WearSec: Towards Automated Security Evaluation of Wireless Wearable Devices

Bernhards Blumbergs; Ēriks Dobelis; Pēteris Paikens; Krišjānis Nesenbergs; Kirils Solovjovs; Artis Rusiņš

Blumbergs, B., Dobelis, Ē., Paikens, P., Nesenbergs, K., Solovjovs, K., Rušiņš, A. (2022). WearSec: Towards Automated Security Evaluation of Wireless Wearable Devices. In: Reiser, H.P., Kyas, M. (eds) Secure IT Systems. NordSec 2022. Lecture Notes in Computer Science, vol 13700. Springer, Cham. https://doi.org/10.1007/978-3-031-22295-5_17

WearSec: Towards Automated Security Evaluation of Wireless Wearable Devices

Bernhards Blumbergs[1][0000−0001−9679−6282] bernhards.blumbergs@lumii.lv, Ēriks Dobelis[1][0000−0002−8691−3614] eriks.dobelis@lumii.lv, Pēteris Paikens[1][0000−0002−5939−5436] peteris.paikens@lumii.lv, Krišjānis Nesenbergs[2][0000−0002−2445−2891] krisjanis.nesenbergs@edi.lv, Kirils Solovjovs[2][0000−0003−1700−3286] kirils.solovjovs@edi.lv, and Artis Rušiņš[2][0000−0001−5292−7201] artis.rusins@edi.lv

[1] Institute of Mathematics and Computer Science, University of Latvia, Raina blvd. 29, Riga, Latvia

[2] Institute of Electronics and Computer Science, 14 Dzerbenes st., Riga, Latvia

Abstract. Wearable devices are becoming more prevalent in the daily life of society, ranging from smartwatches, and fitness bracelets to accessories and headphones. These devices, both from their hardware manufacturing and wireless firmware development perspectives may possess drawbacks. In recent years security researchers have uncovered a series of vulnerabilities. In this paper we introduce the concept and describe the key ideas towards the development of an automated security evaluation prototype for wireless wearable devices using device fingerprinting, as well as passive and active vulnerability identification. Furthermore we describe the technical approaches, challenges, and implementation choices we faced while developing the first stages of the prototype for this concept and handling full-spectrum Bluetooth analysis with software-defined radio.

1 Introduction

In recent years there has been a steady rise in wearable device usage - from 325 million connected wearable devices in the world in 2016 to 1105 million in 2022 [23]. These devices may contain personally identifiable information about the user, electronic health record (EHR) data [7], geolocation information, and may be connected to private networks. Thus, such a combination may be introducing privacy and security risks. Compromised wearable devices may reveal significant personal information, and may facilitate the execution of attacks by an adversary leading to, for example, the breach of private network security, covert account takeover, and performing direct or indirect targeted attacks. Common wearable devices use wireless communication standards, such as Bluetooth Classic (BT), Bluetooth low energy (BLE), or WiFi, to communicate with other devices. Therefore these standards, their implementation, and inherent vulnerabilities are the focus of our research [20].

This paper provides the following contributions:

1. automated wireless wearable device security evaluation prototype conceptual design, its specific functionality requirements, and patent description;
2. the overview of the first stage of the prototype technical implementation.

This paper is structured as follows – Chapter 2 covers the related work on wireless device fingerprinting and security assessment; Chapter 3 describes the proposed system architecture conceptual prototype considerations for performing automated security evaluation of wireless wearable devices; Chapter 4 describes the specific implementation stages, issues encountered during the first stage of the prototype implementation, and solutions to overcome these problems; Chapter 5 discusses future work related to this research.

2 Related Work

For the envisioned approach we are aiming at a dual use system which can be used for unsupervised security review of wireless devices across multiple wireless protocols, identifying them and checking for known vulnerabilities, and also as a supporting testbed for experts performing vulnerability research. For the latter, there is specific research in IoT/wearable area such as the SecuWear platform [17, 18] and others using the popular Ubertooth platform [16], however it has hardware limitations preventing from capturing the full data transmitted (e.g. Bluetooth Classic Enhanced Data Rate packets, and missed packets due to channel hopping), and they focus on manual exploration of vulnerabilities as opposed to automated fuzzing.

In our proposal the first step in a potential attack against wearable devices is the detection and identification of the device, such as fingerprinting to identify either a model or a specific device; followed by scanning for known vulnerabilities and applying fuzzing approaches to identify new potential vulnerabilities.

2.1 Device detection and fingerprinting

Device fingerprinting [40] is a method where the target device is identified using its unique features and imperfections of its hardware components and/or higher layer features. A combination of these unintended features and possibly values derived from these features are called device fingerprints. Fingerprinting process may be split into two parts: feature extraction and classification. The focus of this section is feature extraction. Successful fingerprinting may lead to serious security risks if the identified chipset has publicly known attack vectors with available exploitation proof-of-concept (PoC) code.

