an opportunity to conduct my master's research. In addition, I would like to thank the General Directorate of State Hydraulic Works for supporting me during the pursuit of my master's degree.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

Bluetooth Low Energy (BLE) is a technology of dominant popularity among Micro Controller Unit (MCU) devices due to the small amount of energy it requires, the flexibility it provides, and the capability to work with an application in a smartphone or tablet. It was designed as a tool for very low power operation and provides over 40 channels in the 2.4GHz unlicensed ISM radio frequency band for data transmission. According to the Bluetooth SIG 2022 market update, even only single-mode Bluetooth LE devices are over five billion. Additionally, Bluetooth LE device shipments are forecasted to more than triple growth over the next five years (Bluetooth Technology Website, n.d.). The increase in popularity of this technology has also made it more desirable for attackers and data collectors. Devices that leverage this technology include but are not limited to home security systems, medical devices such as pacemakers, and similar apparatuses whose integrity is crucial for people's safety (Yu et al., 2012). Most devices also collect, hold, and transfer sensitive information. The rapid growth of the Internet of Things has paved the way for many other new applications and services. Internet-of-Things (IoT) devices extensively use BLE since it is an easy-to-deploy and cost-effective low-power wireless solution. However, the increase in BLE devices and the lack of the discovery of many device models would bring security and management challenges, such as vulnerabilities and misconfiguration (Zuo et al., 2019). Identifying the characteristics of BLE devices is an approach that helps detect potentially vulnerable devices in our daily lives. Fingerprinting is a technique for identifying these devices' operating systems (OS), applications,

or network services, which has been widely studied for more than 20 years (Zuo et al., 2019). On the other hand, there are many manufacturers, different device types, and product models, leading to a large number of fingerprints, and it is hard to keep the fingerprinting updated with the addition of new devices day by day. Furthermore, existing fingerprinting studies are generally based on device characteristics information. Zuo et al. (2019) developed an automatic fingerprinting system using the BLE device's static UUIDs (Universally Unique Identifier). However, there are defined UUIDs by the Bluetooth SIG, and multiple devices can use the same UUID with different chips.

Additionally, BLE devices are identified by a Bluetooth MAC address, and the MAC address is uniquely allocated to the device by the manufacturer. Therefore, it may seem like a very convenient piece of data in fingerprinting a device. Although devices with static MAC addresses, such as the Fitbit Inspire, are still available in the market, The BLE protocols force devices to randomize their public MAC address for user privacy and bypass to display of a unique identifier (Jouans et al., 2021). Using MAC address randomization makes MAC addresses unsuitable data to fingerprint a particular device. Moreover, BLE devices do not always prefer to share device names and manufacturer ID information since they are non-mandatory information to be shared in an advertising packet. Since mentioned device information is not accessible for every BLE device and the variety of devices is increasing day by day, a novel approach is required to detect nearby vulnerable devices. A vulnerable BLE device can cause user privacy leaks and even human health threats. They can be a steppingstone toward more serious threats in enterprise environments.

This study proposes a BLE sniffer for the enterprise environment to raise an alarm whenever a nearby vulnerable device is detected. The proposed system includes sniffing BLE

device advertising packets, fingerprinting the BLE chip model, unlike BLE device model fingerprinting, which is a widely used method, vulnerability search based on the device chip model, alerting authorities to the presence of vulnerable devices.

The approach of this proposed system is to fingerprint BLE device chip models, which have less variety, based on the timing information of the device advertising packets, instead of fingerprinting BLE devices whose ever-increasing variety. The fingerprint's timing information is due to the inherent hardware property and controller implementation preferences. It is also argued that focusing on Bluetooth chip models is a more determined approach since there is a limited number of chip vendors, including Nordic Semiconductor, Infineon Technologies, Texas Instruments, and the same chip vendor provides the same upstream SDK (Software Development Kit) for their products. Moreover, different devices could have the same chip model, and this is another idea that facilitates finding if a device is vulnerable based on the chip model. Based on this approach, this study captures advertising packets from nearby devices with the sniffing method and accesses the chip model information using a machine learning models (Decision tree and K-Nearest Neighbor) with the timing and size information extracted from the packets. This study shows that creating chip fingerprints based on timing information is possible, and the machine learning model trained with the time-related information may predict the specific chip model of a real device. Whether the devices whose chip model information is accessed with the machine learning method are vulnerable can be easily learned by querying the databases containing CVE records, such as the CVE list by MITRE Corporation (CVE, n.d.). CVE stands for Common Vulnerabilities and Exposures and is a database of publicly disclosed information security issues.

One of the essential future works is designing and implementing a more advanced machine learning model to identify device chip models. Additionally, the machine learning models can be trained to predict both chip model and firmware versions to detect vulnerable devices with 100% confidence. The other is integrating a database containing CVE records directly to the machine learning project could provide the automatic result of whether a device is vulnerable.

1.1. Motivation

BLE devices are widely deployed in many fields, including security-sensitive healthcare applications. However, they are particularly vulnerable. Especially BLE 4.1 and below are subject to MITM attacks, and it is not fixable (Padgette et al., 2017). Identifying BLE devices' characteristics could help us detect potentially vulnerable devices. For this purpose, many fingerprinting methods have been developed. Fingerprinting is an extensively studied technique for identifying various devices and systems such as computer operating systems (OS), applications, and network services (Yang et al., 2019). However, these methods are generally in the direction of creating device fingerprints by detecting the characteristics of the devices. Since the information such as the MAC address, services, characteristics, device name, and manufacturer ID used for fingerprinting the devices are not always accessible, the fingerprinting methods offered by the existing fingerprinting methods are not always effective enough. For this reason, reaching the chip model information of BLE devices using the packet timing and size information that can always be obtained from an active device's advertising packets will be a more effective method of detecting if a particular device is subject to any vulnerability.

To this end, after collecting the timing and size information of these devices' advertising packets and identifying device chip models using machine learning techniques, the CVEs

database should be queried with the obtained chip model information to see if a particular device is vulnerable. The main focus of this study is to fingerprint the chip models of devices from the captured raw BLE packets. The idea of developing a fingerprinting system for BLE chips using machine learning models will allow device vulnerabilities to be detected more decisively and easily.

1.2. Outline

This thesis consists of five chapters. The remaining thesis is structured as follows. After presenting the recent related studies in chapter two, chapter three includes the data and methodology used in this thesis. The third chapter explains how the data was collected and manipulated and how the machine learning implementation and database querying were performed. Chapter four presents and discusses the corresponding results and explains the vulnerabilities of chip models detected by machine learning models in this study's experimental work. Finally, the last chapter concludes the thesis, discusses the limitations of this study, and suggests future works on this topic.

