Automated Security Assessment Framework for Wearable BLE-enabled Health Monitoring Devices

GHAZALE AMEL ZENDEHDEL, University of New Brunswick, Canada
RATINDER KAUR, Cybersecurity Research Centre, Siemens Canada, Canada
INDERPREET CHOPRA, University of New Brunswick, Canada
NATALIA STAKHANOVA, University of Saskatchewan, Canada
ERIK SCHEME, University of New Brunswick, Canada

The growth of IoT technology, increasing prevalence of embedded devices, and advancements in biomedical technology have led to the emergence of numerous wearable health monitoring devices (WHMDs) in clinical settings and in the community. The majority of these devices are Bluetooth Low Energy (BLE) enabled. Though the advantages offered by BLE-enabled WHMDs in tracking, diagnosing, and intervening with patients are substantial, the risk of cyberattacks on these devices is likely to increase with device complexity and new communication protocols. Furthermore, vendors face risk and financial tradeoffs between speed to market and ensuring device security in all situations. Previous research has explored the security and privacy of such devices by manually testing popular BLE-enabled WHMDs in the market and generally discussed categories of possible attacks, while mostly focused on IP devices. In this work, we propose a new semi-automated framework that can be used to identify and discover both known and unknown vulnerabilities in WHMDs. To demonstrate its implementation, we validate it with a number of commercially available BLE-enabled enabled wearable devices. Our results show that the devices are vulnerable to a number of attacks, including eavesdropping, data manipulation, and denial of service attacks. The proposed framework could therefore be used to evaluate potential devices before adoption into a secure network or, ideally, during the design and implementation of new devices.

CCS Concepts: • **Security and privacy → Systems security**; **Vulnerability management;**

Additional Key Words and Phrases: e-health security, medical devices, security assessment

ACM Reference format:
Ghazale Amel Zendehdel, Ratinder Kaur, Inderpreet Chopra, Natalia Stakhanova, and Erik Scheme. 2021. Automated Security Assessment Framework for Wearable BLE-enabled Health Monitoring Devices. *ACM Trans. Internet Technol.* 22, 1, Article 14 (September 2021), 31 pages.
https://doi.org/10.1145/3448649

Authors' addresses: G. A. Zendehdel, I. Chopra, and E. Scheme, University of New Brunswick, Fredericton, NB, E3B 5A3, Canada; emails: {gamel, Inderpreet.Chopra, escheme}@unb.ca; R. Kaur, Cybersecurity Research Centre, Siemens Canada, Fredericton, NB, E3C 0J1, Canada; email: ratinder.kaur@siemens.com; N. Stakhanova, University of Saskatchewan, Saskatoon, SK, S7N 5C9, Canada; email: natalia@cs.usask.ca.

1533-5399/2021/09-ART14 $15.00
https://doi.org/10.1145/3448649

1 INTRODUCTION

With the digitization of health and personal information, mobile and wearable technologies have become attractive platforms for personal health tracking. Leveraging commodity hardware and the ubiquity of smartphones, these technologies have enabled the rapid and simplified collection of data and analysis from smartphone-linked wearable sensors. The majority of these sensors are **Bluetooth Low Energy (BLE)** enabled. The widespread adoption of these BLE-enabled **wearable health monitoring devices (WHMDs)**, combined with their affordability and convenience, however, has quickly overshadowed security and privacy risks.

In general, medical devices have been and remain vulnerable to cyberattack due to naïve designs and weak security implementations [29, 37, 45, 47]. The security of wearable medical devices is even more dire [39, 48]. These vulnerable devices, when connected to networks, expose entire digital healthcare infrastructures to security threats, and ubiquitous internet connectivity greatly increases the attack surface. Furthermore, standard approaches for security, such as access control, are often unsuitable for medical devices due to the inherent tradeoffs between security and privacy, the safety of patients, and the treatment process. For example, securing implantable defibrillator may require additional verification procedure when abnormal heartbeats are detected. Yet this is likely to delay electrical pulses to restore a normal heart rhythm and may possibly adversely affect patient's health.

There have not yet been any reports of death from maliciously compromised medical devices, but there are many examples of patient injuries or personal information leaks due to security vulnerabilities and a lack of security controls in medical devices [21, 57]. As such, the feasibility of such attacks is no longer in question. For example, in August 2017, the US **Food and Drug Administration (FDA)** agency recalled almost half a million implantable cardiac pacemakers due to concerns over malicious reprogramming [23]. In 2019 and 2020 the FDA issued a number of warnings concerning security vulnerabilities affecting insulin pumps, home monitors, clinical information central stations, and telemetry servers, emphasizing the adverse effects that compromised devices could have on patient health and the risk these devices pose to hospital networks [24–26].

The situation is aggravated by a lack of proper certification and comprehensive guidelines for the secure design of medical devices and health networks. Regulatory guidance for securing medical devices around the world remains limited and is insufficient. Several studies have demonstrated that even devices with existing security vulnerabilities (e.g., susceptible to unauthorized access, tampering) can be approved by the FDA for public use [65, 76]. Similarly, wearable health monitoring device can be except from any regulatory review as a posing low risk to patient safety even though device's mobile related application may acquire sensitive information from sensors [76]. Yet, a recent study showed that 61% of mobile applications of BLE-enabled **Internet-of-Things (IoT)** devices are vulnerable [79].

The lack of an established and validated framework for proper assessment makes the rigorous security analysis of devices already on the market even more challenging. The fact that most medical products are closed source and highly proprietary only serves to exacerbate the problem [13].

A number of studies have shown the need for proper security analysis of medical devices and introduced solutions focusing on the design of secure healthcare systems [11, 22, 31, 56], testing individual products or IoT networks [20, 30, 39, 54, 55, 59], and assessment of threats in such networks [28, 61, 66, 72].

Despite the amount and breadth of research emerging in this field, the research community has yet to offer a comprehensive practical security assessment framework that is applicable to a majority of the devices on the market. Current studies that focus on secure design preclude existing hospital infrastructure and can therefore only be considered for future facilities and devices.

These studies typically lack the ability to guarantee a secure implementation. For example, any inadvertent faults in the implementation of the device firmware, a corresponding application, or public libraries imported in the code can quickly undermine the security of devices, and hence the security of the infrastructure. Manual testing is commonly employed to identify and address security issues in individual devices, but it is costly, time consuming, and requires a breadth and depth of expertise to address the wide variety of emerging technologies. Consequently, attempts have been made to develop generic security assessment frameworks for IoT devices [66, 72]. Many of these studies, however, either exclusively focus on measuring the impact of vulnerabilities on nodes in the network, or assume the presence of traditional WiFi connected devices [8, 28, 72]. While WiFi-enabled medical devices are common in medical settings, a majority of modern and emerging health devices now use the Bluetooth Low Energy communication protocol [62, 69].

Because of its popularity in the IoT, security frameworks that focus on BLE devices, have mainly targeted general consumer "smart" applications such as smart lights or smart locks or highly popular fitness trackers such FitBit [43, 61]. In this work, we focus on the assessment of the growing number of BLE-enabled wearable devices, and propose an advanced security assessment methodology based on **Penetration Execution Standard (PTES)** guidelines [2]. We extend the PTES guidelines to incorporate practical assessment of BLE-enabled devices.

The proposed framework comprises two main stages: information gathering and security assessment. The information gathering module is responsible for fingerprinting a target device, extracting device implementation-specific information and related publicly known vulnerability information. We take advantage of the Bluetooth public product listings database to access device-specification details and leverage this information to scan for known vulnerabilities. The assessment module performs both static and dynamic analysis of the device, aiming to exploit potential vulnerabilities and to evaluate the possibility of occurrence of threats defined by our threat model. To overcome limitations in the existing available data, we perform additional analyses based on fuzz testing to discover previously unknown vulnerabilities. Finally, the implementation of our framework is made publicly available.[1] The main contributions of this work include the following:

- Extending the PTES standard guidelines to incorporate assessment of BLE-enabled devices.
- The design and implementation of an advanced security assessment framework for BLE-enabled wearable devices that incorporates the assessment of known and previously unknown vulnerabilities discovered through testing.
- The evaluation of the developed framework by conducting experiments on three different commercially available WHMDs.

The article is organized as follows: Section 2 presents overview of medical devices, and the current security landscape. Related work is presented in Section 3. Section 4 discusses the proposed assessment methodology, while Section 5 describes the framework modules. In Sections 6 and 7 we explain the implementation of the framework and show results of experiments on three consumer health monitoring devices. We discuss the implications of our analyses and make recommendations in Section 8, before concluding the work in Section 9.

2 OVERVIEW OF MEDICAL DEVICES

Medical devices provide value and benefit in hospitals by enabling more effective and less expensive means of monitoring and treatment. Medical devices can be divided into three main categories [40]:

[1] https://github.com/thecyberlab/safemed.

(1) **Wearable and general health monitoring products** for personal healthcare, such as smart bands, smart scales, and other commercially available smart home products. These devices typically use Bluetooth as a communication medium, and represent a rapidly growing market segment.
(2) **Implantable medical devices** such as pacemakers, insulin pumps, and other invasive medical devices. They typically communicate wirelessly by proprietary protocols, telemetry, or Bluetooth.
(3) **Stationary medical devices** include devices like hospital-based chemotherapy dispensing stations or cardiac monitoring machines that are typically connected via WiFi connection or network cable to a network of other devices.

WHMDs are rapidly becoming widely available as commercial products, as individuals strive to learn more about themselves, their general well-being, and their vital signs. WHMDs are often controlled by a controller device or an application on a smartphone or a tablet. WHMDs are a part of a personal healthcare system, where the wearable body sensors collect different types of data (e.g., EEG, ECG, blood pressure, etc.) and transmit them to an intermediate smartphone device [17]. An application on the smartphone performs the monitoring and analysis (either partly or wholly), allowing the user to see some results immediately. The data are usually transferred to a remote server for processing and analysis, and may be provided to physicians and other healthcare providers for monitoring, feedback, or intervention.

2.1 Security and Privacy Landscape of Medical Devices

Regulatory guidance for securing medical devices around the world is still limited. For example, in the US, the FDA is the main body for medical device certification.[2] Although the FDA provides pre- and post-market cybersecurity review of medical devices, many studies indicated its shortcomings, citing, among other things, a lack of minimum security controls for all devices, exemptions for low risk medical devices, and incomplete security guidance [76]. Indeed, there are a number of reasons securing medical devices is deemed challenging:

Persistent network connectivity: Modern medical devices are increasingly connected and complex. They routinely transmit data over the network to other systems leveraging both wireless (e.g., WiFi, Bluetooth, Zigbee protocols) and wired mediums. This connectivity significantly increases the potential attack surfaces. Although these devices rely on constant or regular connectivity, their design rarely incorporates assurances for secure communication. For example, many hospitals' internal networks are private, yet a hijacked medical device can serve as an entry to internal e-health data and resources [46].

Legacy software and slow updates: The software in medical devices is often outdated, hence, insecure, for, e.g., use of WinXP, which is no longer supported by Microsoft[3] yet it is installed in healthcare products such as GE's CARESCAPE, ApexPro, and Clinical Information Center Systems and Siemens RAPIDPoint 500 Blood Gas Analyzer and some laboratory diagnostics products [15, 16, 71]. There are also few incentives for devices manufactures to provide timely firmware updates and patches. For example, the FDA maintains tight control over approval of any changes to device software (including firmware), as a result, software updates to address security vulnerabilities may only be applied after approval is obtained [27]. In medical facilities, the situation is aggravated by staff reluctance to update devices to avoid disruptions in delivery of medical services that typically incurs additional cost [19].

[2]In Canada, it is Health Canada, and in the European Union it is the Medical Device Directive agency that provide similar regulatory oversight.
[3]https://www.microsoft.com/en-ca/microsoft-365/windows/end-of-windows-xp-support.

Long lifecycle: Many medical devices are expensive and, as such, their service life is long; often reaching 10 years or more. Obtaining FDA approval can add an additional 3–7 years after design before the medical device is even released to the market [74]. Designing any device, let alone a medical device, that will stand up to the ever-changing cyber threat landscape of the next two decades is challenging.

