Pentesting in IoT

Threat Modeling in IoT and Attack Classification

Seminar Paper

Ausgewählte Kapitel der IT-Security

Written by
Agry Zarza

Personal Identifier
1710475113

Submitted on
8 January 2020

Index of Abbreviations

AES	Advanced Encryption Standard
BD_ADDR	Bluetooth Device Address
BLE	Bluetooth Low Energy
BR/EDR	Basic Rate/Enhanced Data Rate
BTBB	Bluetooth Baseband
CMAC	Cipher-based Message Authentication Code
CRC	Cyclic Redundancy Check
CSRF	Cross-Site Request Forgery
DNS	Domain Name System
DOS	Denial of Service
FHSS	Frequency Hopping Spread-Spectrum
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GUI	Graphical User Interface
HCI	Host Controller Interface
IBSG	Internet Business Solutions Group
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
JTAG	Joint Test Action Group
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local Area Network
LAP	Lower Address Part
LTK	Long Term Key
MITM	Man-in-the-middle
NFC	Near-Feld Communication
OS	Operating System
OWASP	Open Web Application Security Project
PAN	Personal Area Network
QR	Quick Response
RFID	Radio-Frequency Identification
SIG	Special Interest Group
SQLi	SQL-Injection
SSL	Secure Sockets Layer
STK	Short Term Key
TK	Temporary Key
TLS	Transport Layer Security
UHF	Ultra High Frequency
UPnP	Universal Plug and Play
USB	Universal Serial Bus
UUID	Universally Unique Identifier
XSS	Cross-Site-Scripting

Contents

1. Introduction

The Internet of Things (IoT) is a collective term for technologies of a global infrastructure comprised of heterogenous devices. With the growing popularity of smart devices and environments, manufacturers are obliged to provide proper security for their devices. An IoT penetration test is the assessment and examination of various components in an IoT device in order to enhance the device's security. Furthermore, its is critical that these devices are built with security-by-design to ensure that they cannot be attacked and exploited.
In the recent years, manufacturers have been trading efficient and well-designed security for usability of the device, mainly for users that are not technically skilled. Due to the increasing awareness of consumers of security questions, manufacturers have to rethink their position. As a result of that, IoT penetration testing business is thriving. The requirements and guidelines for penetration testing would vary from device to device. Nevertheless, a group of information security practitioners have been trying to implement a formal standard for the execution of penetration testing [1].
Additionally, many books on the topic of IoT penetration testing have been published recently. This research focuses on the guidlines provided in [2] and [3].

1.1 Aims and Objectives of this Research

This work focuses mainly on providing an overview of the current state of IoT penetration testing. In order to achieve that it provides background information for IoT and penetration testing approaches in chapter 2. Moreover, this chapter outlines the state-of-the-art of Bluetooth connection-handling and Bluetooth security. This will prepare the reader sufficiently for the practical examination in chapter 3.2, which focuses on Bluetooth sniffing. This research examination is intended to indicate the possibilites of penetration testing, while focusing on the specific aspect of Bluetooth security.
Additionally, this paper lists several relevant attack vectors in different technology contexts. For this purpose a threat model is detailed in chapter 3. This model is based on the Open Web Application Security Project (OWASP) vulnerability reports of 2014 and 2018.
Chapter 4 concludes this paper, discusses its limitations and outlines possibilites of future work.

2. Background Information

This chapter provides background information for technologies that are relevant to this research. Section 2.1 details the Internet of Things. Section 2.2 outlines Bluetooth in general with the main focus on Bluetooth Low Energy. Furthermore it provides an overview for Bluetooth security. Section 2.3 discusses different approaches to penetration testing.

2.1 Internet of Things (IoT)

The Internet of Things enables physical and virtual objects to be networked, cooperated and automated through a broad array of various network protocols. These objects are interconnected by various types of short-range wireless technologies such as Bluetooth (detailed in 2.2), ZigBee, Radio-Frequency Identification (RFID), sensor networks and through location-based technologies [4]. The CISCO Internet Business Solutions Group (IBSG) states that the IoT was "born" in 2008, were the number of devices connected to the internet exceeded the number of people. The number of connected devices grows fast and IBSG also predicts that 50 billion devices will be connected to the Internet in 2020 [5].
With the increasing popularity of IoT technology, smart home devices have become prominent. Examples can range from smart hubs over smart bulbs, to smart sensors for a variety of things. While manufacturers focus on advertising the usability, the "intelligence" and the efficiency of such tools, their security is still questioned. According to AdaptiveMobile, 80 percent of existing IoT devices are not adequately secured [6]. Gartner Inc. predicts that Worldwide IoT Security spending will rise from 1.1 billion dollars in 2017 to 3.1 billion dollars in 2021 [7]. Security is a dominant topic and is becoming vital for further adoption of the technology.

2.2 Bluetooth

Bluetooth is a technology for wireless data exchange over short distances. It uses short-wavelength Ultra High Frequency (UHF) radio waves to transmit data. Devices using Bluetooth communicate on ISM-bands (Industrial, Scientific and Medical radio bands) utilizing a frequency between 2,400 GHz and 2,483.5 GHz. The technology was standadized by IEEE as IEEE 802.15.1 in 2002. The Bluetooth Special Interest Group (SIG) maintains the standard and oversees developement of the specification. Manufactured IoT devices must meet those standards to be advertised as Bluetooth

devices [8].
Bluetooth can handle many devices at the same time by using a technique called Frequency Hopping Spread-Spectrum (FHSS). This technique enables transmitters change the frequency 1,600 times per second. Therefore, many devices can make use of the full radio spectrum. Consequently, it is highly unlikely for two devices to interfere with each other on the same frequency [8].
If a device connects to another via Bluetooth they form a piconet, also called a personal area network (PAN). The general topology of such a connection is a master/slave connection, where one master can form a piconet with up to 7 slaves. The devices involved in the piconet hop the frequency together in a pseudorandom manner, seeded by the master's mac address. The timing of the hops is determined by the master's clock [8].
There are several ways for devices to initiate a connection. For the preliminary work the potential slave device has to be put in discovery mode. This enables devices to listen on the 32 designated inquiry frequency channels for messages and respond to them. The inquiry process is assymetrical. The potential master device sends out inquiry packets on those inquiry frequency channels. After exchanging messages the devices eventually reach connected mode and form a piconet, while both devices may already be connected to other Bluetooth devices in a piconet [8].

2.2.1 Bluetooth Low Energy (BLE)

BLE was introduced in Bluetooth 4.0 is one of the most common technologies utilized by smart devices. In fact, BLE is explicitly designed for devices with resource and power constraints. BLE is effectively using lower data rate and less power than Bluetooth classic, thus significantly reducing battery consumption on smart devices. Table 2.1 shows the key differences between Bluetooth and BLE.