CHAPTER 2

LITERATURE REVIEW

Numerous studies have been implemented to evaluate and improve BLE. These previous works include case studies to emphasize the importance of BLE device security, identify BLE devices using various methods and generate various vulnerabilities data. Fingerprinting and analyzing BLE device characteristics, detecting vulnerable BLE devices, and some experiments that proved how important it is to detect device vulnerabilities received much attention in the scientific literature. The literature reviewed throughout this study will be summarized in this chapter.

According to previous studies, various compromises have been made in Bluetooth Low Energy communication structure compared to the Bluetooth classic to simplify the low energy protocol. Some of these compromises were made in packet whitening, channel hopping rate, and the key exchange protocol since the low-power devices have limited input and computing capabilities. These decisions made the privacy of the data transmitted in BLE low secure than the Bluetooth classic. Ryan (2013) provided the first BLE sniffer that is able to monitor packets in Bluetooth Low Energy. Based on the evaluation of the encryption of the link layer in this study, it is argued that BLE devices are more vulnerable than Bluetooth Classic devices. In the following times, many more important studies and findings were made about the privacy and security of Bluetooth Low Energy devices. According to Kaspersky's report in 2021, medical equipment is already being hacked in many hospitals. Patient monitoring videos were watched illegally, and x-ray machines have delivered dangerously high radiation levels (Bracken, A. B.,

n.d.). Antonioli and Payer (2022) conducted a study on five different vehicle infotainment units (Bluetooth hardware to exchange data); by attacking those units, an attacker may access sensitive information about the driver along with the ability to send malicious commands to the unit itself. The attacker can even remotely control the vehicle using Bluetooth packets since the unit is connected to the vehicle's internal Controller Area Network (CAN bus allows microcontrollers and devices to communicate and transmits the data). Based on another report conducted by security firm Armis in 2018, on the domain of vulnerable BLE devices, the vulnerability of Texas Instruments' BLE chips leads to an attacker using the compromised device as a springboard for further internal attacks. The issue impacts millions of Wi-Fi access points, accounting for a sizable percentage of hardware used in corporations (Spring, A. T., n.d.).

Before Bluetooth 2.0+EDR (Enhanced Data Rate) pairing process's security feature was the exchange of a four-digit secret key which is easily guessed and makes it relatively easy to perform passive and active eavesdropping (Man-In-The-Middle attacks). The introduction of SSP (Secure Simple Pairing) with Bluetooth 2.1+EDR and the LE Privacy in Bluetooth 4.0 (i.e., the first version of Bluetooth LE) decreased the security concerns. However, SSP is still the standard pairing method many Bluetooth devices use. Albahar et al. (2016) stated in their study that LE privacy uses the advertisement method for the communication between Bluetooth devices. This method advanced Bluetooth Security compared to other methods in the field. However, according to their research, there is still no prevention against all attacks. That means we are surrounded by many devices that have vulnerability. Therefore, specifying which BLE devices have vulnerabilities has vital importance. To emphasize the importance of this topic, it can be said that implementing the BLE protocol in the eHealth sector permits various attacks because of the lack of authentication and integrity protection among the devices. Yaseen et al.

(2019) presented a novel framework named MARC to analyze BLE security features and mitigate MITM attacks against medical sensors using Bluetooth Low Energy pairing mechanisms in the eHealth sector. Their solution to mitigate possible attacks targeting BLE healthcare sensors focused on Received Signal Strength Indicator (RSSI) level, advertisement interval, advertised Bluetooth address, and malicious scan requests. They used Texas Instruments'CC2540 USB dongle to analyze BLE packets in their testbed. As a result, they could detect attacks and the cloned nodes using Bluetooth low energy characteristics, classes, devices architecture, security features, pairing methods, and multiple roles at different layers.

On the other hand, the more critical point is detecting vulnerable devices before their privacy is compromised. Thus, manufacturers and developers can take precautions on various hardware and firmware. One of the popular studies in detecting vulnerable devices is the fingerprinting study by Celosia and Cunche. Celosia and Cunche (2019) created fingerprints using the GATT profile of BLE devices. Based on their dataset, they analyzed the content of a GATT profile and the potential of these fingerprints to identify several devices uniquely. A GATT profile is a data structure containing many elements subject to variation between devices and thus holds potential for fingerprinting. In the light of such information, they considered handles and UUIDs of services and handles, UUIDs, properties, and values of characteristics to fingerprint devices. They also showed that the content of a GATT profile, some services, and characteristics could be exploited to track and infer sensitive information on the user.

Additionally, Zuo et al. (2019) developed a mobile app analysis tool BleScope for automatic fingerprinting of vulnerable BLE devices with static UUIDs from mobile apps to raise public awareness of BLE device fingerprinting and uncover these vulnerable BLE devices before attackers. They showed that an attacker could fingerprint a BLE device with static UUIDs since

there is a flaw in the existing design and implementation of the communication protocols between a BLE device and a mobile app. In a typical device connection scenario, a device broadcasts advertisement packets with UUIDs, and by leveraging these UUIDs, a companion app can identify the device. They also performed a field test and found that among around 6000 BLE devices, 94.6% of these devices are fingerprintable, and 7.4% of them are vulnerable to attacks. But these studies have mostly focused on fingerprinting devices from their UUIDs. According to the approach of this study, device UUIDs may not be helpful to detect a vulnerable BLE device chip model since many devices can use a defined UUID with different chips. give

In another study, fingerprints were created by applying device features to artificial neural networks. Yang et al. (2019) proposed an approach to fingerprint IoT devices using neural networks algorithms because the manual fingerprinting approaches are long-term processes. Since device manufacturers implement different network systems on their products, they first explored the features of devices in three network layers: network layer, transport layer, and application layer. By examining the features of these network protocols and using neural network algorithms, they generated 12,880 device fingerprints in a fine granularity label. Their implementation showed that the device fingerprints applied classification can be used to discover 15.3 million network-connected devices and analyze their distribution characteristics. Thus, they allowed extensive and rapid fingerprinting studies to be conducted. Similarly, this study aims to train a machine learning model with timing fingerprints of chips and make a broader range of vulnerable device detection since chip models have a limited variety. Results of another recent research on fingerprinting BLE chipsets instead of BLE devices found that an attacker could use unique physical-layer fingerprints to detect hardware imperfections of BLE chipsets on transmissions of devices. Physical-layer fingerprints can reliably differentiate many kinds of

BLE chipsets. This study of 162 BLE devices concluded that physical-layer identification using Carrier Frequency Offset (CFO) and I/Q offset is viable for an attacker to track mobile devices (Givehchian et al., 2022). It is clearly indicated that the presence of vulnerable BLE chips threatens user privacy and even physical security.