No proper security testing: Many health monitoring agencies follow FDA-defined steps and approach medical device certification from a risk mitigation perspective. This has led to the proliferation of medical devices that are deemed to pose little risk to patient health and are, as such, exempt from regulatory review. In reality, this leads to lack of security testing and controls embedded in these devices. A study conducted by the Ponemon Institute identified that only 9% of manufacturers and 5% of **Healthcare Device Organizations (HDOs)** test medical devices at least once a year; while 43% of manufacturers and 53% HDOs do not test devices at all [42]. As a result, low risk medical devices that contain vulnerabilities become stepping stones, allowing easier access to health infrastructure.

Closed and constrained nature: The closed and constrained nature of medical devices raises issues. First, proprietary software relies on the original manufacturer to develop and review the code, often ignoring security requirements [49]. Second, it is impractical for developers not trained in security to implement the required security features in the product. In these scenarios, vulnerabilities are commonly addressed using compensating controls that are independent of device firmware, but have their own limitations. Because they are unable to inspect vulnerable sessions encrypted by proprietary protocols, this leads to incompatibility and integration issues with existing security monitoring systems.

New security testing boundaries: Traditionally, medical devices were thought to be inaccessible to outside adversaries. With the wide adoption of WHMD, the locality and physical access to the device have drastically changed. Wearable devices now regularly come in close contact with other people (e.g., neighbours, mass transit) and commonly cross network boundaries (e.g., home network, office environment, coffee shops). This new reality has changed the threat landscape of medical devices.

2.2 BLE Technology

Most newer wearable devices have adopted the Bluetooth Low Energy protocol [62]. According to a recent report [69], the number of BLE-enabled devices on the market, worldwide, is now almost double the number recorded in 2018. Due to this explosion in popularity, in this research, we specifically focus on BLE enabled devices.

Bluetooth is a wireless communication technology that is known as a short range and power efficient wireless communication protocol for exchanging data, mostly between resource-constrained devices. Since its initial application in wireless headsets in 1999, BLE has rapidly become one of the leading IoT infrastructure technologies [12]. Figure 1 shows the BLE protocol stack architecture, incorporating the following layers, interfaces, and profiles:

Physical Layer: Responsible for translating digital signals over the air and providing services for the Link Layer. It uses the 2.4-GHz ISM (Industrial, Scientific, Medical) radio band for communication and divides it into 40 channels for advertising and data communication.

Link Layer: Responsible for data encryption and decryption, and is where advertising, scanning and connection initiation happens. The Link Layer deals with channels and packet types. Out of 40 channels, 3 (RF channels 0, 12, and 39) are used for advertising (broadcast transmission, device discovery and connection establishment) and 37 (RF channels 1–11 and 13–38) are data channels. To reduce interference with other devices, it uses an adaptive frequency hopping mechanism, requiring both BLE devices to decide to agree to a number of specific channels.

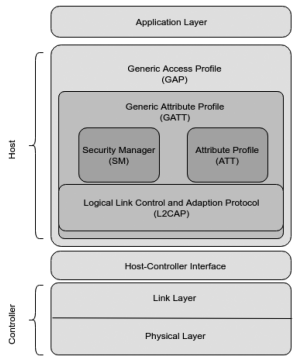


Fig. 1. BLE protocol stack.

Host-Controller Interface (HCI): Provides the standard interface for communication between the host and the controller.
Generic Access Profile (GAP): Defines roles and operational/security modes and procedures. Roles include broadcaster, observer, peripheral, and central. The broadcaster-Observer pair implements unidirectional connection-less communication with the broadcaster periodically sending advertising packets and the observer scanning for the broadcaster and listening to advertising packets. The peripheral-central pair establishes bidirectional connection-oriented communication. A peripheral device (slave in link layer) advertises by broadcasting connectable advertising packets. It is designed to consume the least power and processing possible (as are wearable devices). The central device (the master in the link layer), such as a smartphone, manages connection(s) with peripheral device(s).
Generic Attribute Profile (GATT): Introduces profiles to organize data as services and characteristics. Each service can include multiple characteristics and each characteristic contains properties and values. Device Information Profile, Heart Rate Profile, Glucose Profile, and Blood Pressure Profile are few examples of profiles defined in GATT that are relevant in this work.
Attribute Protocol (ATT): Defines the client and server roles. The server maintains a set of attribute-value pairs and sends indication or notification messages to the client, which is allowed to discover, read or write these attributes. The server is usually the peripheral device and the client is typically the central device.
Security Manager Protocol: Responsible for pairing, binding, key generation and distribution.

2.2.1 BLE Security. BLE security includes five distinct security features: pairing, bonding, device authentication, encryption, and message integrity.

Pairing and Bonding: Pairing includes authenticating the identity of the two devices to be linked, usually by sharing a key. Once authenticated, the link is encrypted, the long-term key is generated by one of the devices and then distributed to other devices.

Device Authentication: Verifies that the peer devices have the same keys to protect against **man in the middle (MITM)**. It can be achieved through secure pairing methods.

Pairing Methods: The pairing procedure enables two BLE devices to establish an encryption key (a **long-term key (LTK)**) for the rest of their communication. The BLE format supports both general Bluetooth legacy pairing methods, known to be vulnerable to several attacks [64], and the

more recent v.4.2, LE **Secure Connections (SC)** pairing. Importantly, the LE SC pairing method offers protection against eavesdropping and MITM attacks [3, 4].

To establish keyed communication, the pairing process goes through several stages to exchange and authenticate keys. Legacy pairing methods first generate a Temporary key and a Short Term Key to derive the LTK, while SC establishes the LTK directly using **Elliptical Curve Diffie-Hellman (ECDH)** key exchange and **Advanced Encryption Standard–Cipher-based Message Authentication Code (AES-CMAC)** algorithms. The pairing process can use one of the following association models:

- **Just Works:** This is the simplest, and hence the least secure, pairing method that is used for devices that have no input/output capability (which is the case for many wearable devices). The method uses a fixed value to generate an encryption key.
- **Passkey:** A type of pairing where a six-digit value is provided by the peripheral device and for pairing, the user must enter the value on the central device. Thus, for this pairing method the device must have a I/O mechanism to display the number.
- **Out Of Bound:** This type of pairing uses outside channels (such as **near-field communication (NFC)**) to exchange values.
- **Numeric comparison:** association scheme is introduced specifically for LE SC pairing process. The user must confirm that two values match to finish pairing. As with the passkey method, for numeric comparison, the device must have the capability of displaying the six-digit number. In addition, the device must have input capabilities to confirm or reject the pairing.

Encryption: Provides data confidentiality. BLE uses AES-CCM cryptography to encrypt data in-transit between paired or bonded devices.

Message Integrity: Protects against message forgeries and replay attacks. BLE supports signing the data with a Connection Signature Resolving Key.

Security Modes: BLE connections may operate in either Security mode 1 or 2, each containing several security levels. The higher the level, the more secure the connection is deemed [3].

(1) Security Mode 1 is meant to provide data confidentiality through encryption using FIPS-compliant algorithms (AES-CMAC [68] and ECDH [50]).
   - Level 1 – The least secure method, requiring no authentication or encryption. In essence it provides no security.
   - Level 2 – Allows unauthenticated pairing with encryption, hence no MITM protection is provided at this level.
   - Level 3 – Authenticated pairing with encryption using AES-CMAC [68].
   - Level 4 – (Starting BLE v.4.2) requires authenticated LE Secure Connection pairing with encryption using AES-CMAC [68] and ECDH [50] to establish a LTK. This is currently the most secure method.

(2) Security Mode 2 supports only data integrity by means of data signing using AES CBC-MAC [68]. Because it provides no encryption, Security Mode 1, levels 3 or 4, are generally preferred.
   - Level 1 – Unauthenticated pairing with data signing
   - Level 2 – Authenticated pairing with data signing

Device Address: A Bluetooth device ID is a six-byte value, displayed like FE:23:45:67:AC:DB:FE, where the three most significant bytes indicate Organizations Unique Identifier and the last three bytes are product specific [4].

There are four different forms that a device address can take: Public (Bluetooth device ID), Random static, Random private non-resolvable, Random private resolvable.

The public or static address of a device are publicly advertised during the discovery stage. If this address remains the same, then the device can be easily tracked. This concern is resolved through the use of private addresses. Both methods of generating private address (random private non-resolvable and random private resolvable) are designed to preserve the privacy of the device. The **Identity Resolution Key (IRK)** generates a pseudorandom value that acts as a private device address and can be resolved to the public device address by the IRK.

3 RELATED WORK

Since the early 2010s, the security and privacy issues of medical devices, and the entire e-health domain, have been widely studied [14, 63, 65]. Studies in this field range from secure design practices for healthcare system architectures and methods to assess the impact of vulnerabilities on the IoT network, to testing of individual devices and applications. Table 1 presents a comparative overview of the existing studies in this field.

The earliest studies were mostly focused on manual testing of individual products. One of the earliest works showing the vulnerabilities of medical devices is the seminal study by Halperin et al. [36, 37], which introduced several successful attacks on an **Implantable Cardiac Defibrillator (ICD)**, compromising the confidentiality, integrity, and availability of the device. Similar attacks were also later shown in insulin pumps [59], Fitbit trackers [20, 30, 60, 73], medical infusion pumps [55]. Several studies have also explored information disclosure in Bluetooth-enabled wearable devices [39].

The proliferation of vulnerabilities in medical devices triggered increased research on the secure design of IoT-based healthcare systems [11], secure transmission of medical data [22, 31, 56], and general security recommendations to protect medical devices from potential attacks [10, 38, 58].

Although security by design is essential, it is only effective when implemented carefully *as* designed. Consequently, a number of studies have proposed security modelling and security assessment frameworks for medical devices that range from assessment of the entire IoT network (at the network layer) to the assessment of the devices or mobile healthcare applications [52, 54].

For example, Atamli and Martin developed a generic threat model for IoT systems to help determine where efforts should be invested to secure these systems. The model considered three use-cases (power management, smart cars, and smart healthcare systems), and for each, defined potential sources of threats and attack classes (device tampering, information disclosure, privacy breach, denial of service, spoofing, elevation of privilege, signal injection, side-channel attacks) [8]. This model was theoretical and generalized to suit any IoT system, and thus did not offer practical guidance, yet laid out a path for further research.

A more comprehensive approach to the security evaluation of IoT networks was offered by Ge et al. The proposed graphical security framework was able to model potential attack scenarios and analyze various defence strategies given the topology of an IoT network and a list of vulnerabilities of the individual devices [28]. Although the verification of the proposed framework was left to future work, the practical applicability of this framework is limited to the quality and quantity of security information available prior to the attack modelling. Our approach can be seen as complementary to this model as it aims to automatically discover both known and unknown vulnerabilities of individual medical devices.

