The workprogram for penetration testing of ZigBee enabled IoT devices

January 2018
Abstract
The Internet of Things is changing our society. The increasing amount of “smart devices” that are being connected to the Internet is attracting everyone’s attention. However, for the sake of the usability, IoT devices frequently have poor security. With the rapid development of the IoT, comes the need to secure the devices and thereby protect organisations and citizens against cyber-attacks.

The first step to achieve a higher maturity of connected devices is to conduct penetration tests, which are means of verifying the level of security of an IoT system. However, there are not many frameworks specific for the IoT realm. This research adds to the collection of penetration testing frameworks, by creating a workprogram, specifically targeted to testing IoT devices with the ZigBee protocol.

To achieve this goal, IoT security experts are interviewed and available penetration testing workprograms examined. The ZigBee protocol, which is one of the widespread IoT protocols, is analyzed for potential vulnerabilities and attack vectors, by hands-on assessment of a smart light bulb system and the ZigBee network.

The final product of the research is an open source workprogram, which will standardize the process of conducting IoT penetration tests in both corporate and small businesses. It contains six steps, which include formal, mandatory steps, ZigBee protocol analysis and optionally hardware and firmware analysis.
### List of abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2FA</td>
<td>Two-Factor Authentication</td>
</tr>
<tr>
<td>ACL</td>
<td>Access Control List</td>
</tr>
<tr>
<td>AES</td>
<td>Advances Encryption Standard</td>
</tr>
<tr>
<td>APL</td>
<td>Application Layer</td>
</tr>
<tr>
<td>APS</td>
<td>Application Support Sublayer</td>
</tr>
<tr>
<td>CCM</td>
<td>Counter with CBC-MAC</td>
</tr>
<tr>
<td>CE</td>
<td>Coordinator</td>
</tr>
<tr>
<td>DH</td>
<td>Device High</td>
</tr>
<tr>
<td>DL</td>
<td>Device Low</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things</td>
</tr>
<tr>
<td>JTAG</td>
<td>Joint Test Action Group</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>NIST</td>
<td>National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NWK</td>
<td>Network Layer</td>
</tr>
<tr>
<td>OTA</td>
<td>Over-the-air updates</td>
</tr>
<tr>
<td>OWASP</td>
<td>Open Web Application Security Project</td>
</tr>
<tr>
<td>PAN ID</td>
<td>Personal Area Network Identifier</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>RF</td>
<td>Radio Frequency</td>
</tr>
<tr>
<td>RZUSB</td>
<td>RZ RAVEN USB</td>
</tr>
<tr>
<td>SKKE</td>
<td>Symmetric-Key Key Establishment</td>
</tr>
<tr>
<td>SME</td>
<td>Subject Matter Expert</td>
</tr>
<tr>
<td>SSL</td>
<td>Secure Sockets Layer</td>
</tr>
<tr>
<td>TLS</td>
<td>Transport Layer Security</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>TPM</td>
<td>Trusted Platform Module</td>
</tr>
<tr>
<td>UART</td>
<td>Universal Asynchronous Receiver-Transmitter</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WP</td>
<td>Workprogram</td>
</tr>
<tr>
<td>XCTU</td>
<td>Xbee Configuration and Test Utility</td>
</tr>
<tr>
<td>ZHA</td>
<td>ZigBee Home Automation</td>
</tr>
<tr>
<td>ZBA</td>
<td>ZigBee Building Automation</td>
</tr>
<tr>
<td>ZLL</td>
<td>ZigBee Light Link</td>
</tr>
<tr>
<td>ZigBee WP</td>
<td>ZigBee Workprogram</td>
</tr>
</tbody>
</table>
# Table of contents

1 Introduction .......................................................................................................................... 1

2 Research design and methodology ....................................................................................... 2
   2.1 Problem statement .......................................................................................................... 2
   2.2 Goal of the research ...................................................................................................... 2
   2.3 Research design ............................................................................................................ 3
   2.4 Methodology ................................................................................................................ 5
      2.4.1 RQ1: What is the architecture of the ZigBee protocol and what are its security features? 5
      2.4.2 RQ2: How should IoT devices be secured? ................................................................. 6
      2.4.3 RQ3: Which IoT pentesting standards exist and how can they be adapted for the ZigBee workprogram? ........................................................................ 6
      2.4.4 RQ4: What should an IoT workprogram for a ZigBee supported device include? .......... 6
   2.5 Conclusions .................................................................................................................. 8

3 Literature review ..................................................................................................................... 9
   3.1 Related work .................................................................................................................. 9
   3.2 Drivers for pentesting .................................................................................................... 10
   3.3 Pentesting frameworks .................................................................................................. 10
      3.3.1 OWASP .................................................................................................................. 11
      3.3.2 NIST ..................................................................................................................... 12
      3.3.3 Online publications about IoT pentesting ................................................................. 12
   3.4 Conclusions .................................................................................................................. 14

4 Setting up the experiment ...................................................................................................... 15
   4.1 ZigBee protocol theory .................................................................................................. 15
      4.1.1 ZigBee protocol stack ............................................................................................. 15
      4.1.2 ZigBee roles .......................................................................................................... 17
      4.1.3 ZigBee security ..................................................................................................... 17
      4.1.4 ZigBee Keys ......................................................................................................... 18
      4.1.5 Key Management .................................................................................................. 18
      4.1.6 ZigBee vulnerabilities ............................................................................................ 19
   4.2 Testing ZigBee enabled device ......................................................................................... 20
      4.2.1 Preparation - installing KillerBee .......................................................................... 21
      4.2.2 Protocol hacking .................................................................................................. 23
      4.2.3 Hardware hacking ................................................................................................. 26
   4.3 Creating ZigBee network ................................................................................................. 29
      4.3.1 Configuration steps ............................................................................................... 29
      4.3.2 Security on the XBee ............................................................................................ 30
4.3.3 Security analysis ........................................................................................................... 30
4.4 Conclusions ....................................................................................................................... 33
5 Analysis of hands-on knowledge .......................................................................................... 35
  5.1 Interviews .......................................................................................................................... 35
  5.2 Analysis of available workprograms ............................................................................... 36
  5.3 Good practices for securing IoT devices ......................................................................... 38
  5.4 Conclusions ....................................................................................................................... 40
6 Creating the ZigBee workprogram ..................................................................................... 42
  6.1 Preliminary version of the ZigBee WP .............................................................................. 42
  6.2 Final version of the ZigBee WP ....................................................................................... 43
  6.3 Conclusions ....................................................................................................................... 44
7 Discussion ........................................................................................................................... 46
  7.1 Research accountability statement ................................................................................. 46
  7.2 Ethical considerations ....................................................................................................... 47
  7.3 Evaluation of the workprogram ...................................................................................... 47
    7.3.1 Financial aspect of the ZigBee WP ......................................................................... 48
  7.4 Future work ....................................................................................................................... 49
8 Bibliography ....................................................................................................................... 51
Appendix A: Final version of the ZigBee WP ....................................................................... 1
Appendix B: Interview questions ........................................................................................... 12
1 Introduction

In 2015, Fiat Chrysler recalled 1.4 million vehicles after a group of researchers remotely seized control of the vehicle and were able to shut down the engine on the highway, cut the car breaks and potentially overtake the steering wheel. The patch was released on USB flash drive and the company has taken network level security measures to prevent remote manipulation of the car (Welch, 2015). Same year, researcher Billy Rios reported that he has found vulnerabilities in a drug infusion pump, which allow a hacker to raise the dosage limit of medication delivered to the patient (Zetter, 2015). However, so far the most severe in consequences was the attack from autumn 2016, where the biggest distributed denial-of-service (DDoS) attack took place. Mirai, the botnet behind the attack, spread to IoT devices with factory default usernames and passwords, took control over them and attack Internet services. A major DNS provider was hit by Mirai, which resulted in disrupting websites worldwide (Symantec, 2017).

We are entering a phase in digital evolution, where everything around us will be connected to the Internet. The term to describe this phenomena is IoT, short for Internet of Things. It is growing exponentially and is attracting everyone’s attention, starting from academia, through biggest tech companies to end customers. Gartner predicts that there will be more than 20 billion IoT devices in the world by 2020 (Gartner, 2017). The IoT market and its customer acceptance is growing. However, for the sake of usability, IoT devices have often poor security, which leaves the customers vulnerable (Symantec, 2015). Because of the small size and their cheap price, IoT devices might be designed with limited hardware resources such as small memory and CPU’s (Broadband Internet Technical Advisory Group, 2016). Security is often not a priority for the device manufacturers, which results in using default passwords or leaving unnecessary ports open (Symantec, 2017). The devices usually don’t have built-in mechanisms to receive updates, which leaves the vulnerabilities unpatched. To make the situation even worse, manufacturers and owners often are not aware of security risks or have little incentive to improve the security by for example changing default passwords or downloading updates (Symantec, 2017).

One way to achieve higher security maturity, is to measure level of security in IoT devices, by conducting penetration tests also called pentests and based on the result come up with the plan to mitigate the vulnerabilities. Over the years, there has been quite a big number of pentesting standards published, outlining step-by-step on how to perform a penetration test. Standards published by NIST, OSSTMM, OISSG and PCI are just some examples.

With the rise of the IoT, there are some efforts in standardizing the procedure for pentesting of IoT devices as well. The most well-known is Open Web Application Security Project (OWASP, 2017a). OWASP created a checklist for testing IoT devices, however it is not protocol specific.

Due to the lack of established standards we have a need to develop an IoT workprogram with a focus on a specific protocol. ZigBee, low-cost and low-power protocol, is one of the most widespread wireless technologies used to connect IoT devices (Zillner, 2015). Since the protocol is universal the amount of clients approaching Deloitte and asking for penetration tests on a ZigBee enabled device is steadily growing. For this reason we decided to create a workprogram for IoT devices with the ZigBee protocol.
2 Research design and methodology

This chapter elaborates on the research design and methodology applied in this thesis, including an explanation of the goal of the research, research design, research model and methods for data collection.

2.1 Problem statement

Penetration tests are performed on a daily basis at Deloitte. Workprograms and various methodologies exist to guide testers and make sure that good quality work is being delivered. Each workprogram has a specific scope, such as mobile, application, network or infrastructure pentesting among others. They structure work of a pentester, by outlining a list of detailed steps to be performed and tools to be used during security tests.

In recent years, the amount of customers requesting penetration tests specifically on IoT devices is increasing. This can include, but is not limited to biometric scanners, cameras, medical devices or home automation products. Pentesters have access to workprograms that have common elements of an IoT pentest; there is a web penetration, network interface or firmware testing workprogram. They can be partially used during an IoT pentest, however, there is no standard IoT methodology or a workprogram available that allows testers to structure their work from beginning until the end.

Every pentest on IoT devices differs from the other, due to the complexity of the IoT landscape. There is a variety of different protocols and ways to implement them in IoT devices. That makes the tests cumbersome and time-consuming, since for each test, ethical hacker needs to do preliminary research, decide on the tools to be used, purchase the hardware, investigate the implementation and then perform the actual test. It is not scalable in case of bigger engagements with multiple IoT devices at once.

To achieve the level of efficiency and standardization of penetration tests similar to the general pentesting team, three milestones have to be reached. A workprogram should be created, which will focus on one specific protocol with detail steps. Then, the work should be expanded, by adding new protocols. Finally, automatization of parts of pentesting process such as reporting should take place. The focus of this thesis is achieving the first milestone – creating a workprogram for one specific protocol, which will then be the basis for further development.

2.2 Goal of the research

The goal of this research is to develop a workprogram for pentesting IoT devices with the ZigBee protocol to help improve the existing process of conducting IoT penetration testing at Deloitte. ZigBee, a mesh network standard is one of the most widespread wireless technologies, used to connect IoT devices due to its low-cost and low-power design (Flynn, 2016). That is why, the goal of this thesis is to standardize the process of testing ZigBee devices for vulnerabilities, by creating a workprogram that can be followed by testers during pentests. For the purposes of this research, we will refer to a workprogram as to a list of actions that need to be executed in order to call a given pentests finished and completed.

The ZigBee workprogram (ZigBee WP) should provide guidance on minimal set of activities to be performed during a pentest and it should be used by every ethical hacker who performs testing of a ZigBee enabled device. It is a checklist, which has to be filled with the findings and observations and attached to the official documentation of the project. At the same time the workprogram needs to be easy to follow and to understand, even for a pentester who has no previous experience with the ZigBee protocol.
The goal of creating the ZigBee WP will be achieved by analyzing generally available pentesting standards and Deloitte workprograms, providing an overview of the stakeholders’ opinions and by conducting hands-on pentests to validate the assumptions. ZigBee protocol was chosen as the topic of the workprogram, as the most widespread IoT protocol (Zillner, 2015). The document will help testers perform pentesting thoroughly and time-efficiently by following list of steps with special focus on ZigBee protocol. Adhering to workprogram will introduce structure to the work of pentesters, by making sure that no testing phase is omitted in the process. The pentesting process will also become more efficient and repeatable. In consequence, penetration tests will run more smoothly and bring more consistent results. On top of that Deloitte clients will be confident of the validity of the results.

Creating a ZigBee workprogram is the first milestone in the process of automatizing the pentesting process to make it more manageable during large scale client engagements. A ZigBee WP will be a basis for further expansion. As a next step, other IoT protocols and guidelines for pentesting will be created, followed by creation of automatic tools such as reporting. Achieving the first milestone - creating the ZigBee workprogram is the objective of this research. Further expansion is out of scope of this research and could be a topic for future work.