Feature extraction can start from the moment when the target device is powered and starts broadcasting or responding to incoming signals. For example, there are cases when a device may be identified by its signal level ramp time while the transmitter is turning on or off (i.e., transient-based radio frequency fingerprinting) [22]. The start and stop time of the transient signal can be found using Bayesian detection, variance fractal dimension threshold detection, phase

detection, mean change point detection, or permutation entropy and generalized likelihood ratio test [38]. Further analysis of transient signal frequency and phase spectrum can also be done, however, the drawback of this method is that since the transient time is short high sample rates are required for accurate analysis. This method requires capturing to be started before the device (or its transmitting circuit) is powered and activated.

The implemented wireless standard determines carrier frequencies, which shall be used by the device. Since this frequency is generated by the use of a non-ideal radio frequency (RF) circuit in the transmitter some carrier offset is expected. This offset can also be used as one item of the RF fingerprint. To achieve a reliable data transmission it is important to deal with this expected CFO (carrier frequency offset) and it is usually solved in the transmission standard itself. For example, in Bluetooth, every advertising data packet contains some bits at the beginning of transmission, which the receiver uses to estimate and correct the CFO. The longer these training sequences are, the more precise the CFO estimate can be achieved.

Devices, which have both Bluetooth and WiFi functionality often use a combo chipset to combine both of these standards. Wireless protocol bits are encoded in I and Q channels, resulting in hardware imperfections such as IQ (In-phase and quadrature component) imbalance, IQ offset, and previously mentioned CFO [14]. IQ offset is the shift from the center of the constellation diagram [37]. IQ imbalance can be split into two imperfections: amplitude error and phase error. To quantify both of these errors EVM (error vector magnitude) [28] can be calculated. IQ imbalance occurs because the carrier oscillator signal at the Q channel is never perfectly offset by 90 degrees when compared to the I channel, also it is very hard to achieve identical gain for both I and Q channel oscillator signal. IQ offset occurs because the baseband signal has a DC component and carrier frequency signal leaking through the mixer into the baseband signal. The usage of a combined chipset means that even though Bluetooth uses a GFSK (Gaussian frequency-shift keying) modulated signal, which might be implemented without the separation of I and Q signals, it will still have IQ imperfections.

There have also been successful attempts at fingerprinting using machine learning technology for image recognition, where the used image is compiled from RF samples. For example, waterfall images, constellation diagrams, and wavelet coefficients projected onto time scale plane or multiple windowed time diagram images compiled into one [26] [2] [24].

It is possible to fingerprint devices using features, which are present at higher levels of communication. There is no need for expensive specialized hardware to perform higher-layer fingerprinting. One could use a device, such as, Ubertooth [16], various WiFi dongles, or some other commercial-off-the-shelf tools. Researchers at [6] have used freely available open-source tools to extract GATT (Generic attribute profile), which contains a UUID (Universally unique identifier) of the device, which may reveal information about the device, used protocols as defined by the Bluetooth specification [3], service classes and profiles, GATT services, and manufacturer of thhe device.

There has been related work done on using response to non-standard IEEE 802.11 frame with modified MAC header as a WiFi fingerprinting method. Different devices responded differently to frames with control flags and frame types prohibited by IEEE 802.11 standards. This method yielded good results for fingerprinting access points, but not so good for client devices [4].

Researchers in [31] propose the following network parameters to be used as IEEE 802.11 device fingerprints:

1. Transmission rate as different NICs (Network interface cards) have a different distribution of used transmission rates;
2. Frame size, which depends on the data being sent;
3. Access time of the medium, the time device waits after transmission before going idle and starting to broadcast frames;
4. Time between received frames.

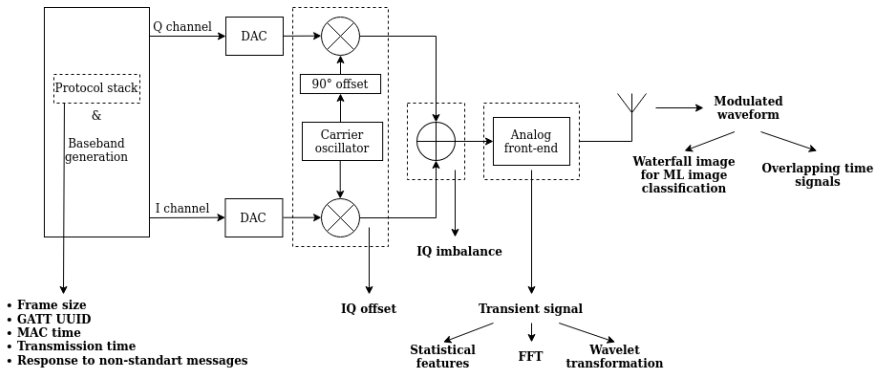

Fig. 1. Overview of extractable fingerprints

Figure 1 summarizes all of the previously described features, which may be used as fingerprints by their origin.