Table 1. Summary of Related Work

Reference	Technology analyzed	Focus	Shortcomings
Secure design studies			
Binu et al. [11]	Bluetooth	LEACH key exchange protocol	No device assessment
Elhoseny et al. [22]	—	Securing data in medical images	No device assessment
Griggs et al. [31]	—	Blockchain-based solution for PHI data privacy	No device assessment
Pham et al. [56]	—	Blockchain-based solution for PHI data privacy	No device assessment
Security assessment studies			
Halperin et al. [36, 37]	Radio Frequency	Radio-based attacks on ICDs	Manual device-specific attacks
Radcliffe [59]	Radio Frequency	Attacks on insulin pumps	Manual device-specific attacks
Park et al. [55]	Infrared (IR)	Sensor spoofing, injection attack on medical infusion pump	Manual device-specific attacks
Trippel et al. [73]	Analog signals	Analog acoustic injection attacks	Specific to accelerometers
O'Loughlin et al. [52]	Mobile health applications	Privacy analysis	No device assessment
Papageorgiou et al. [54]	Mobile health applications	Static & dynamic analysis of mobile applications	No device assessment
Rahman et al. [60]	HTTP, Bluetooth	Feasibility of attacks on Fitbit devices	Device-specific attacks
Atamli and Martin [8]	Generic	Generic threat model for IoT networks	Theoretical, no practical guidance
Ge et al. [28]	Generic	Model potential attack scenarios & defence strategies in IoT network	Theoretical, relies on available in advance security information
Goyal et al. [30]	WiFi, Bluetooth	Security and privacy of health trackers	Manual device-specific attacks
Tekeoglu and Tosun [72]	WiFi, Bluetooth	Generic testbed for IoT devices	No automation, preliminary experiments with WiFi-enabled devices
Siboni et al. [66]	WiFi, Bluetooth	Generic testbed for wearable IoT devices	Lack details on testing procedure WiFi-based assessment
Hassan et al. [39]	Bluetooth	Survey of the existing threats	-
Hassan et al. [39]	Bluetooth	Overview of threats	BLE is not discussed
Lonzetta et al. [48]	Bluetooth	Overview of security threats	BLE is not discussed
Haataja and Toivanen [33]	Bluetooth	Novel MITM attacks	BLE is not discussed
Sun et al. [70]	Bluetooth	Vulnerabilities of Secure Simple Pairing	BLE is not discussed
Cusack et al. [20]	BLE	Attacks on four tracking devices	Manual testing of devices
Guo et al. [32]	BLE	Detection & prevention of battery exhaustion attack	No device assessment
Pallavi and Narayanan [53]	BLE	Feasibility of spoofing, eavesdropping and firmware reverse engineering attacks	Manual testing, No details on execution of attacks
Ryan [64]	BLE	Eavesdropping attack	Manual testing
Sivakumaran and Blasco [67]	BLE	Static analysis Android applications	No device assessment
Zhang et al. [78]	BLE	Securing the communication between devices	Requires modification of BLE architecture stack
Yaseen et al. [77]	BLE	Anomaly-based detection of MITM attacks	No comprehensive device assessment

Contrary to these studies, several other researchers have focused on the practical implementation of security assessment frameworks. Tekeoglu and Tosun, for instance, developed a testbed to investigate security and privacy issues of WiFi or Bluetooth enabled IoT devices [72]. The testbed was equipped with open source security scanners and tools (e.g., Wireshark,[4] Kismet,[5] Kali Linux,[6]

[4]Wireshark: a multi-platform network protocol analyzer [18].
[5]Kismet: an open-source packet capturing and analysis tool for wireless networks [6].
[6]Kali Linux: an open source Linux distribution aimed at advanced penetration testing and security auditing [51].

OpenVAS,[7] and binwalk [1]), but the work did not provide many details about the testbed design beyond these tools. Although the stated focus of the testbed was both WiFi and Bluetooth enabled devices, the employed tools were primarily geared toward WiFi communication, with preliminary experiments only conducted with WiFi-enabled devices. Siboni et al. targeted wearable IoT devices [66], described design requirements that a security testbed should follow, and introduced the high-level design of their framework. This study offered a comprehensive view of what components and interactions should be tested but lacked details on how this testing can be conducted. A proof-of-concept prototype of the framework was again solely focused on WiFi communication and offered some preliminary assessment of two devices in a simulated WiFi network environment.

In general, automation of the security assessment of IoT devices—and especially wearable devices—remains a challenging problem. As with most computer devices, they are exposed to the traditional attacks (e.g., MITM attacks, DoS), yet they rarely include internal resources to combat these attacks. Furthermore, traditional tools and techniques designed to facilitate rapid security assessment are not applicable to wearable devices. Our approach goes beyond *what* should be tested and gives a roadmap for *how* wearable devices can be tested. Our framework leverages the general guidelines offered by the PTES. Yet the key principles of security assessment offered by the existing frameworks [8, 66, 72] align with PTES.

Whereas the vulnerabilities of WiFi-enabled IoT devices have received a some research attention, research on the security of Bluetooth and especially BLE-enabled medical devices remains limited. A general overview of Bluetooth security threats was presented in [39, 48]. A majority of the studies in this area have focused on the feasibility of various attacks on the Bluetooth protocol [33, 70] and its BLE version (e.g., battery exhaustion [32], spoofing, eavesdropping and firmware reverse engineering [53], eavesdropping [64], and unauthorized access to data stored on the phone [67]). Several studies have looked at the detection and prevention of BLE attacks through customized solutions [77], and securing the communication protocol [78]. All these solutions, however, are fragmented efforts to explore the security state of BLE-enabled devices. A more holistic approach to the vulnerability assessment of wearable devices was proposed by Hale et al. [34, 35]. SecuWear is a software and hardware vulnerability testing and risk mitigation platform built using open source technologies. This is one of the most comprehensive attempts to offer a testing environment, yet the framework was designed to simulate the capabilities of BLE-enabled wearable technology through one hardware module. As such it can primarily serve as a generic research platform rather than a practical assessment framework for various individual BLE-enabled devices. Conversely, here, we strive to offer a plug-and-play assessment framework applicable for testing the security of individual devices.

4 AUTOMATED SECURITY ASSESSMENT FRAMEWORK

The security assessment methodology for BLE devices proposed in this work is based on the PTES [2], whose guidelines define procedures to follow during penetration testing. These guidelines are comprehensive and introduce numerous instructions and tools for testing web applications and IPv4-based communication. Yet, they provide limited support for Bluetooth devices and applications. We therefore take PTES guidelines as a basis and adopt it to provide a comprehensive security assessment of wearable BLE health monitoring devices. The original PTES guidelines incorporate the following core stages: Pre-engagement Interactions, Intelligence Gathering, Threat Modeling, Vulnerability Analysis, Exploitation, Post Exploitation, and Reporting. We exclude from our analysis Pre-engagement Interactions and Reporting stages, which deal with customer engagement and communication of testing results, respectively, and the Post Exploitation stage, which

[7] OpenVAS: a vulnerability scanning and management framework [7].

is primarily focused on adversarial behavior on compromised devices. Our proposed framework therefore incorporates the following stages.

Stage 1: Information Gathering. A security assessment framework should be able to identify assets, how they interact, and the overall system workflow. The following assets and information are required for a general wearable device security analysis:

- *Wearable Device Specifications:* Wearable devices are usually proprietary devices with closed-source software (or firmware) and no (or the least possible) information available about the hardware and structure of the device. Thus, gaining access to the *physical device* is required to perform the assessment. Other device specification requirements include: **system-on-chip (SoC)** or *microcontroller unit*, *operating system* type and version (open-source or proprietary), memory and *storage* (internal or external), and *communication channel* (Bluetooth, Wi-Fi, USB or propriety protocol).
- *Companion Application Specifications:* The application source code of a wearable device is usually not accessible. Thus, the application is therefore required to interact with the device and make the system work. Specifications about supported operating system(s) and storage access are essential as well.
- *Functional Specifications:* Documentation and demonstration of how the whole system works is required to identify abnormal behaviour and responses from the system.
- *General System Configuration:* If there is a specific environmental condition required for the device to work or a specific system configuration is required, then this should be provided.

Stage 2: Threat Modelling. Traditionally, the computer security literature defines adversaries based on their capabilities. For example, adversaries can be passive or active. *Passive attackers* silently eavesdrop on signals transmitted or received by a device. Conversely, *Active attackers* may additionally interfere with legitimate communications and initiate malicious data transfer. Consequently, they may modify, steal, spoof, destroy or replay device data, thus compromising the integrity of device operations, leading to an invasion of patient's privacy, and/or preventing legitimate actions.

With respect to the target device, an adversary can take on either internal role (e.g., manufacturer, patient, or physician who has access to re-program a device), or external (e.g., neighbour, relative, a stranger on mass transit). Since wearable medical devices regularly cross network boundaries (e.g., home network, office environment, coffee shops), devices previously thought to be inaccessible to outside adversaries, are now open to third-party entities who would not otherwise be authorized to access them.

Our assessment framework assumes the presence of an active adversary with external (illegitimate) access to the device. Nevertheless, we explore attacks that require both passive or active roles. An adversary is assumed to have a basic knowledge of the BLE protocol and mobile applications. In our analysis, we primarily focus on attacks targeting data and on system confidentiality (patient privacy) and integrity.

Confidentiality attacks, such as eavesdropping on a communication channel or extracting application logs from mobile device (passive traffic logs), could reveal (1) the presence of a device and its type, which indirectly indicates the existence of a certain type of medical condition or therapy of the wearer patient (e.g., wearable glucose monitor), (2) vulnerabilities of the devices that can lead to further attacks, and, finally, (3) sensitive information such as medical information (e.g., heart rate, glucose level) and information required for authentication and access to the device (e.g., device pin).

Attacks on the integrity of devices and data are more sophisticated than eavesdropping and include modifying or forging communication (data replay, data tampering, data fuzzing, MITM). This can lead to (1) transmitting erroneous device configuration parameters, (2) gaining access to device and modifying therapy settings, or (3) mimicking the medical device and sending false data on behalf of the patient.

The proposed framework also incorporates attacks on *application and device availability* known as denial-of-service attacks (e.g., denying application access to a device, making device and application unusable, crashing the device application), but these attacks can be easily detected and mitigated by the patient, and thus their impact is limited.

The primary focus of the security assessments enabled by our framework are the software aspects of wearable medical devices. A remote server is considered to be a part of a larger digital health ecosystem, and thus do not considered in this work. Similarly, hardware components that are vulnerable to physical alterations of the device (e.g., circuit-level attacks) and embedded sensors are beyond the scope of the assessment. The following attack surfaces are therefore considered in this work:

- *Communication channel* (in this work BLE): Threats relevant to passive eavesdropping and active tampering with information transfer. In our analysis we consider the following threats: data eavesdropping, data tampering, replay, spoofing, and communication forging.
- *Software:* These are primarily threats related to code vulnerabilities in software and firmware, and overall lack of code protection that might lead to reverse engineering of software. In the security assessment of wearable devices, the framework should support static analysis of mobile application source or binary code, and device firmware at the source and binary code level.
- *Data:* Wearable devices typically collect and transfer user data between a device and mobile application, to be stored (at least temporarily) on the mobile device. Analysis should therefore incorporate examination of both stored and in-transit data. Data analysis should be able to confirm data privacy by investigating data encryption of both private user/patient information registered in the mobile app (including username and password) and health data captured by the monitoring device and transmitted to the mobile application.

Stage 3: Vulnerability Analysis. This analysis is focused on identifying any weaknesses or misconfigurations that can be leveraged during the exploitation stage. Identifying known vulnerabilities and testing for those not yet publicly known in any constituent part of a component is a critical factor in security assessment and the framework should be able to discover them.

Stage 4: Exploitation. The security assessment framework should be able to support different security tests to assess the level of security from different aspects. Test scenarios should address different threats and the impacts on the target component.

The proposed framework's design maps directly to these PTES guideline stages. The following section offers technical details for each of these stages and describes techniques that can be employed in a BLE environment.

5 FRAMEWORK DESIGN

The proposed framework is logically divided into two parts: information gathering and targeted testing. Figure 2 illustrates the framework design. The information collection module that corresponds to the PTES information gathering stage is responsible for identifying the device and extracting specific information that is then leveraged for vulnerability scanning (vulnerability

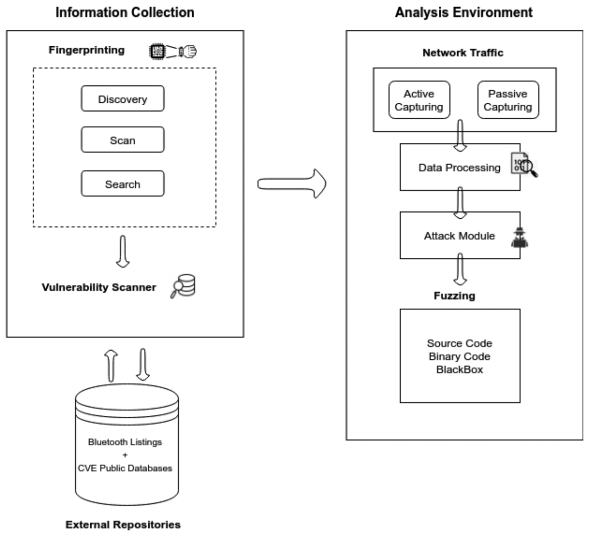


Fig. 2. Framework architecture.

analysis stage). The collected information serves as a foundation for the security assessment conducted by the analysis module, corresponding to the exploitation stage of PTES guidelines.