Neumann et al. (2012) analyzed the performance of network parameters for fingerprinting wireless devices. In this previous study, network parameters such as transmission time, frame inter-arrival time, and frame size were compared to show the best performance for fingerprinting wireless devices. The transmission time performs one of the best parameters compared to the other network parameters for device identification.

Overall, the review of existing literature revealed a hilarious number of the security risks of BLE devices. Sensing vulnerable devices can prevent many security problems, especially in enterprise environments. To detect vulnerable BLE devices, fingerprinting methods were informed in the review of the literature. Compared to existing studies, this study includes machine learning models trained with advertising packet timing information collected by sniffing nearby BLE devices to predict BLE devices' chip models.

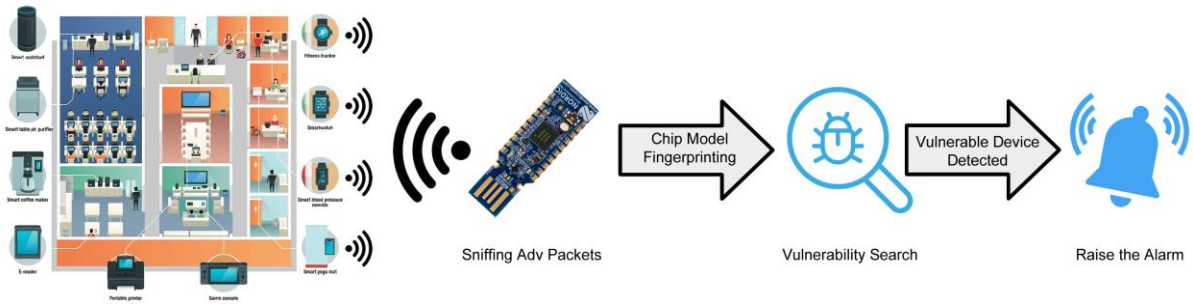
CHAPTER 3

DATA AND METHOD

The proposed system is to build a BLE sniffer for the enterprise environment to raise an alarm whenever a nearby vulnerable device is detected. The main focus of this system is to fingerprint the chip models of devices using only timing and size information collected from the captured raw BLE packets. The workflow of the proposed system is shown in Figure 3.1. below. In the system implementation, these steps were followed; the first step is sniffing BLE device advertising packets and then labeling these packets' timing and size information. Secondly, fingerprinting BLE devices' chip models via machine learning classification algorithms. After that, device chip models were searched on the vulnerability database for the possible device vulnerabilities. If a vulnerable device nearby is detected, the system would raise an alarm to warn the enterprise authorities. This chapter presents different steps taken to gather and manipulate the data used in this proposed system and provides information on the implementation to acquire the results.

Figure 3.1.

Proposed System Workflow for Enterprise Environment



3.1. Data Gathering

The data analyzed and used in this study were collected by sniffing method from eight different fitness trackers using the nRF52840 USB Dongle and the nRF Sniffer Tool developed by Nordic Semiconductor (Nordic Semiconductor, n.d.b). The device manufacturer, device model, device model number, chip vendor, chip model, and Bluetooth core specification version information of these devices are listed in table 3.1.

Table 3.1.

List of Devices

Brand Name	**Device Model Name**	**Device Model Number**	**Chip Vendor**	**Chip Model**	**Core Spec Version**
Fitbit	Inspire	FB412	Infineon	PSoC6	5.0
Garmin	Fenix 3 HR	-	Nordic	nRF51822_CE- S110	4.0
Huawei	Honor Band 5	CRSB59S	-	-	4.2
Jawbone	UP24	JH02	Nordic	nRF52840-S140	5.0
Samsung	Galaxy Fit	SMR370	Dialog	DA1469x	5.0
Sony	SWR10	-	-	-	4.0
Xiaomi	Mi Band 3	XMSH05HM	Dialog	DA14680	4.2
Xiaomi	Mi Band 4	XMSH07HM	Dialog	DA1469x	5.0

As seen in Table 3.1, eight different peripheral devices were used in the data collection setup. All these devices are in the fitness tracker category. Six of these devices have five different chips belonging to Cypress Semiconductor (i.e., Infineon Technologies), Nordic Semiconductor, and Dialog Semiconductor, the industry's leader chip vendors. Huawei Honor Band 5 and Sony SWR10's advertising packet data was not included in the study since their chip model information was not accessed on the Launch Studio, which is the listing search is provided

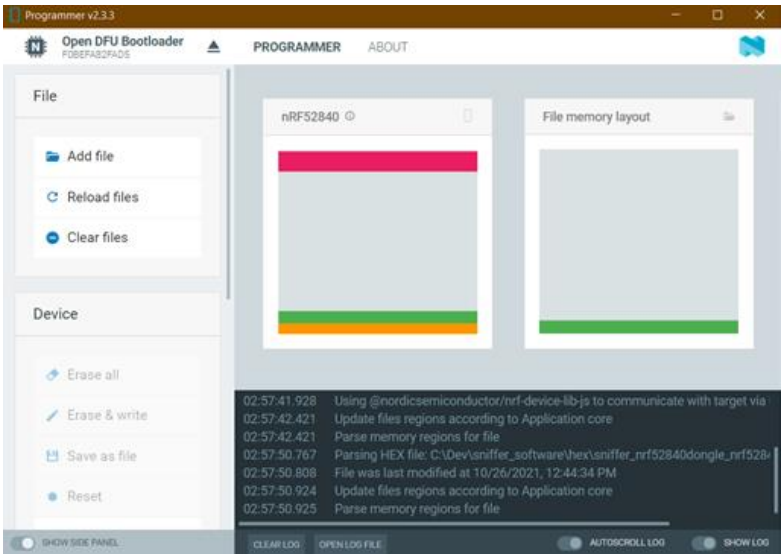
by Bluetooth SIG to help find previously qualified designs and declared products (Bluetooth, n.d.a).