Characteristic	Bluetooth	BLE
Network topology	Scatternet	Star Bus
Power consumption	less than 30 mA	less than 15 mA
Speed	700 Kbps	1 Mbps
Range	less than 30 meters	up to 150 meters
Frequency Channels	79 channels from 2.400 GHz to 2.4835 GHz with 1 MHz spacing	40 channels from 2402 MHz to 2480 MHz
Latency in data transfer	up to 100ms	up to 3ms
message size in bytes	up to 358	8 to 47
Error detection	16 bit CRC (8 bit CRC for headers)	24 bit CRC
Security	64b/128b	128 bits AES
Application throughput	0.7 to 2.1 Mbps	less than 0.3 Mbps
Nodes per master	7	Unlimited

Table 2.1: Differences between Bluetooth Classic and Bluetooth Low Energy [8]

As detailed in table 2.1, BLE uses a slightly different FHSS scheme, because BLE consists of 40 different channels, 3 advertisement and 37 data channels. Fig. 2.1 shows the BLE stack and its three main layers: Application, Host and Controller. The second of two interact through the Host Controller Interface (HCI). The Controller's LE Physical Layer (PHY) is responsible for signal modulation and demodulation. In addition to that it also calculates the hopping pattern for FHSS. The Link Layer (LL) manages several things, including the device's Bluetooth address, encryption and connection initiation [2].

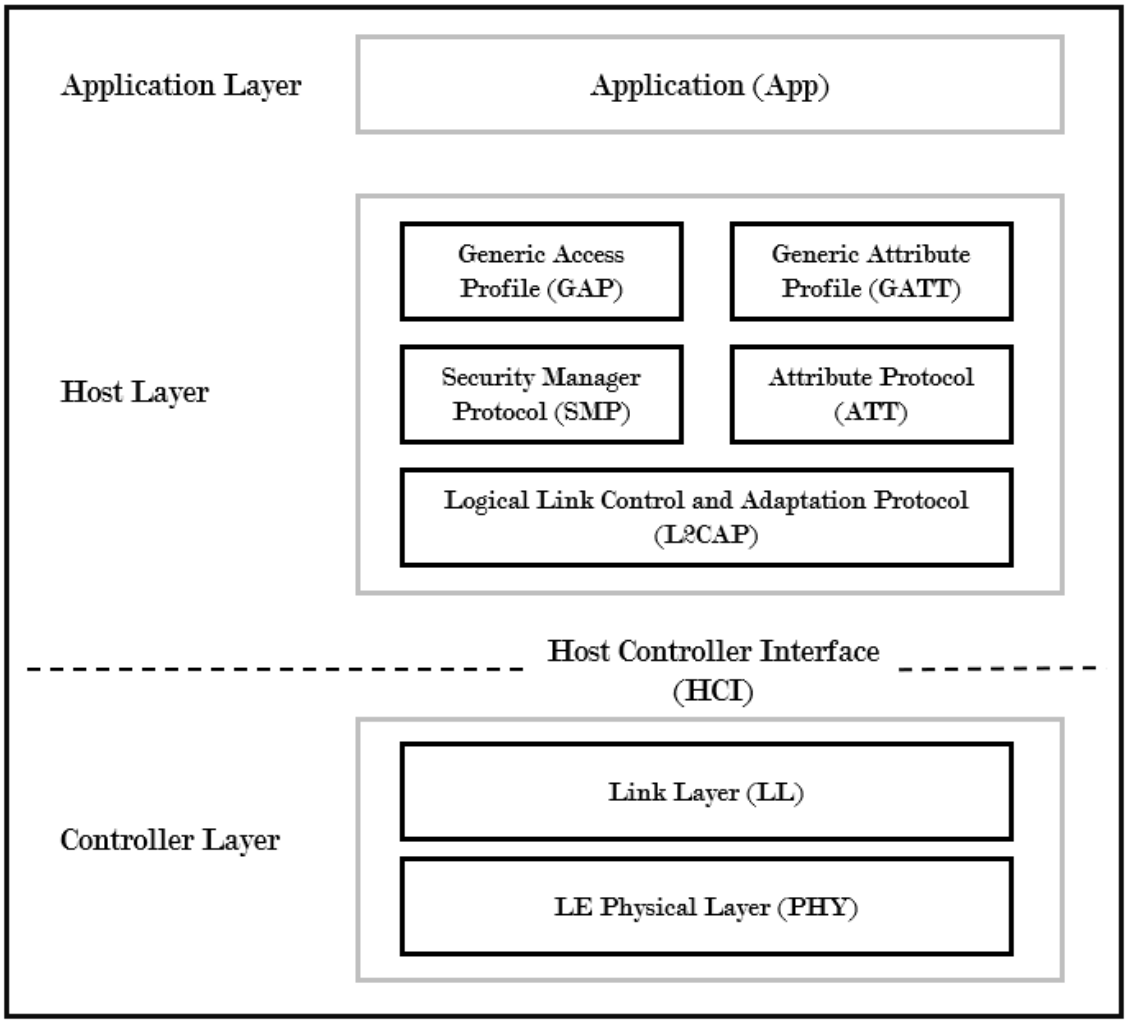


Figure 2.1: Three Main Layers of the BLE Stack [2]

The most relevant components of the Host layer for this research are Generic Access Profile (GAP), Generic Attribute Profile (GATT) and Logical Link Control and Adaptation Protocol (L2CAP). The task of L2CAP is encapsulating data from other layers in a proper packet structure. GAP is controlling a majority of the advertisements and also handles the role of the device in a specific connection. GATT is the component which manages the data exchange and performs operations like read, write and error handling. GATT manages the data by categorizing it as shown in Figure 2.2.

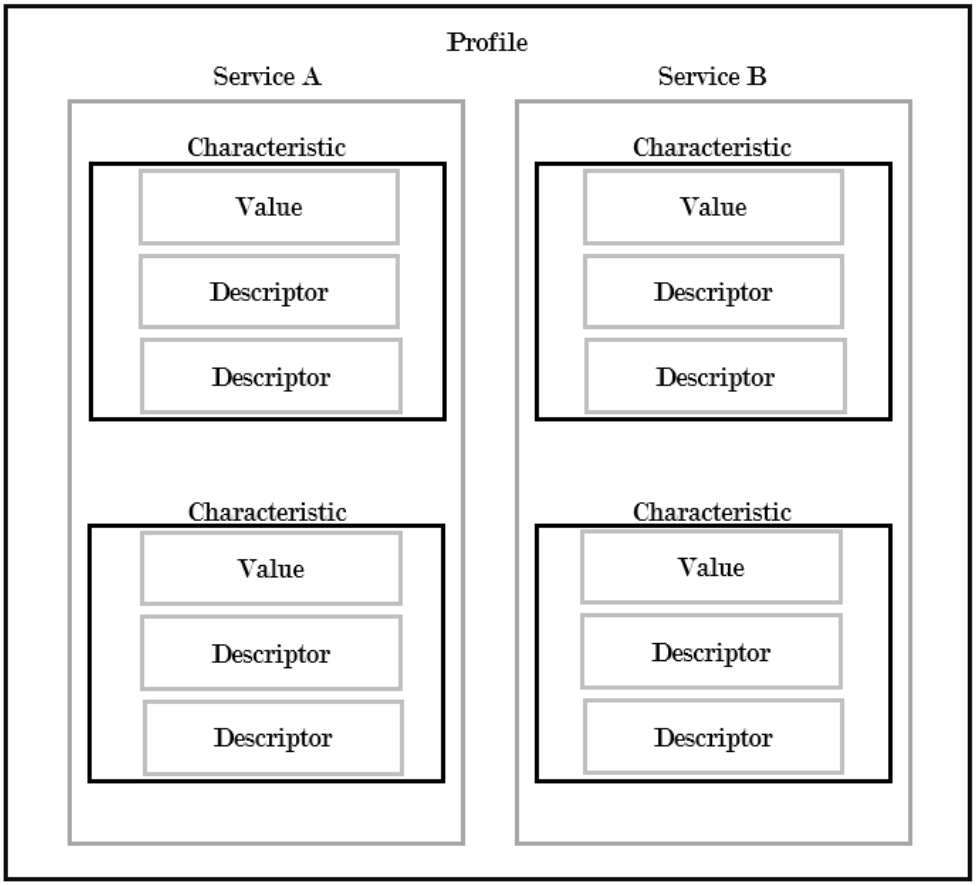


Figure 2.2: GATT Data Management and Attribute Allocation [2]

The entire data is enclosed within profiles, each containing services. Services have a specific universally unique identifier (UUID) for reference purposes. A service encloses characteristics. Characteristics also have UUIDs which can be referenced. A characteristic holds a value and several descriptors [2] .