5.1 Information Collection

5.1.1 Fingerprinting. The fingerprinting module aims to identify the type of device under test, and to collect device specific technical information. The information is gathered from the device itself and external Bluetooth product listing, which contains information about all registered Bluetooth devices.[8] The product listing contains devices that completed the *Bluetooth Qualification Process*, the process that is required for all devices that use any of the Bluetooth trademarks. Generally, this process ensures that implementation of the Bluetooth technology meets Bluetooth qualification but does not focus on security [9]. Logically, the fingerprinting process is organized in three steps:

Step 1: Discovery. This component investigates the detectability of the target device. Due to characteristics of the protocol communication, BLE devices in advertising mode are visible at this stage, i.e., the devices that are in the communication range, not in sleep mode, and are not connected to another device (so they are possibly available for a connection request). The discovery component searches for nearby BLE devices and attempts to identify them by their MAC address.

Step 2: Scanning. All discovered devices (or rather MAC addresses) are scanned to obtain services and characteristics by performing service discovery requests to the device.

The list of all possible GATT services that can be provided by the devices is defined by the Bluetooth specifications.[9] The individual device vendors are, however, only required to implement some of the services that provide essential information about the device, such as the device name, while other information (e.g., manufacturer name or device model number) is optional. The information about service UUID, characteristics, values, permissions, and device information or battery

[8] https://launchstudio.bluetooth.com/Listings/Search.
[9] https://www.bluetooth.com/specifications/gatt/services/.

Table 2. Vulnerability Scanning Details for TrackR Bravo Device

CVE ID: CVE-2016-6541
CVSS: 5.8
Impact: {integrity: partial, availability: partial, confidentiality: partial}
Access: {authentication: none, vector: adjacent_network, complexity: low}
CWE: 200 - Information Exposure - Likelihood: none
vulnerable configuration: ['cpe:2.3:o:thetrackr:trackr_bravo_firmware:*:*:*:*:*:android:*:*', 'cpe:2.3:o:thetrackr:trackr_bravo_firmware:*:*:*:*:*:iphone_os:*:*']
Last Modified: 2019-10-09 23:19:00
Summary: TrackR Bravo device allows unauthenticated pairing, which enables unauthenticated connected applications to write to various device attributes. Updated apps, version 5.1.6 for iOS and 2.2.5 for Android, have been released by the vendor to address the vulnerabilities in CVE-2016-6538, CVE-2016-6539, CVE-2016-6540 and CVE-2016-6541.

services are examples of more generic, and thus optional, services that vendors may or may not provide. The scanning stage aims to discover which services are implemented in a tested device to guide the following security assessment.

Step 3: Search. Since the information provided by the vendor might not be comprehensive, we leverage the Bluetooth product-listing. This product-listing necessarily includes manufacturer name, device serial number, hardware/firmware/software revision and the product's referenced qualified design that are later used in vulnerability scanning.

5.1.2 Vulnerability Scanner. Based on the device specifications extracted by the fingerprinting module, the vulnerability scanner searches for known vulnerabilities and weaknesses in the software or hardware of the device as reported by the CVE, **National Vulnerability Database (NVD),** and **Common Weakness Enumeration (CWE)** databases. For each of the vulnerabilities found, we extract a CVSS Score that indicates the severity of the vulnerability, its corresponding impact and required access, a **Common Platform Enumeration (CPE)** configuration that indicates known software configurations affected by this vulnerability, and summary information that contains further vulnerability details.

As an example, consider a result of a vulnerability analysis conducted on a small tracker device (TrackR bravo) that can be attached to any item, such as car keys to help the owner find this item, when lost. Table 2 shows the details of this device and the device's corresponding vulnerabilities.

5.2 Analysis Environment

The security testing and corresponding analysis are performed in the Analysis Environment. This process is logically broken into several components: network traffic analysis, execution of known attacks, and fuzzing to discover unknown vulnerabilities.

5.2.1 Network Traffic Analysis. This component monitors and extracts the traffic between the medical device and mobile application, or between connected end devices. The traffic includes the advertising data as well as the data from data channels (e.g., read, write, notification and command-response data). There are two approaches to accessing the traffic between devices: passive and active capturing.

Passive capturing takes advantage of a smartphone's developers options that enable the capture and storage of Bluetooth traffic in logs on internal phone memory. For the Android platform, capturing is accomplished by enabling the Bluetooth HCI Snoop log, and log files are accessed

using the Android Debug Bridge tool. For the iOS platform, Apple provides Bluetooth profiles that, when installed, allow users to store logs as system diagnostic files.[10] It should be noted that passive traffic capturing requires user interaction for dumping the logs and feeding them into the framework for analysis.
Active capturing is essentially an eavesdropping attack (performed as part of a man-in-the-middle attack) that listens to and captures the traffic while the device and the corresponding application are interacting with each other.

5.2.2 Attack Module. Vulnerability scanning is based on official CVEs released for a component that is used in the target device, given the device's specific platform and configuration. The list of vulnerabilities is not exhaustive. As such, we also explored a device's vulnerability by examining its response to known attacks. The attack types were adopted to investigate the security of the device against the threats defined in our threat model.

In the following sections, we briefly explain the execution process of these adopted attack types.

Firmware code and data protection. We use Shannon's entropy [75] to inspect the device's firmware binary and sensor's data stream to check whether it is encrypted (or compressed) or not. The entropy value represents the randomness of a sequence of bytes in a file. If the entropy is high, then the file or a stream of data is deemed to be encrypted (or compressed), whereas if it is low, then the file/data are likely to be unencrypted (uncompressed). Entropy E is calculated for the file or stream of data for data blocks of size 1,024 bytes following the formula:

$$E = -\sum_{x} P(x) * log\, P(x),$$

whereas $P(x)$ is given as a probability of byte x appearing in the block of 1,024 bytes.

Spoofing. BLE devices broadcast advertising packets (which contain device address and advertised services) on an advertising channel. Spoofing the device is done by copying the advertising packet—so as to clone the device—and broadcasting it from a BLE transmitter/receiver device, on behalf of the target device. The companion application on the smartphone (or more accurately any BLE scanner device) receives spoofed device advertising packets and, if successful, sends connection initiation to the spoofed device. Therefore, to be successful a clone device should increase frequency of advertising packets to dominate the channel and force an original application to establish a connection with a clone device instead of the original tested device.

Eavesdropping. The framework collects the device address and other advertising information by listening on the advertising channel. If the spoofing is successful, then the emulated application initiates interaction with the original device capturing the exchanged traffic.

Replay. If eavesdropping and spoofing are successful, then we perform replay attacks on the device and on the application. For replaying to the application, we trick the mobile application into connecting to the spoofed device, and from the spoofed device (peripheral) we send previously captured "data" packets to read and notify commands. For replaying to the device, we connect to the device as the central device (counterfeiting the mobile application) and send previously captured "command" packets from the application to the device.

Tampering. Tampering with data is done on both advertising and data packets. Tampering differs depending on the services that the vendor decided to include in the advertising data. For example, the firmware revision string value (from Device Information Service) can be modified to force the

[10]The profile is only supported by iOS 13 and higher versions [5].

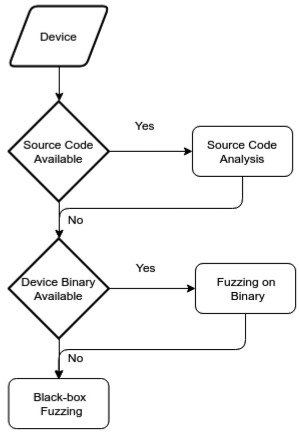


Fig. 3. Device fuzz testing flowchart.

device to request a firmware update. This could result in the mobile application sending a patch for a higher version of the firmware, enabling a downgrade attack. During the process of this over-the-air update, it is possible to obtain the device binary, to which there was no prior access. Similarly, manipulating battery level data (e.g., increasing it to a higher value) can lead the device to believe it has more power than it does, which could consequently result in an unexpected shut-off and lead to patient injury (e.g., in the case of a cardiac pacemaker).

Man-in-The-Middle. Spoofing the device allows us to perform a MITM attack. Since the target device accepts connection requests coming from any peer device, only the connection between the smartphone application and the forged device is needed to perform a MITM attack. In this work, we used the Gattacker tool to accomplish this attack. The central and peripheral devices (fake mobile application and forged device) and the web-socket to link them are provided by the tool. Advertising was started on behalf of the original wearable device; when the user connects to the forged device, a connection is then made from the central system to the original target device. As a result, all traffic goes through the established link; allowing it to receive all data from the wearable device, to manipulate it, and to then forward it to the application. For this attack, it is desired that the device and application remain functional (neither app nor the device should crash in this attack) and so only the payload of the data packets are modified, corresponding to read or notify commands with valid data. MITM attacks are an essential stepping stone for many other attacks, thus if the device is susceptible to MITM, then its security should immediately be questioned.

5.2.3 Fuzzing. As a final step, the device undergoes fuzzing. Fuzz testing is a software engineering technique initially developed for input validation. Over time, the approach was adopted in security research for automatic vulnerability analysis. In essence, feeding a device crafted malformed inputs and monitoring how the target device responds allows us to discover unreported vulnerabilities. The quality of the fuzzing depends on how much is known about the devices and its components (e.g., application, communication channel). This amount of information allows us to advance from simplistic dumb fuzzing that generates and applies inputs regardless of input and code semantics, to smart fuzzing that produces inputs based on an understanding of the code, its functionality, and context.

Here, we leverage the obtained background information about BLE communication and device sensors and use a mutation technique to generate fuzzed inputs. For example, captured traffic that carries device sensor data is extracted and processed to identify common patterns (e.g., common

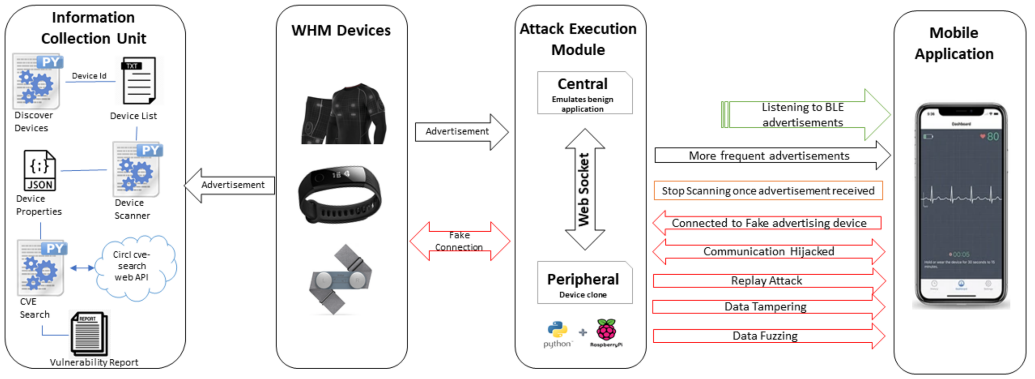


Fig. 4. Experimental setup.

groups of bytes by length, and common sequences of bits that appear in every byte array with the same length).

Figure 3 represents a decision flowchart of how testing proceeds based on the obtained knowledge of the system and the accessible sources from each component.

6 EXPERIMENTAL SETUP

The experimental setup of the proposed framework is illustrated in Figure 4. It is comprised of the devices to be tested, an Information collection unit, an Attack execution module, and a Mobile application. The framework relies on some human intervention as the initial step requires an installation of mobile applications on a phone environment. The information collection unit is set up on a Ubuntu-based machine with the BLE sensor available. This unit comprises various scripts implemented using the BLEsuite Python package. The Discover Devices script scans visible Bluetooth devices and records all discovered addresses. The Device Scanner script then identifies the device manufacturer names, services, descriptors and the characteristics of the discovered BLE devices, and requests that these devices be checked in the Bluetooth product-listing database through its web API.