2.2 Vulnerability identification

Within this project, the two main directions for vulnerability identification will be pursued:

1. known and publicly released vulnerability applicability, based on the DUT (device-under-test) passive fingerprinting and intercepted communication analysis;
2. active attempts to trigger unknown vulnerabilities by the use of *black-box* fuzz testing techniques.

To allow the identification of applicable publicly released vulnerabilities, enough details, such as, operating system and firmware build versions, have to be collected through the process of passive DUT fingerprinting. A vulnerability database assembled by researchers or available databases, such as, the well-known Exploit-DB [32], may be used to search for the best match of applicable vulnerabilities based on the extracted fingerprinting information. In cases, when existing databases do not include known vulnerabilities and related proof-of-concept code, the researchers should evaluate the database's completeness and perform its enrichment, if necessary. This approach to vulnerability identification heavily depends on two key factors: 1) implementing a set of efficient methods for device fingerprinting, and 2) developing a complete and structured database of applicable vulnerabilities. Within this research, a custom-structured database of identified and publicly released vulnerabilities related to wearable devices, their firmware, and their protocol stack implementations, will be attempted to be assembled to be used in the initial stages of the wireless DUT security assessment.

Discovery of the vulnerabilities would mostly be done through one or a combination of multiple approaches, such as, fuzz testing, reverse engineering, or source code analysis. However, in the scope of this research, only black-box fuzz testing is assessed due to having limited or no prior information related to the DUT implementation. Fuzz testing or fuzzing is a software testing technique aimed at uncovering issues, such as, coding errors and security vulnerabilities, by the use of random or malformed data to trigger an unexpected behavior [11, 13]. In a nutshell, the two most commonly used fuzzing methods have been recognized [29]:

1. mutation-based (also referred to as coverage-guided or dumb fuzzing) - is aimed at introducing changes by modifying the existing values blindly (e.g., random values, bit-flips, and other binary modifications), that may keep the input valid, but trigger new or unexpected behavior. The injected test cases are derived from known good data, such as, captured Bluetooth communication frames;
2. generation-based (also referred to as behavior-based or intelligent fuzzing) - focuses on describing data and state models based upon the communication protocol specification (e.g., RFC).

The most recent relevant related work on Bluetooth fuzzing over the air is the Braktooth publication [9], detailing an approach and an engineering solution based on the modified firmware of an Espressif ESP32 development kit for instrumenting interventions in an otherwise well-behaved Bluetooth stack. There is also a similar work on Bluetooth Low Energy testing [10]. Attacks on BT (Bluetooth) and other accessory firmware are especially insidious, because in some popular configurations (e.g., iPhones) the Bluetooth chip will be powered and can be engaged in activities even when the primary device is turned off [8]. This is a relevant consideration within the risk profile of this project.

In contrast to the static vulnerability identification through DUT fingerprinting and lookup in a database of known vulnerabilities, the fuzz testing will

heavily rely on three core aspects: 1) valid test case sample acquisition or their specification in accordance with the standard, 2) identification of protocol fields and their mutation strategies, and 3) required larger time-span for fuzz testing process execution and appropriate means of its orchestration to identify a likely anomaly or recover from a dead-end state. Within this research, both mutation-based and generation-based approaches may be employed, depending on the availability and openness of the wireless protocol specification. With the primary effort being set on mutation-based fuzz testing the loss of code coverage is anticipated [29]. The use of mutation-based fuzzing would permit the test case automated generation based on the captured frames during the device fingerprinting process or when performing an active interrogation, while maintaining limited or no knowledge of the underlying communication protocols. This, in turn, may provide support for a wider range of wearable DUT testing, while considering a likely increased time and process context management penalties. The main goal for the initial active vulnerability identification is focused on identifying anomalous conditions, and not on in-depth assessment and zero-day vulnerability analysis for exploit development.

There has been significant work in the field of software vulnerability identification with fuzz testing technique development and tool implementations, both from academic [25], industry [12], and community [33] perspectives. The publication [21] lists and assess 32 research papers, which introduce novel fuzzing strategies and their tool implementation, however, by no means this is a conclusive list. Furthermore, it has to be noted, that the field of vulnerability assessment and fuzz testing is dynamic and constantly changing, with new projects appearing, and existing ones either being deprecated, abandoned, or reworked. To mention a few notable projects with a primary focus on mutation-based strategies – AFL (American Fuzzy Lop) [15], Boofuzz [35], Bbuzz [27], and Zzuf [5]; and with a primary focus on generation-based strategies – Peach [34] (now part of GitLab), Spike [19] (abandoned), and Defensics [39] (commercial solution).