2.2.2 Bluetooth Security

As Bluetooth is a wireless networking setup, security is a great concern. Devices are sending data over a wireless connection. This data can contain sensitive information and therefore precautions are needed to make sure those signals aren't intercepted. Especially the automated connection handling nature of Bluetooth benefits adversaries to gain information without permission.
BLE offers several security levels and security modes. There are four different security levels ranging from 1 through 4, with 4 being the most secure [8]:

- Security Level 1 supports unpaired communication with no security at all.
- Security Level 2 supports Advanced Encryption Standard Cipher-based Message Authentication Code (AES-CMAC) encryption for unpaired communication.

- Security Level 3 supports encryption and requires pairing.
- Security Level 4 supports Elliptic Curve Diffie-Hellman Ephimeral (ECDHE) exchange instead of AES-CMAC.

There are two BLE security modes and two additional modes, which are [8]:

- Security Mode 1 is referencing all security levels without data signing
- Security Mode 2 is referencing all security levels with data signing in paired as well as unpaired communications
- Mixed Security Mode is when a device needs to support both signed and unsigned data.
- Secure Connection Only Mode is the combination of Security Level 4 with Secure Mode 1, meaning all traffic involve authenticated connections and encryption only.

In the pairing process the devices agree upon everything related to security. The security measures are initiated by the Security Manager of the master. The slave may request the master to initiate pairing or other security procedures. That means that the devices exchange their capabilities and consequently decide on a specific pairing method, which are detailed in table 2.2.

Pairing Method	Description
Numeric Comparison	Both devices present the same Temporary Key (TK) on their respective displays. The user is asked to verify that the values match on both devices.
Just Works	This mode is especially for devices which have no display or input mechanism, such as earphones or a computer mouse. Technically, it is the same as Numeric Comparison, but the TK value is set to zero. Obviously there is no Man-in-the-middle (MITM) attack protection for this mode.
Passkey Entry	With this mode the TK value is displayed on one device, and the user is promped to enter the value into the other device.
Out of Band (OOB)	In this mode the key is exchanged through a different protocol than BLE, i.e. utilizing Near-Feld Communication (NFC) or using Quick Response (QR)-codes.

Table 2.2: BLE Pairing Methods [8]

The device's capabilities are exchanged through L2CAP values, which are not encrypted. Following that, the devices agree upon a Temporary Key (TK). The value of TK is an integer between 0 and 999999. The user is asked to verify the generated TK using a display for example when pairing a phone to a car. If the Just Works mode is selected, the value of TK is 0. The TK is ultimately used for generation of a

Short Term Key (STK). The STK itself is never transmitted between devices and is actually used to establish the Long Term Key (LTK). The LTK is then used by the paired devices for subsequent connections in most cases [8].
In [9] the author describes vulnerabilities of BLE security and exposes a critical security isse. The key exchange protocol is vulnerable to brute force attacks. It details that once a TK value is successfully found, the STK and LTK keys can also be otained by decrypting the corresponding keys.

2.3 Pentesting Approaches

Approaches to penetration testing can vary depending on the information that is available to the testers when examining a device. Therefore, the tests can be categorized as follows [2]:

- Black Box: The assessment is performed with no prior knowledge of the device's technology. These kind of tests are very common and can be performed for a relatively low cost.
- White Box: Testers are provided full access to the source code and have full knowledge of the technology of a device. These tests are more expensive due to its comprehensive testing nature. White Box assesments are typically performed by manufacturers themselves or third-party consulting firms, who are granted full access to the device and the infrastructure.
- Grey Box: These tests are a hybrid form of Black Box and White Box. Testers have partial knowledge of the device's technology when performing assessments

Ideally, a thorough assessment should also include the infrastruture the IoT device operates in and not only the gadget itself. It is important to note that the costs of the examination increase with the scope of the test.

3. Security of IoT devices

The first section 3.1.1 in this chapter provides a threat model and describes possible exploits. Additionally, it discusses implications of several vulnerabilities. In section 3.2 this research outlines a practical example of penetration testing based on the Ubertooth device, which is essentially a passive Bluetooth sniffer.

3.1 Threat Modeling

The Open Web Application Security Project (OWASP) is an organization that focuses on web security and application security. In this regard, it provides extensive documentation, including vulnerability assessment. The following section 3.1.1 outlines the top IoT vulnerabilities of 2014. In 2018 OWASP updated its classification which is shown in section 3.1.2. Furthermore, to offer the latest viewpoint on IoT security, the vulernability properties of each assessment are compared and mapped to one another.

3.1.1 OWASP Top 10 IoT Vulnerabilities (2014)

In 2014 OWASP published the Top 10 IoT Vulnerabilities [10]. They present vulerabilities, attack vectors, threat agents, security weaknesses, possible exploits and their respective impacts, and countermeasures against the threats. Additionally, they provide example attacks for each vulnerability. Table 3.1 shows attack surfaces and their characteristics, with Insecure Web Interface representing the biggest vulnerability.

This research continues to present brief explanations for each vulnerability. By listing countermeasures that need to be taken in order to secure the attack surface this work will indicate possible attack vectors implicitly.

A) Insecure Web Interface

Web Interfaces are tools that enable interaction between users and software running on a web server. These tools are very popular, since they can be accessed from many devices and require hardly any technical knowledge. Most IoT devices can be configured and mantained through web interfaces, for example a router is accessible using a default IP in the private LAN. Most of these interfaces are password-protected. However, those credentials may be inadequately secured.
OWASP recommends the following actions to secure the web interface [10]:

Attack Surface Area	Detectability	Exploitability	Technical Impact
Insecure Web Interface	easy	easy	severe
Insufficient Authentication or Authorization	easy	average	severe
Insecure Network Services	average	average	moderate
Lack of Transport Encryption	easy	average	severe
Privacy Concerns	easy	average	severe
Insecure Cloud Interface	easy	average	severe
Insecure Mobile Interface	easy	average	severe
Insufficient Security Configurability	easy	average	moderate
Insecure Software/Firmware	easy	difficult	severe
Poor Physical Security	average	average	severe

Table 3.1: OWASP Top 10 IoT Vulnerabilities (2014) [10]