Hardware & Software

In this experimental work, the nRF52840 Dongle introduced by Nordic Semiconductor has been utilized as sniffing hardware. The nRF52840 Dongle is a small, low-cost USB dongle using nRF52840 SoC that integrates Bluetooth Low Energy (LE), Bluetooth mesh, Thread, Zigbee, 802.15.4, ANT, and 2.4 GHz proprietary applications (Nordic Semiconductor, n.d.a). The nRF52840 Dongle is supported by most nRF Connect for Desktop applications and can be programmed with the nRF Connect for Desktop's Programmer application. nRF Connect for Desktop is a cross-platform framework for development applications, and it supports most operating systems. This tool contains apps for testing Bluetooth Low Energy, monitoring LTE links, power optimization, programming, etc. (Nordic Semiconductor, n.d.c). Figure 3.2. shows the user interface of Programmer app v2.3.3 which the Nordic SoCs can be programmed, read, written, or erased.

Figure 3.2.

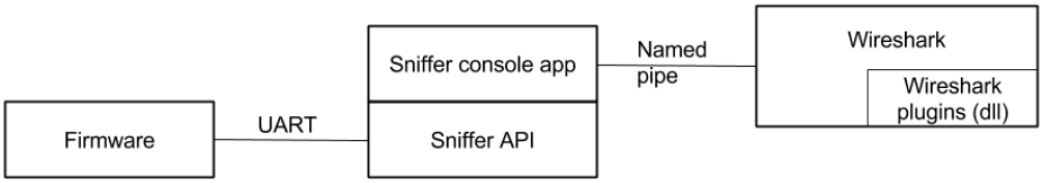
Programmer Application GUI



Another software used for the nRF52840 Dongle to capture advertising packets is the nRF Sniffer for Bluetooth LE Development tool v4.1.0, also developed by Nordic Semiconductor. The nRF Sniffer includes Python API that allows scripted use of BLE Sniffer and real-time display of Bluetooth LE packets. The nRF Sniffer lists all nearby Bluetooth Low Energy devices that are advertising, providing the Bluetooth address and address type, complete or shortened name, Received Signal Strength Indication (RSSI), and timing information for each captured packet. The sniffer consists of three parts where the API replaces the console app as the controller and hub of communication. The parts of the Sniffer are as shown in Figure 3.3.

Figure 3.3.

The Parts of the Sniffer



Source: (Nordic Semiconductor, 2014, p. 2)

The nRF Sniffer for Bluetooth LE software consists of firmware programmed onto a Development Kit (DK) or dongle and a capture plugin for Wireshark (Wireshark, n.d.) that records and decodes the captured raw data. Raw data decoded with the Sniffer tool can be viewed and exported with the Wireshark plugin included in the API. To use the nRF Sniffer software and view the captured data, the user must have the nRF Connect Programmer and locate the firmware HEX file for the supported hardware. The nRF Sniffer tool also includes some Python requirements. The user should have Python v3.6 and Pyserial v3.5 which is a third-party Python library. The instructions to meet other requirements are clearly stated in the nRF Sniffer user guide (Nordic Semiconductor, 2021).

Packet Sniffing Setup

In the packet sniffing setup, a smartphone that has the device applications, the nRF52840 Dongle as a sniffing and scanning hardware with a laptop running sniffing software that is noticed in the hardware & software part, and eight BLE smartwatches that broadcast advertising packets as a peripheral device are utilized.

Figure 3.4.

Packet Sniffing Setup

The methodology of this data gathering experiment is as follows. Firstly, each device was placed in the specified locations, as shown in Figure 3.4. After the nRF52840 Dongle, which was programmed with the hex file of the Sniffer tool, was connected to a computer via a universal serial bus (USB) interface, the Wireshark was launched on the computer, and the sniffer scanned channels (37, 38, 39) continuously. The advertiser and the smartphone were positioned at a close distance to the nRF52840 Dongle to allow the connection to be established. The device began to send the advertisements periodically. After packets were captured by the plugin interface of the nRF Sniffer tool, a connection was established between the advertiser and the smartphone to collect and analyze varied sizes of device advertising packets' timing parameters. The raw and

decoded data captured can be viewed simultaneously in the Wireshark. When collecting diverse types of packets before and after pairing is completed (which results in capturing an average of 6,000 BLE packets per device.), data was recorded and exported from the Wireshark in .csv format for analysis and manipulation to be used in the machine learning model. The mentioned steps are repeated for every single device listed in Table 3.1. Explanations of the obtained data are given below.

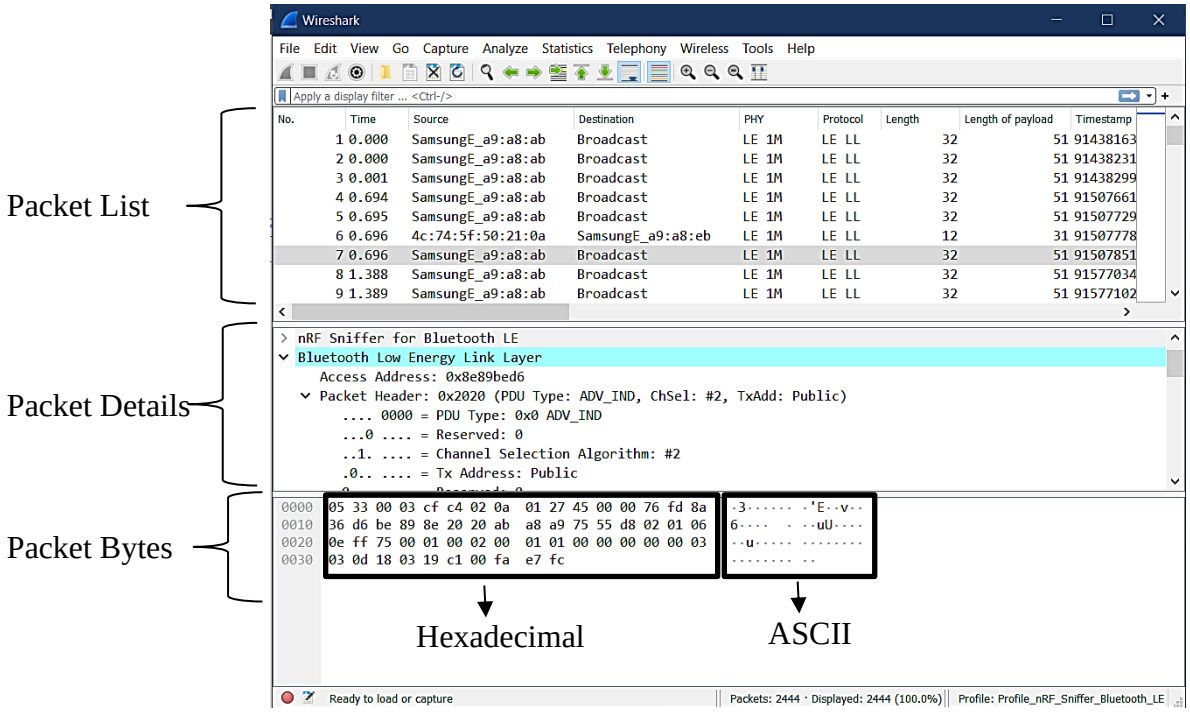
Data Description

A BLE device periodically sends a set of consecutive advertising packets. An advertiser may transmit only its advertising information or payload information within the advertising packets (Ghamari et al., 2018). In this study, approximately 40,000 advertising packets were gathered from eight devices with the previously mentioned packet sniffing setup. These packets contain the advertiser device's broadcasted data, communication between the peripheral and the central devices, and packet information added by the sniffing tool. The raw data packets were decoded by the nRF Sniffer tool, and the packet list, packet details, and packet bytes were displayed on the Wireshark, as seen in Figure 3.5.