2.3 Research design

The research is designed based on Verschuren and Doorewaard research model (Verschuren & Doorewaard, 2010). The model is discussed in the book “Designing a Research Project”, which main objective is to instruct researches on how to set up an adequate research (Verschuren & Doorewaard, 2010). The volume explains different aspects of designing a research project such as defining project context, and research objective, giving clear definitions of key concepts, designing research model and formulating research questions. The recommendations from the book were followed to design this research.

The steps to be taken in the course of this research project are formulated as follows:

(a) A study of theories of ZigBee, Xbee, secure IoT devices and externally available pentesting standards,

(b) supplemented by analysis of Deloitte workprograms, information obtained during the interviews with Subject Matter Experts (SME’s) and analysis of ZigBee enabled devices and self-created ZigBee network are to be used as a basis for formulating a list of good practices for securing IoT devices, analysis of ZigBee security and preliminary version of the ZigBee WP,

(c) which will be validated by experienced IoT pentesters by hands-on penetration test and

(d) results in final workprogram for ZigBee pentesting.
The workprogram for penetration testing of ZigBee enabled devices | Report

Based on the research model we defined four research questions:

RQ1: What is the architecture of the ZigBee protocol and what are its security features?

RQ2: How should IoT devices be secured?

RQ3: Which IoT pentesting standards exist and how can they be adapted for the ZigBee workprogram?

RQ4: What should an IoT workprogram for a ZigBee supported device include?

The research model (Figure 1) describes four stages of the research. Activities in each stage contribute to answering one or more research questions, which then results in the final product – the ZigBee workprogram.

The first column of the research model relates to the theoretical phase. During this stage, preliminary research will be conducted to get an understanding of the topic and its context. Subsequently the theory on ZigBee technical specifications, architecture, protocol security and threat landscape will be conducted. This will be followed by Xbee theory, which is necessary to set up a working ZigBee network. Furthermore, pentesting standards published by known governmental organizations, associations and private researchers will be analyzed for applicable information which could be used when creating an IoT pentesting workprogram for ZigBee devices. Moreover, the theory of securing the IoT devices will be collected and analyzed, which includes reports and articles from security organizations, private researchers and companies. This phase of the research model answers the questions RQ2 and RQ3, which is purely theoretical. Furthermore, it contributes to answering question RQ4, by enumerating existing IoT standards.

IoT is a wide topic, which can apply to almost anything connected to the Internet. Therefore we scoped down the research and discarded IoT theory as a separate subject due to limited time available. The focus of literature research will be only on ZigBee architecture, protocol specifications and security, followed by theory Xbee and IoT pentesting standards.

Next, analysis part begins, which refers to column B of the research model. It consists of analyzing Deloitte pentesting workprograms that are currently in use by the ethical hacking team. The goal is to understand how pentesting is currently conducted and what the layout and structure of a
The workprogram for penetration testing of ZigBee enabled devices | Report

workprogram is, as some of the steps can be adapted to the ZigBee WP as well. Additionally, interviews with Deloitte professionals will be held and insights will be collected and analyzed. The objective of the interviews is to receive information on how the process of IoT pentesting usually looks like and what are the lessons learned that could be used to improve the process of pentesting in the future.

Moreover, a ZigBee network will be created with Xbee modules. The goal of creating the ZigBee network is to understand the logic of the protocol and to test different security levels. This knowledge will help understand how the ZigBee protocol and its security works and how ZigBee workprogram should be designed to be as thorough as possible.

After the analysis phase is completed, the creation of an intermediate product starts. This activity is graphically represented in column C of our research model. List of good practices for securing IoT devices will be created, based on relevant theory. The list defines how manufactures, developers or even end-users should protect the IoT devices. It is a valuable input to the workprogram, since it will help identify the focus area of pentesters when testing IoT devices and in consequence improve the workprogram. Furthermore, the security of ZigBee enabled device and self-created ZigBee network will be analyzed.

Based on the theory, list of good practices and findings from the analysis of the ZigBee network, the preliminary version of the workprogram will be shaped. The draft will be validated by experienced pentesters from Deloitte by means of hands-on hacking and tabletop exercise. This exercise will be followed by a round of feedback and discussion of lessons learned. This will in consequence be incorporated in the workprogram and will result in the final ZigBee WP which is the intended result of the research project.

2.4 Methodology

In this section we give an explanation and justification of our research methodology. The section contains four sections which correspond to four research questions, defined in the previous paragraph. For each question, data collection techniques and methods of analysis are described.

2.4.1 RQ1: What is the architecture of the ZigBee protocol and what are its security features?

In order to have a good understanding of the ZigBee protocol and the level of security it provides, this question will be answered. It refers to the theoretical column of the research model (column A) and it describes ZigBee theory. The question can be answered based on theoretical approach, so called desktop research. It will consist of two steps:

- Identification of literature
- Literature review

In the identification phase, publicly available literature will be selected to describe the ZigBee protocol. The literature will be obtained from the Internet, from the ZigBee Alliance website. The alliance is the creator of the ZigBee protocol and therefore the documentation of the protocol specification is the most thorough and complete. ZigBee implementations in IoT devices are often based on implementation created by the ZigBee Alliance, which provides another argument for the reliability of the source.

For the security aspect of ZigBee on the other hand, publicly available work of security researchers from universities all over the globe will be used. The documents will be obtained via Internet search from IEEE.org, hva.nl/bibliothek and Google Scholar. ZigBee Alliance does explain security principles
in one of their documents (ZigBee Alliance, 2016), but it does not provide insights into vulnerabilities and possible attack vectors. That is why the work of independent researchers from universities will be selected, as a reliable source of information on vulnerabilities of ZigBee.

During literature review, selected documents will be read and analyzed for relevant information. This information will be provided in the practical analysis section in chapter 4, when it is needed to understand the practical experiment. The answer to this question is a basis for further research and allows to understand the principles of the ZigBee protocol.

2.4.2 RQ2: How should IoT devices be secured?
Similar to RQ1, this question also refers to the theoretical column of the research model (column A) and can be answered based on a desktop research. The knowledge of how IoT devices should be secured will contribute to the creation of the workprogram, by indicating the areas which should be tested by ethical hackers.

To answer this question, reports from technology companies, technology associations and governmental institutions will be selected and analyzed for relevant information on how to secure IoT devices. Bruce Schneier, an internationally renowned security technologist, has gathered major IoT security and privacy guidelines (Bruce Schneier, 2017). The list will be reviewed and relevant documents containing information about endpoints, network and protocol protection will be selected for further analysis. This includes documents created by organization such as OWASP, IEEE or Homeland Security.

The information from the literature review about securing IoT devices will be cross referenced and verified for analogous information. This information will then be inserted into a list of good practices. The list of practices is a mid-product which can be seen in column C of the research model.

2.4.3 RQ3: Which IoT pentesting standards exist and how can they be adapted for the ZigBee workprogram?
This question also refers to the theoretical column of the research model. To answer this question, desk research will be conducted. First part of the question is about researching existing, public standards for pentesting IoT devices. This is done with an Internet search for governmental institutions and associations that created such frameworks. After first stage is completed, analysis will be performed. The goal of the analysis is to extract information, which will be crossed references and searched for common denominators – pentesting steps, relevant for the ZigBee WP.

2.4.4 RQ4: What should an IoT workprogram for a ZigBee supported device include?
RQ4 is the most complex question and to get the answer, several research methods are required. Theory used to answer questions RQ1, RQ2 and RQ3 will also contribute to answering question RQ4.

Activities from column B – Analysis and column C – Intermediary products will be performed with the aim of finding an answer.

The first task is the analysis of workprograms that are currently in use at Deloitte. Pentesters have access to various workprograms; there is a web penetration, network interface or firmware testing workprogram. Analyzing those documents, will give an overview on what a workprogram for ethical hackers should contain regarding the structure and the depth of the information. Moreover, some of the information from the workprograms can be reproduced in the ZigBee WP. Three workprograms are selected for the analysis: infrastructure, hardware and firmware. Infrastructure workprogram is widely used during almost every pentesting engagement and for this reason its structure and formal
steps of conducting pentesting will be examined. Hardware and firmware workprograms will be studied for relevant content, which should be used in the ZigBee workprogram.

Next, interviews will be conducted with the goal of gathering information on the process and approach of performing an IoT pentest. For this research, we will use small, non-random sample of participants. There is a group of SME’s at Deloitte, who work with IoT on a regular basis and have hands-on experience with conducting pentesting on IoT devices. Those professionals will be invited for an interview. Interview questions consist of two aspects. First, questions about conducting generic pentesting assessments will be asked, to understand the procedures of conducting such project. Next, questions about IoT pentesting will be asked, which includes the current process and the challenges. The list of interview questions can be found in Appendix B.

The interviews are scoped down to discussions with Deloitte pentesters, because they are the ones who will use the workprogram during engagements on a later stage. They understand client’s needs and the workflow of a pentest at Deloitte. Whereas choosing external people or asking the Internet community, could lead to confusing input due to different principles. Obtained information during the interviews will be used as a starting point for developing the preliminary version of the workprogram for testing ZigBee devices.

Then, the ZigBee network will be created and different levels of security will be tested. The goal is to get acquainted with the ZigBee protocol. Setting up a new network will help to understand device connectivity and mesh networking, which is one of ZigBee features. The knowledge of ZigBee protocol is needed to create a well-designed workprogram.

The last activity from analysis column (column B) hands-on pentesting of ZigBee enabled devices will be performed. A smart light bulb system will be tested with ZigBee specific tools. The goal is to obtain knowledge about the ZigBee protocol and its implementation, as well as learn about the software and hardware tooling, which can be used during pentesting.

The next step will be the creation of the list of good practices for securing IoT devices. For this purpose, documents created by governments and technology associations will be analyzed for relevant information about securing IoT devices. The information will be selected based on two requirements. The first requirement is the relevancy of the information according to the scope of the ZigBee WP, which is explained later in this chapter. The second requirement is that the advice should be mentioned in all the analyzed sources. If it is, it will be included in the list of good practices. This rule applies to all the information, but the ZigBee specific penetration testing advice, which is only included in one paper.

In this stage, theoretical research, document analysis, interviews, practical experiments and list of good practices will be completed and based on gathered information first draft of the workprogram will be created.

Top 10 IoT security considerations created by OWASP, can be divided into six testing areas (OWASP, 2017b):

1. Hardware analysis
2. Firmware analysis
3. Wireless protocol analysis
4. Mobile application
5. Web application

6. Cloud services/infrastructure

The focus of the ZigBee WP will be on the first three phases - hardware, firmware & OS and wireless protocol. The reason is that hardware, firmware & OS are specific for testing IoT devices, which means there is no structured workprogram available. Wireless protocol analysis, in this research will focus on ZigBee protocol, for which such workprogram doesn’t exist either. The other three phases - mobile, web application, and cloud services/infrastructure are very similar to a generic pentest, for which there is already documentation and guidance available. Testing guidance will be adapted to the ZigBee workprogram.

In order to confirm that the quality of the workprogram is sufficient and can be used during client engagement, the validation phase will start. Three SME’s from Deloitte were chosen to verify the preliminary workprogram, which combined have 12 years of experience with IoT pentesting at different clients. They know what steps are important and what should not be omitted in the workprogram. Moreover, they have already worked with other workprograms and methodologies before, so they know what the goal and focus of a workprogram is. This is precisely the reason why we have not chosen for wider public like the Internet community. They might have experience with hacking, but they won’t understand the reality and requirements of performing an IoT pentest for a client. Processing the feedback of pentesters from Deloitte and modifying workprogram according to their remarks, will then result in the final product - final version of the ZigBee workprogram.

2.5 Conclusions

There are four questions defined for this research project. First research question (RQ1) is: What is the architecture of ZigBee protocol and what are its security features? The answer to this question can be found in chapter 4 “Setting up the experiment”. Second question (RQ2) is: How should IoT devices be secured? The answer to this question in form of a list of good practices for securing IoT devices can be found in chapter 5 “Analysis of hands-on knowledge”. Third question (RQ3) is: Which IoT pentesting standards exist and how can they be adapted for the ZigBee workprogram? The answer is explained in chapter 3 “Literature review”. Forth question: What should an IoT workprogram for a ZigBee supported device include? (RQ4), is the most complex one and several chapters contain information that contribute to answering it. However chapter 6 “Creating the ZigBee workprogram” contains the description of the final product and hence provides the answer to question RQ4.
3 Literature review

This chapter focuses on introducing the theory used in this research. First, there is a short summary of previously conducted research on the ZigBee protocol and its security. Next, an explanation is given on what motivates companies to perform pentesting on their IoT devices. It is followed by a literature review on available generic and IoT pentesting standards, which will contribute towards creating the ZigBee workprogram.

The chapter refers to the theoretical column of the research model and it answers question RQ3: Which IoT pentesting standards exist and how can they be adapted for the ZigBee workprogram?

Other concepts like ZigBee, XBee and securing IoT theories from column A of the research model, will be explained in Chapter 4: Setting up the experiment.