- Enforcing a change of credentials upon setup of the device - never allow for default passwords and usernames
- Prevent information gathering, in particular ensuring robust password recovery mechanisms, which do not supply an attacker with information indicating a valid account
- Ensuring the web application code is not susceptible to vulnerabilities such as Cross-Site-Scripting (XSS), Cross-Site Request Forgery (CSRF), SQLi (injection) and others
- Ensuring credentials are not exposed in network traffic
- Enforcing strong passwords, i.e. minimum amount of characters with mandatory uppercase letters and special characters
- Prevent effective Brute Forcing by limiting login attempts to a maximum of five tries and ensuring account lockout after maximum amount of attempts

B) Insufficient Authentication or Authorization

This section overlaps with the previous one, but addresses all of the interfaces. Authentication can be insufficient when weak passwords are used or when credentials are poorly protected. Examples of poor credentials are usernames like "admin" or "user" and passphrases like "admin", "password" or "1234". In fact many vendors use these default values for an easy installation process and transfer the responsibility to the user to secure credentials sufficiently. Adversaries can use weaknesses of authentication or authorization mechanisms to gain access to the device. The proposed countermeasures are [10]:

- Enforcing strong passwords

- Ensuring granular access control, meaning the proper handling of permissions of who can access which interfaces
- Implement two factor authentication
- Ensuring secure password recovery mechanisms
- Enforcing password expiration and history checks, forcing users to reset their passwords
- Ensuring revocation of credentials
- Enforcing the requirement of app, device and server authentication while ensuring the uniqueness of the respective IDs

C) Insecure Network Services

Security components like firewalls and Intrusion Detection Systems protect most business network environments. Smart home networks of cosumers also need additional security properties. Insecure network services may be susceptible to a variety of attacks, i.e. buffer overflow and Denial of Service (DoS). Successful attacks can cause data loss or corruption of devices in the network. OWASP recommends these actions [10]:

- Ensuring that unused ports are closed and therefore not exposed
- Ensuring services are resistant against buffer overflow, fuzzing and DoS attacks
- Ensuring network ports or services are properply protected and inaccessible by services like UPnP
- Enforcing unknown or unassignable traffic requests to be blocked on service gateway layer

D) Lack of Transport Encryption

Communication is the characteristic that makes IoT devices smart. Those devices send and receive data constantly in order to perform a varietyof actions. As data travels over the network a lack of transport encryption causes the data to be viewed in plaintext. Therefore, sensitive information can be exposed to asversaries performing MITM attacks. To ensure transport encryption these measures need to taken [10]:

- Ensuring data is encrypted utilizing secure transport protocols, i.e. SSL and TLS
- Ensuring the latest commonly accepted encryption standards are used
- Ensuring secure encryption key handshaking and message payload encryption
- Ensuring received data integrity verification

E) Privacy Concerns

More devices are connected to the IoT every day. As identity theft is on the rise it becomes more dangerous to store personally identifiable information. The lack of proper protection of that data generates privacy concerns. To prevent such concerns, OWASP recommends [10]:

- Ensuring collection of data using need-to-know principle and storing the data using need-to-retain principle
- Ensuring any data collected is properly protected with encryption and therefore anonymized
- Ensuring proper permission handling for accessing collected information

F) Insecure Cloud Interface

Clouds facilitate storage and accessibility of data of IoT devices. They also introduce an additional attack surface with several attack vectors. Consequently, the cloud constitutes a new interface with new risks and vulnerabilities. In addition to the countermeasures that apply for the web interface 3.1.1.A a secure cloud interface requires [10]:

- Implement two factor authentication if possible
- Ensuring cloud systems use transport encryption

G) Insecure Mobile Interface

In 2011 "the total annual global shipments of smart phones exceeded those of client PCs (including Pads) for the first time"[11]. That means that mobile devices are the most popular smart device. Considering that these devices hold probably most of a user's sensitive data, they need to be adequately secured. Additionally, a variety of smart devices are controlled through mobile applications, therefore the impact of a compromised mobile interface is severe. The recommended countermeasures are (in addition to the mentioned actions regarding credentials in 3.1.1.A and 3.1.1.B) [10]:

- Ensuring mobile app code hardening: obfuscation and encryption
- Implementing mobile app anti-tempering mechanisms, using i.e. checksums and digital signatures
- Preventing the mobile app's execution on tempered Operating System (OS) environment

H) Insufficient Security Configurability

The administrator should be able to design and enforce specific security regulations which prevent modifications without proper approval. Otherwise adversaries can use the lack of proper permission-handling to access data on a device. The appropriate countermeasures for a secure configuration are [10]:

- Ensuring the separation of users with high privileges (administrators) and users with restricted permissions
- Ensuring encryption of data in all states
- Implementing strong password policies
- Ensuring documentation and notification of security events

I) Insecure Software/Firmware

Like any other software, the firmware of a device might be exposed to vulnerabilities. As technology advances the spectrum of exploits a device's firmware is exposed to grows bigger and faster. Firmware updates are needed to protect the device against current and future threats. However, it is as important to ensure that these updates are validated, as attackers might perform their own malicious update via Domain Name System (DNS) hijacking. These countermeasures are to consider [10]:

- Ensuring the device has the capability for updates
- Ensuring the update file and the transmission channels are encrypted properly
- Ensuring the updates are validated by using correctly signed files before allowing the update to be uploaded and applied
- Ensuring software authenticity by implementing secure boot when possible

J) Poor Physical Security

Physical access to a device is probably the easiest way to compromise it, or at least gather valueable information. A device may have Universal Serial Bus (USB) ports or other external ports, which are intended for configuration or maintenance. Attackers can use these external ports to gain access to the data stored on the device. To physically secure a device one must take these actions [10]:

- Ensuring external ports can not be used to maliciously access the device
- Preventing easy and effective device disassembly
- Utilizing a minimal number of device access ports
- Ensuring encryption of the stored data

3.1.2 OWASP Top 10 IoT Vulnerabilities (2018) and IoT Top 10 2018 Mapping Project

In the recent years, technology and therefore security requirements have changed rapidly. In 2018 OWASP delivered an update for its vulnerability assessment in order to offer an up-to-date threat classification. The OWASP Top 10 IoT Vulnerabilities of 2018 are shown in table 3.2.

	OWASP Top 10 IoT Vulnerabilities (2018)
1.	Weak, Guessable, or Hardcoded Passwords
2.	Insecure Network Services
3.	Insecure Ecosystem Interfaces
4.	Lack of Secure Update Mechanism
5.	Use of Insecure or Outdated Components
6.	Insufficient Privacy Protection
7.	Insecure Data Transfer and Storage
8.	Lack of Device Management
9.	Insecure Default Settings
10.	Lack of Physical Hardening

Table 3.2: OWASP Top 10 IoT Vulnerabilities (2018) [12]

It can be seen that the rankings have changed, though the attack vectors remain similar in respect to their corresponding categories. Therefore, OWASP provides a mapping between their assessments of 2014 and 2018, which is detailed in table 3.3.