The collected packets may contain advertiser device information such as device name, manufacturer id, chip vendor, Bluetooth specification version, subversion number, 16-bit service class UUIDs, 128-bit service class UUIDs, device appearance, advertising address, access address, Bluetooth MAC address, and more. The data may also contain packet information such as payload size, packet size, packet time, delta time, Received Signal Strength Indication (RSSI), packet counting, data direction, transmitting channel, PDU type.

Figure 3.5.

Wireshark Interface



Due to this study's adopted and advocated approach, only the timing information, including packet time (start to end) and delta time (end to start), was selected for the machine learning model inputs to identify if a device is vulnerable. Since the calculated timing information varies according to the packet and payload size, and this size information is directly related to the packet transmission time, the packet size and payload size were also included in the data as the inputs of the machine learning model. The packet size represents the packet length in bytes, and payload size is the payload length in bytes. The packet time (start to end) is the time of transmission of the current packet in microseconds, and the delta time (end to start) is the time in microseconds from the end of the previous data packet to the start of the current packet. It is the idling time between packet data transmission.

In addition to the four inputs collected from the devices, the chip model information was added to the data as the output of the machine learning model. Several semiconductor vendors offer BLE transceiver chips containing the BLE radio and a full-featured, low-power microcontroller capable of running the BLE protocol stack and a user-specific application (Schrader et al., 2016). Therefore, the chip model information indicates which Bluetooth hardware the BLE device is operating. Thus, as stated in the Bluetooth Core Specification, we can detect hardware-related vulnerabilities of a device with the chip model information. As seen in Table 3.1, the packets captured from Sony SWR10, and Honor Band 5 were not included in the research's data because of the inaccessibility of their chip model information.

The Bluetooth address, referred to as BD_ADDR in the Bluetooth specification, is an extended 48-bit Bluetooth MAC address value that uniquely identifies the Bluetooth device. A Bluetooth device must use this type of address. IEEE offers registries that maintain lists of unique identifiers under standards and issue unique identifiers to those wishing to register them (The Institute of Electrical and Electronics Engineers (IEEE), 2017). However, unlike registered addresses, some developers may assign random addresses for the devices so that the MAC address may be randomized. For this reason, the Bluetooth MAC address information will not always be consistent in identifying a device and accessing the chip model. Therefore, the Bluetooth MAC address information is also excluded from the data.

Despite much information contained in the packets, it has been observed that each device's device name and company ID information cannot always be accessed with the sniffing method. For this reason, data related to the device name and company ID are not included in this study, even though some packets have this information.

The same UUID can be used by multiple devices having different chip models. This means that the UUIDs found in the data cannot fully identify the chip model. Considering this reason, UUID data collected from the Bluetooth devices are not included in the data used for the machine learning model.

The delta time (start to start) found in the packets was also eliminated since it means the time delay of the start of the previous data packet to the start of the current packet. In other words, it is the sum of the packet time and delta time (end to start) which are already existing timing values in the data.

Considering all these descriptions, the timing info, which is inherent to the chip, and the size information, which is effective in calculating the timing of packets, were selected to identify whether a device has hardware-related vulnerabilities. The table 3.2. showing that the timing and size information of four different chip models belonging to two different chip manufacturers visibly differ is as follows. The time and size information in microseconds specified in the table represent typical values resulting from captured the ADV_IND type advertising packets from four devices.

Table 3.2.

Typical Time and Size Information of Chip Models

Device Model	**Chip Model**	**Chip Vendor**	**Packet Time (µs)**	**Delta Time (µs)**	**Packet Size (byte)**	**Payload Size (byte)**
Fenix 3 HR	nRF51822	Nordic	344	655	33	52
UP24	nRF52840	Nordic	408	704	27	46
Galaxy Fit	DA1469x	Dialog	336	129	32	51
Mi Band 3	DA14680	Dialog	376	1124	37	56

3.2. Data Manipulating

For the reasons mentioned in the data definition section, two devices' packet data were deleted from approximately 40,000 packets collected from eight devices. In addition, due to the reasons mentioned in the study's approach, only timing and size information was separated from each devices' CSV files containing the data packets. Packet size (bytes), payload size (bytes), packet time (μs), and delta time (μs) information of the remaining six devices were combined in a different CSV file under the headings LengthOfPacket, LengthOfPayload, PacketTime, and DeltaTime, respectively. Besides, the device chip model numbers found by manually searching the Launch Studio website, which is the listing search is provided by Bluetooth SIG to help find previously qualified designs and declared products (Bluetooth, n.d.a), are added to the last column of the file under the heading ChipModel. Afterward, duplicate lines were deleted. Thus, the data file was prepared as the machine learning dataset.

3.3. Machine Learning Model

In this study, it was aimed to access the chip model information of a real Bluetooth device by using only its packet timing and timing-related information. For this purpose, a common machine learning model, Decision Tree, and K-Nearest Neighbor (KNN) learning model were used to make chip model predictions. Decision Tree model predicts the value of a target variable by learning simple decision rules derived from data features, and The KNN algorithm is an instance-based learning algorithm. Jupyter Notebook was used to apply the decision tree model to the obtained data. The Jupyter Notebook is a web-based interactive computing platform, and it allows to run the code in Python language from the browser (Jupyter, n.d.). With the motivation that information other than timing cannot be used to determine whether a device is vulnerable or not, the first four features in the dataset described above were

used for machine learning inputs. The following steps were followed in the implementation of the model.

First, the dataset was imported in the project created in Jupyter Notebook using the Pandas Python package. After the rows with null values are cleared with the dropna function, the first four columns of the imported dataset are assigned as the inputs of the model (x) and the last column as the model's output (y). Then the dataset was split into random train and test subsets using the train_test_split method of the Sklearn library. The original dataset is divided into the training set and test set according to the ratio of 80:20. The models were trained with approximately 3600 advertising packet data (X_train and y_train). The trained models were expected to predict device chip models using the timing information of the devices. The accuracy scores were calculated by comparing the model predictions with the y_test, including exact chip model information for two machine learning models.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter presents the machine learning model test results, a brief description of the performance values, and the results of the querying chip models on the vulnerability database.