For the literature review, publicly available literature was chosen. This literature was created by technology alliances such as the ZigBee Alliance, governmental organizations, national institutes, industry associations and private researches. The majority of the documents was obtained from IEEE.org, hva.nl/bibliotheek and Google Scholar. Some of them were acquired from websites of associations and governmental bodies and a small amount was found on blogs and websites of private researchers. The materials include reports, white papers for conferences and presentations. The focus of the literature review is twofold, first it concentrates on the ZigBee protocol and IoT pentesting standards. It is necessary to understand the protocol in order to perform a successful test and IoT available pentesting frameworks are evaluated to extract relevant information for pentesting of ZigBee.

Section 3.1 explains previously conducted research on the ZigBee protocol and its security, section 3.2 describes the motivation of different companies to perform pentesting of IoT devices and section 3.3 analyzes theory of generic and IoT pentesting standards.

3.1 Related work

A considerable amount of documents about ZigBee security is available. Li et al. did an analysis of ZigBee security, with special focus on data transmission security services, encryption techniques and security keys (Li, Jia, & Xue, 2010).

Radman et al. presented a way of conducting security assessment based on compromised cryptographic keys (Radmand et al., 2010). The researchers were able to obtain cryptographic keys via remote and physical attack. Additional attacks such as eavesdropping, spoofing and replay were also discussed.

A group of researchers from Finland (Olawumi, Haataja, Asikainen, Vidgren, & Toivanen, 2003) described similar three attacks that were carried out in a laboratory environment, which included scanning, eavesdropping and replaying captured data. ZigBee security background and its vulnerabilities were also described by Whitehurst et al. where they discussed network key management and unauthenticated encryption amongst others (Whitehurst, Andel, & McDonald, 2014).

Morgner et al. investigated the state of security in connected lighting systems, that all operate using the ZigBee protocol (Morgner, Mattejat, Benenson, Müller, & Armknecht, 2017a). They analyzed security and threat model for ZigBee implementation at three vendors which produce smart light bulbs.
Not only theoretical research has been done on ZigBee, but also attempts to create a framework for conducting penetration tests. Wright (Wright, 2009) published KillerBee framework containing tools for exploiting ZigBee networks, which allows to sniff and analyze the traffic of ZigBee and any other IEEE 802.15-4 based network. Later, Zillner (Zillner, 2015) presented SecBee, tool for attacking ZigBee, based on scapy-radio and KillerBee. SecBee enhances the functionality of the original framework, by enabling testers to check encrypted networks and automatically perform specific tasks such as network leaves/joins, reset to factory default or search for unsecure key transport. There is also Z3sec framework, which can be used to send arbitrary touchlink commands and hence exploit security weaknesses in the touchlink commissioning procedure (Morgner, Mattejat, Benenson, Müller, & Armknecht, 2017b). In their paper, the authors describe methods of extracting key material by passive eavesdropping and taking over the ZigBee devices.

3.2 Drivers for pentesting

IoT is an emerging market, with a new wave of use cases and business models (Buyya & Dastjerdi, 2016). It is growing at a tremendous rate and Gartner predicts it will reach almost 20.4 billion devices by 2020 (Gartner, 2017). Companies also spotted the trend and increasing number of them is building and deploying IoT solutions. Boston Consulting Group estimates that by 2020, € 250 billion will be spent on IoT technologies, product and services (Hunke Nicolas et al., 2017). Here is where the security aspect comes into play. With the adoption of IoT, the attention of attackers is shifting from servers towards end devices (Buyya & Dastjerdi, 2016). It is easier to hack end devices, because they are less protected, better accessible and their number supersedes the number of servers. Organizations want to make sure that their day-to-day business is not threatened by attack of any sort. For Dutch economy only, value loss due to cyber-attacks is estimated to be € 10 billion (van Wieren, van Luit, Estourgie, Jacobs, & Bulters, 2016). That’s where professional pentesters come in place and check if organization’s systems are deployed with sufficient security and pose no threat to existing infrastructure. Defining the weak points in infrastructure and giving advice on mitigating vulnerabilities, can prevent financial losses, downtime and/or confidential data leakage (Tang, 2014). To perform an effective and complete pentest, a methodology such as a workprogram is needed. From the document, ethical hackers can get pointers on necessary actions and make sure that nothing gets omitted during the pentesting process. Such a workprogram specific for ZigBee devices, does not yet exist within Deloitte.

3.3 Pentesting frameworks

According to NIST, pentesting is a process of determining how effectively an entity being assessed (e.g. host, system and network) meets specific security objectives (Scarfone, Orebaugh, Amanda, & Orebaugh, 2008). The goal of this chapter is to analyze publicly available IoT pentesting standards and extract information relevant for creating ZigBee pentesting workprogram.

The preliminary phase of the research proved that there are not many pentesting standards available, which would be targeted specifically at IoT devices. We found mainly projects of security researchers or frameworks designed by private companies in the topic of IoT pentesting. There are on the other hand, many blogs and articles online, which include instructions on how to perform IoT pentesting and give advice on focus areas. Moreover, OWASP - Open Web Application Security Project has developed IoT testing guidance with the goal to help manufacturers build more secure products in IoT space. It helps to identify the risks by enumerating a list of actions that need to be taken in order to test IoT devices. OWASP is a non-profit organization, which means it does not endorse commercial products or companies (OWASP, n.d.). It is an advantage, because their advice is impartial and objective. Frameworks designed by private companies, as well as blogs and articles can be a very valuable source of information, but they have to be looked at critically with the acknowledgement that they cannot be treated as verified scientific resources. A collection of articles about IoT pentesting found has been
examine. Additionally, a decision has been made to analyze a generic pen testing framework, which is created by National Institute of Standards and Technology (NIST) (Scarfone et al., 2008). Framework was analyzed for information which is also relevant for pen testing IoT devices and was cross-referenced with papers from private companies and articles online.

### 3.3.1 OWASP

The OWASP IoT project is designed to help manufacturers, developers, and consumers better understand the security problems of IoT and make conscious security decisions when developing or using IoT products (OWASP, 2017a). The project consists of a list of IoT Top 10 vulnerabilities and IoT security guidance for manufacturers and testers. The guidance for manufacturers will be discussed in section 5.3 “Good practices for securing IoT devices”. The IoT Top 10 vulnerabilities are (OWASP, 2016b):

- I1 - Insecure Web Interface
- I2 - Insufficient Authentication/Authorization
- I3 - Insecure Network Services
- I4 - Lack of Transport Encryption
- I5 - Privacy Concerns
- I6 - Insecure Cloud Interface
- I7 - Insecure Mobile Interface
- I8 - Insufficient Security Configurability
- I9 - Insecure Software/Firmware
- I10 - Poor Physical Security

Actions I1, I5, I6 and I7 do not fall into the scope of the ZigBee WP, because they focus on mobile, cloud, web interfaces, and privacy.

The list provides not only guidance into security problems of IoT devices, but also ways to mitigate them. This can be mapped to the workprogram as areas that need to be tested during pen tests.

Additionally, OWASP has created a specific checklist for conducting pen tests in Tester IoT Security Guidance (OWASP, 2017b), which relate to the categories listed in the Top 10 vulnerabilities. The guidance is meant to be a basic set of guidelines, covering fundamental areas and improving security of IoT devices. For insufficient authentication/authorization (I2), a pen tester should assess whether the device uses strong passwords, password recovery, and password expiration mechanisms. Then, in Insecure Network Services (I3), the solution should be checked for test ports and possibilities of buffer overflow, fuzzing or denial of service attacks. Next in I4, the device should be checked whether encryption and firewall are enabled. Then in I8, the tester should verify whether password security options and encryption options are available, alongside logging of security events. Insecure software/firmware (I9) assesses the device with regards to update capability, as well as encryption and signing of those updates. Lastly on the list is poor physical security (I10), which validates the number of physical external ports, unused ports, and limiting administrative capabilities.

The information in the guides developed by OWASP is very relevant for creating the ZigBee workprogram. It shows which actions should be performed within the scope of the test. They form a good basis for a pentest, which then needs to be expanded with specific actions to test the ZigBee protocol.
3.3.2 NIST
National Institute of Standards and Technology published a Technical Guide to Information Security Testing and Assessment, which documents explain basic technical aspects of conducting penetration testing (Scarfone et al., 2008), to assess vulnerabilities and assist organizations in improving their security posture. It divides actions that need to be taken during a pentest into:
- Review techniques
- Target identification
- Target vulnerability validation techniques

The first step Review techniques, can be broken down to several actions. First is the documentation review. It determines if the technical aspects of policies and procedures are in place, because the documents provide the basis of an organization’s security. Then the purpose of log review is to check if an organization is logging appropriate data. System configuration review, on the other hand is spotting weaknesses in configuration controls. Another method of review is network sniffing technique, which monitors network communication and examines headers and payloads in the communication protocol.

Target identification includes network discovery, investigating which active devices, there are on the network and what is the network architecture; network port and service identification, which discovers open ports and services associated with those ports; vulnerability scanning, which identifies known vulnerabilities and finally wireless scanning, which discovers wireless signals in the area.

Target vulnerability validation techniques consist of password cracking, penetration testing and social engineering. Password cracking identifies weak or default passwords and password policies. Penetration testing verifies vulnerabilities and demonstrates how they can be exploited and social engineering allows for testing human element in security.

Even though NIST is not a specific framework for testing IoT devices, it does provide basic steps of pentest, which can be used in the ZigBee workprogram. Reviewing documentation about the architecture of a solution, needs to be done even in case of IoT devices, to understand how the protocol is implemented and which security measures are taken. The same applies to configuration of the system and event logging mechanisms. Discovering vulnerabilities on the network, checking for open ports and cracking passwords is also applicable to testing of IoT devices.

The only aspect of the framework, which is not relevant for the ZigBee WP is social engineering. With IoT devices, there is no human factor involved, therefore this step can be skipped.

3.3.3 Online publications about IoT pentesting
IoT is a complicated environment of different protocols, architecture and operating systems and that is why IoT penetration testing needs a different approach than regular pentesting.

In an article called “How to conduct an IoT pentest”, the author has explained that IoT environment usually consists of network, applications, firmware, encryption and hardware and IoT pentest should encompass all components mentioned above (Ryan, 2017). Hardware pentesting consists of deconstructing the device, identifying debugging interfaces or storage chips and obtaining firmware. The firmware is analyzed, configurations and executables extracted from it. Pentester needs to develop a blueprint of the device: what kind of hardware, software language, and communication protocols are used. Based on the blueprint and assessment plan can be developed. Hardware analysis is the evaluating of physical and hardware controls and inspecting interfaces (such as Joint Test Action Group (JTAG), or Universal Serial Bus (USB). In the firmware & OS analysis stage, tester checks whether the manufacturers implemented security best practices within firmware and OS. Testing security of the
firmware, update distribution process (e.g. cryptographically signing firmware updates). Boot sequence, code execution, application core dumps and data confidentiality protections need to be inspected. The goal of the wireless protocol analysis is to define its weaknesses by identifying device roles, cryptographic primitives, encryption keys, authentication and other algorithms. Also running analysis attacks like man-in-the-middle, replay, and unauthorized network commissioning belongs to this phase.

Attify, a company providing penetration testing for Internet of Things devices and applications has developed a guide for IoT pentesting (Attify Security, 2016). In the document they advise to map an attack surface before starting with pentesting an IoT device, for which an understanding of the IoT device architecture is needed. This includes analyzing product manuals and documentation, building an architecture diagram which shows all the components of the device and to think as an attacker and define weak spots that can be exploited.

Attify defines three potential attack surfaces. First is the embedded device, which can contain vulnerabilities such as communication over serial port, which gives root access, or the possibility to extract the firmware. Next is the software and applications. It can include firmware of the device, but also web and mobile applications, which are out of scope of this research. The last is the radio communication, which refers to the usage of communication protocols.

Rapid7 is a technology company, which also developed guidance to IoT security testing services. Just like Attify, they also claim that for an IoT pentest, the whole IoT solution should be investigated (Heiland, Deral, Sevier Nathan, & Littlebury Chris, 2017). The company has created an IoT security testing methodology, which consists of nine steps: functional evaluation, device reconnaissance, cloud focused testing, mobile application/control system focused testing, network testing, physical inspection, physical device attacks and radio-focused testing. We will only analyze the steps which are within the scope of this research and discard cloud and mobile application testing. Functional evaluation is the concept of setting up a regular environment with the IoT device that needs to be tested. This is needed to fully understand the product and map out the attack surface. If the budget and time allows, two environments could be set up, one regular and one altered to make comparisons. Device reconnaissance is enumerating available information, as also mentioned by Attify. In Rapid7 methodology, network testing consists of running vulnerability scans on TCP and UDP ports in the IoT infrastructure. Physical inspection assesses pins for power, data, serial pin-outs, debug/serial console access, USB access, and efforts required to disassemble the device. The next step, which is physical device attacks, aim at exploiting found vulnerabilities. The last step – radio testing verifies whether the IoT device has encryption, allows for unauthorized access or is vulnerable to replay attack.

The last guide which was selected, was the IoT pentesting guide (Gupta Aditya, 2017). The document contains a list of concrete steps on how to perform the test within five focus areas: embedded devices, firmware hacking, hacking communications, mobile and web endpoint, radio penetration test. The majority of the information is in line with Networkworld, Rapid7 and Attify documents. However is does contain a special section on ZigBee testing in the radio penetration test phase. The list consists of six steps as defined by Aditya Gupta:

1. Use ZigBee sniffer to sniff the ZigBee communications happening.
2. Identify the channel on which ZigBee devices are communicating.
3. Verify whether the communication is encrypted.
4. Verify whether you are able to capture the key exchange or whether the key can be found on the device or firmware.
5. Try to decrypt the communication.
6. Try to replay the communication packets to make the device act again. The test steps from this list will be inserted into the ZigBee WP.