OWASP Top10 2014	OWASP Top10 2018
1. Insecure Web Interface	3. Insecure Ecosystem Interfaces
2. Insufficient Authentication or Authorization	1. Weak, Guessable, or Hardcoded Passwords 3. Insecure Ecosystem Interfaces 9. Insecure Default Settings
3. Insecure Network Services	2. Insecure Network Services
4. Lack of Transport Encryption	7. Insecure Data Transfer and Storage
5. Privacy Concerns	6. Insufficient Privacy Protection
6. Insecure Cloud Interface	3. Insecure Ecosystem Interfaces
7. Insecure Mobile Interface	3. Insecure Ecosystem Interfaces
8. Insufficient Security Configurability	9. Insecure Default Settings
9. Insecure Software/Firmware	4. Lack of Secure Update Mechanism 5. Use of Insecure or Outdated Components
10. Poor Physical Security	10. Lack of Physical Hardening

Table 3.3: IoT Top 10 2018 Mapping Project [13]

As can be concluded from this table, a single category was introduced in 2018 that was not mentioned before, namely “Lack of Device Managment”. This property points

at a lack of security support on devices deployed in production. This is consistent with the observations of the IoT device market, where usability was prioritized at the cost of proper security.

3.2 Practical Examination: Wundertooth & Ubertooth

The Ubertooth One is an open-source Bluetooth monitoring and testing device by GreatScottGadgets [14]. The Wundertooth is a remix of Ubertooth One and is manufactured by Rysc Corporation. It enhances several capabilities, including the addition of a Joint Test Action Group (JTAG)-header and enclosure. According to the Frequently Asked Questions (FAQs) of Rysc Corp. the Wundertooth can be used by applying the same guide as with Ubertooth One [15] . However, this research was not able to achieve a working setup with the Wundertooth device. After following the instructions carefully, the device was not detected. The fact that the Ubertooth One worked with the same setup leads to the conclusion that additional measures need to be taken to achieve a working solution. Therefore, this paper uses in its practical implementation the Ubertooth One device.

The following describes a step-by-step guide to successfully capture Bluetooth packets.

Step 0 - Hardware Specifications

- HP Elitebook with Intel Core i5-7200U 2.50GHz 64-Bit processor
- OS: Kali Linux 64-Bit v2019.4 native install
- Wundertooth
- Ubertooth One
- Bose Soundlink Revolve (Bluetooth Speaker)
- Xiaomi Mi 9T Pro smartphone

Step 1 - Prerequisites

Ubertooth One offers a well-documented GitHub-repository [16]. Follwowing its instructions, the device's tools require several components. To fulfill the demands, one has to install:

```
$ sudo apt-get install cmake libusb-1.0-0-dev make gcc g++ \
  libbluetooth-dev pkg-config libpcap-dev python-numpy python-qt4
```

In contrast to the guide, the *python-pyside* needs to be installed separately using the *pip-installer*:

```
$ pip install pyside2
```

In order to for Ubertooth tools to decode Bluetooth packets the Bluetooth baseband library (*libbtbb*) needs to be downloaded and installed:

```
$ wget https://github.com/greatscottgadgets/libbtbb/archive
  /2018-12-R1.tar.gz -O libbtbb-2018-12-R1.tar.gz
$ tar -xf libbtbb-2018-12-R1.tar.gz
$ cd libbtbb-2018-12-R1
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig
```

Following that, the Ubertooth tools can be downloaded from the GitHub repository. They include host code, which enables sniffing Bluetooth packets. Additionally, they provide means to configure the Ubertooth device, including a simplified method for a firmware update.

```
$ wget https://github.com/greatscottgadgets/ubertooth/releases/
  download/2018-12-R1/ubertooth-2018-12-R1.tar.xz
$ tar xf ubertooth-2018-12-R1.tar.xz
$ cd ubertooth-2018-12-R1/host
$ mkdir build
$ cd build
$ cmake ..
$ make
$ sudo make install
$ sudo ldconfig
```

The Wireshark Bluetooth Baseband (*BTBB*) and Basic Rate/Enhanced Data Rate (*BR/EDR*) plugins facilitate the analysis of Bluetooth baseband traffic that has been captured within the Wireshark GUI. The plugins need to be installed both separately from the *libbtbb* library. For the *BTBB* plugin the following commands have been used:

```
$ sudo apt-get install wireshark wireshark-dev libwireshark-dev
  cmake
$ cd libbtbb-2018-12-R1/wireshark/plugins/btbb
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-gnu/
  wireshark/libwireshark3/plugins ..
$ make
$ sudo make install
```

It is important to state that the *MAKE_INSTALL_LIBDIR* directory can vary depending on the OS used and the wireshark installation. However, it should be the directory of existing Wireshark plugins. The procedure needs to be repeated for the *BR/EDR* plugin.

```
$ cd libbtbb-2018-12-R1/wireshark/plugins/btbredr
$ mkdir build
$ cd build
$ cmake -DCMAKE_INSTALL_LIBDIR=/usr/lib/x86_64-linux-gnu/
    wireshark/libwireshark3/plugins ..
$ make
$ sudo make install
```

Step 2 - Firmware Update & Verification of the Installation

After installing the prerequisits, the Ubertooth One device was plugged in through a USB port. It is of utmost importance to operate the Ubertooth One with the antenna attached to it. Otherwise, there is a risk of damaging the device. After inserting the device, it has been verified that the system detects it:

```
$ lsusb
```

As can be seen from the output of the command, the OS recognized the USB:

```
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 005: ID 04f2:b595 Chicony Electronics Co., Camera
Bus 001 Device 004: ID 138a:003f Validity Sensors, Inc. VFS495
Bus 001 Device 003: ID 8087:0a2b Intel Corp.
Bus 001 Device 002: ID 1ea7:0064 SHARKOON Technologies 2.4G Mouse
Bus 001 Device 014: ID 1d50:6002   OpenMoko, Inc. Ubertooth One
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Upon first operation of the Ubertooth One it is necessary to update its firmware. The tools, which were downloaded before, facilitate a simplified way to achieve this task. The directory was changed to the firmware directory of the Ubertooth tools and the following command was executed:

```
$ ubertooth-dfu -d bluetooth_rxtx.dfu -r
```

Since the update was successful, this output was produced:

```
Switching to DFU mode...
Checking firmware signature
.....................................
.....................................
.....................................
Detached
```

To verify the firmware-version this command was utilized:

```
$ ubertooth-util -v
```

The GitHub guide recommends to validate the functionality of the Ubertooth device via the spectrum analyzer, which is a tool to analyze the 2.4GHz band. Specifically, it provides a Graphical User Interface (GUI) tool named *ubertooth-specan-ui*, which visually monitors the frequencies. This command started the spectrum analyzer:

```
$ ubertooth-specan-ui
```

Once successful, one can see the analyzer work its task through a powerful GUI. Figure 3.1 shows the spectrum analyzer in action. Furthermore, it shows several 802.11b networks in different channels, represented by the green amplitudes. The white amplitudes depict beacons that are visible during scanning. The red lines measure the center frequency of Channel 8 in the 2.4GHz radio band.

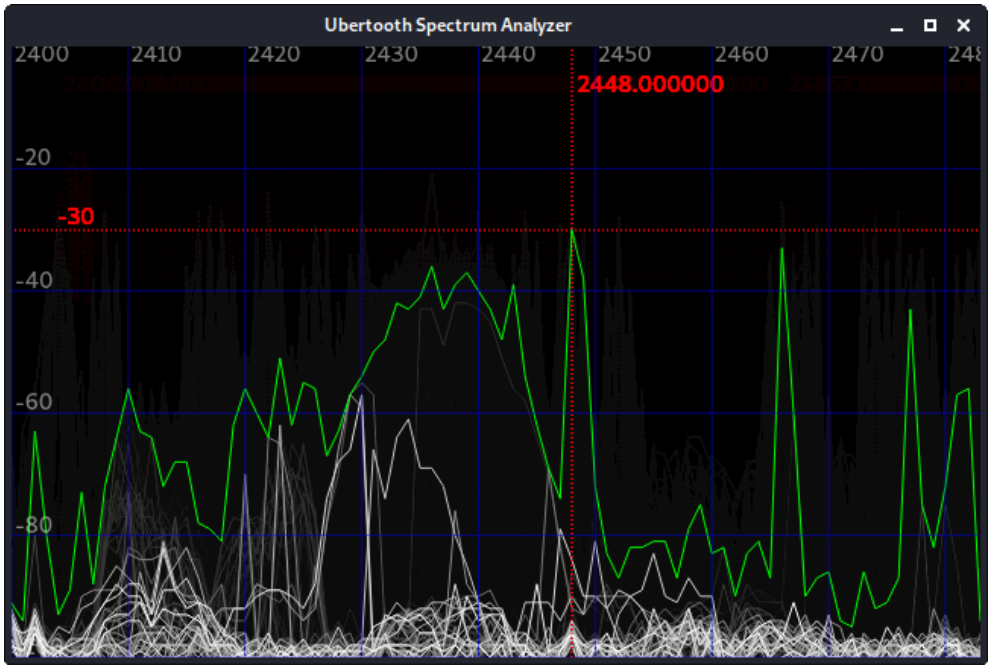


Figure 3.1: Ubertooth One Spectrum Analyzer

Step 3 - Intercepting Lower Address Part (LAP) Packets

The six byte Bluetooth Device Address (*BD_ADDR*) is comprised of three components, which are outlined in table 3.4.

Part	Byte	Description
NAP	2	Non-significant Address Part - assigned by IEEE and therefore publicly on [17]
UAP	1	Upper Address Part - also assigned by IEEE, publicly available on [17]
LAP	3	Lower Address Part - transmitted with every packet as part of the packet header

Table 3.4: Components of a Bluetooth Device Address

The *BD_ADDR* makes the allocation of sniffed Bluetooth packets possible. The Ubertooth One can start the LAP scan with this command:

```
$ ubertooth-rx
```

An ommited output of the scan is shown below:

```
systime=1578428685 ch=26 LAP=3a5138 err=1 clkn=8355 clk_offset=5199
    s=-80 n=-55 snr=-25
systime=1578428688 ch=26 LAP=3a5138 err=1 clkn=18479 clk_offset
   =5628 s=-81 n=-55 snr=-26
systime=1578428692 ch=43 LAP=3a5138 err=2 clkn=28967 clk_offset
   =6029 s=-80 n=-55 snr=-25
systime=1578428692 ch=47 LAP=3a5138 err=2 clkn=30327 clk_offset
   =6069 s=-81 n=-55 snr=-26
systime=1578428694 ch=52 LAP=3a5138 err=1 clkn=36948 clk_offset=87
   s=-79 n=-55 snr=-24
systime=1578428696 ch=53 LAP=3a5138 err=0 clkn=42360 clk_offset=302
    s=-77 n=-55 snr=-22
```

In the output *ch* represents the channel used by the device referenced in the *LAP* value. The channel hopping is clearly comprehensible. *Clkn* indicates the master's clock, while *s* references the signal strength and *n* states the value of noise. Following that the *snr* value represents the signal-to-noise ratio. This mode is especially helpful for undiscoverable devices, because it can calculate a *BD_ADDR* through the LAP in combination with the other parameters.

Combined with the scan techniques provided by *hcitools*, much information of the Bluetooth sourroundings can be aquired as detailed below:

```
$ sudo hcitool lescan
```

```
CC:B1:1A:DF:D3:25 [TV] Samsung 5 Series (40)
44:79:DD:B8:98:A2 (unknown)
2C:41:A1:6E:F8:44  LE-Bose Revolve SoundLink
C0:28:8D:88:D8:E3
FC:8F:90:16:06:A6 [TV] UE48JU7580
74:26:4F:E4:A7:AA (unknown)
69:9A:5A:B4:C7:DB (unknown)
00:24:E4:20:E2:1B (unknown)
E0:DC:FF:EB:80:D2  Mi Phone
```

The highlighted parts in the output represent the *BD_ADDR* of the soundbox and of the smartphone.

Step 4 - The Ubertooth-BTLE Tool

One of the most powerful tools the Ubertooth One provides is the Bluetooth Low Energy sniffing mode. Among other things, it can sniff and follow connections and even interfere with them.
In the "follow" mode, Ubertooth listens on one of the three advertising channels. Once a BLE connection is established, Ubertooth will follow the hops along the data channels capturing the transmissions between the devices. Per default, Ubertooth can be used to follow any connection it observes randomly. Naturally, the device can be restricted to observe a specific device by providing the *BD_ADDR* of the device in question. The general syntax of the corresponding command for "follow" mode looks like:

```
$ ubertooth-btle -f <BD_ADDR>
```

Furthermore, it is possible to redirect the output into a file or a pipe. To achieve this, the syntax requires this generic command:

```
$ ubertooth-btle -f <BD_ADDR> -c <file or pipe>
```

In the case of this examination the following command was used to follow the packets of the soundbox device:

```
$ ubertooth-btle -f -t 2C:41:A1:6E:F8:44 -c capture.pcap
```

As can be seen from the omitted output below, Ubertooth immediately started to capture advertisment messages of the soundbox device.

```
systime=1578433812 freq=2402 addr=8e89bed6 delta_t=551.234 ms rssi=-23
00 1b 44 f8 6e a1 41 2c 02 01 1a 03 03 be fe 0d ff 10 03 40 10 01 31 e0 dc ff eb 80 d2 52 1a 0b
Advertising / AA 8e89bed6 (valid)/ 27 bytes
    Channel Index: 37
    Type:  ADV_IND
    AdvA:  2c:41:a1:6e:f8:44 (public)
    AdvData: 02 01 1a 03 03 be fe 0d ff 10 03 40 10 01 31 e0 dc ff eb 80 d2
        Type 01 (Flags)
           00011010
               LE General Discoverable Mode
               Simultaneous LE and BR/EDR to Same Device Capable (Controller)
               Simultaneous LE and BR/EDR to Same Device Capable (Host)

        Type 03 (16-bit Service UUIDs)
           febe
        Type ff (Manufacturer Specific Data)
           Company: SGL Italia S.r.l.
           Data: 40 10 01 31 e0 dc ff eb 80 d2

    Data:  44 f8 6e a1 41 2c 02 01 1a 03 03 be fe 0d ff 10 03 40 10 01 31 e0 dc ff eb 80 d2
    CRC:   52 1a 0b
```

Step 5 - Wireshark Analysis

It is also possible to capture BLE packets in Wireshark. For that a pipe was created through the command:

```
$ mkfifo /tmp/mypipe
```

Following that, a new interface needs to be added to Wireshark in order to use the just created pipe. For this, the custom pipe */tmp/mypipe* was added to the list of

interfaces. Furthermore, the *Ubertooth-btle* tool was used with the output redirected to the capture interface */tmp/mypipe*:

```
$ ubertooth-btle -f -t 2C:41:A1:6E:F8:44 -c /tmp/mypipe
```

As shown in Fig. 3.2 it was possible to analyse and dissect BLE packets in Wireshark. This feature enhances the visibility of the captured data and provides a comfortable way to analyse an intercepted connection.

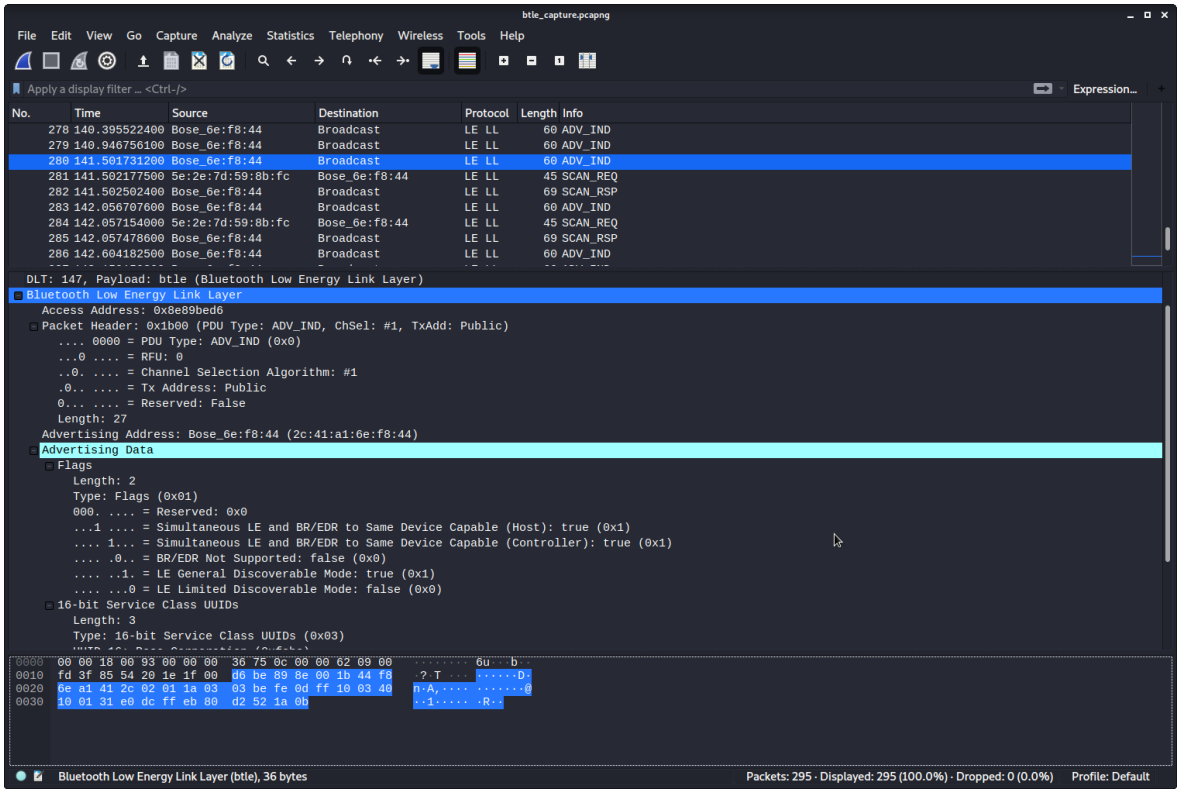


Figure 3.2: Display of BLE packets in Wireshark

Step 6 - Using crackle to Discover Keys

Finally, it is possible to use additional third party tools in combination with Ubertooth to obtain critical information. In this examination the *crackle* tool was used to discover the keys of a sample dataset. In order to achieve this, the output file from a *ubertooth-btle* scan was used to decrypt the captured data. The tool was installed through this command:

```
$ sudo apt install crackle
```

After the installation process, the following command was issued to obtain the keys:

```
$ crackle -i capture.pcap -o result.pcap
```

```
 TK found    : 726273
 LTK found   : 7aa64997a3a84831ecdd5ef6a84a3e0d
Done, processed 172 total packets, decrypted 36
```

In section 2.2.2 it was described that Bluetooth keys are vulnerable, because of their inherent properties. The output from the *crackle* command confirms this fact. Moreover, it is possible to combine Ubertooth with additional third party tools like *bettercap* and *gatttool*. The former is a powerful framework, which facilitates in the context of Bluetooth Low Energy: devices scanning, characteristics enumeration, and reading & writing. The utility *gatttool* can be used to manipulate characteristics attribute-values.

4. Conclusion

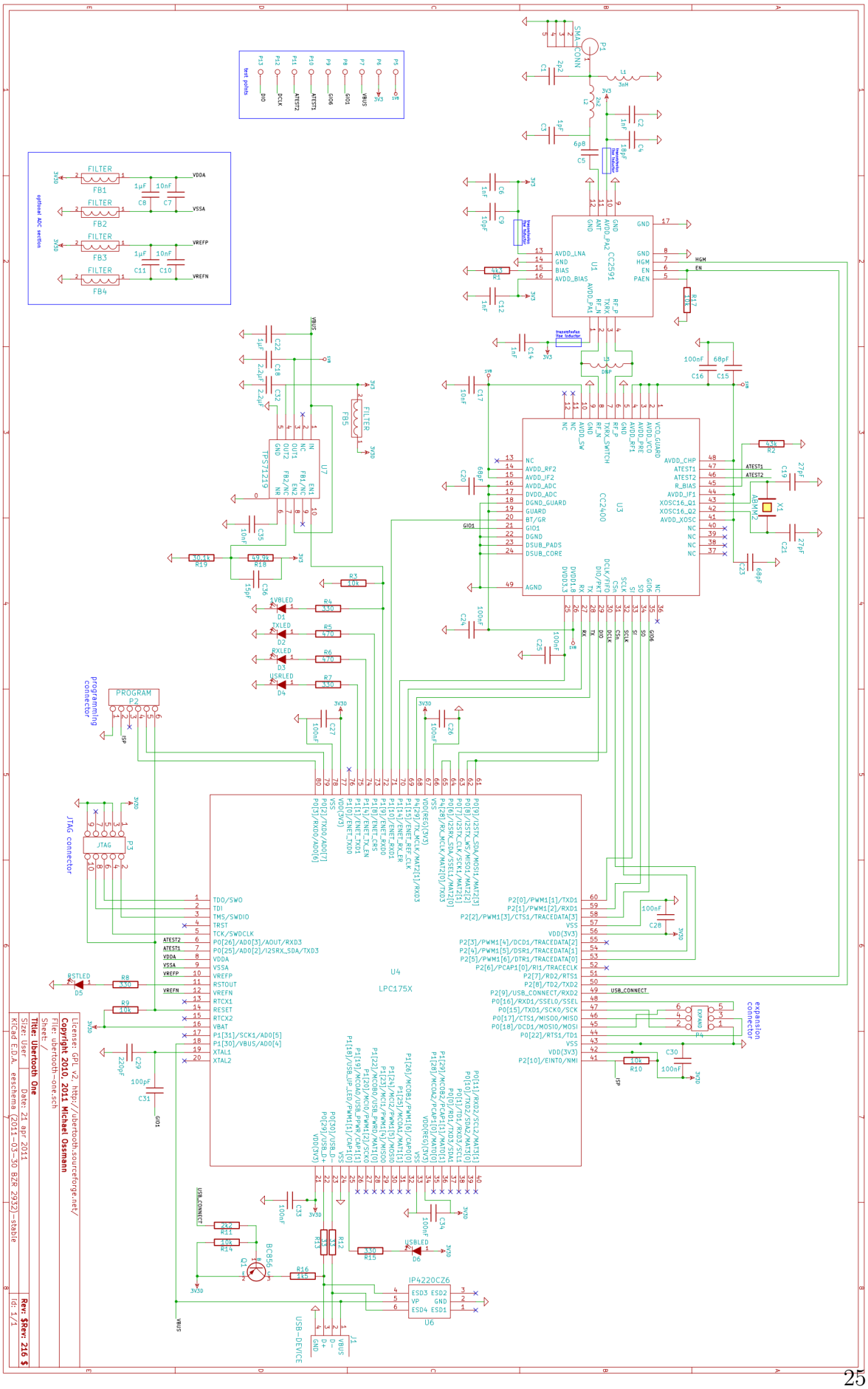
This research outlined the necessity of penetration testing in order to sufficiently secure IoT devices. Through the threat model provided in this work, one can comprehend how important it is to take appropiate countermeasures to provide adequate security. Furthermore, the practical examination offered insights to the implications of insufficient security. The devices aren't safe in any way against the various forms of Bluetooth sniffing presented in the examination. Information was obtained, including keys for connection-handling that had detrimental effects on the devices' security.
Manufacturers have to keep up with the ever-changing security requirements. Current models like the threat model presented in this research provide an overview of attack vectors and guidlines to take precautionary measures. In this regard, it is very important that device manufacturers do not place the usability of a device in the foreground, as has been the case to date. It is to be hoped that the actions and measures taken to make devices more secure are not superficial.

4.1 Discussion, Limitations & Future Work

The author took an explorative approach to the topic of this paper. While penetration testing is a significant process to examine the security properties of a given device, it lacks standardization. Other than the open documentation and reports from OWASP, there is no standardized information available. In this regard the threat model provided is not intended to cover all possible attack vectors. Some properties and countereasures may be slightly redundant. This is due to the fact that the author would not want to contradict the OWASP information. To provide a threat model detailed from scratch was beyond the scope of this paper. However, this can be a possibility for future research.
The practical examination was intended to provide an overview of penetration tests in the specific context of Bluetooth security. It has been pointed out in this research that a thorough assessment should include the whole infrastruture the IoT device operates in. This approach was dispensed with due to the lack of technical opportunities in order to perform such a test. Nevertheless, future work could analyse a device in different contexts and provide an assessment applied to all layers of a given device.

A. Appendix

Ubertooth schematics [18]


Title: Ubertooth One
Date: 21 apr 2011
Size: User
KiCad E.D.A. eeschema (2011-03-30 BZR 2922)-stable
Sheet: /
File: ubertooth-one.sch
Rev: $Rev: 216 $
Id: 1/1
Copyright 2010, 2011 Michael Ossmann
License: GPL v2 http://ubertooth.sourceforge.net/
U1 CC2591
U3 CC2400
U4 LPC175X
U6 IP4220CZ6
U7 TPS72219
P1 SMA-CONN
P2 PROGRAM
programming connector
P3 JTAG
JTAG connector
P4 EXPAND
expansion connector
J1 USB-DEVICE
X1 26MHz
FILTER
FB1
FB2
FB3
FB4
FB5
VDDA
VSSA
VREFP
VREFN
optional ADC section
test points
TDO/SWO
TDI
TMS/SWDIO
TRST
TCK/SWDCLK
RSTOUT
RESET
RTCX1
RTCX2
VBAT
XTAL1
XTAL2
USB_CONNECT
USBLED
RSTLED
VBUSLED
TXLED
RXLED
USRLED
BC856
HGM
EN
PAEN

List of Figures

List of Tables

Bibliography

[1] "Penetration Testing Execution Standard," [Accessed 22 Nov. 2019]. [Online]. Available: http://www.pentest-standard.org/index.php/Main_Page 1

[2] A. Guzman and A. Gupta, *IoT Penetration Testing Cookbook: Identify vulnerabilities and secure your smart devices.* Packt Publishing Ltd, 2017. 1, 4, 5, 7, 26

[3] N. Dhanjani, *Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts.* " O'Reilly Media, Inc.", 2015. 1

[4] M. A. Feki, F. Kawsar, M. Boussard, and L. Trappeniers, "The Internet of Things: The Next Technological Revolution," *Computer*, vol. 46, no. 2, pp. 24–25, Feb 2013. 2

[5] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," April 2011, white Paper. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf 2

[6] (2016) AdaptiveMobile estimates up to 80% of 'connected' devices do not have adequate security measures. [Accessed 22 Nov. 2019]. [Online]. Available: https://www.adaptivemobile.com/newsroom/press-release/adaptivemobile-estimates-up-to-80-of-connected-devices-do-not-have-adequate 2

[7] R. Contu, D. Kish, P. Carpenter, S. Deshpande, and L. Pingree, "Gartner Predict 2016: Security Solution," 2015, [Accessed 22 Nov. 2019]. [Online]. Available: https://www.gartner.com/newsroom/id/3869181 2

[8] Bluetooth SIG, "Bluetooth Core Specification v5.1," 2019, [Accessed 22 Nov. 2019]. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/ 3, 5, 6, 7, 26

[9] M. Ryan, "Bluetooth: With Low Energy Comes Low Security," in *Presented as part of the 7th USENIX Workshop on Offensive Technologies.* Washington, D.C.: USENIX, 2013. [Online]. Available: https://www.usenix.org/conference/woot13/workshop-program/presentation/Ryan 7

[10] "Top 10 IoT Vulnerabilities (2014)," 2014, [Accessed 22 Nov. 2019]. [Online]. Available: https://www.owasp.org/index.php/Top_10_IoT_Vulnerabilities_(2014) 8, 9, 10, 11, 12, 26

[11] "Smart Phones Overtake Client PCs in 2011," 2012, [Accessed 22 Nov. 2019]. [Online]. Available: https://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011 11

[12] "Top 10 IoT Vulnerabilities (2018)," 2018, [Accessed 22 Nov. 2019]. [Online]. Available: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project 13, 26

[13] "OWASP Top10 IoT Vulnerabilities Mapping Project," 2018, [Accessed 22 Nov. 2019]. [Online]. Available: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=OWASP_IoT\OT1\ss_Top_10_2018_Mapping_Project 13, 26

[14] "Ubertooth One - Great Scott Gadgets," [Accessed 8 Jan. 2020]. [Online]. Available: https://greatscottgadgets.com/ubertoothone/ 15

[15] "Frequently Asked Questions - Rysc Corp," [Accessed 8 Jan. 2020]. [Online]. Available: https://store.ryscc.com/pages/faq 15

[16] T. Saul, "Build Guide · greatscottgadgets/ubertooth Wiki - GitHub," [Accessed 8 Jan. 2020]. [Online]. Available: https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide 15

[17] Institute of Electrical and Electronics Engineers (IEEE), "List of Organizationally Unique Identifiers - IEEE," [Accessed 8 Jan. 2020]. [Online]. Available: http://standards-oui.ieee.org/oui/oui.txt 18

[18] "Ubertooth Schematics Datasheet," [Accessed 8 Jan. 2020]. [Online]. Available: https://cdn.sparkfun.com/datasheets/Dev/ARM/SchematicUbertooth.pdf 24