3 Prototype Concept Description

To perform an automated security evaluation of wireless wearable devices, we introduce the key principles, define functional requirements and design an operational prototype of the technology. The anticipated design of a finalized prototype is similar to a security gate and utilizes an array of sensors, performing a variety of functional tests and assessments in a fully automated manner on devices being carried through this gate. Such a conceptual approach has been chosen to facilitate the security assessment of the wearable devices while being worn openly or being hidden by their wearer. Such an approach would permit the achievement of at least the following high-level effects: 1) assessing the security level of wearable devices worn by their user, 2) evaluation of identified vulnerabilities and their severity level, 3) prohibition of wearing and use of wearable devices at places with a certain security level, and 4) identification of hidden or clandestine wearable devices.

The core automated functionality tests and assessment of the wireless wearable devices are directed toward the following activities:

1. passive fingerprinting of wearable devices based on their radio spectrum broadcast interception. In this stage, the received broadcast data is being interpreted to identify at least the hardware chip manufacturer and its model and, if possible - the device itself, based on the transmitted data, its unique patterns, and signatures;
2. passive fingerprinting-based identification of known vulnerabilities of wearable devices. In this stage, the collected fingerprint data is queried against a database, which combines the known and publicly disclosed generic or specific vulnerabilities and their metadata for wireless wearable devices;
3. active interaction with wearable devices to validate the identified vulnerability, acquire further data for fingerprinting or perform automated tests to attempt triggering unknown vulnerabilities. In this stage, the identified vulnerabilities are further probed to validate their exploitability in case there is a publicly disclosed proof-of-concept exploit code. Secondly, active interaction evokes responses from the device, which allows for further fingerprinting. And thirdly, automated vulnerability identification and probing are done by the use of fuzz-testing approaches at the wireless protocol level.

The prototype process flow has the following blocks in a hierarchical order (Fig. 2):

1. wireless device-under-test (DUT). One or multiple commercial-off-the-shelf (COTS) wearable wireless devices under test;
2. wireless sensors with related hardware-protocol driver (HPD) modules. The prototype design includes a set of necessary hardware to cover the respective radio frequency bands used by the COTS wearable devices. This can be implemented using a chip designed for the protocols or generic SDR (software-defined radio). To enable the interaction with the DUT, related HPD integrations are required at the prototype model's operating system level (e.g., kernel modules and libraries). HPD modules are supervised by relevant wrappers to ensure the interaction with higher abstraction levels of the overall automated interaction process. It is important to note that the prototype needs to be able to gather not only data, which is relevant for normal device operation, but, also, side information (e.g. signal amplitude, frequency offset, etc.) for fingerprinting, and it needs to generate radio protocol variations, which sometimes might be outside normal bounds of the protocol specification. The wrapper permits interaction with a specific wireless sensor through the message queuing process;
3. asynchronous message queuing and management. This block allows the higher abstraction layers to inject and receive data intended for one or multiple HPDs;
4. domain-specific language (DSL) is a common API-based construct permitting to form the messages for injection or receiving data from the messaging queue. This layer permits either creation and interpretation of raw data packets or standard messages according to the wireless protocol specifications;

5. traffic recording and filtering capture all raw traffic as well as other parameters of the RF signal traversing HPD. This permits interaction with recorded wireless packet or frame data via DSL construct for purposes, such as, validation of delivered message compliance and received data interpretation for passive fingerprinting or assessing the results of fuzz-testing operations;
6. The user scripts are the top-level entity permitting the definition and conduct of activities by interacting via the DSL API. The user script may be written in any scripting or programming language allowing the JSON-based API interaction with the DSL.

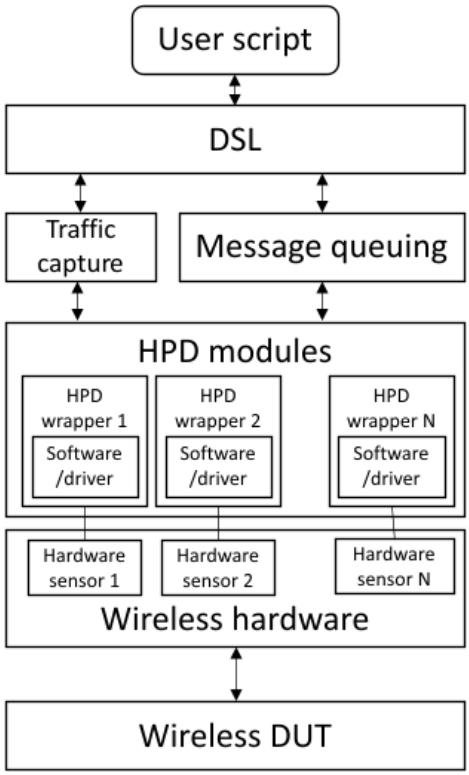

Fig. 2. Prototype concept process flow

The stages of the operational technological prototype development are addressed in the following order:

Stage 0. identification and acquisition of wearable DUTs and prototype hardware specification definition;

Stage 1. recording of the wearable device wireless communications in a broad spectrum range;

Stage 2. assessment and fingerprinting of the wireless communication patterns;
Stage 3. labeled data-set creation of recorded wireless communications for machine learning-assisted automated fingerprinting;
Stage 4. active device probing and assessment.

In this paper, the scope is kept within stage 1 of the prototype implementation, with respective implementation considerations and identified issues described in section 4, while the overall high-level concept is described in submitted patent application [30].

4 Implementation Considerations

This chapter focuses on the Stage 1 implementation of the described prototype and aims to provide the attempted approaches taken, identified challenges and issues, and describe any applicable solutions.

The prototype will include both protocol-specific chips and a generic SDR mechanism. Readymade chips allow an easy way to communicate using standard protocol mechanisms without complex implementation efforts. At the same time, they are limited only to those functions that are allowed by the manufacturer and will neither allow significant deviations from the protocol nor allow the gathering of nuances of the radio signal. SDR allows any conceivable data collection and generation of any signal variations as long as we can implement those. The rest of this chapter elaborates on the choice and implementation of the SDR mechanism as readymade chip usage is relatively straightforward.

4.1 Frequency bandwidth consideration

To capture all the channels of Bluetooth and Bluetooth LE simultaneously a recording at an 80 MHz wide spectrum band is required. Within our tests, none of the low-cost SDR devices offered a bandwidth as wide as 80 MHz, and the use of multiple devices was considered to ensure the capture of the required bandwidth combined between the devices.

4.2 Choice of SDR device

The first device we tested was a widely known hobbyist SDR device *HackRF One* by Great Scott Gadgets. It provides up to 20 million complex samples per second, thus we would need four such devices to record the whole 80 MHz bandwidth.

Another positive side to HackRF was previous references on using multiple hardware synchronized devices for recording (such as described on this blog: `https://olegkutkov.me/tag/hackrf/` and a different method implemented in firmware [1].

Following initial tests with HackRF, we came to the conclusion that described method of hardware synchronization does not easily lead us to results. Synchronization of the HackRF devices on the firmware level was clearly described only

for two devices, but we needed four. Though from the description we could guess that it is extendable to more devices, the way to do this was not obvious.

A major issue we encountered with HackRF was the differing quality of the four devices. Signal level, signal-to-noise-ratio (SNR), and other characteristics differed widely across the devices, sometimes by more than 10 dB. This introduced an inconvenience in data processing, as it was clear that for each recording the four streams have to be put through calibration routines to be convenient for analysis, and some issues (e.g. differing SNR) cannot be easily taken away even with calibration. Also, this creates a risk that our recordings might not be easily repeatable by other HackRF devices.

Of course, this is likely not something that is wrong with HackRF One device design itself, but more of different manufacturing and supply chains for each of the devices. As HackRF is an open hardware design, they are produced not only by the original manufacturer but, also, by other unlicenced manufacturers with possibly different quality assurance processes and possible variations in hardware components.

We had an option to procure other four HackRF One devices with additional emphasis on ensuring they come from the original manufacturer. This was complicated from a procurement perspective as the original manufacturer is not represented locally, and we did not have control of the procurement processes of local suppliers. Thus we made a decision to utilize existing HackRF One devices by the team for individual exploratory experiments and look for alternative SDR devices.

We identified 3 manufacturers providing devices within our budget and quality criteria:

- Ettus Research USRP B series
- Nuand BladeRF
- Lime Microsystems LimeSDR

After limited experiments with BladeRF we chose Ettus Research products due to better software support in GNU Radio, which was part of the plan for the software stack. Also, "UHD: USRP source" (the block providing integration with Ettus Research SDRs from GNU Radio) provides tagging of the data capture overruns, which turned out to be a repeating issue in our experimentation when capturing the full Bluetooth spectrum. We did not identify comparable features in competing products.

The eventually chosen hardware setup was combining two Ettus Research USRP b205mini devices with synchronized clocks, which provide 56 Msps each, covering the required 80 MHz bandwidth.

4.3 Data capture bandwidth issues

The first issue we encountered during experiments was - the limited throughput of the USB connection. Initial recordings had a mismatch of data file size compared to the expected value (based on recording time and sample size in bytes).