3.4 Conclusions

In this chapter, the motives for conducting penetration tests and open sourced penetration frameworks are discussed.

First, the drivers for pentesting were defined. With the growing amount of the IoT devices in every organization, companies realize that it can have an impact on their security and their business continuity. The attention of attackers is shifting towards the endpoints such as IoT devices, because they are easier to compromise than servers. Pentesting verifies whether there are any weak points or vulnerabilities in the infrastructure. The knowledge of the vulnerabilities is the first step to mitigate them.

Then, OWASP IoT project with a checklist called “Tester IoT Security Guidance” was described. The list is targeted at pentesters of IoT devices and it points out which actions should be taken to test a device and its infrastructure and it is grouped per vulnerability. There are 10 vulnerabilities specified, but four of them are out of scope of this research, because they focus on mobile, web and cloud infrastructure and privacy.

Next, NIST framework is discussed, which provides a generic baseline for pentesting. NIST is a high-level framework, which indicates areas to be assessed during pentesting, but it doesn’t give much detail. Therefore, it can also be adapted by the ZigBee workprogram as a list of basic actions that have to take place. The only action which is not relevant for the workprogram is the social engineering, as it is not applicable for IoT devices.

Then, online publications about IoT pentesting are discussed. The approach to IoT pentesting of Networkworld, Attify, Rapid7 and IoT pentesting guide were described. All of the sources agree that IoT pentesting requires looking at the problem from the broader perspective of an ecosystem. There are similarities in their approach to testing IoT devices. All of the methodologies contain hardware, firmware and protocol analysis phases when conducting pentesting on IoT devices. The naming convention varies, but the actions stay the same. This means that these steps should also be included in the ZigBee WP.

Only one pentesting guide, specified actions for pentesting device with the ZigBee protocol, the other methodologies adhere to IoT devices in general.
4 Setting up the experiment
The goal of this chapter is to get practical experience with the ZigBee protocol and understand its design principles by conducting a security analysis on ZigBee enabled device and setting up a ZigBee network.

The chapter is divided into three sections. First, section 4.1 discusses the available literature on a ZigBee protocol and its security, which will help the reader understand the practical analysis section. This chapter refers to the theoretical column of the research model (column A) and its answers question RQ1: What is the architecture of the ZigBee protocol and what are its security features?

Next, in section 4.2 hands-on testing of ZigBee enabled smart light bulb system is described with the explanation of the selected hardware, software and the pentesting frameworks used. The chapter outlines the different stages of testing: sniffing the network traffic, interacting with the protocol and hardware hacking. The goal of this chapter is to get hands-on experience with the ZigBee protocol and understand the steps of testing the device. Learning by doing allows to comprehend how a real pentest will look like, which steps in the workprogram are necessary to add and which mistakes pentester should avoid. The chapter refers to the analysis part of the research model (column B) and it contributes to answering question RQ4: What should an IoT workprogram for a ZigBee supported device include?

The last section - 4.3 shows the efforts of creating a ZigBee network with the purpose of getting acquainted with the protocol and evaluating its security. The chapter relates to the analytical and intermediate product columns of the research model. It contributes to answering a final question, RQ4: What should an IoT workprogram for a ZigBee supported device include?

4.1 ZigBee protocol theory
ZigBee is a wireless communication standard based on IEEE 802.15.4. It was created by the ZigBee Alliance, an association established in 2002, as a collaboration of experts from businesses, universities and governments. It is a low-cost, low-power wireless technology, which can be embedded in a variety of applications such as home security, energy management, lighting controls and many more. The ZigBee is one of the most widely used IoT standards for wireless communication, adopted by many big players on the market such as Samsung and Philips (Fan, Susan, Long, & Li, 2017), according to which ZigBee is a global standard with mass market potential (ZigBee Alliance, 2012).

ZigBee Alliance has three network specifications: ZigBee PRO, ZigBee RF4CE and ZigBee IP. ZigBee PRO is designed to provide the foundation for IoT and it supports large networks with thousands of devices. ZigBee RF4CE is meant for simple control applications, which don’t require full mesh networking capabilities and ZigBee IP is the standard for IPv6 based wireless mesh networking solution. The focus of this research is on ZigBee PRO, due to its wide coverage and scalability.

4.1.1 ZigBee protocol stack
ZigBee stack architecture is built out of four layers: physical, medium access control, network and application layers as shown in the figure below. Physical layer (PHY) and medium access control layer (MAC) are based on the IEEE 802.15.4 standard, which is a protocol for lightweight wireless networks. The ZigBee Alliance has created a network layer (NWK) and the application layer (APL) on top of PHY and MAC (ZigBee Standards Organization, 2012).
IEEE 802.15.4 Physical and MAC Layer

Physical layer (PHY) provides an interface between Medium Access Control (MAC) layer and physical radio channel. It operates in two frequency ranges: 868/915 MHz and 2.4GHz. The first frequency covers 868MHz in Europe and 915MHz in the USA and Australia. The 2.4GHz frequency is used virtually world-wide (ZigBee Standards Organization, 2012). ZigBee standard has access to 16 separate 5MHz channels in the 2.4GHz band. The MAC layer is responsible for providing reliable communications, helps to avoid collisions and improves efficiency.

ZigBee Network Layer and Application Layer

Network layer (NWK), created by ZigBee, performs services such as starting the network, assigning network addresses, adding/removing network devices, routing messages, applying security and implementing route discovery. It also enables peer-to-peer, multi-hop mesh networking (Gislason, n.d.).

The Application Layer (APL) on the other hand, provides interoperability for application profiles, which are agreements over messages, formats and actions adopted by applications running on different devices (Francesco, 2012). Home Automation (ZHA), Light Link (ZLL), Building Automation (ZBA) are examples of application profiles. As announced by the ZigBee Alliance, they will be unified into a single solution, called ZigBee 3.0 (ZigBee Alliance, 2017). The application profiles are interoperable with devices from different manufactures (ZigBee Alliance, 2012).
4.1.2 ZigBee roles
There are three types of roles in the ZigBee network (ZigBee Standards Organization, 2012):

- **Coordinator** - single device in the network responsible for establishing, executing and managing the ZigBee network. Responsible for configuring security of the network and appointing address of the Trust Center (by default, address of the Coordinator, but can be set for alternate device). The coordinator needs to be powered on at all times and cannot enter sleep mode.

- **Router** - intermediary between coordinator and end-devices, relays packets for other devices.

- **End Devices** - devices that transmit or receive messages, but cannot perform routing operations. They often enter sleep mode to save energy.

4.1.3 ZigBee security
ZigBee implements security in three layers: MAC, NWK and APS. **MAC layer** security is based on the IEEE 802.15.4 standard. AES encryption with CCM is applied, which allows to use either authentication or encryption. To make sure that data is not altered in transit, Message Integrity Code (MIC) is appended to the messages (Libelium, 2009). At the **NWK layer** encryption and integrity is also provided by applying AES CCM encryption. **APS layer** bases the security of data packets on the link or network key and is responsible for securely establishing and managing cryptographic keys (Fan et al., 2017). An important feature of ZigBee protocol is that cryptographic protection such as AES is applied only between devices, and not between every protocol layer, because ZigBee is based on the “open trust” model (Zillner, 2015).

ZigBee standard offers two security models: centralized and distributed (Fan et al., 2017).

<table>
<thead>
<tr>
<th>Centralized security network</th>
<th>Distributed security network</th>
</tr>
</thead>
<tbody>
<tr>
<td>For high security applications</td>
<td>Destined for easier to configure systems</td>
</tr>
<tr>
<td>Nodes must support install code</td>
<td>Nodes must be pre-configured with a link key</td>
</tr>
<tr>
<td>Supports Trust Center, routers and end devices</td>
<td>Supports routers and end devices</td>
</tr>
<tr>
<td>Only ZigBee Coordinator/Trust Center can start centralized network</td>
<td>Routers are able to start distributed network</td>
</tr>
<tr>
<td>Nodes join, receive network key and establish unique trust center link key</td>
<td>Nodes join and receive network key</td>
</tr>
<tr>
<td>Trust Center periodically creates, distributes and switches to new network key</td>
<td>Network key is not periodically changed</td>
</tr>
</tbody>
</table>

Table 1 ZigBee security models

**Centralized model** provides higher security, but it is more complex, because it includes the Trust Center (network coordinator). The Trust Center forms the network and authenticates devices. It establishes unique link key for each device on the network. Similarly to distributed model, each device needs to be pre-configured with a link key, which is used to send an encrypted network key when sending it to another device.
In a **distributed model**, there is no central node such as Trust Center. Routers can start a distributed network and issue network keys. To participate in the distributed model, each device has to be pre-configured with a link key, which is used to encrypt a network key when sending it to another device. The network key is the same for all the devices.

The Trust Center can operate in two modes (ZigBee Standards Organization, 2012):

**The high security mode** is designed for commercial applications. Trust Center maintains and controls a list of devices, master, link and network keys. It also enforces policies of network key updates and admitting devices to the network, implements key establishment using Symmetric-Key Key Establishment (SKKE) and entity authentication. In this mode, the memory required for the Trust Center grows with the number of devices.

**Standard mode** applies to lower security, such as residential applications. Trust Center keeps only the network key and controls network access. Other information is stored on each node. The memory used, does not depend on the size of the network.

ZigBee standard provides over-the-air (OTA) updates, which allow the manufacturer to mitigate discovered security vulnerabilities, fix issues with the product and add new features. According to the ZigBee standard, all image transfers, over-the-air are encrypted, signed with a unique key (Fan et al., 2017).

### 4.1.4 ZigBee Keys

There are three types of symmetric keys in the ZigBee standard.

**Network Key** is an AES 128-bit key used for broadcasting communication and is shared amongst all the devices (Zillner, 2015). It is generated by the Trust Center and updated at different intervals.

**Link Key** is used for unicast communication and shared between two devices on the network, e.g. device and the Trust Center (Zillner, 2015). It is derived from the master key and managed by the APS layer. There are two types of link keys (Fan et al., 2017):

- **Global link key** - the same key for each device, the advantage is that the memory required for the Trust Center doesn’t grow with the number of devices. However, it was leaked in 2015 and has a default value of 5A 69 67 42 65 65 6C 6C 69 61 6E, which is hex encoding of ZigBee Alliance09 (Ronen, O’Flynn, Shamir, & Weingarten, 2017). Because it is public, the default value shouldn’t be used anymore.

- **Unique link key** - it is unique for each device, hence the security is better compared to the global link key model, where the key was leaked.

**Master Key** is a shared secret between devices with the purpose of establishing the trust. It can be pre-installed in the factory, installed by the Trust Center or can be based on data entered by a user e.g. PIN (ZigBee Standards Organization, 2012). Their function is to keep Link Keys exchange between two nodes confidential during Symmetric-Key Key Establishment (SKKE), which is explained in the next chapter.

### 4.1.5 Key Management

ZigBee keys are 128-bits in length and are acquired by means of pre-installation, key establishment or key transport (Zillner, 2015).
Pre-installation only applies to master keys, which are installed on a device by the manufacturer in a factory.

Key transport is achieved when the device makes a request to the Trust Center for the key to be sent. It is valid for requesting any of the three types of keys in high security mode. In standard mode, however, Trust Center only holds the network key. It is recommended to use an out-of-band transport.

Key establishment is a method of generating link keys based on the master key, hence the devices involved in the process need to be in possession of the master key (obtained by pre-installation, key transport or user input). The procedure is based on Symmetric-Key Key Establishment (SKKE) protocol (CERTSI, 2016), which is a key agreement scheme employed in the ZigBee mechanism (Yuksel, 2010).

4.1.6 ZigBee vulnerabilities

Even though the ZigBee Alliance made an effort in providing confidentiality and integrity for ZigBee enabled devices, vulnerabilities in the protocol or in the implementation of the protocol still exist. The significance of a vulnerability depends, however, on the profile of an attacker.

For low skilled attackers such as script kiddies, unencrypted keys or disclosure of configuration details will already be enough to exploit a system. In case a ZigBee deployment has higher maturity and those vulnerabilities are mitigated, the attacker might not be able to proceed with more advanced methods.

However, more structured and organized attackers, such as state actors have enough skills and persistence to exploit a vulnerability, such as breaking cryptographic protection or exploiting vulnerability in the implementation of the protocol.

This chapter explains most common vulnerabilities in ZigBee.

Disclosure of configuration details

First of all, ZigBee networks within range can be discovered together with its configuration details. It can be considered a vulnerability, because it could be a starting point for more complex attacks on ZigBee devices once initial intelligence is gathered. Also, it poses risk for privacy.

Default link keys

To provide interoperability of ZigBee devices, default link keys are used. They are not specified in the standard and shared only by ZigBee Alliance members developing ZigBee certified products, however default key for Home Automation and Light Link profiles was leaked in 2015 (Ronen et al., 2017). It can be now used to decrypt the network key and in consequence communication between two devices.

Unencrypted keys

Another potential vulnerability in ZigBee protocol is sending unencrypted network keys. When a new device joins the ZigBee network in a standard mode (residential mode), network key is send unencrypted, and hence it can be easily sniffed by an attacker and used to decrypt the communication afterwards.