We adopted the CVESearch Python tool, a wrapper around *Circl cve-search* web API,[11] to search publicly known information about security vulnerabilities in software and firmware through the following repositories:

- NIST NVD,
- CPE,
- CWE,
- CIRCL incident statistics and threat ranking,
- toolswatch/vFeed

The Attack execution module includes a clone representation of the device (the fake device) that is established using information gathered by the collection unit, and an emulation of its application (the fake application). Following GAP role naming standards, we refer to a clone representation of the device as a *peripheral device* and the emulation of its application as a *central device*.

The presence of the clone device and an emulation of the application, apart from the originals, are necessitated by the nature of the security assessment. The original application on the smart-

[11] https://cve.circl.lu/api/.

phone does not discover and/or make connection to any device other than the one(s) statically defined in the application code. The clone representation of the tested device therefore mimics the original device, allowing us to modify commands and communicate altered data to the legitimate application. The emulated version of the application permits us to deceive an original device and to send it forged commands. This includes arbitrary commands that a normal user may not have permissions to send through the true application interface.

The peripheral and central devices were implemented on a Raspberry PI model 3 B+ platform due to its portability. They were both built on Linux OS and support Bluetooth 4.2. Although, theoretically, both the peripheral and central roles could have been implemented on one physical device (with two Bluetooth dongles), the switch between roles would require a restart of Bluetooth services whose delay would be unsuitable for real-time capturing. The traffic capture, attacks and fuzz testing were implemented using the modified Gattacker tool [44].

The entire proposed security assessment framework was implemented as a stand alone tool and is publicly available for others to use at https://github.com/thecyberlab/safemed.

For validation we selected three Bluetooth-based wearable health monitoring devices. Since our framework targets assessment of individual devices, we selected commercially available devices that presumably underwent necessary certification and testing activities before being released to the market. Since most wearable devices lack input/output capabilities, we selected devices that implement *just works* pairing method. This is the least secure method developed for BLE devices, as such additional security measures are expected to be provided by the vendors.

The framework was validated by testing the following devices: a *heart rate monitoring device* that provides visualization of recorded ECG signals to assist identifying abnormal heart rhythm (DuoEK personal ECG tracker), a *smart personal fitness tracker* for general daily activity monitoring (Purifit B521A smartband), and a *smart garment* for muscle activity monitoring during workouts (Athos Smart Apparel). All of these devices share the data collected with the user through a mobile application and use Bluetooth connectivity. Two of them require authentication and store the user's personal information on the smartphone and on the cloud.

7 ASSESSMENT RESULTS

A summary of the security assessment for all devices is given in Table 5.

7.1 Athos Smart Apparel

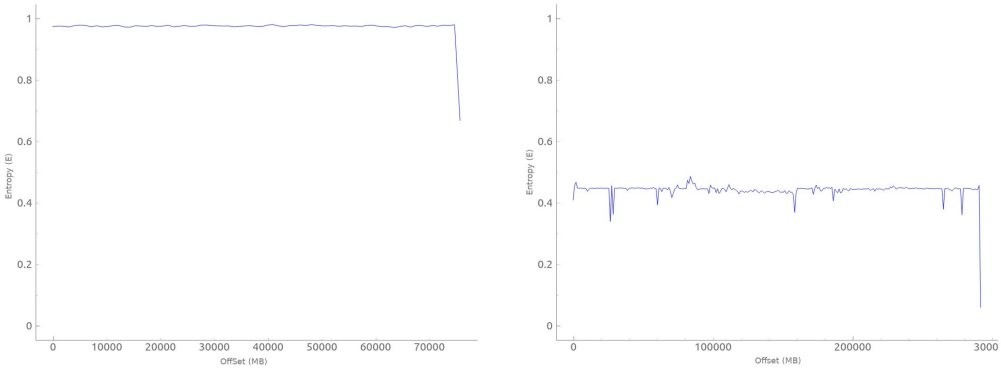
Athos is a smart wearable system that integrates **surface electromyography (sEMG)** electrodes into the athletic apparel. The product consists of a garment (a shirt and shorts) in which a layer of sensors is embedded, and a small device (called the "core") that snaps into the garment to transmit the biosignal data read by the sensors in near real time to a mobile application. The mobile application displays which muscles are firing and how much they are being exerted. It supports iOS versions 12.4 and higher. The sensors are comprised of 18 sEMG sensors, 10 in the shirt and 8 in the shorts, and 4 heart rate sensors in the shirt. Our fingerprinting further showed that the device has an nRF51x22_QF built-in chip with BLE version 4.1.

Fingerprinting: The device provides device discovery service, thus scanning the device address, our framework collected device name, services, characteristics, and properties. Table 3 shows the information about the Athos device that is registered in the Bluetooth product listing. The GATT services implemented on the device are shown in Table 5.

Vulnerability Scanning: The scan revealed no publicly reported vulnerabilities for this device, which is not entirely unexpected as the device firmware is close sourced.

Table 3. Athos Apparel Information Extracted from Bluetooth Product Listing

Company Name	MAD Apparel, Inc.
Marketing Name	Athos Core
Device Model	A100
Referenced Qualified Design Company	Nordic Semiconductor ASA
Model Number	nRF51 × 22_QF with updated S × 10 stack
Date	2017-10-17 17:35:00



(a) Firmware binary (b) Workout data

Fig. 5. Entropy of Athos firmware binary and data stream.

```
...
2019.10.16 15:30:59.572 | > R | 180f (Battery Service) | 2a19 (Battery Level) | 40 (@)
2019.10.16 15:30:59.676 | > R | 180a (Device Information) | 2a25 (Serial Number String) | 4143323533383230304e5a ( AC2538200NZ )
2019.10.16 15:30:59.752 | > R | 180a (Device Information) | 2a26 (Firmware Revision String) | 424c452076302e392e39 (BLE  v0.9.9 )
2019.10.16 15:30:59.811 | > R | 180a (Device Information) | 2a28 (Software Revision String) | 4453502076302e382e31 (DSP  v0.8.1 )
2019.10.16 15:31:00.878 | > R | 1800 (Generic Access) | 2a00 (Device Name) | 4174686f7320436f7265 (Athos Core)
2019.10.16 15:31:13.045 | < C | 4e137a00c29d436f8190733fc4b08abc | 4e137a01c29d436f8190733fc4b08abc | 73 (s)
2019.10.16 15:31:13.765 | > N | 4e137a00c29d436f8190733fc4b08abc | 4e137a01c29d436f8190733fc4b08abc | 0014140
      d009a00be007e007e009600eb00c700c3 (          ~ ~          )
...
```

Listing 1. Athos Captured Traffic.

Attack Execution

- **Passive traffic logs:** The mobile application requires a user to create profile and authenticate themselves when accessing application, yet log files are stored in clear text.
- **Firmware code and data protection:** The Entropy of the firmware binary file was close to 1.00, allowing the framework to infer that the binary file is encrypted. However, the entropy of the sensors' data stream is around 0.4 on average, which is interpreted as being unencrypted. Figure 5 shows the entropy graphs for the firmware binary and sensor's data.
- **Spoofing:** The advertising packets were captured and rebroadcasted successfully, initiating a connection setup between an original device and an application and their fake representations in the framework.
- **Eavesdropping:** Once the system was set up, interaction with the garment as a normal user, began. The implemented commands available in the application were connection, disconnection, find core, calibration, workout start, pause, stop. For each command, emulated applica-

tion sent the requests to the core, and responses were sent back from the core to the emulated application, this essentially in initiated a simulated 'workout' that lasted for a few minutes.

- **Replay:** Since eavesdropping and spoofing were successful, the replay attack was carried out with (1) randomly chosen static data and (2) the data from the previous session. Listing 1 shows an excerpt of the captured traffic that was used in the replay attacks.

  In both cases, the replay attack was successful. It was determined that, once changed, the application could no longer show an actual data and displayed the replayed data instead.
  A replay attack was also performed on the device with write commands being successfully sent on behalf of the app to the device. Since Athos only collects and forwards data from sensors, and there are no actuators on the device (such as inflation actuators for blood pressure monitoring devices), there were no critical implications of this attack for this device.
- **Tampering:** Data tampering was conducted by changing the value of the *Software Revision String* attribute from the *Device Information Service*. This change forced the original application to request the update for the tested device. The update binary was sent from the original application to the peripheral (clone) device, and was then transferred to the emulated application through the web socket to the true Athos device. Consequently, we were able to capture an updated binary while the device was forcefully updated. Thus, the tampering attack was deemed successful.
  The battery level value was also successfully changed, causing the application to persistently display the false static values sent by the attack conductor. For this particular application, the implications of such an attack are not critical to the user's safety, but it could shorten the life of the product or interfere with the device and research or training results. Listings 2 and 3 demonstrate the altered *Battery Level* and *Software Revision String* values.
- **MITM:** Following spoofing of the original application and device, the framework was able to perform data replay and manipulation (tampering). After the data tampering attack, the device and application continued to function and, as a result, a MITM attack on the Athos suit was successful.

```
{
    "uuid": "180f",
    "name": "Battery Service",
    "characteristics": [
        {
            "uuid": "2a19",
            "name": "Battery Level",
            "properties": [
                "read",
                "notify"
            ],
            "descriptors": [
                {
                    "handle": 15,
                    "uuid": "2902",
                    "value": ""
                }
            ],
            "hooks":{
                "staticValue":"64"
            }
        }
]},
```

Listing 2. Battery Level Tampering.

```
{
    "uuid": "180a",
    "name": "Device Information",
        {
            "uuid": "2a28",
            "name": "Software Revision
                String",
            "properties": [
                "read"
            ],
            "asciiValue": "DSP v0.8.0",
            "hooks":{
                "staticValue":"44535020
                    76302e382e30"
            }
        }
}
```

Listing 3. Downgraded version of Software Revision information.

- **Fuzzing:** From the data patterns found during data analysis, the indices of bytes in the input were obtained and available for mutation. To perform fuzzing, the proposed framework replaced fixed values with special case values (0, 1, F) and random values generated by a

Table 4. Athos Device Information

Company Name	MAD Apparel, Inc.
Marketing Name	Athos Core
Device Model	A100
Referenced Qualified Design Company	Nordic Semiconductor ASA
Model Number	nRF51 × 22_QF with updated S × 10 stack
Date	2017-10-17 17:35:00

random generator. Sending these fuzzed data to the application resulted in the application crashing and failing to function normally consequently resulting in a denial of service.

Summary. The device is registered in the *Bluetooth product listing*, which provided basic information about the device (Table 4). Although no known vulnerabilities were reported for this device, the results of the proposed framework showed that it is vulnerable against a number of adopted attacks.

7.2 Smart Fitness Tracker

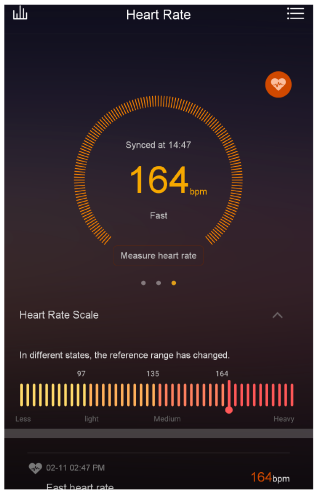
The Purifit B521A smartband is a fitness tracker that, among other features, monitors sleep quality and heart rate. It contains built-in sensors for these functions, a Nordic52832 SoC, and Bluetooth 4.0 for connectivity, and supports both Android (version 4.4 and higher) and iOS (version 8.0 and higher). The tracker interacts with the Purifit mobile application to synchronize data between the device and the application and displays monitored data to the user.

Fingerprinting: The device does not appear in a search result of the Bluetooth product listings, which means the *Bluetooth Qualification Process* has not been completed for this device. The GATT services implemented on the device are shown in Table 5. In addition to standard service, the device provides two vendor-specific proprietary services about device and battery information.