Two machine learning algorithms, Decision Tree and K-Neighbors were applied to the labeled data as described in Chapter 3, where all data collecting setup and methods were explained. Machine learning algorithms' results are placed below.

The data used in the training and testing of the machine learning models consist of packet timing and timing-related information (packet and payload size in bytes) collected by sniffing method from six different BLE devices. As shown in Table 3.1, since Xiaomi Mi Band 4 and Samsung Galaxy Fit have the DA1469x chip model produced by Dialog Semiconductor, the machine learning models were trained with packet timing and timing-related information of five different chip models. Additionally, four different performance metrics: Accuracy, Precision, Recall, and F1-score were calculated in this implementation. While the accuracy refers to the degree of closeness of a measured quality to that quality's actual value, the precision value, namely the positive prediction value, calculates the fraction of correct positive identifications. Recall performance metric describes the portion of correctly identified positives, and F1-score measures a test's accuracy considering both precision and recall (Newaz et al., 2020). Table 4.1 shows the precision, recall, F1-score values of the Decision Tree classification model.

Table 4.1.

Performance Values of Decision Tree Model

Chip Model	Precision	Recall	F1-score
DA1469x	0.89	0.91	0.90
nRF51822_CE- S110	0.94	0.95	0.94
PSoC6	0.72	0.54	0.62
nRF52840-S140	0.61	0.57	0.59
DA14680	0.54	0.58	0.56

Table 4.2 illustrates the precision, recall, F-1 score values of the K-Neighbors model according to labeled classes, in other words, chip models.

Table 4.2.

Performance Values of K-Neighbors Model

Chip Model	Precision	Recall	F1-score
DA1469x	0.84	0.89	0.87
nRF51822_CE- S110	0.92	0.95	0.93
PSoC6	0.74	0.64	0.69
nRF52840-S140	0.74	0.64	0.69
DA14680	0.63	0.54	0.58

As a result of testing the model trained using the Decision Tree, the accuracy score was 87.5%, whereas the accuracy score of the K-Neighbors classification model was 86%.

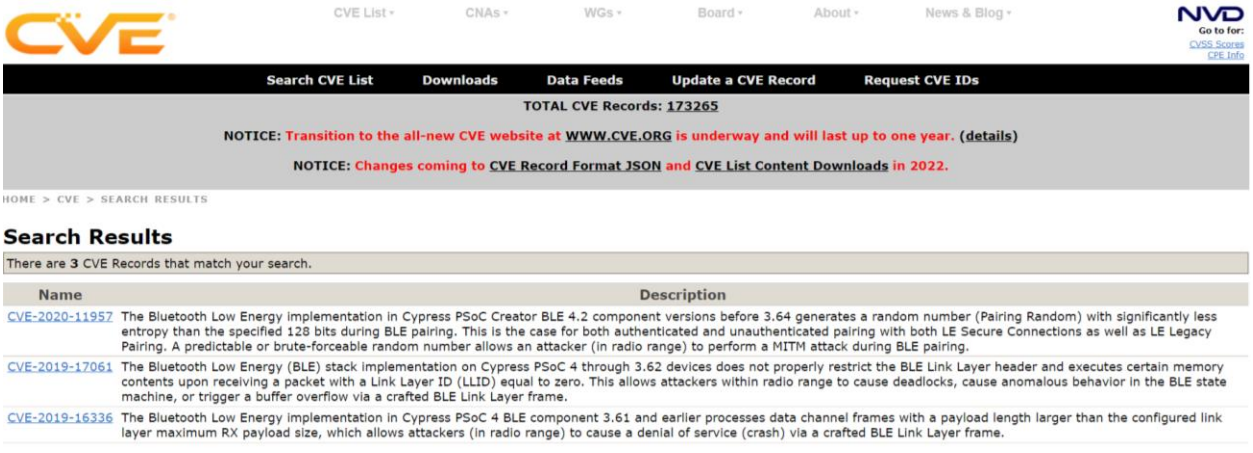
This study achieved high accuracy in detecting different types of BLE device chip models, which is a piece of crucial information to identify device vulnerabilities. One predicted observation is that chip models with more timing information provided higher performance values. For instance, nRF51822_CE- S110 by Nordic Semiconductor had approximately 93% precision, 95% recall, and 94% F-1 scores because there was more captured data for this chip

model. In addition, it has been revealed that using the packet time information gives accurate results even in identifying chip models by the same chip vendor since the data included the information of different chip models belonging to the same manufacturer.

As machine learning models extracted features from advertising packet timing values, it is expected that training machine learning models with a wider variety of chip model data will have a positive impact on the accuracy. To enlarge the dataset, advertising packets can be sniffed with different firmware versions of existing devices since it affects the timing calculations of a packet, or the number of devices in the testbed can be increased. To learn whether a BLE device is vulnerable, the chip models were queried on the public vulnerability database. To build a chip model vulnerability database, there are two sources available. First is the CVE list by MITRE. Bluetooth and BLE keywords were searched on that website to collect BLE-related vulnerabilities, as shown in Figure 4.1. The Launch Studio website, which provides the listing search by Bluetooth SIG to find qualified designs and products, was used to collect the chip model details of devices.

Figure 4.1.

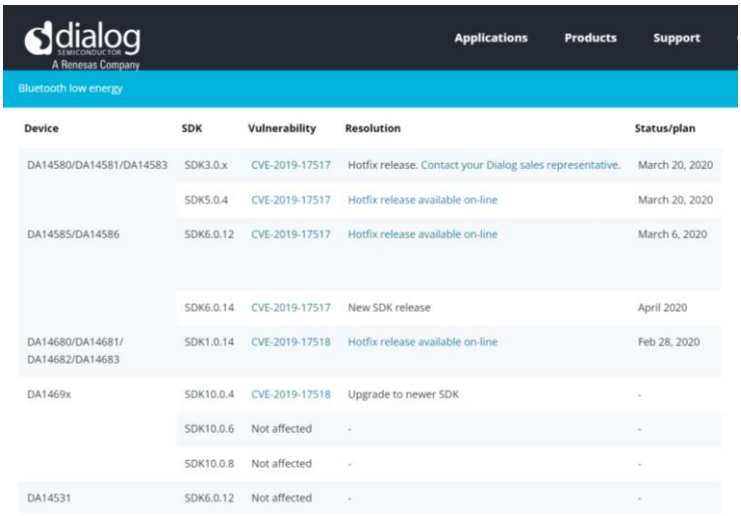
Search Results for PSoC 6 (Fitbit Inspire)



The second source was the chip vendors' web pages. Chip vendors mostly share chip vulnerabilities that influence their products. Figure 4.2. shows that the Dialog Semiconductor website for shared vulnerabilities and resolutions for their products.