Insecure key storage

Moreover, keys used in ZigBee devices are saved in memory, which means an attacker can extract them if he/she has physical access to the device.
Unauthorized network commissioning

ZigBee enabled device connects to the first network made available. This presents potential vulnerability, in case an attacker would force the device to leave the current network and join a fake one by sending command “Reset to factory default”. This has been performed in a paper “IoT Goes Nuclear” (Ronen et al., 2017) and was a starting point for another attack.

During the experiment, we will check for vulnerabilities mentioned above. We will verify if a device discloses configuration details, to get information about the ZigBee network e.g. channel number. Then, we will verify whether the messages are sent unencrypted over the network. In case encryption is enabled, we will test if it can be decrypted with default link keys. We will also perform hardware hacking, to check whether it is possible to extract key information.

4.2 Testing ZigBee enabled device

The goal of this section is to describe the efforts of setting up a test environment and hands-on testing of a ZigBee enabled device. Those activities have the objective to get acquainted with the software, hardware and frameworks used for testing ZigBee devices, as well as to understand the design of the protocol.

The experience of working with a ZigBee protocol, is essential in order to comprehend which pentesting activities ZigBee workprogram should contain. Hands-on pentesting allows to grasp how the ZigBee protocol works and how the testing environment should be prepared to make pentest as efficient as possible. Those insights will improve the quality of the ZigBee workprogram.

Installation of the Killer Bee framework is explained and it is followed by a chapter on protocol and hardware hacking. All sub-chapters contain the explanation of hardware and software used, accompanied by screenshots. The chapter relates to the analytical part of the research model and partially answers question RQ4: What should an IoT workprogram for a ZigBee supported device include?

One of the popular “smart lighting” products on the market was selected as a test objective, due to a big number of publicly known security research about the device. It is built on top of the ZigBee Light Link (ZLL) protocol, which is one of many, interoperable Application Profiles developed by the ZigBee Alliance.

As shown on the picture below, test set consists of a ZigBee enabled bridge and a light bulb. Devices communicate through the ZigBee protocol. The hub is also connected to the home router with an Ethernet cable and can be connected to the vendor mobile app.

The test will be divided in two parts: protocol and hardware hacking. During the protocol hacking, the security of the overall network and the implementation of the ZigBee protocol will be checked. The hardware hacking has a goal of verifying whether no sensitive data such as key material is stored on the bridge.

The device will be checked for the vulnerabilities described in the previous chapter, which include disclosure of configuration details, no encryption enabled, use of default link keys, insecure key storage and unauthorized network commissioning.
4.2.1 Preparation - installing KillerBee

There are several frameworks available for pentesting ZigBee devices. The most well-known is KillerBee published by Wright (Wright, 2009), which contains tools for exploiting ZigBee networks. It allows to sniff and analyze the traffic of ZigBee and any other IEEE 802.15-4 based networks. Two frameworks are based on KillerBee. One is Z3sec framework, which is also an open-source penetration testing framework (Morgner et al., 2017b) and SecBee, tool for attacking ZigBee, based on scapy-radio (Zillner, 2015). SecBee enhances the functionality of the original framework, by enabling testers to check encrypted networks and automatically perform specific tasks such as network leaves/joins, reset to factory defaults or search for unsecure key transport.

For the purpose of this research the KillerBee framework was selected. The goal is to set up a ZigBee testing framework in a form of a workprogram, in which the environment is able to intercept and interact with ZigBee traffic. KillerBee is the more advanced and community supported framework that provides the possibility for both sniffing the traffic and interacting with the network e.g. sending replay attacks. Moreover, both SecBee and Z3sec are based on the KillerBee framework and its core capabilities, hence it is more reliable to work with the original framework.

Several steps are needed to then deploy a KillerBee framework. The framework is supported on Linux OS. We used a clean Ubuntu 17.04 image on HP EliteBook 840 G3 with Intel ® Core ™ i5-63000U CPU @ 2.40 GHz and with 16GB memory.

To install the KillerBee environment, these commands were run from the terminal:

- `sudo apt-get install python-gtk2 python-cairo python-usb python-crypto python-serial python-dev libgcrypt-dev`
- `sudo hg clone https://bitbucket.org/secdev/scapy-com`
- `cd scapy-com`
- `python setup.py install`
- `sudo git clone https://github.com/riverloopsec/killerbee`
KillerBee requires hardware to sniff and interact with ZigBee. We used a RZ RAVEN USB (RZUSB) from Atmel, because it is recommended by the KillerBee creators and it is also easily accessible. The RZUSB stick with default firmware only supports passive functionality of KillerBee. In order to use extended capability and support injection, RZUSB stick has to be flashed with specific KillerBee firmware. It is recommended to use 2 RZUSB sticks, one for passive listening and one for attacking the ZigBee network. Detailed instructions for flashing the sticks and the firmware can be found here: https://github.com/riverloopsec/killerbee/tree/master/firmware.

To flash the sticks, we had to purchase:
- Atmel AVR Dragon On-Chip Programmer (ATAVRDRAGON)
- Downloader Cable for ZigBee

We downloaded AVRDUDE software and KillerBee firmware. Then, we connected Atmel AVR Dragon On-Chip Programmer via Downloader Cable for ZigBee to RZUSBstick. Picture of the connection can be seen below.

![Figure 4 Flashing RZUSB stick](image)

The following command was used to flash firmware:

```
avrdude -P usb -c dragon_jtag -p usb1287 -B 10 -U flash:w:kb-rzusbstick-001.hex
```

The RZUSB stick was flashed with the newer version of the firmware: `kb-rzusbstick-002.hex`. However, the commands from KillerBee framework did not work, so as troubleshooting activity, the stick was flashed with the first version of the firmware: `kb-rzusbstick-001.hex`, after which KillerBee commands started working.

The same procedure was executed on two RZUSB sticks, because it is recommended to use two devices, so that one can be used for transmitting and the second one for eavesdropping.
4.2.2 Protocol hacking

Hardware

In order to eavesdrop on the ZigBee network, a hardware device, which supports the IEEE 802.15.4 standard is needed. For this research CC2531 USB sticks and RZUSB were used. CC2531 in combination with software is able to sniff ZigBee packets on the network. RZUSB with stock firmware also allows only for passive functionality such as sniffing or receiving packets. However, when flashed with KillerBee firmware (discussed in more detail in the previous chapter), it can perform replay attacks, packet injection and impersonating other devices on the network. Both of the hardware devices can be plugged in directly into the USB slot of laptop or computer. Below is the picture of the dongles.

Software

Regular software tools for sniffing network like nmap, won’t work for a ZigBee protocol, because they operate on a different level. To sniff ZigBee traffic, special hardware which supports the IEEE 802.15.4 standard is needed. Texas Instruments Inc. created SmartRF Protocol Packet Sniffer, which can be downloaded from their website for free. It displays and stores, radio packets. Figure below shows an example of an output while sniffing ZigBee enabled device communication.
Another software product is Perytons Protocol Analyzer, which is commercial software for sniffing and analysis of ZigBee and IEEE 802.15.4, but also 6LoWPAN. It has a user-friendly interface, which can be seen on the screenshot below.

The KillerBee framework also has an option for sniffing ZigBee traffic. It is possible to collect information about devices on the network with the \texttt{zbstumbler} command. The tool hops between channels and transmits beacon request frames, displaying information that could provide useful (Wright, Cache, & Liu, 2010).
Additionally, ZigBee traffic can be captured and saved to a .pcap file with the use of the `zbdump` command. The picture below shows the execution of the commands.

Traffic capture saved in a .pcap file can be then opened in Wireshark, as shown in the picture below. Here we can see that two devices were communicating and exchanging data amongst each other.
The KillerBee framework has the tool to perform replay attacks. The idea of a replay attack is to use captured data, retransmit the frames as if the original sender was transmitting them again. Only devices with no encryption and no frame counter are vulnerable to replay attacks, so the assumption is that the updated smart light bulb system won’t be susceptible to this attack. It was verified with the experiment and indeed, the security measures protected the device from the replay attack.

The framework also provides tools to perform attacks on the encryption. Zbdsniff tool processes the packet capture file and searches for configuration of APS frames to retrieve and display the contents of network keys. However, it also didn’t return any key.

It is possible to emulate attacks identified with the literature with KillerBee framework. Due to the time constrains, we have focused however on the basic commands which allow to verify whether ZigBee system is secure or not. This is the goal of penetration testing engagements and the objective of every workprogram, including ZigBee WP. Extending the penetration test with other commands included in the KillerBee framework, should be part of future work.

4.2.3 Hardware hacking
The goal of this chapter is to describe the process of verifying whether hardware of a device contains any sensitive information, for instance passwords or encryption keys. Such information could help planning an attack on the ZigBee protocol, therefore hardware should be evaluated during a pentest. Hardware hacking can be a complement of ZigBee penetration testing, because shared keys or other sensitive information can be embedded in the hardware and identified during such tests.

As in the previous chapter, a smart light bulb system will be assessed. It has a ZigBee Light Link (ZLL) application profile and it consists of two devices: the light bulb and a bridge. The bridge is a central node, which controls all the light bulbs. For hardware hacking, bridge will be opened and analyzed. As part of the future work, the light bulb itself could also be analyzed for sensitive information.

There is a lot of information available about hardware hacking methods, tools and techniques online. Accessing this information helps understanding the architecture of the design, obtain datasheets and guidelines for more experienced people from the hacking community.

When assessing hardware, the first step is to open the device and verify whether it has any tamper proof mechanisms. In case of the bridge, there are hidden screws, but they can be easily unveiled. Once the device is open, physical inspection of the device needs to be done. The goal is to identify different components and services on the board.

The figure below shows the inside of the bridge. Pins were checked with the multimeter to identify which protocol they are using and whether there was any voltage on them, which could indicate data flowing through the pins. Red line marks the Joint Test Action Group (JTAG) and the blue line is the Universal Asynchronous Receiver-Transmitter (UART) protocol.
There was an activity on pin four and five counting from the right on UART (blue line). To verify if there is data flow, logic analyzer was used. It is a piece of hardware that allows to visualize digital signs and identify what is happening on a pin. The picture below shows connecting UART pin to Saleae Logic Analyzer and running dedicated Logic software.

No sensitive data appeared in the Logic Analyzer software, so the next step was to try opening a serial port connection. The serial port can be used for debugging purposes. It can reveal sensitive information, if not protected with a password or protected with a weak password.

To accomplish this, the bridge was connected to FTDI Basic Breakout, which then was connected via a USB cable to the laptop. We ran Putty software and selected Connection type: Serial, Speed: 9600 and Serial line: COM1.

![The inside of a bridge](image)

*Figure 10 The inside of a bridge*

![Running logic analyzer](image)

*Figure 11 Running logic analyzer*
This is the output from serial port opened in Putty.

As seen in the picture below, debugging mode was protected with a password. Easy to guess passwords were not successful, neither was an Internet search for leaked login credentials. During a penetration test, a password dictionary file such as rockyou.txt, can be used to verify whether the system uses common set of passwords. The list can be found in folder /usr/share/wordlists on Kali Linux or can be downloaded from the Internet.

**Figure 12 Serial port in Putty**

**Figure 13 Password protection in debugging mode**
4.3 Creating ZigBee network
The manufacturer of the light bulb system took a number of precautions to reduce the risk of hacking the device, which makes it more difficult to get acquainted with the protocol and understand how it works. Therefore a decision has been made to create a ZigBee network. This way it was possible to set the level of security and then test the response of the device. This method gives better understanding of the way the protocol works, which allows to better determine testing steps that should be included in the workprogram.

In this chapter, efforts made to create a ZigBee network will be described. Setting up the network and testing it will allow to gain deep understanding of the ZigBee protocol, its architecture and its features. This knowledge is needed to create a well-designed ZigBee workprogram, which is understandable for pentesters.

This chapter contributes to question RQ4, as it provides lessons learned and attention points which should be included in the workprogram, improving its quality.

The first idea was to use CC2531 USB dongles to create a ZigBee network. It is USB enabled, system-on-chip solution (SoC) manufactured by Texas Instruments for IEEE 802.15.4 and ZigBee applications (Texas Instruments Inc., 2010). The dongles would have to be flashed with Z-stack firmware and then configured with the use of Z-tool software. However, in reality the process was complicated, cumbersome and it did not succeed within the given time. The Z-stack user community is very small and there are not a lot of guidelines on how to successfully configure the devices to communicate. This is when the idea pivoted and Xbee modules were purchased to create a ZigBee network instead.

Xbee modules are radio frequency devices that transmit and receive data over air using radio signals (Digi International Inc., 2017). They are manufactured by Digi International Inc. and are based on the IEEE 802.15.4 networking protocol. They are designed for high-throughput applications requiring predictable communication timing. Multiple protocols, including ZigBee are supported (Digi International Inc. & Mouser Electronics, n.d.). Model S2C supports the ZigBee protocol and therefore it is selected for setting up the network.

4.3.1 Configuration steps
To create a ZigBee network, two laptops with Windows 10 Enterprise 64-bit and two Xbee modules were used. Xbee modules are not USB enabled, therefore a USB board was needed to connect Xbee to the computer.

XCTU software, developed by Digi International Inc. is used to set up, configure and test Xbee modules. It can be downloaded for free from the Digi International website. Once this is done, the modules have to be added to the program by clicking Select radio modules and then flashed with ZigBee firmware: XB24C, ZigBee TH Reg, and version 4060. It can be done from Configuration settings.