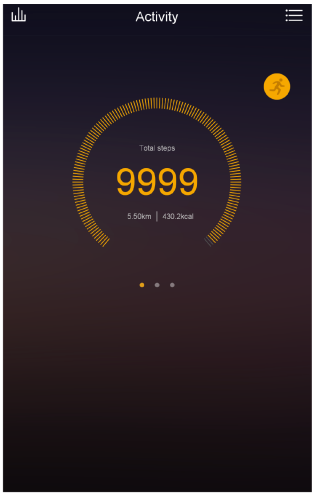
Vulnerability Scanning: As with the Athos suit, no officially reported vulnerabilities were found for this device or the built-in software components.

Attack Execution:

- **Passive traffic logs:** The mobile application requires a user to create a profile and authenticate themselves before accessing the application. The log files are stored in clear text and easily accessible.
- **Firmware code and data protection:** The entropy analysis yielded a high value for the firmware binary file (0.75) suggesting that the binary is encrypted (or compressed). The data stream entropy is lower (0.4) indicating that data stream is not encrypted. Figure 7 shows the entropy graph for the Purifit firmware binary and sensor data.
- **Spoofing:** As in the previous case, the framework successfully initiated connections between the original device and the emulated application, and between the clone device and the original application.
- **Eavesdropping:** Although proprietary services were embedded within the device, the device was communicating in clear-text, allowing our framework to acquire usable information.
- **Replay:** It was possible to replay data previously recorded during transfer of the sensor data to the application. In response, the application displayed only this replayed workout data as if it were new. Surprisingly, it was also possible to send replayed data back to the



(a) Heart-rate



(b) Pace

Fig. 6. Tampered values.

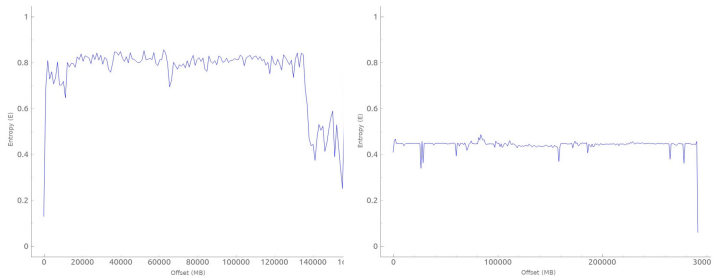
tracker, which successfully replaced the current data on the tracker itself with the replayed (previous session) data. In this case, if the previous data had not yet been synchronized with the application, then it would have been permanently lost.

- **Tampering:** This device does not receive firmware version information from a public GATT service, however, since the data are in clear-text, it was identified that the application requests the firmware version by sending a command including the text "BT+VER" to the device. The device responds with the message "BT+VER:*122.043.018*," which indicates the firmware version and manufacturing code. It was found, however, that altering the value of the version does not trigger an update procedure. This suggests that either the device uses a channel other than Bluetooth to transfer the update or that it may have no update procedure at all.

  To indicate battery level, the device sends "AT+BATT:*val*," where *val* shows the battery percentage. Thus, by changing this value, the application could be easily tricked to show a false battery level. Again, although tampering with the battery level for this device may not be a safety hazard, it could compromise the utility and lifespan of the device. To send other monitored data (pace data and heart rate), the device uses a specific message format in clear-text. Thus, it was intuitive to tamper with the workout data and heart rate by simply replacing the integer value assigned to "AT+HEART" and "AT+PACE" in the corresponding packets. Figure 6 shows the results of tampering with these values on the application.

  In addition to sensor data, the device also has the capability to send command messages to the phone to *take photo* and *find phone*. Thus, as long as the camera was open on the phone, it was possible to request a phone to silently take pictures. The photos are stored on the smartphone's storage and are not immediately sent to the cloud, which is an appropriate privacy preserving measure. Yet this capability can be easily leveraged by malicious applications installed on the phone to collect and transfer pictures to a third party.

  These two features are easily exploited to create a more noticeable denial-of-service attack. Sending commands to take a significant number of pictures on behalf of a tested



(a) Firmware binary (b) Data stream

Fig. 7. Entropy of Purify tracker firmware binary file and data stream.

device could fill the phone's memory and thus prevent it from functioning. Similarly, sending the *find phone* request from an original application results in the smartphone producing a beep sound. Several commands sent in a short period of time could be used to distract the user or other people and effectively force them to disable the device.

- **MITM:** Both data replay and data tampering were conducted without interrupting normal device operations; as such MITM attack was deemed to be successful.
- **Fuzzing:** It was found that sending fuzzed input to the application increased the time it took for the application to synchronize the data. The application accepted some fuzzed data (for example, it accepted the device serial number and user profile configurations as sensor data), which suggests that it does not validate the input data before displaying it. Nevertheless, the application was able to handle the data fuzzing attempts without crashing and, as such, the fuzzing did not result in denial-of-service attack. It is possible that more rigorous fuzzing attempts, however, could still break the application.

Summary. The Purifit device is not registered in the *Bluetooth product listings* and no *device information* is revealed about the advertising data. Although there are no reported vulnerabilities for this device or application, the proposed framework found that the device is vulnerable to most attacks.

7.3 ECG Tracker

The DuoEK personal ECG tracker is a small device with two electrodes that record ECG signals to identify irregular heart beats (a symptom of various heart problems). The device contains built-in memory, a rechargeable battery, and Bluetooth 4.0 technology and supports both Android (version 5.0 and higher) and iOS (version 9.0 and higher). The companion application, ViHealth, is responsible for logging and visualizing data received from the device.

Fingerprinting: The device provides a number of standard Bluetooth services including the heart rate service to measure heart rate and a battery service to read the battery level (Table 5). Additionally, the analysis revealed that the device uses a proprietary Nordic Semiconductor SoC and firmware that provides a proprietary **device firmware update (DFU)** service that performs a typical firmware update on an nRF5 device.

The device does not appear in a search result of the Bluetooth product listing, which means the *Bluetooth Qualification Process* has not been completed for the DuoEK ECG Tracker.

Vulnerability Scanning: As with the previous devices, the vulnerability scanner did not find any published vulnerabilities for this device or its built-in software components.

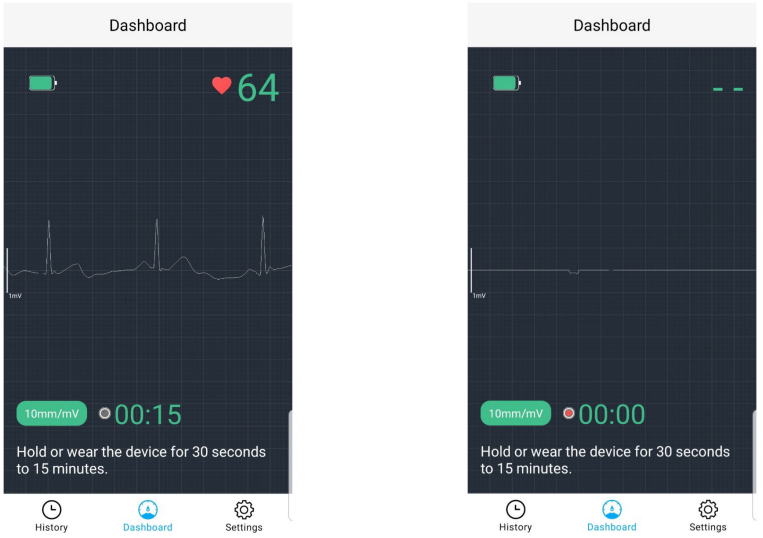
Attack Execution

- **Passive traffic logs:** The mobile application does not provide any authentication and, similarly to the other devices, the log files are stored on the phone in clear text.
- **Firmware code and data protection:** Because the device uses a proprietary DFU service for firmware updates, we were unable to access the device binary. The analysis of the ECG data stream showed that transferred traffic has rather low entropy (around 0.4) and is, therefore, likely unencrypted. Figure 9 shows the entropy graph for a one minute sample of ECG data from the DuoEK device.
- **Spoofing:** The ECG tracker was successfully spoofed.
- **Eavesdropping:** The framework was able to successfully monitor and capture traffic between the device and application. In spite of lack of encryption, parsing transferred data was not straightforward. Additional analysis showed that the tracker embedded a dynamic timestamp value in the communication, which made interpretation of the data representation less clear, and requiring additional steps to translate and manipulate the data.
- **Replay:** We were able to capture heart rate data. Replay attacks, however, were not successful on the device. Since traffic representation was not clear, automatic execution of a replay attack was not feasible.
- **Tampering:** The device did not expose the device version information through public services. It did, however, provided battery level information under *Battery Service* and granted access to tamper with this value. When doing so, it was possible to make the mobile application view the device's battery as fully charged, regardless of its actual status. The tracker has no on-board IO capabilities such as displays to show the legitimate value, therefore, the user would never know the actual state of the battery on the device. This could cause problems in a scenario when the user or the patient is in a serious condition and no warning could be given that the ECG tracker battery is dead.
- **MITM:** Data tampering, although not possible for all services, was successfully executed for the battery level service. Hence, the device should be considered vulnerable to MITM attack.
- **Fuzzing:** The application normally represents ECG data by illustrating a graph of incoming data as shown in Figure 8(a). Fuzzing the data breaks the normal representation of data that is defined by the application, rendering the application unable to read and display it, as shown in Figure 8(b). Nevertheless, because the device includes built-in storage, the original record is restored and is re-synchronized in the log history on the application each time the device is connected.

Summary. No records of the DuoEK ECG tracker were found in the *Bluetooth product listing*, which means that this device had not passed the Bluetooth SIG standard qualification tests. There were also no reported vulnerabilities for this device, the built-in SoC, or the mobile application. Unlike the previous devices, the DuoEK was more resilient to the attacks attempted as part of the proposed framework.

8 DISCUSSION

The comprehensive security evaluation of any device is a challenging task. The sources difficulties range from the proprietary nature of the protocols and undisclosed device configurations, to a simple lack of access to the devices. These significantly limit the ability of researchers to properly analyze the exposure of devices to security threats, and consequently, provide assurances of device security. Many of these concerns have been raised by the security community, emphasizing the fact that obscuring details about software and hardware components alone is insufficient to secure these devices [63, 76]. Nevertheless, medical devices are commonly perceived as "black boxes" even



(a) Normal heart rate signal (b) Fuzzed signal

Fig. 8. ECG signals.

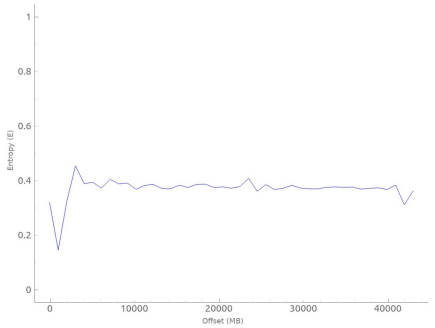


Fig. 9. Entropy of DuoEK ECG data.

if some information about their security weaknesses (e.g., via CVE, CVSS) is known. As such, our proposed framework aims to provide a pseudo black box-style security assessment of wearable health monitoring devices. That is, it requires no specific information to be known about the device ahead of time.

The framework is designed to follow a testing roadmap, collect the necessary information, and execute attack scenarios with minimal human intervention. The framework was designed to minimize the amount of human intervention, yet like in other research, the testing requires a manual installation of mobile applications and the necessary adaptations to connect health monitoring device. Some attack testing operations might also require human involvement. For example, this happens in cases when the framework discovers vulnerabilities in a process of fuzzing, and we need to further analyze logs to understand the device flaws and how an attack scenario that can be embedded into the framework to be handled automatically.