Figure 4.2.

Dialog Semiconductor Webpage for Shared Vulnerabilities



Device	SDK	Vulnerability	Resolution	Status/plan
DA14580/DA14581/DA14583	SDK3.0.x	CVE-2019-17517	Hotfix release. Contact your Dialog sales representative.	March 20, 2020
	SDK5.0.4	CVE-2019-17517	Hotfix release available on-line	March 20, 2020
DA14585/DA14586	SDK6.0.12	CVE-2019-17517	Hotfix release available on-line	March 6, 2020
	SDK6.0.14	CVE-2019-17517	New SDK release	April 2020
DA14680/DA14681/ DA14682/DA14683	SDK1.0.14	CVE-2019-17518	Hotfix release available on-line	Feb 28, 2020
DA1469x	SDK10.0.4	CVE-2019-17518	Upgrade to newer SDK	-
	SDK10.0.6	Not affected	-	-
	SDK10.0.8	Not affected	-	-
DA14531	SDK6.0.12	Not affected	-	-

Device vulnerabilities learned as a result of queries made in the vulnerability database are as follows:

CVE-2019-17518 record shows that DA1469x and DA14680 (Samsung Galaxy Fit, Mi Band 4, and Mi Band 3) have the BLE implementation imperfection. According to this record, these devices respond to link-layer packets with a payload length larger than expected, allowing attackers in radio range to cause a buffer overflow via a crafted packet.

For PSoC 6 chip model, there are three existing vulnerability records. CVE-2020-11957 reveals that PSoC 6 (Fitbit Inspire) generates a random number (Pairing Random) with significantly less entropy than the specified 128 bits during BLE pairing. A predictable or brute-forceable random number allows an attacker (in radio range) to perform a MITM attack during BLE pairing. The second record for PSoC 6 CVE-2019-17061 shows that PSoC 6 allows

attackers within radio range to cause deadlocks, cause anomalous behavior in the BLE state machine, or trigger a buffer overflow via a crafted BLE Link Layer frame. Additionally, according to the CVE-2019-16336 record, PSoC 6 allows attackers (in radio range) to cause a denial of service (crash) via a crafted BLE Link Layer frame.

For Garmin Fenix 3 HR and Jawbone UP24 fitness trackers, one record is available on the vulnerability database, CVE-2020-10069. As this record informs, nRF51/52 allows attackers to trigger the vulnerability by simply sending a connection request with the cleared channel map field.

As a result, this study demonstrates that by sniffing only the timing information of a BLE device advertising packet, the chip model information of that device can be accessed, and it can be determined whether this device has a vulnerability.

CHAPTER 5

CONCLUSION

This study proposed a sniffer system for fingerprinting the chip models of BLE devices based on the captured advertising packets for the enterprise environment to raise an alarm whenever a nearby vulnerable device is detected. The system is built using nRF52840 Dongle as a sniffer and nRF Sniffer for Bluetooth LE software that can process received raw packets and display them on Wireshark. Such advertising packets' labeled datasets of five different chip models are merged, and four input types which are packet size (bytes), payload size (bytes), packet time (μs), and delta time (μs), were chosen fed into two basic Machine learning classification models. The described data set was preprocessed, removing two devices (Honor Band 5 and Sony SWR10) captured packets. The preprocessed data were used to train two machine learning algorithms, Decision Tree and k-Nearest Neighbor, and tested with a dataset consisting of 20% of all data that was split up from the database before training the models. The maximum results were achieved by the Decision Tree classifier, scoring 87.5%. Machine learning models provided higher performance values for the chip models with more timing information.

The results obtained confirm the validity of the proposed machine learning approach, which can perform accurate BLE device chip model estimation by using only time data of BLE packets. This was done by using a sniffing process consisting of eight fitness-tracker devices, gathering data, preprocessing data, training two Machine Learning classification models, Decision Tree, and k-Nearest Neighbor, for performing chip model estimation. Finally, the

results highlight that the Decision Tree algorithm works slightly better than the K-Nearest Neighbor model.

5.1. Limitations

In this study, there were two main limitations. The first limitation was the poor number of BLE devices. To gather data, real BLE devices must be acquired and only 8 devices have been studied. Since the number of nearby devices affected the size of the dataset and caused the training of the machine learning models with a small amount of data for each chip model and the lack of fingerprinted chip model variety, it can be said that this is the most significant limitation. However, two ways to enlarge the dataset were discussed in chapter four.

On the other hand, the difficulty of training machine learning models for each firmware version was the second limitation. Since chip vulnerabilities are defined for different firmware versions, each firmware version must be trained separately. However, chip vendors generally update firmware versions when they realized a vulnerability as a resolution. Therefore, the models must be updated when there is a new firmware version and it burdens training the machine learning model for each updated firmware version.

5.1. Future Work

In this section, ideas for future works and the development of this study are shared. First, the manual process after obtaining the chip model information from the machine learning algorithm can be eliminated. Automated vulnerability database querying by integrating vulnerability database containing CVE records directly to the machine learning project could automatically determine whether a nearby device is vulnerable. The second idea is the performance improvement of machine learning models. The performance improvement can be achieved by working on the optimization of the labeled data, increasing the data size fed into

machine learning models, and scaling the data variables. The third future work would be machine learning model output improvement. In the current work, the ML model outputs chip model information for four devices and chip model + firmware version information for two devices. In future work, the ML model can be trained to predict both chip model and firmware version for all devices to detect vulnerable devices with 100% confidence for the reasons why explained in the limitations section.

REFERENCES

Albahar, M. A., Haataja, K., & Toivanen, P. (2016). Bluetooth MITM vulnerabilities: A literature review, novel attack scenarios, novel countermeasures, and lessons learned. *International Journal on Information Technologies & Security*, *8*(4).