Next step is the configuration of radio modules. As discussed in chapter 4.1.2, there are three device types in a ZigBee network (ZigBee Standards Organization, 2012). Coordinator is a single device in each network that manages it. Acts as a Trust Center, providing security control of the network. Router relays packets for other devices. It is an intermediary between coordinator and end-devices. End devices transmit or receive messages, but cannot perform routing operations.

For the ZigBee network, decision has been made to configure one Coordinator and one Router. The following is needed to pair the devices:
- 1 Xbee setup as Coordinator (C)
- 1 Xbee setup as Router (R)
- The same PANID in both devices. It can range from 0000 to FFFF e.g. 2017.
- JV enabled for R
- CE enabled for C
- DL = FFFF for C

The figure below shows the setup that was created for testing. Besides two laptops with Xbee modules, there are also two laptops with RZUSB stick flashed with KillerBee firmware. One stick is used for performing attacks and the other one is sniffing network traffic for documentation purposes.

4.3.2 Security on the XBee
According to Digi International Inc. there are 3 parameters that can enable security on Xbee (Digi International Inc., 2017). First is EE – Encryption Enable parameter. When the parameter is set to 1, data transmissions are encrypted with the network key. All devices need to have the same value of the parameter in order to communicate.

Network Encryption Key (NK) sets the network security key. The Coordinator sends the network key to other devices, encrypted with the preconfigured link key when they join the network.

Encryption Key (KY) parameter sets the APS trust center link key. The default mode allows the devices to join the network without having a preconfigured link key.

For the purpose of this research, EE parameter was set to different values and security of the ZigBee network was tested accordingly.

4.3.3 Security analysis

Encryption disabled
Traffic sniffing is the first action to be performed when testing ZigBee devices, because it reveals information on what is happening on the network. First, PANID on the Coordinator was changed to 2017 and parameter EE set to 0. This means there is no encryption enabled. Then RZUSB was connected to Linux OS, zbdump command on channel 16 called and the output saved to a .pcap file. While KillerBee was sniffing for packers, parameter EE=0 was set and Router PANID was configured to 2017, pairing it with the ZigBee network. Then message “Hello” was sent from the Coordinator to the Router. After 5 minutes the capture was stopped and the .pcap file opened in Wireshark.
The Wireshark output below, shows that the message “Hello” is displayed in clear text and can be easily sniffed. Hence, if a pentesters observes the plaintext, it means that the device is vulnerable for sniffing attacks and communication cannot be trusted.

Figure 15 No encryption enabled Wireshark

Tshark is a terminal based version of Wireshark designed for capturing packets. Below is the output from Tshark, which shows that the message is in clear text. This means that encryption is not enabled, and anyone on the network can view them.

Figure 16 No encryption enabled Tshark

As next, replay attack was attempted without changing configuration settings of XBee modules. KillerBee provides a possibility to implement a replay attack with zbreplay command, which reads from a specified .pcap file and retransmits the frames. Pcap file captured in previous step will be used for the purpose of the attack. The screenshot below shows the command used. The attack succeeded and the packets were transmitted successfully. The message “Hello” is shown in clear text in the data payload, see above in Figure 17.

Figure 17 Replay attack, no encryption

Encryption enabled

When security in a ZigBee network is enabled, then key sniffing is one of the possible attack vectors. Capturing shared key could be used for decrypting messages and sending commands to devices. This can be done by capturing ZigBee traffic during pairing of devices.

To perform this analysis, PANID on the Coordinator was changed to 2018 and parameter EE set to 1, enabling the encryption – see screenshot below.
The work program for penetration testing of ZigBee enabled devices | Report

Security
Change security parameters

![Encryption Enable](image)

EE Encryption Enable: Enabled

Then RZUSB was connected to Linux OS, `zbdump` command on channel 16 called and the output saved to .pcap file. With the KillerBee sniffing for packets, Router PANID was configured to 2018, which paired it with the ZigBee network and parameter EE=1. Then message “Hello” was sent from the Coordinator to Router. After 5 minutes the capture was stopped and .pcap file opened in Wireshark.

When we look at the payload of the messages transmitted over the network, it is visible that they are not displayed in clear text, but are encrypted. Figure 21 shows the traffic capture of the ZigBee network. Packet 15 displays Association Request, sent from the Router to the Coordinator. After the successful association, APS: Command is sent to the Router. This command is encrypted with Key-Transport Key, also called Default Trust Center Link Key and is known as hex encoding of string “ZigBeeAlliance09” (Zillner, 2015). In principle the Network Key can be decoded with the Default Trust Center Link Key and when obtained, the entire communication between the devices could be decrypted.

This however did not work in practice and didn’t result in obtaining the key. Potential reason can be that Default Trust Center Link Key was different from the one used online. “ZigBeeAlliance09” is used for Home Automation, which includes smart light bulbs. For ZigBee network with XBee modules, the key might have been different.

Another approach for obtaining the key is using a specific command from the KillerBee framework – `zbdsniff`. It captures ZigBee traffic, looking for NWK frames and key provisioning (Wright, 2009). If a key is found, it will print the results. However, this method was not successful with XBee setup, as seen in the picture below. The file was processed, by no results were given.

---

Figure 18 EE=1, encryption enabled

Figure 19 Traffic capture
Figure 20 Searching for the key with the zbdbsniff command

It might be due to different encoding scheme, however further investigation should be done in order to receive a clear answer. Next possible step to decrypt the network key from the communication would be to involve crypto researchers, who are more experienced and see whether they could extract it.

4.4 Conclusions

The focus of this chapter was to explain ZigBee theory and use that information in practice by conducting hands-on assessment of the ZigBee enabled device and newly created ZigBee network. The goal was to get acquainted with the tools and the protocol itself, which is necessary to create the workprogram.

ZigBee theory

The theory explained the basics of the ZigBee protocol, which is a low-cost, low-power wireless technology. ZigBee stack is built of four layers: physical, medium access control, network and application layers. There are two security models. The centralized model is intended for high security applications and the distributed model was created for easier to configure systems. ZigBee uses the AES 128-bit encryption and three types of keys: network, link and master key. The keys can be managed by means of pre-installation, key transport and key establishment.

Testing ZigBee enabled device

In the next section, this information was used to assess the smart light bulb system which operates on the ZigBee protocol. It allowed to learn how to work with hardware and software tools, used for ZigBee testing. As in opposite to literature analysis alone, hands-on experience gave an impression of what pentesters will face during an engagement and this knowledge will be used to design an applicable ZigBee WP.

In the beginning phase of the test, the test environment was prepared by means of downloading software and ordering hardware. Software, which is used for testing ZigBee is KillerBee, SmartRF Protocol Packet Sniffer or Perytons Protocol Analyzer.

The protocol hacking entails passive test such as checking for disclosure of sensitive information and sniffing the network to verify whether messages are sent encrypted. Active attacks include the use of publicly known encryption keys to decrypt messages and performing replay attacks. During hardware hacking the bridge was closely investigated with the goal of verifying whether it is possible to extract key material or passwords. The UART protocol was connected to the logic analyzer, however, no sensitive data was flowing through the pins. Debugging mode opened via a serial port opened in Putty was protected with a password.

It turns out that the manufacturer has provided security measures against most of the attacks by using encryption and non-default passwords. More advanced techniques would have to be used to accomplish the goal of extracting sensitive information from the hardware and decrypting the traffic. This could include firmware analysis or advanced hardware hacking.
Creating ZigBee network

Therefore, the decision has been made to create a ZigBee network and test different security levels. The network was created with Xbee S2C modules and XCTU software. The test was performed on the network with and without encryption enabled. The messages with the encryption enabled were not readable, whereas without encryption were visible in plain text in the network capture. The default link key was used to try to decrypt the messages, however this did not succeed. It could be possible that Xbee uses a different link key.

The goal of getting hands-on experience with software, hardware and ZigBee protocol was accomplished, which led to better understanding of the ZigBee protocol and the approach to the penetration tests. The experiment allowed to acquire the knowledge of the protocol, hardware and tools needed to create the ZigBee workprogram. Understanding the perspective of a pentester and the challenges he/she will face, helps to create good quality workprogram, which satisfies needs of the target group.
5 Analysis of hands-on knowledge
This chapter describes the activities and efforts which contribute to the development of the workprogram for testing ZigBee enabled devices.

Chapters 5.1 and 5.2 cover analytical part of the research model (column B) and they contribute to answering question RQ4: What should an IoT workprogram for a ZigBee supported device include? Chapter 5.3 on the other hand, refers to intermediate products from column C of the research model. It fully answers question RQ2: How should IoT devices be secured?

First, chapter 5.1 explains the outcome of the interviews conducted with Deloitte experts. Next, in chapter 5.2 workprograms are analyzed and an explanation on how they can be adapted for the ZigBee workprogram is given. Then, chapter 5.3 presents the list of good practices for IoT devices.

5.1 Interviews
Interviews were chosen as a data collection method for several reasons. Pentesters working at Deloitte have vast knowledge about pentesting procedures and techniques and above that they have practical experience with pentesting IoT devices. Getting to know their experience, can significantly improve the quality of the workprogram. Moreover, they can share formal procedures of conducting pentest, which also need to be included in the ZigBee WP. Finally pentesters will be end-users of the document, therefore it is essential to understand their needs and wishes with regard to the structure of the workprogram.

The interviews were held in an unofficial setting. Small, non-random sample of participants was gathered. There is a group of SME’s at Deloitte, who work with IoT on a regular basis and have hands-on experience with conducting pentesting on IoT devices. Those professionals will be invited for an interview. The main goal of each discussion was to understand how an IoT penetration test looks like, including formal procedures and how the structure of the workprogram should be. The same set of questions was asked of each interviewee. Questions can be found in Appendix C.

1. Penetration test workflow
Interviewees explained how a formal penetration test looks like, from the kick off meeting with the client, through the signing of scoping documents and using a workprogram, for writing observations and the report. The information was very helpful to understand the process and structure of pentest and it will be included in ZigBee WP.

The interviews proved that so far IoT pentests were not conducted in a structured manner. There is no workprogram available, thus pentesters used parts of other workprograms, which were relevant for the scope of the engagement. If pentesters were not acquainted with the IoT protocol, they had to learn about the protocol on the spot and verify which hardware or software can be used. In essence these projects were conducted ad-hoc and without structure.

2. Mandatory steps of a penetration test
Even though no structured process, nor any IoT workprogram was available, formal processes such as signing contracts, defining the scope, reporting and communicating with the client were always conducted with due care. It is only the content aspect that lacks standardization, not the formal process. This includes information about given IoT protocol, which needs to be tested, along with the software and hardware tools needed. Once such document is in place, no time needs to be lost for researching the matter and educating the pentesters.
3. Structure of the workprogram

Expectations for the ZigBee WP were also discussed during interviews. Interviewees shared more practical tips on how to design a workprogram which is well structured and concise. Colleagues mentioned that steps should be clearly listed and explained. It should contain links to relevant websites with information and software. References to other workprograms should also be included. This way the ZigBee WP is user-friendly for pentesters. Even pentesters with no prior knowledge of ZigBee will be able to follow the workprogram. One of the interviewees mentioned an internal hacking database in the cloud that pentesters use to share knowledge, which might be a useful reference for some steps in the ZigBee workprogram.

The interviews confirmed the need for creating an IoT workprogram, as currently there is no specific IoT workprogram available yet. The standardization of IoT pentesting in form of a workprogram is needed, to improve the efficiency and scalability of conducting IoT penetration assessments. Currently, pentesters spend too much time researching and learning, before they are able to begin an IoT assessment. However, mandatory actions for conducting penetration test are in place and can be used in the ZigBee workprogram as well.

The workprogram will be the first step towards standardization of IoT pentesting processes and later automating and scaling up.

5.2 Analysis of available workprograms

The goal of analyzing workprograms used at Deloitte, was to understand what the structure of a workprogram is and which formal actions need to be taken during penetration test. Another objective was to verify whether any content from the available workprograms would be also relevant for the ZigBee WP.

Three workprograms were analyzed:

- Infrastructure Testing
- Firmware Reversing
- Hardware testing

“Infrastructure Testing” WP is one of the most widely used workprograms in the pentesting team, which is the reason why it was selected for the analysis. The content of the workprogram is not relevant for this research, however, it contains valuable information about mandatory steps that need to take place before starting and after finishing an official pentest.

1. Initial Steps

The first step in the workprogram are the Initial Steps. It contains the preparation phase, such as verifying whether the latest version of the workpaper is used, uploading scoping documents to internal systems, making sure that all prerequisites are satisfied. These are formal activities which are obligatory during an official penetration test, therefore the ZigBee workprogram should contain them as well. This step also includes information gathering, which is searching for publicly available client/vendor/product related information, such as passwords, encryption keys and any other sensitive data. It is also relevant for the ZigBee workprogram, because it allows to look for potentially sensitive data, which can then be used during a pentest.
2. Finalize Testing and Reporting

The last two steps of the “Infrastructure Testing” document are relevant. These are Finalize Testing and Reporting. Step Finalize Testing includes activities such as making sure that all evidence and logs are signed off and uploaded to the internal system. Reporting on the other hand consists of creating detailed observations for each finding in the workpaper. Both steps should be included in the ZigBee workprogram, as part of the formal framework.

“Firmware reversing” and “Hardware testing” also contain the same formal steps as “Infrastructure Testing”. However, some of the content is also relevant for the ZigBee WP.