Unfortunately, none of the software tools (GNU radio and HackRF native CLI tools) we used had an in-built capability to warn us of the issue. We concluded that data loss is due to loss in USB connection as data loss further down the processing shall be identified by the software processing stack.

The second issue was the disk writing speed. We wanted to use GNU radio for initial processing steps as it provided the convenience of visual feedback (via built-in frequency and waterfall sinks) and a library of built-in filters and other tools. The minimal data rate for raw data from the device is 2x12bitsx80Msps, which equals 240 MBps with optimal data packing, which could match the SSD writing speed of a single drive. GNU Radio typically uses 2x32bit floating point numbers to represent a complex data stream and saves them into the file in this format. Thus the size of the stream becomes 8 bytes x 80 Msps, which equals 640 MBps. This in turn exceeded the writing speed of our commodity SSD hardware.

Theoretically, we could have used a more optimal data recording format, but this would decrease the convenience of using the default format and would require additional work on creating data packing code. Also, we knew that we cannot predict all the specific needs for the next stages, and we might need to store processed data for which converting to 2x12bit complex numbers might lead to loss of precision, therefore it would create additional software incompatibility risks and overhead, which cannot be fully estimated at the moment. Ultimately, we chose to keep the default data format of GNU radio and to overcome disk writing speed limitations we chose a RAM-based disk for the recording process. If we would need to decrease stored sampled sizes, we can add postprocessing in the future.

Another gain of this approach was faster data processing speed as reading and writing data to/from RAM-based disk is significantly faster than commodity SSD.

The only drawback (considered minor by us) was another procurement need - additional RAM for the processing computer. Fortunately, this is a common procurement need and was easy to implement. As a result, the following workstation for data collection and analysis was acquired for data acquisition and processing:

- CPU: Intel Xeon Silver 4208 (32) @ 3.2 GHz;
- RAM: 1 TB;
- HDD: 10 TB;
- SSD: 512 GB;
- Kernel: 5.15.0-25-generic;
- OS: Ubuntu 22.04 LTS.

4.4 Interference by surrounding signals

As the 2.4GHz band is full of other signals, such as, WiFi, Zigbee, and other BT devices, which are out of the scope of this research, one of the challenges in preparing a radio data stream dataset is making a clean recording without the noise. Even though the most obvious solution would be a Faraday cage, it

is not sufficient, because the signals would be reflected within the cage multiple times leading to interference, thus for this purpose, it was decided that we need to design an electronic anechoic chamber environment. Such an environment insulates external transmissions as a Faraday cage but also has internal absorbers, that absorb any internal signal without reflecting it. The dimensions of such environment and absorbers must be comparable to wavelengths of interest, thus the dimensions cannot be too small. We opted for external dimensions of 1m, that satisfy this requirement. Also, in order to reduce any leaking of external signals all data lines for conducting the experiment are optically disconnected to isolate internal and external signals of this environment. This also allows us to test malicious transmissions, which should not be allowed in public environments. Additionally, the power supply for test equipment must also be filtered in order to isolate external signals further.

The electronic anechoic chamber we are currently acquiring has the following technical properties:

- Outside dimensions: 1.0m x 1.0m x 1.0m;
- Inside dimensions (from the edge of absorbers): 0.8m x 0.8m x 0.8m;
- RF parameters: 100 dB attenuation from 150 kHz to 10 GHz;
- Connections: 4 optically decoupled USB 3 connectors, power sockets that supply filtered 220V 16A;
- Material: Stainless steel with foam pyramid absorbers.

By using an electronic anechoic chamber we are ensuring that devices under test are sources of all activity in the radio frequency spectrum, no external signals will interfere with the experiments, and also the experiments will not interfere with external signals.

4.5 Lack of open source, instrumentable low-level protocol stack

A key limitation for protocol analysis and interaction (especially, using SDR) is the availability of the open-source, instrumentable low-level protocol stack. Bluetooth is a complex set of protocols, which are not described clearly thus it is not trivial to implement correct behavior. Our requirements include particular modifications to the protocol to attempt the exploitation of the protocol vulnerability, and the available implementations of lower protocol layers are all closed source compiled firmware, which will perform only standards-compliant behavior.

The key part is the ability to receive and send arbitrary, modified, or noncompliant Bluetooth frames without having to reimplement the full Bluetooth protocol functionality. The approach proposed by Braktooth [9] relies on a reverse-engineered and modified version of the existing Espressif firmware for ESP32 systems that allows it to intercept BT packets and forward them to a computer controlling logic, which may craft an appropriate BT response packet in the time that is required to respond. Other authors propose an approach that instruments and observes the firmware on a phone or computer [36], which allows more effective fuzzing but is limited to the specific chipsets used on phones and computers.