Antonioli, D., & Payer, M. (2022). On the Insecurity of Vehicles Against Protocol-Level Bluetooth Threats. In *17th Workshop on Offensive Technologies, co-located with IEEE S&P (WOOT 2022).*

Bluetooth Technology Website. (n.d.). *2022 market update*. *https://www.bluetooth.com/2022-market-update/*

Bluetooth. (n.d.a). *Launch studio: Listing search.* https://launchstudio.bluetooth.com/Listings/Search

Bluetooth. (n.d.b). *Specifications: Assigned numbers.* *https://www.bluetooth.com/specifications/assigned-numbers/*

Bracken, A. B. (n.d.). *Unpatched security bugs in medical wearables allow patient tracking, data theft*. Threatpost English Global threatpostcom. https://threatpost.com/unpatched-security-bugs-medical-wearables-patient-tracking-data-theft/178150/

Celosia, G., & Cunche M. (2019). Fingerprinting Bluetooth-Low-Energy Devices Based on the Generic Attribute Profile. *IoT S&P 2019 - 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things, Nov 2019, London, United Kingdom*. (pp. 24-31). https://doi.org/10.1145/3338507.3358617

CVE. (n.d.). *Search CVE List*. https://cve.mitre.org/cve/search_cve_list.html

Ghamari, M., Villeneuve, E., Soltanpur, C., Khangosstar, J., Janko, B., Sherratt, R. S., & Harwin, W. (2018). Detailed examination of a packet collision model for Bluetooth Low Energy advertising mode. *IEEE Access*, *6*, (pp. 46066-46073). https://doi.org/10.1109/ACCESS.2018.2866323

Givehchian, H., Bhaskar, N., Herrera, E. R., Soto, H. R. L., Dameff, C., Bharadia, D., & Schulman, A. (2022). Evaluating Physical-Layer BLE Location Tracking Attacks on Mobile Devices. *2022 IEEE Symposium on Security and Privacy (SP).* (pp. 507-521). https://doi.org/10.1109/SP46214.2022.00030

Jouans, L., Viana, A. C., Achir, N., & Fladenmuller, A. (2021, January). Associating the Randomized Bluetooth Mac Addresses of a Device. In *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)* (pp. 1-6). https://doi.org/10.1109/CCNC49032.2021.9369628

Jupyter. (n.d.). https://jupyter.org

Neumann, C., Heen, O., & Onno, S. (2012). An empirical study of passive 802.11 device fingerprinting. In *2012 32nd International Conference on Distributed Computing Systems Workshops* (pp. 593-602). https://doi.org/10.48550/arXiv.1404.6457

Newaz, A. I., Sikder, A. K., Babun, L., & Uluagac, A. S. (2020, June). Heka: A novel intrusion detection system for attacks to personal medical devices. In *2020 IEEE Conference on Communications and Network Security (CNS)* (pp. 1-9). https://doi.org/10.1109/CNS48642.2020.9162311

Nordic Semiconductor. (n.d.a). *nRF52840 dongle*. https://www.nordicsemi.com/-/media/Software-and-other-downloads/Product-Briefs/nRF52840-Dongle-product-brief.pdf?la=en&hash=8DDD17CCF2E574021A06A05C71B46C505D492361

Nordic Semiconductor. (n.d.b). *nRF sniffer for Bluetooth LE.* *https://www.nordicsemi.com/Products/Development-tools/nRF-Sniffer-for-Bluetooth-LE*

Nordic Semiconductor. (n.d.c). *nRF connect for desktop*. https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-desktop

Nordic Semiconductor. (2014). *Nordic semiconductor sniffer API guide: Version 0.2.* *https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjhloPI0OH2AhWzmWoFHeUzAVMQFnoECAYQAQ&url=https%3A%2F%2Fdevzone.nordicsemi.com%2Fcfs-file%2F__key%2Fsupport-attachments%2Fbeef5d1b77644c448dabff31668f3a47-72c3e4570fc34e4c8a736438d5f8aa2c%2F3240.Nordic-Semiconductor-Sniffer-API-Guide.pdf&usg=AOvVaw3Zqm207cgQf3ixEpkUbneU*

Nordic Semiconductor. (2021). *nRF sniffer for Bluetooth LE v4.1.0.* *https://infocenter.nordicsemi.com/pdf/nRF_Sniffer_BLE_UG_v4.1.0.pdf*

Padgette, J., Bahr, J., Batra, M., Holtmann, M., Smithbey, R., Chen, L., & Scarfone, K. (2017). Guide to Bluetooth Security, Special Publication (NIST SP), *National Institute of Standards and Technology, Gaithersburg, MD, [online]*. https://doi.org/10.6028/NIST.SP.800-121r2

Ryan, M. (2013). Bluetooth: With low energy comes low security. In *7th USENIX Workshop on Offensive Technologies (WOOT 13)*.

Schrader, R., Ax, T., Röhrig, C., & Fühner, C. (2016, September). Advertising power consumption of Bluetooth low energy systems. In *2016 3rd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced*

Computing Systems (IDAACS-SWS) (pp. 62-68). https://doi.org/10.1109/IDAACS-SWS.2016.7805787

Spring, A. T. (n.d.). *Two Zero-day bugs open millions of wireless access points to attack*. Threatpost English Global threatpostcom. https://threatpost.com/two-zero-day-bugs-open-millions-of-wireless-access-points-to-attack/138713/

The Institute of Electrical and Electronics Engineers (IEEE). (2017). *Guidelines for use of extended unique identifier (EUI), organizationally unique identifier (OUI), and company ID (CID).* https://standards.ieee.org/wp-content/uploads/import/documents/tutorials/eui.pdf

Wireshark. (n.d.). https://www.wireshark.org

Yaseen, M., Iqbal, W., Rashid, I., Abbas, H., Mohsin, M., Saleem, K., & Bangash, Y. A. (2019). MARC: Anovel framework for detecting MITM attacks in ehealthcare BLE systems. *Journal of medical systems*, *43*(11), 1-18. https://doi.org/10.1007/s10916-019-1440-0.

Yang, K., Li, Q., & Sun, L. (2019). Towards automatic fingerprinting of IoT devices in the cyberspace. *Computer Networks*, *148*, 318-327. https://doi.org/10.1016/j.comnet.2018.11.013

Yu, B., Xu, L., & Li, Y. (2012, June). Bluetooth Low Energy (BLE) based mobile electrocardiogram monitoring system. In *2012 IEEE International Conference on Information and Automation* (pp. 763-767). IEEE. https://doi.org/10.1109/ICInfA.2012.6246921

Zuo, C., Wen, H., Lin, Z., & Zhang, Y. (2019). Automatic Fingerprinting of Vulnerable BLE IoT Devices with Static UUIDs from Mobile Apps. *In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS '19). Association for*

Computing Machinery, New York, NY, USA. (pp. 1469–1483).

https://doi.org/10.1145/3319535.3354240