Our experimental results support several points for consideration by the community:

Table 5. Summary of Devices' Security Assessment

Activity	Athos Smart Apparel	Purifit B521A smartband	DuoEK ECG tracker
Information collection			
Bluetooth product listings	Registered	Not registered	Not registered
Fingerprinting *Public GATT services:* Device Information service Battery service Heart rate service	 Manufacturer Name Serial Number Firmware Revision Software Revision Battery level	 	 Manufacturer Name Battery level Heart Rate Measurement Body Sensor Location
Proprietary GATT services:		Firmware Revision Manufacturer code Battery level Heart Rate Measurement Pace Rate Measurement	Device firmware update
Vulnerabilities scan	No publicly listed vulnerabilities	No publicly listed vulnerabilities	No publicly listed vulnerabilities
Network traffic analysis			
Passive traffic logs	Available	Available	Available
Mobile app authentication	Required	Required	No authentication
Attacks			
Firmware code protection source code binary code	 Not available Encrypted	 Not available Encrypted	 Not available No access
Data encryption	Not encrypted	Not encrypted	Not encrypted (yet dynamic values are added for protection)
Spoofing	Successful	Successful	Successful
Eavesdropping (Active traffic capture)	Successful	Successful	Successful
Replay	Successful - Replayed previously captured traffic (command and data) to device and application.	Successful - Replayed previously captured traffic (command and data) to device and application.	Unsuccessful - Application does not accept previous data as new record.
Tampering	Successful - Tampered device information (battery level and software revision string).	Successful - Tampered device information (firmware revision, battery level, heart and pace rate data).	Partly successful - Tampered device information (only battery level).
MITM	Successful	Successful	Successful
Fuzzing	Successful - Application does not validate incoming data.	Successful - Application does not validate incoming data.	Unsuccessful - Both device and application are protected against invalid data.
Denial-of Service attack	Application crashed as a result of fuzzing	Application did not crush as a result of fuzzing. Data tampering attempts ('find phone', 'take photo') resulted in phone being disabled.	Unsuccessful

- *Lack of certification:* In spite of efforts to regulate and manage security vulnerabilities present in current medical devices, many vendors either see them as hindrance or fail to see their value. For example, despite being "mandatory" for products using Bluetooth, two of the devices that we tested are not certified by the *Bluetooth SIG*.
- *Disregard of available security mechanisms:* In spite of security measures available within the BLE protocol, the tested devices ignored most of the standardized built-in mechanisms, instead compensating using add-on techniques. All three devices used the just-works pairing method, which is a default pairing method that provides almost no security. To add protec-

tion, all devices implemented a proprietary handshake procedure after pairing. Nevertheless, the exchanged values were static, hence we were still able to intercept the communication. The lack of necessary protection allowed us to mount various attack scenarios, including eavesdropping, replay, and tampering.

- *Lack of encryption:* Eavesdropping, enabled by a lack of communication encryption, is one of the common security problems in medical devices. Most devices transmitting health data do so in clear text [41]. Our data analysis confirms this; none of the devices encrypted the traffic, which allowed unauthorized eavesdropping, capturing and altering of user data. Although also transmitted in clear-text, the ECG tracker was the only device that embedded a timestamp in the communication, which (even if inadvertently) made it more resilient to attacks.
- *No proper input validation:* The lack of proper input validation in the application for the Athos device resulted in our ability to cause application failure. Thus, validating input data and discarding malformed data may be used to prevent denial of service attacks and potential misdiagnosis based on incorrect data.
- *Firmware binary protection:* Storing the firmware binary in the memory of the smartphone (as was done with the Athos suit and the Purify tracker) allowed access to the binary file and consequently made analysis substantially easier. It is recommended that storing the firmware binary on the device be avoided. If not possible, then the binary file should minimally be encrypted, to prevent tampering with device functionality and to make reverse engineering more difficult.

9 CONCLUSION

Evaluating the security of IoT devices is a challenging task, as they are characterized by a vast number of mobile devices, components across different platforms with diverse connectivity, and face numerous threats. This is especially challenging with medical and health technologies, as the implications of vulnerabilities could include privacy, financial, and health threats. In this work, we presented a novel generalizable framework for evaluating the security of BLE-enabled wearable monitoring devices and demonstrated its feasibility of implementation by validating it on three commercial BLE-enabled devices and applications. The proposed framework encompasses four main phases—fingerprinting, vulnerability scanning, data analysis, and attack execution—which are designed to follow our security assessment methodology.

We were able to (1) scan the tested devices for services and characteristics, (2) obtain additional details about the devices from the online *Bluetooth public product listings*, (3) inspect the device for known vulnerabilities based on the information obtained, (4) capture and store the traffic between the device and the application, (5) perform analysis of the data and the device binary, (6) examine encryption of data and firmware, and (7) execute various attacks including spoofing, eavesdropping, replay, tampering, and fuzzing attacks on the devices and their applications. The results of our case studies show that all three tested devices are subject to man-in-the-middle attack, and consequently information disclosure, tampering, and fuzzing. Potential counter-measures identified through this process included adopting a secure pairing method to protect against a MITM attack, encrypting the data or adding dynamic elements such as time to the data to protect against information disclosure and replay attack, and considering input validation to protect against fuzzing.

The proposed framework could be adopted as a basis for testing and certification of personal wearable devices, or during the design of new devices. Although this work focused on implementation for BLE devices, it could be deployed for other protocols as well. The attack conductor module could be enhanced by adding additional attack types that may be identified in future works. With

these additional attack use cases, we plan to determine quantitative metrics such as Attack Code Coverage, Attack Surface Area Coverage, Number of zero day vulnerability detected, and Severity Score of zero-day vulnerability, with a larger set of devices. Furthermore, any improvement in fuzz testing (code coverage and input generation/mutation) could improve the ability of the framework to detect vulnerabilities and, consequently, assess security. Finally, the framework could also be extended to support devices with IO capabilities that adopt other pairing methods, such as passkey and numeric comparison.

Despite the rapid growth and proliferation of technology that is impacting and improving our daily life, it is increasingly important to address concerns about privacy and safety - especially when it comes to healthcare. Maintaining a balance between security and design goals remains a challenging task and requires closer collaboration between manufacturers, security researchers, and clinicians. Doing so will be essential in developing proper mechanisms for designing secure and reliable devices for users and patients, alike.

REFERENCES

[1] 2010. Binwalk. Retrieved November 29, 2019 from https://www.refirmlabs.com/binwalk/.

[2] 2014. High Level Organization of the Standard, Penetration Testing Execution Standard (PTES). Retrieved November 3, 2019 from http://www.pentest-standard.org/index.php/Main_Page.

[3] John Padgette, John Bahr, Mayank Batra, Marcel Holtmann, Rhonda Smithbey, Lidong Chen, and Karen Scarfone. 2017. Guide to Bluetooth Security, NIST Special Publication 800-121 Revision 2.

[4] 2019. Bluetooth Core Specification, ver. 5.1, Bluetooth SIG. Retrieved from https://www.bluetooth.com/bluetooth-resources/bluetooth-core-specification-v5-1-feature-overview/.

[5] 2019. Bug Reporting, Profiles and Logs. Retrieved November 3, 2019 from https://developer.apple.com/bug-reporting/profiles-and-logs/?platform=ios.

[6] 2019. Kismet. Retrieved November 29, 2019 from https://www.kismetwireless.net.

[7] 2019. OpenVAS—Open Vulnerability Assessment Scanner. Retrieved November 29, 2019 from https://www.openvas.org/.

[8] Ahmad W. Atamli and Andrew Martin. 2014. Threat-based security analysis for the internet of things. In *Proceedings of the International Workshop on Secure Internet of Things (SIoT'14)*. IEEE, 35–43.

[9] Attestation of Global Compliance. [n.d.]. Buetooth Qualification Process.

[10] Taha Belkhouja, Amr Mohamed, Abdulla K. Al-Ali, Xiaojiang Du, and Mohsen Guizani. 2018. Light-weight solution to defend implantable medical devices against man-in-the-middle attack. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM'18)*. IEEE, 1–5.

[11] P. K. Binu, Karun Thomas, and Nithin P. Varghese. 2017. Highly secure and efficient architectural model for IoT based health care systems. In *Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI'17)*. IEEE, 487–493.

[12] Inc. Bluetooth SIG. 2018. Bluetooth market update.

[13] A. J. Burns, M. Eric Johnson, and Peter Honeyman. 2016. A brief chronology of medical device security. *Commun. ACM* 59, 10 (2016), 66–72.

[14] Carmen Camara, Pedro Peris-Lopez, and Juan E. Tapiador. 2015. Security and privacy issues in implantable medical devices. *Journal of Biomedical Informatics* 55, (2015). 272–289

[15] Siemens CERT. 2019. Vulnerability in Laboratory Diagnostics Products from Siemens Healthineers. Retrieved from https://cert-portal.siemens.com/productcert/txt/ssa-832947.txt.

[16] Siemens CERT. 2020. SSA-616199: BlueKeep Vulnerability Identified in RAPIDPoint 500 Operating on Windows XP. Retrieved from https://cert-portal.siemens.com/productcert/pdf/ssa-616199.pdf.

[17] Hsinchun Chen. 2011. Smart health and wellbeing [Trends & Controversies]. *IEEE Intell. Syst.* 26, 5 (2011), 78–90.

[18] Gerald Combs et al. 2019. Wireshark. Retrieved November 29, 2019 from https://www.wireshark.org/.

[19] IHE PCD Technical Committee. 2015. Medical Device Software Patching. Retrieved from https://www.ihe.net/uploadedFiles/Documents/PCD/IHE_PCD_WP_Patching_Rev1.1_2015-10-14.pdf.

[20] Brian Cusack, Bryce Antony, Gerard Ward, and Shaunak Mody. 2017. Assessment of security vulnerabilities in wearable devices. In *the Proceedings of 15th Australian Information Security Management Conference, 5–6 December, 2017, Edith Cowan University, Perth, Western Australia*. 42–48.

[21] Office of Public Affairs Department of Justice. 2010. Medical Device Manufacturer Guidant Charged in Failure to Report Defibrillator Safety Problems to FDA. Retrieved from https://www.justice.gov/opa/pr/medical-device-manufacturer-guidant-charged-failure-report-defibrillator-safety-problems-fda.

[22] Mohamed Elhoseny, Gustavo Ramírez-González, Osama M. Abu-Elnasr, Shihab A. Shawkat, N. Arunkumar, and Ahmed Farouk. 2018. Secure medical data transmission model for IoT-based healthcare systems. *IEEE Access* 6 (2018), 20596–20608.

[23] FDA. 2017. Firmware Update to Address Cybersecurity Vulnerabilities Identified in Abbott's (formerly St. Jude Medical's) Implantable Cardiac Pacemakers: FDA Safety Communication. Retrieved from https://www.fda.gov/MedicalDevices/Safety/AlertsandNotices/ucm573669.htm.

[24] FDA. 2019. Cybersecurity Vulnerabilities Affecting Medtronic Implantable Cardiac Devices, Programmers, and Home Monitors: FDA Safety Communication. Retrieved from https://www.fda.gov/medical-devices/safety-communications/cybersecurity-vulnerabilities-affecting-medtronic-implantable-cardiac-devices-programmers-and-home.

[25] FDA. 2019. Cybersecurity Vulnerabilities in Certain GE Healthcare Clinical Information Central Stations and Telemetry Servers: Safety Communication. Retrieved from https://www.fda.gov/medical-devices/safety-communications/cybersecurity-vulnerabilities-certain-ge-healthcare-clinical-information-central-stations-and.

[26] FDA. 2019. SweynTooth Cybersecurity Vulnerabilities May Affect Certain Medical Devices: FDA Safety Communication. Retrieved from https://www.fda.gov/medical-devices/safety-communications/sweyntooth-cybersecurity-vulnerabilities-may-affect-certain-medical-devices-fda-safety-communication.

[27] Center for Devices and Radiological Health. 2017. Deciding When to Submit a 510(k) for a Software Change to an Existing Device, FDA-2016-D-2021. Retrieved from https://www.fda.gov/regulatory-information/search-fda-guidance-documents/deciding-when-submit-510k-software-change-existing-device.

[28] Mengmeng Ge, Jin B. Hong, Walter Guttmann, and Dong Seong Kim. 2017. A framework for automating security analysis of the Internet of Things. *J. Netw. Comput. Appl.* 83 (2017), 12–27.

[29] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. 2011. They can hear your heartbeats: Non-invasive security for implantable medical devices. In *Proceedings of the ACM SIGCOMM Computer Communication Review*, Vol. 41. ACM, 2–13.