5 Conclusions and Future Work

The main conclusion from the current implementation stage is that the identified engineering challenges are not an obstacle to implementing the proposed prototype concept and this would be a usable framework for automated security evaluation of wireless wearable devices. The development of the prototype is currently a work in progress, with the key hardware and software infrastructure working as a proof of concept, but still needing development on the DSL/HPD interface part. We hope that the lessons learned from our practical experimentation will be useful for other researchers working on low-cost RF protocol analysis.

The immediate future work includes recording and publishing an RF dataset for analysis and direct experimentation with protocol fuzzing, replicating the existing experiments, and extending it to a broader set of devices, focusing specifically on wearable devices. The dataset should be open and expandable in the future in a standardized way.

Future work towards automated vulnerability discovery should focus on the following aspects:

1. identification, implementation, and validation methods and approaches for wireless wearable device fingerprinting;
2. assembly of a dedicated vulnerability database, based on known and publicly released vulnerabilities affecting wireless wearable devices;
3. evaluation, design, and validation of applicable black-box fuzz testing approaches and methods for mutation-based and generation-based strategies;
4. implementation of the fuzz testing test case generation, process management, and physical wireless wearable device state identification and recovery, with limited feedback or interaction capabilities.

6 Acknowledgements

This research is funded by the Latvian Council of Science, project "Automated wireless security analysis for wearable devices", project No. LZP-2020/1-0395.

References

1. Bartolucci, M., del Peral-Rosado, J.A., Estatuet-Castillo, R., Garcia-Molina, J.A., Crisci, M., Corazza, G.E.: Synchronisation of low-cost open source sdrs for navigation applications. In: 2016 8th ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing (NAVITEC). pp. 1–7. IEEE (2016)
2. Bertoncini, C., Rudd, K., Nousain, B., Hinders, M.: Wavelet fingerprinting of radio-frequency identification (rfid) tags. IEEE Transactions on Industrial Electronics **59**(12), 4843–4850 (2011)
3. Bluetooth SIG, Inc.: Assigned numbers. `https://www.bluetooth.com/specifications/assigned-numbers/`, accessed: 26.08.2022

4. Bratus, S., Cornelius, C., Kotz, D., Peebles, D.: Active behavioral fingerprinting of wireless devices. In: Proceedings of the first ACM conference on Wireless network security. pp. 56–61 (2008)
5. Caca Labs: zzuf - multi-purpose fuzzer, http://caca.zoy.org/wiki/zzuf, accessed: 2022-08-30
6. Celosia, G., Cunche, M.: Fingerprinting bluetooth-low-energy devices based on the generic attribute profile. In: Proceedings of the 2nd International ACM Workshop on Security and Privacy for the Internet-of-Things. pp. 24–31 (2019)
7. Cilliers, L.: Wearable devices in healthcare: Privacy and information security issues. Health Information Management Journal **49**(2-3), 150–156 (2020). https://doi.org/10.1177/1833358319851684, https://doi.org/10.1177/1833358319851684, pMID: 31146589
8. Classen, J., Heinrich, A., Reith, R., Hollick, M.: Evil never sleeps: When wireless malware stays on after turning off iphones. In: Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks. p. 146–156. WiSec '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3507657.3528547, https://doi.org/10.1145/3507657.3528547
9. Garbelini, M.E., Chattopadhyay, S., Bedi, V., Sun, S., Kurniawan, E.: Braktooth: Causing havoc on bluetooth link manager (2021)
10. Garbelini, M.E., Wang, C., Chattopadhyay, S., Sumei, S., Kurniawan, E.: {SweynTooth}: Unleashing mayhem over bluetooth low energy. In: 2020 USENIX Annual Technical Conference (USENIX ATC 20). pp. 911–925 (2020)
11. Garg, P.: Fuzzing: Mutation vs. generation, https://resources.infosecinstitute.com/topic/fuzzing-mutation-vs-generation/, accessed: 2022-08-28
12. GitLab: Devsecops with gitlab, https://about.gitlab.com/solutions/dev-sec-ops/, accessed: 2022-08-30
13. GitLab DEVSECOPS blog: What is fuzz testing?, https://about.gitlab.com/topics/devsecops/what-is-fuzz-testing/, accessed: 2022-08-28
14. Givehchian, H., Bhaskar, N., Herrera, E.R., Soto, H.R.L., Dameff, C., Bharadia, D., Schulman, A.: Evaluating physical-layer ble location tracking attacks on mobile devices. IEEE Symposium on Security and Privacy (SP) (2022)
15. Google: american fuzzy lop, https://github.com/google/AFL, accessed: 2022-08-28
16. Great Scott Gadgets: Ubertooth one. https://greatscottgadgets.com/ubertoothone/, accessed: 26.08.2022
17. Hale, M.L., Ellis, D., Gamble, R., Waler, C., Lin, J.: Secu wear: An open source, multi-component hardware/software platform for exploring wearable security. In: 2015 IEEE International Conference on Mobile Services. pp. 97–104. IEEE (2015)
18. Hale, M.L., Lotfy, K., Gamble, R.F., Walter, C., Lin, J.: Developing a platform to evaluate and assess the security of wearable devices. Digital Communications and Networks **5**(3), 147–159 (2019)
19. ImmunitySec: Spike, https://www.kali.org/tools/spike/, accessed: 2022-08-30
20. Ken Research: Worldwide wearable devices cybersecurity market. https://www.kenresearch.com/defense-and-security/security-devices/worldwide-wearable-devices/179018-16.html, accessed: 28.08.2022
21. Klees, G., Ruef, A., Cooper, B., Wei, S., Hicks, M.: Evaluating fuzz testing. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 2123–2138. CCS '18, Association for Computing Ma-

chinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243804, https://doi.org/10.1145/3243734.3243804
22. Köse, M., Taşcioğlu, S., Telatar, Z.: Rf fingerprinting of iot devices based on transient energy spectrum. IEEE Access **7**, 18715–18726 (2019). https://doi.org/10.1109/ACCESS.2019.2896696
23. Laricchia, F.: Number of connected wearable devices worldwide from 2016 to 2022. https://www.statista.com/statistics/487291/global-connected-wearable-devices/, accessed: 28.08.2022
24. Li, B., Cetin, E.: Waveform domain deep learning approach for rf fingerprinting. In: 2021 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5. IEEE (2021)
25. Liang, J., Wang, M., Chen, Y., Jiang, Y., Zhang, R.: Fuzz testing in practice: Obstacles and solutions. In: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER). pp. 562–566 (2018). https://doi.org/10.1109/SANER.2018.8330260
26. Liu, D., Wang, M., Wang, H.: Rf fingerprint recognition based on spectrum waterfall diagram. In: 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP). pp. 613–616. IEEE (2021)
27. Lockout: Bbuzz: a bit-aware network protocol fuzzing and reverse engineering framework, https://github.com/lockout/Bbuzz, accessed: 2022-08-28
28. Mahmoud, H.A., Arslan, H.: Error vector magnitude to snr conversion for nondata-aided receivers. IEEE Transactions on Wireless Communications **8**(5), 2694–2704 (2009)
29. Miller, C., Peterson, Z.N.: Analysis of mutation and generation-based fuzzing. DefCon 15 (2007), https://defcon.org/images/defcon-15/dc15-presentations/Miller/Whitepaper/dc-15-miller-WP.pdf
30. Nesenbergs, K., Paikens, P., Blumbergs, B., Rusins, A., Dobelis, E.: Apparatus and method for wireless security analysis of wearable devices (2022), lV Patent application No. EPLV202200000033380
31. Neumann, C., Heen, O., Onno, S.: An empirical study of passive 802.11 device fingerprinting. In: 2012 32nd International Conference on Distributed Computing Systems Workshops. pp. 593–602. IEEE (2012)
32. Offensive Security: Exploit-DB, https://www.exploit-db.com/, accessed: 2022-08-26
33. OWASP: Fuzzing, https://owasp.org/www-community/Fuzzing, accessed: 2022-08-30
34. Peach: Peach fuzzer community edition, https://peachtech.gitlab.io/peach-fuzzer-community/, accessed: 2022-08-30
35. Pereyda, J.: boofuzz: Network protocol fuzzing for humans, https://github.com/jtpereyda/boofuzz, accessed: 2022-08-30
36. Ruge, J., Classen, J., Gringoli, F., Hollick, M.: Frankenstein: Advanced wireless fuzzing to exploit new bluetooth escalation targets. In: 29th USENIX Security Symposium (USENIX Security 20). pp. 19–36. USENIX Association (Aug 2020), https://www.usenix.org/conference/usenixsecurity20/presentation/ruge
37. Sköld, M., Yang, J., Sunnerud, H., Karlsson, M., Oda, S., Andrekson, P.A.: Constellation diagram analysis of dpsk signal regeneration in a saturated parametric amplifier. Optics Express **16**(9), 5974–5982 (2008)
38. Soltanieh, N., Norouzi, Y., Yang, Y., Karmakar, N.C.: A review of radio frequency fingerprinting techniques. IEEE Journal of Radio Frequency Identification **4**(3), 222–233 (2020)

39. Synopsys: Defensics fuzz testing, https://www.synopsys.com/software-integrity/security-testing/fuzz-testing.html
40. Xu, Q., Zheng, R., Saad, W., Han, Z.: Device fingerprinting in wireless networks: Challenges and opportunities. IEEE Communications Surveys & Tutorials **18**(1), 94–104 (2015)