3. Firmware analysis

Since practical experiments with firmware were out of scope of hands-on experiments, the steps to analyze firmware listed in the workprogram were the basis for creating a firmware pentesting section in the ZigBee workprogram. However the workprogram doesn’t discuss the way of obtaining firmware, but it immediately discusses the analysis. There should be an extra step added in the ZigBee WP clarifying how it can be done. Obtaining firmware can happen in several ways. It can be downloaded from the Internet, or extracted from traffic capture or extracted from the hardware. However, mentioned approach requires significant effort and firmware analysis is not the main focus of the ZigBee workprogram. Therefore, the advice for pentesters is to request firmware from the client during the kick off meeting to reduce the cost of the engagement.

The objective of the firmware analysis is to extract sensitive data such as passwords or encryption keys to attack the ZigBee protocol and the workprogram advises several programs to analyze firmware, such as Jefferson, Binwalk, Strings, Scalpel or Foremost. This software will be included in the ZigBee WP in the firmware analysis section.

4. Hardware testing

“Hardware testing” workprogram enumerates several steps to analyze hardware. Steps like understanding the architecture, identifying interfaces and services are relevant for any sort of device and are crucial for pentester to prepare plans of attack and gather initial information. Then the document discusses analyzing memory for sensitive information and connecting to identified ports.

Advanced injections such as SQL, XML and XSS are discussed as the attack factors on hardware, but they are not relevant for the ZigBee WP. The goal of hardware testing is to obtain sensitive information such as encryption keys and not try to verify whether the device has fail safe mechanisms in case of advanced injections.

5. Summary

The analysis of the selected workprograms proved to be of great value for creating the ZigBee workprogram. The examination of the documents gave an overall impression of the structure of a workprogram and allowed to understand the expectations of pentesters. The ZigBee workprogram should be created in a similar format, with the same structure in order for the pentesters to use it without confusion.

The “Infrastructure Testing” workprogram provided information about the template of the workprogram, along with information about mandatory steps that need to take place before and after every penetration test. The other two workprograms: “Firmware testing” and “Hardware testing”, contained relevant content for the ZigBee WP. Both firmware and hardware might be tested during a
ZigBee penetration test, for sensitive data such as encryption keys or passwords and the workprograms explain how it needs to be done, which can be replicated in the ZigBee WP.

5.3 Good practices for securing IoT devices

The goal of this section is to present specific guidelines for securing IoT devices in a form of a list of “good practices”. The term “good practices” is used deliberately here, because it means using recommended practices, whereas “best practices” implies that it is a state-of-the-art practice. This chapter refers to column C with intermediate products and tries to answer the question RQ2: How should IoT devices be secured?

The list was created based on the literature analysis and it will work as an input for the first draft of the pentesting workprogram. In recent years there has been an increase in published materials regarding good practices for IoT security. Bruce Schneier, an internationally renowned security technologist, has published a list of IoT security and privacy guidelines (Bruce Schneier, 2017). The list contains frameworks and guidelines for securing IoT devices, created by governmental organizations, technology alliances and security working groups.

The list has been scoped down to the documents containing information about endpoint, network and protocol security in IoT devices. Guidelines regarding privacy and specific type of IoT devices such as medical or industrial devices has been discarded.

Good practices for securing IoT devices have been created based on the analysis of the documentation. The list shows where companies, developers or even end users should focus on, to provide a sufficient level of security for their IoT devices. Those guidelines will be incorporated into the ZigBee workprogram, as aspects that need to be checked during a pentest.

1. Hardware

Hardware plays an important part in making a device secure. The hardware should be scoped to the minimum requirements and features required for the device to be operational. Such hardening can be accomplished by implementing port locks and camera covers, which lock out USB and Ethernet ports (OWASP, 2016a). In case some functionality is needed, e.g. USB port, its use should be monitored and logged. IEEE Internet Technology Policy Community (IEEE, 2017) goes along with this thinking and stresses that devices should also be tamper-proof. Mechanisms to detect physical tampering, such as opening the device cover or removing part of the device should be deployed. Vulnerabilities like open TCP/UDP ports, open serial ports, password prompts, places to inject code like web servers, unencrypted communication and radio connections should be protected. Debugging and informational interfaces, running on the device should be restricted to users with physical access to the device and protected with a strong password (U.S. Department of Homeland Security, 2016). Packaging of the device can also be tamper-evident, showing if the package has been open before the arrival.

2. Boot

In order for the IoT device to run securely, the boot process must be assured and trusted (Internet of Things Security Foundation, 2016). If a hacker finds a way to replace the executable file with malware and the device runs it, the device is compromised. Secure booting entails that when power is introduced to the device, the integrity of the software is verified using cryptographically generated digital signatures. Keys can be stored on dedicated chips such as Trusted Platform Module (TPM). Only the software that is authorized to run on the device will be loaded. Additionally, strong boot-level passwords should be implemented (IEEE, 2017).
3. Updates and patches

Another security aspect is the patching process, which should be well planned to mitigate potential risk factors. There should be an automated mechanisms for addressing vulnerabilities in place, which process information from critical vulnerability reports and hacker communities in real time (U.S. Department of Homeland Security, 2016).

The updates and patches should be encrypted and signed with a digital signature (OWASP, 2016a), and preferably send over-the-air (OTA) to limit manual work.

4. Network

Minimizing device bandwidth – limiting the amount of network traffic IoT devices can generate, should also be considered (IEEE, 2017). This could limit DDoS attacks, during which IoT devices are used as zombie systems. The amount of network traffic that can be generated by IoT devices should be minimized to the level needed only to perform their function. Devices should operate on a minimal number of required ports active and they should have a firewall or the deep packet inspection capability to filter specific data destined to that device (OWASP, 2016a).

Network with IoT devices should be separated by a firewall or a gateway from the rest of the network. This way the security parameter is more isolated (Broadband Internet Technical Advisory Group, 2016). In case a ZigBee network needs to be connected to the network, the traffic should be filtered (Masica, 2007). This could be done in the MAC layer via Access Control List (ACL), which would accept MAC frames only from authorized nodes (Masica, 2007).

5. Device authentication

Default passwords on IoT devices should be disabled as well as debug mode settings and backdoor’s (IEEE, 2017). Each device should have a unique password, complex enough to resist simple brute force attacks (OWASP, 2016a). Recommended method is a two-factor authentication (2FA), applied whenever it is possible. Device should be able to establish trust with other devices by means of authentication, regardless whether the device is on the local or external network (Broadband Internet Technical Advisory Group, 2016). There should also be a password expiration policy in place (OWASP, 2016a).

6. Encryption and protocols

Encryption should be deployed on an IoT device, to prevent eavesdropping and capturing sensitive information. Communication in transit should be protected with strong cryptography protocol like Transport Layer Security (TLS) or Secure Sockets Layer (SSL) (IEEE, 2017). The device should use the encryption security service, based on the AES algorithm. Out-of-band method should be applied for loading cryptographic keys onto IoT devices, which entails using a different method than wireless channel. An example, could be a serial port on the device through which key could be loaded (Masica, 2007).

7. ZigBee protocol

Security features on the lower layers of the stack should be enabled. IEEE 802.15.4 provides MAC security. The security architecture of the ZigBee deployment, should provide maximum security for the Trust Center, which is responsible for distribution of cryptographic keys (Masica, 2007).
5.4 Conclusions
The goal of this chapter was to analyze the results from conducted interviews, examine selected workprograms used at Deloitte and describe a list of good practices for securing IoT devices.

Interviews

Interviews were held with selected pentesters from the hacking team, who have experience with IoT pentests. They gave advice on the structure of the workprogram, which should be concise and easy to understand, with links to software tools, references to other useful workprograms and hacking knowledge database.

Interviewees described the process flow of a penetration assessment. Every pentest starts with a kick off meeting, signing the scoping documents, following the workprogram and writing the report.

Based on the interviews, it appears that there is no standard workprogram for conducting a ZigBee IoT penetration test. For IoT penetration tests, pentesters have to do research and create ad-hoc workprograms, which consumes a lot of their time and is inefficient. The pentesters admitted that there is a need for creating an IoT workprogram.

There is a list of mandatory actions, which need to take place before and after each test. This includes initial steps, finalizing the test and writing the report. They should be included in the ZigBee WP.

Analysis of available workprograms

Those mandatory steps are also outlined in every workprogram, which is in use at Deloitte. Three workprograms were analyzed for relevant content and information about the structure.

“Infrastructure testing” workprogram gave information about the mandatory actions, which are obligatory during each pentest. Every pentest starts off with signing documents, discussing scope with the client, gathering initial information and finishes with finalizing the pentest, uploading evidence, writing observations and the report. These mandatory steps will be included in the ZigBee WP.

“Hardware testing” workprogram includes the steps of analyzing hardware, which consist of understanding the architecture, identifying interfaces, analyzing memory and connecting to identified ports.

“Firmware reversing” has a goal of extracting sensitive information and can be done with several tools. Jefferson, binwalk, strings, scalpel and foremost are software tools, which can help achieve the goal.

Even though hardware and firmware hacking is not the main objective of the ZigBee workprogram, it can help test the ZigBee protocol by extracting encryption keys or other sensitive data like passwords, which then can be used to decrypt communication or log in to serial port and take control of the device. Therefore, these actions will be also used in the ZigBee WP.

Good practices for securing IoT devices

The last section discusses the list of good practices for securing IoT devices, which was created based on publicly available literature, published by technology alliances, governmental institutions and security research groups.

The list gives advice on securing different areas of an IoT device to prevent attacks. These areas include securing hardware of a device, boot process, correct implementation of updates and patches, securing network, device authentication and encryption and protocols.
The list of good practices for securing IoT devices provides valuable input for the ZigBee WP. It points out areas of IoT devices that should be secured, which can be mapped to the ZigBee workprogram in a form of steps for conducting pentest. The list can be also useful for manufacturers, developers and end users.
6 Creating the ZigBee workprogram

This chapter describes the creation of the preliminary version of the workprogram, which is the intermediate product from the column C of the research model. In total there are three intermediate products. Good practices for securing IoT devices and analysis of ZigBee security are described in previous chapters. This chapter focuses on preliminary version of the ZigBee WP, which is the last intermediate product from the research model.

It is followed by the validation process of the preliminary version of the workprogram, resulting in the final version of the ZigBee workprogram, which is the desired outcome of this research. Based on input from previous chapters, this chapter answers the question RQ4: What should an IoT workprogram for a ZigBee supported device include? For the final version of the ZigBee workprogram please refer to Appendix A.

6.1 Preliminary version of the ZigBee WP

A first draft of the workprogram was created based on the results from previous chapters. Summaries and conclusions drawn from literature review, setting up the experiment and analysis of hands-on knowledge will be used as an input for the preliminary version of the ZigBee WP.

The workprogram starts with a list of pre-requisites, which contain descriptions of hardware and software needed to perform the pentest. There is also a link to the location on a share drive where the test VM’s are stored. This list was derived from lessons learned from setting up the experiment described in Chapter 4 and can be seen on the picture below.

![Pre-requisites](image)

Pre-requisites

For testing ZigBee devices two ISO with Linux OS are needed; one for sniffing and one for attacking ZigBee enabled device. Depending on your choice of hardware and software, some programs only run on Windows machine, so it is good to have one Windows machine as well.

Prepare hardware

- You can get the hardware from the asset shop xxx.
- Tip: If asset shop doesn’t have the hardware you need, ask xxx.

List of hardware:

- Multimeter
- Saleae Logic Analyzer
- 2 x RZ RAVEN USB sticks flashed with KillerBee firmware
- CC2531 dongle for sniffing traffic

Prepare software

- Download ZigBee Windows ISO and ZigBee Linux ISO from the network drive, path: xxx
- Windows ISO contains Texas Instruments SmartRF Packet Sniffer and Peryton ZigBee analyser
- Linux ISO has KillerBee installed

Figure 21 Preliminary version of the ZigBee WP: list of pre-requisites

Usually, for IoT pentesting, more preparation time is needed than in the case of a regular pentest, because specific hardware might be required to perform the test. Delivery of hardware can take up to several weeks. When planning the IoT engagement, the aspect of relevant tools and hardware needed, has to be taken into account and the list of pre-requisites assist in this process.
In total there are six test activities in the preliminary version of the workpaper: initial steps, hardware analysis, firmware analysis, ZigBee analysis, finalize testing and reporting.

**Initial steps** covers a list of mandatory actions that a pentester has to perform before starting the penetration test. These actions include informing the client about the start of the test, signing documents or uploading them to the internal system. List of activities is based on paragraph “Mandatory steps of a penetration test” from section 5.1 and “Initial Steps” which can be found in section 5.2.

**Hardware analysis** section builds on hands-on practical experience described in chapter 4 and paragraph “Hardware testing” from section 5.2. It includes the following activities: understanding the architecture of the device, identifying interfaces and services, connecting to them, checking for keys in the memory. The scope of this section however is limited to a specific goal, which is extracting sensitive information to attack a ZigBee protocol.

**Firmware analysis** was not part of the hands-on pentesting and it was created based on literature review and workprograms available at Deloitte. It derives from the “Firmware analysis” paragraph in section 5.2. The goal is similar as with hardware hacking and it focuses on extracting sensitive information such as encryption keys or passwords to attack ZigBee. The steps include obtaining firmware, splitting firmware, checking files.

**ZigBee analysis** section builds on the information gathered during the literature review presented in section 4.1, along with a practical analysis section in 4.2 and 4.3. It contains several steps: identifying channel, sniffing network traffic, verifying if communication is encrypted, sniffing the key, checking for reusing of the keys and performing replay attack.