[30] R. Goyal, N. Dragoni, and A. Spognardi. 2016. Mind the tracker you wear: A security analysis of wearable health trackers. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. 131–136.

[31] Kristen N. Griggs, Olya Ossipova, Christopher P. Kohlios, Alessandro N. Baccarini, Emily A. Howson, and Thaier Hayajneh. 2018. Healthcare blockchain system using smart contracts for secure automated remote patient monitoring. *J. Med. Syst.* 42, 7 (2018), 130.

[32] Z. Guo, I. G. Harris, Yutong Jiang, and L. Tsaur. 2017. An efficient approach to prevent Battery Exhaustion Attack on BLE-based mesh networks. In *Proceedings of the International Conference on Computing, Networking and Communications (ICNC'17)*. 1–5.

[33] Keijo Haataja and Pekka Toivanen. 2010. Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures. *Trans. Wireless. Comm.* 9, 1 (Jan. 2010), 384–392.

[34] M. L. Hale, D. Ellis, R. Gamble, C. Waler, and J. Lin. 2015. SecuWear: An open source, multi-component hardware/software platform for exploring wearable security. In *Proceedings of the IEEE International Conference on Mobile Services*. 97–104.

[35] Matthew L. Hale, Kerolos Lotfy, Rose F. Gamble, Charles Walter, and Jessica Lin. 2019. Developing a platform to evaluate and assess the security of wearable devices. *Digit. Commun. Netw.* 5, 3 (2019), 147–159.

[36] Daniel Halperin, Thomas S. Heydt-Benjamin, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. 2008. Security and privacy for implantable medical devices. *IEEE Perv. Comput.* 7, 1 (2008), 30–39.

[37] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. 2008. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'08)*. IEEE, 129–142.

[38] Jeremy A. Hansen and Nicole M. Hansen. 2010. A taxonomy of vulnerabilities in implantable medical devices. In *Proceedings of the 2nd Annual Workshop on Security and Privacy in Medical and Home-care Systems*. ACM, 13–20.

[39] Shaikh Shahriar Hassan, Soumik Das Bibon, Md Shohrab Hossain, and Mohammed Atiquzzaman. 2018. Security threats in bluetooth technology. *Comput. Secur.* 74 (2018), 308–322.

[40] Jason Healey, Neal Pollard, and Beau Woods. 2015. *The Healthcare Internet of Things: Rewards and Risks*. Atlantic Council Report, March 18 (2015).

[41] Kit Huckvale, José Tomás Prieto, Myra Tilney, Pierre-Jean Benghozi, and Josip Car. 2015. Unaddressed privacy risks in accredited health and wellness apps: A cross-sectional systematic assessment. *BMC Med.* 13 (09 2015), 214.

[42] Ponemon Institute. 2017. Medical Device Security: An Industry under Attack and Unprepared to Defend. Retrieved from http://www.counciltoreduceknowncybervulnerabilities.org/wp-content/uploads/2017/05/Ponemon-Synopsys-Report-Final.pdf.

[43] Jennifer Ann Janesko. 2018. *Bluetooth Low Energy Security Analysis Framework*. Technical Report. RHUL–ISG–2018–5. 5 April 2018. Information Security Group.
[44] Slawomir Jasek. 2016. Gattacker. Retrieved from https://github.com/securing/gattacker.
[45] Niraj K. Jha, Anand Raghunathan, and Meng Zhang. 2013. Securing medical devices through wireless monitoring and anomaly detection. September 19 2013. US Patent App. 13/839,768.
[46] Kaspersky Lab. [n.d.]. Kaspersky Security Bulletin: Kaspersky Lab Threat Prediction for 2018. Retrieved from https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2018/03/07164714/KSB_Predictions_2018_eng.pdf.
[47] Chunxiao Li, Anand Raghunathan, and Niraj K. Jha. 2011. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *Proceedings of the IEEE 13th International Conference on e-Health Networking, Applications and Services*. IEEE, 150–156.
[48] Angela M. Lonzetta, Peter Cope, Joseph Campbell, Bassam J. Mohd, and Thaier Hayajneh. 2018. Security vulnerabilities in Bluetooth technology as used in IoT. *J. Sens. Actuator Netw.* 7, 3 (2018), 28.
[49] Eduard Marin, Dave Singelée, Flavio D. Garcia, Tom Chothia, Rik Willems, and Bart Preneel. 2016. On the (in) security of the latest generation implantable cardiac defibrillators and how to secure them. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*. 226–236.
[50] Svetlin Nakov. 2018. *Practical Cryptography for Developers*.
[51] Offensive Security. [n.d.]. Kali OS. Retrieved November 29, 2019 from https://www.kali.org/.
[52] Kristen O'Loughlin, Martha Neary, Elizabeth C. Adkins, and Stephen M. Schueller. 2019. Reviewing the data security and privacy policies of mobile apps for depression. *Internet Intervent.* 15 (2019), 110–115.
[53] Sode Pallavi and V. Anantha Narayanan. 2019. An overview of practical attacks on BLE Based IOT devices and their security. In *Proceedings of the 5th International Conference on Advanced Computing & Communication Systems (ICACCS'19)*. IEEE, 694–698.
[54] Achilleas Papageorgiou, Michael Strigkos, Eugenia Politou, Efthimios Alepis, Agusti Solanas, and Constantinos Patsakis. 2018. Security and privacy analysis of mobile health applications: The alarming state of practice.*IEEE Access* 6 (2018), 9390–9403.
[55] Youngseok Park, Yunmok Son, Hocheol Shin, Dohyun Kim, and Yongdae Kim. 2016. This ain't your dose: Sensor spoofing attack on medical infusion pump. In *Proceedings of the 10th USENIX Workshop on Offensive Technologies (WOOT'16)*.
[56] Hoai Luan Pham, Thi Hong Tran, and Yasuhiko Nakashima. 2018. A secure remote healthcare system for hospital using blockchain smart contract. In *Proceedings of the IEEE Globecom Workshops (GC Wkshps'18)*. IEEE, 1–6.
[57] Vernessa Pollard and Mahnu Davar. 2017. FDA's Evolving Civil Money Penalty Authority: Simple Violations Can Lead to Major Costs. Retrieved from https://www.mastercontrol.com/gxp-lifeline/civil_money_penalty_authority_0609/.
[58] Laurie Pycroft and Tipu Z. Aziz. 2018. Security of implantable medical devices with wireless connections: The dangers of cyber-attacks. *Expert Rev Med Devices* 15, 6 (2018), 403–406.
[59] Jerome Radcliffe. 2011. Hacking medical devices for fun and insulin: Breaking the human SCADA system. In *Black Hat Conference Presentation Slides*, Vol. 2011.
[60] Mahmudur Rahman, Bogdan Carbunar, and Madhusudan Banik. 2013. Fit and vulnerable: Attacks and defenses for a health monitoring device. In *Proceedings of the 6th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs'13)*.
[61] Apala Ray, Vipin Raj, Manuel Oriol, Aurelien Monot, and Sebastian Obermeier. 2018. Bluetooth low energy devices security testing framework. In *Proceedings of the IEEE 11th International Conference on Software Testing, Verification and Validation (ICST'18)*. IEEE, 384–393.
[62] VynZ Research. [n.d.]. Global Network Connected Medical Devices Market Was Valued at USD 20.0 Billion in 2018, Observing a CAGR of 24.0 during 2019–2024: VynZ Research. Retrieved from https://www.globenewswire.com/news-release/2019/12/23/1964239/0/en/Global-Network-Connected-Medical-Devices-Market-was-valued-at-USD-20-0-billion-in-2018-Observing-a-CAGR-of-24-0-during-2019-2024-VynZ-Research.html.
[63] Michael Rushanan, Aviel D. Rubin, Denis Foo Kune, and Colleen M. Swanson. 2014. SoK: Security and privacy in implantable medical devices and body area networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'14)*. IEEE Computer Society, 524–539.
[64] Mike Ryan. 2013. Bluetooth: With Low energy comes low security. In *Proceedings of the 7th USENIX Workshop on Offensive Technologies (WOOT'13)*. USENIX Association.
[65] S. Seneviratne, Y. Hu, T. Nguyen, G. Lan, S. Khalifa, K. Thilakarathna, M. Hassan, and A. Seneviratne. 2017. A survey of wearable devices and challenges. *IEEE Commun. Surv. Tutor.* 19, 4 (2017), 2573–2620.
[66] Shachar Siboni, Asaf Shabtai, Nils O. Tippenhauer, Jemin Lee, and Yuval Elovici. 2016. Advanced security testbed framework for wearable IoT devices. *ACM Trans. Internet Technol.* 16, 4 (2016), 26.
[67] Pallavi Sivakumaran and Jorge Blasco. 2019. A study of the feasibility of co-located app attacks against {BLE} and a large-scale analysis of the current application-layer security landscape. In *Proceedings of the 28th USENIX Security Symposium (USENIX Security'19)*. 1–18.

[68] J. H. Song, R. Poovendran, J. Lee, and T. Iwata. 2006. *The AES-CMAC Algorithm.* Technical Report 4493. Internet Society Requests for Comment (RFCs).

[69] Statista Research Department. [n.d.]. Bluetooth Low Energy (BLE) Enabled Devices Market Volume Worldwide, from 2013 to 2020. Retrieved August 2, 2020 from https://www.statista.com/statistics/750569/worldwide-bluetooth-low-energy-device-market-volume/.

[70] Da-Zhi Sun, Yi Mu, and Willy Susilo. 2018. Man-in-the-middle attacks on secure simple pairing in bluetooth standard V5.0 and its countermeasure. *Pers. Ubiq. Comput.* 22, 1 (Feb. 2018), 55–67.

[71] CyberMDX Research Team. 2020. Vulnerability in GE CARESCAPE, ApexPro, and Clinical Information Center (CIC) Systems CISA Advisory (ICSMA-20-023-01). Retrieved from https://www.cybermdx.com/vulnerability-research-disclosures/cic-pro-and-other-ge-devices.

[72] Ali Tekeoglu and Ali Şaman Tosun. 2016. A testbed for security and privacy analysis of iot devices. In *Proceedings of the IEEE 13th International Conference on Mobile Ad Hoc and Sensor Systems (MASS'16).* IEEE, 343–348.

[73] Timothy Trippel, Ofir Weisse, Wenyuan Xu, Peter Honeyman, and Kevin Fu. 2017. WALNUT: Waging doubt on the integrity of mems accelerometers with acoustic injection attacks. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'17).* IEEE, 3–18.

[74] Gail A. Van Norman. 2016. Drugs, devices, and the FDA: part 2: An overview of approval processes: FDA approval of medical devices. *J. Am. Coll. Cardiol.* 1, 4 (2016), 277–287.

[75] Yue Wu, Yicong Zhou, George Saveriades, Sos Agaian, Joseph P. Noonan, and Premkumar Natarajan. 2013. Local Shannon entropy measure with statistical tests for image randomness. *Inf. Sci.* 222 (Feb. 2013), 323–342. https://doi.org/10.1016/j.ins.2012.07.049

[76] T. Yaqoob, H. Abbas, and M. Atiquzzaman. 2019. Security vulnerabilities, attacks, countermeasures, and regulations of networked medical devices—a review. *IEEE Commun. Surv. Tutor.* 21, 4 (2019), 3723–3768.

[77] Muhammad Yaseen, Waseem Iqbal, Imran Rashid, Haider Abbas, Mujahid Mohsin, Kashif Saleem, and Yawar Abbas Bangash. 2019. MARC: A novel framework for detecting MITM Attacks in eHealthcare BLE Systems. *J. Med. Syst.* 43, 11 (2019), 324.

[78] Qiaoyang Zhang, Zhiyao Liang, and Zhiping Cai. 2019. Developing a new security framework for bluetooth low energy devices. *Comput. Mater. Contin.* 59, 2 (2019), 457–471.

[79] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. 2019. Automatic fingerprinting of vulnerable BLE IoT devices with static UUIDs from mobile apps. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'19).* Association for Computing Machinery, New York, NY, 1469–1483.

Received May 2020; revised October 2020; accepted January 2021