**Finalize testing and reporting.** The test is concluded with gathering the evidence and writing reports with observations. Similar as with initial steps, those sections of the workprogram were designed based on current workprograms and discussions with the SME’s. The information can be found in paragraph “Mandatory steps of a penetration test” in section 5.1 and “Finalize testing and reporting” in section 5.2.

**6.2 Final version of the ZigBee WP**

Once a preliminary version of the ZigBee workprogram was finished, validation process has started. The main goals were to verify whether the activities listed in the workprogram are clear for pentesters with little or no experience with the ZigBee protocol, whether pentesters can follow the workprogram without any problems and whether it satisfies their needs. Processing the feedback resulted in the final version of the ZigBee WP, which can be found in Appendix A.

Validation process was divided into two activities:

- Tabletop assessment
- Hands-on assessment

**Tabletop assessment**

As part of the tabletop assessment feedback request was sent to three technical colleagues to analyze the document, with the goal of assessing the overall quality of the workprogram. This includes verification whether the document is readable and understandable.
**Hands-on assessment**

Hands-on assessment, however, was a practical session, with an experienced hacker from the IoT team, who was invited to perform a pentest on the ZigBee enabled device according to the steps listed in the workprogram.

The goal was to check whether the pentester with no previous experience with the ZigBee protocol, can reproduce the activities enumerated in the workprogram and whether the structure, layout and content are logical. The workprogram will serve as an obligatory manual for pentesters during testing ZigBee enabled devices. Therefore, it needs to be understandable and cover the most important areas which have to be tested.

**Results of the hands-on assessment**

The session resulted in extremely valuable feedback. The structure of the workprogram and the order of the testing steps were challenged. The preliminary version of the workprogram starts and finishes with mandatory formal steps, which remain unchanged. After initial steps, hardware and firmware hacking is performed and as a last step the ZigBee protocol analysis is performed, which is explained in more detail in the previous chapter. However, hardware and firmware testing has only one goal, to get access to passwords and/or encryption keys, which will allow to decrypt ZigBee communication. There is no reason for performing those two steps, if ZigBee implementation doesn’t include encryption. Therefore, these steps should be moved to the end of the workprogram and be marked as optional. In the final version of the ZigBee workprogram, after initial steps, the pentester goes directly into ZigBee protocol analysis, which includes references to hardware and firmware testing sections in case additional attack vector is needed to perform a full pentest.

The steps listed in the workprogram should have a clear rationale outlined. This prevents pentester from skipping steps, which might seem unnecessary without an explanation. Moreover, it was pointed out that the workprogram should contain information on what observations to report, in order to guide pentesters and help them to come to the right conclusions. Another advice, was to create a glossary of ZigBee terms at the end of the workprogram, as a reference for pentesters who are not familiar with the ZigBee terminology. For better usability, links to downloading software should be placed in the document, in case testers want to use their own machine instead of prepared ISO’s.

Appropriate testing locations should be specified in the workprogram. Ideally, it should be no noise location, such as a Faraday cage. However, it is expensive to build and cumbersome to conduct pentest in. A cheaper and easier option is creating a self-made version of the Faraday cage made of a plastic box where the devices are placed, covered with aluminium foil. Alternatively pentesters should be in a safe noise environment, where he/she knows which traffic comes from which device. This information should be included in the workprogram to prevent receiving random signals when sniffing the ZigBee protocol.

Gathered feedback was processed and preliminary version of the ZigBee WP was changed according to the remarks. This resulted in the final version of the ZigBee workprogram, which has six steps: initial steps, ZigBee analysis, finalize testing, reporting, hardware analysis (optional) and firmware analysis (optional).

**6.3 Conclusions**

Chapter 6 introduces the preliminary version of the workprogram, which is an intermediate product from the research model. Furthermore, it discusses the validation process and the outline of the final version of the ZigBee workprogram, which is the desired outcome of this research. The complete version of the ZigBee workprogram can be found in Appendix A.
Preliminary version of the ZigBee WP is the first attempt towards designing a workprogram for testing ZigBee enabled devices, which will be used during IoT pentesting engagements. It should be logical, concise and straightforward, so that pentesters are able to work with it on a daily basis. The structure of the document should be similar to the structure of workprograms used currently at Deloitte. This can avoid the confusion of pentesters who will work with the ZigBee WP.

The preliminary version was validated by means of the tabletop and hands-on assessment. Based on the received feedback, the conclusion is that the workprogram has to be more self-explanatory, in such a way that pentesters with no experience with the ZigBee protocol, would be able to perform pentest on ZigBee enabled devices. A glossary with ZigBee specific terms should be created to help pentesters understand the abbreviations. The structure of the workprogram has to change to reflect the main focus, which is the ZigBee protocol, whereas hardware and firmware hacking should be listed as optional steps. The workprogram should also include links to websites where software can be downloaded and information about testing location to prevent invalid results from pentesting in a noisy location.

Received feedback was processed and resulted in the final version of the ZigBee workprogram, which can be found in Appendix A. The ZigBee WP has six steps: initial steps, ZigBee analysis, finalize testing, reporting, hardware analysis (optional) and firmware analysis (optional). It contains an explanation of the pentest environment with regards to used hardware and software. In the document, pentesters can find references to helpful documents such as internal workprograms or Github repositories. Each step has two columns to be filled in by a pentester: results of the test and the evidence.

The workprogram allows to structure the process of pentesting of IoT devices with the ZigBee protocol, by providing clear instructions with regards to the tools needed and steps of the penetration test.
7 Discussion
The growing presence of IoT devices in our lives poses a major challenge to not only make them compatible with other devices, but more importantly, secure. The focus of this research is to develop a methodology, in a form of a workprogram for conducting penetration tests on IoT devices with the ZigBee protocol. The result is a first milestone in achieving standardization in the process of testing IoT devices.

The research methodology is based on Verschuren and Doorewaard model (Verschuren & Doorewaard, 2010) and it consists of three activities: theory, analysis and intermediate products, which result in creating the final version of the ZigBee workprogram (see Appendix A). During the course of the research, there have been minor improvements made to the research model and the methodology, but the final result of the research, the ZigBee workprogram stayed the same.

The first stage of the research was the theoretical phase from column A of the research model. During the preliminary phase, the scope of the research has been defined, as well as the sources of literature. The literature review provided relevant information about the ZigBee protocol, its architecture and its security. This information is needed to conduct a successful penetration test. Furthermore, available pentesting frameworks for performing pentests on the IoT devices were discussed. It gave information about potential steps which should be included in a workprogram for pentesting IoT devices and its structure.

The next phase was the analysis, which consisted of four activities. First interviews were held, during which information was gathered on how pentesting assignments are performed at Deloitte. Then, available workprograms were analyzed to understand the structure of such documents and extract information relevant also for the ZigBee WP. Next, an experiment was set up, during which a smart light bulb and ZigBee network were assessed.

The development of the intermediary products was the third phase of the research. List of good practices for securing IoT devices was developed. Then, an analysis of the ZigBee security was described and finally the preliminary version of the ZigBee workprogram was created.

Gathered input from the intermediary products together with the validation process, resulted in the final version of the ZigBee WP, which can be found in Appendix A.

7.1 Research accountability statement
The methodology was followed throughout the course of the research, nevertheless some ideas pivoted. Each change to the research model was discussed and validated with the supervisors.

One of the major additions to the research model was the decision to create a ZigBee network and analyze its security, complementary to testing a smart light bulb. It was motivated by the fact that security in self-created ZigBee network can be set manually and differences in the behavior of the protocol can be well understood, in opposite to well protected smart light bulb system, which is a challenging device to start learning about the ZigBee protocol. To create a ZigBee network, Xbee modules were used. Hence the theory about Xbee modules was also added to the research model.

Moreover, a decision was taken to remove the IoT theory from the research model as a separate subject, due to the broad scope of this topic, which doesn’t bring added value to assessing the ZigBee protocol and creating the workprogram.
Finally, analysis of available reports from IoT pentests was removed from the research model after the preliminary research. This is due to the fact that those reports only discussed observations from the tests and did not provide relevant information such as steps to be taken during an assessment.

In order to obtain reliable results, the preliminary version of the ZigBee workprogram was validated. The goal of this procedure was to check whether the product meets the requirements of the pentesters and fulfills its purpose. Two methods of validation were used to assess the workprogram: table top and hands-on assessment. The first one, is an assessment of the document with the goal of checking the overall quality of the workprogram. The second one, has a goal of performing a hands-on assessment on the ZigBee enabled device according to the preliminary version of the workprogram to verify its structure and contents. Both types provided a valuable feedback, which was processed and resulted in the final version of the ZigBee workprogram.

7.2 Ethical considerations
The main topic of this research is ethical hacking of IoT devices with the ZigBee protocol. There are some ethical considerations that need to be taken into account when performing ethical hacking.

Hacking IoT devices, can encompass manipulation of the radio spectrum, such as capturing the traffic and replaying it by broadcasting a radio signal. Radio spectrum is a limited public resource. Modifying it could interfere with critical systems, homeland security and army equipment.

In case of hacking ZigBee enabled devices, pentesters have to be careful not to break the law. It is advised to use a Faraday cage (RF cage). However, building an RF cage is quite impractical and expensive to use. An alternative could be a self-made RF cage. Broadcasting device can be placed in a plastic box and covered with aluminium foil. This way the broadcasting distance will be limited to several meters, not impacting other devices and breaking the law. However, this makes the process of pentesting harder, as the device cannot be easily reached and is not sustainable for large scale engagements.

In July 2017, there has been an update published to telecommunication regulation from 1998 (“Telecommunicatiewet,” 2017), according to which special permission from the government is needed in order to work with public frequencies. On one hand, new broadcasting legislation is a clever move, because it proposes penalties for broadcasting illegal radios, which can cause damage or disrupt the work of airports, army and such. On the other hand, it also makes it very difficult for security researchers and professionals to perform their job, which is finding vulnerabilities in the hardware sold to end customers.

To improve the security and safety of the customers and prevent large scale attacks, there needs to be a consensus between the regulator and security specialists, in which a method of testing will be established and agreed upon. This method should be created according to the law and should allow testers to maintain high professional standards.

One of the methods to differentiate security research from hacking is creating a procedure of responsible disclosure, which researches could use to make their findings public.

7.3 Evaluation of the workprogram
The goal of this research is to create a workprogram for pentesting ZigBee enabled devices, which is the first step to automate the process of pentesting IoT devices.

The ZigBee workprogram structures the process of conducting pentests at Deloitte and makes it more time efficient and scalable. Ethical hackers with no previous experience with the ZigBee protocol, will
be able to perform a penetration test on a ZigBee enabled device, when following the activities listed in the workprogram. The document contains basic information about the ZigBee protocol and the test environment, which includes hardware and software needed to successfully pentest a device. Thus, it provides a quick introduction into the world of ZigBee.

Moreover, the ZigBee workprogram has the same document format as other workprograms used at Deloitte, so that pentesters will be able to follow it without confusion, since they are familiar with the structure.

Despite its clear advantages, the workprogram has also several limitations. First of all, the workprogram covers only the KillerBee framework. There are other frameworks which can be used for pentesting ZigBee, such as SecBee or Z3sec, however, they were not discussed in the document. SecBee enhances KillerBee framework, by adding commands to automatically test for the usage of default link key and insecure re-join. Z3sec provides commands attacking ZigBee touchlink commissioning, which is a user-friendly way of adding new devices to a ZigBee network.

Secondly, the workprogram does not cover all possible attack vectors. The scope was narrowed down to verify whether the most common ones discussed in the literature are present (e.g. replay attack). Lastly, the hardware mentioned in the workprogram is not the only hardware which can be used for testing a ZigBee device. There are alternatives, which could be added to the workprogram. Expanding the workprogram with new attacks and new tools, should be part of the future research. The framework should be under constant development to continuously improve its quality and enhance the scope of testing. Although we did not obtain the encryption key during practical, hands-on testing, we could verify that tooling used for ZigBee pentesting is correct and well-functioning and we were able to understand its logic.

### 7.3.1 Financial aspect of the ZigBee WP

#### Costs

To use the ZigBee workprogram, initial investment from the management is needed. Table 2 shows the list of hardware and software components and its prices.

The basic set of hardware for testing the ZigBee protocol has to be purchased. Hardware for hacking purposes can be time-consuming to obtain, due to governmental regulations and can take up to several months. The hardware should be available to pentesters before the actual engagement is signed, to prevent a situation where a client goes to the competition, due to long waiting time for the equipment and hence for the start of a project.

<table>
<thead>
<tr>
<th>Component</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 x RZ RAVEN USB</td>
<td>€80,-</td>
</tr>
<tr>
<td>1 x Multimeter</td>
<td>€30,-</td>
</tr>
<tr>
<td>1 x Saleae Logic Analyzer</td>
<td>€200,-</td>
</tr>
<tr>
<td>1 x Perytons Protocol Analyzer</td>
<td>€620,-</td>
</tr>
<tr>
<td><strong>Investment</strong></td>
<td>€930,-</td>
</tr>
</tbody>
</table>

Table 2 Investment in hardware and software tools
Moreover, software licenses have to be purchased, which includes Perytons Protocol Analyzer. Other software is open sourced and can be downloaded for free. According to the calculations in the Table 2, the investment in tools is 930, - euros.

In order to scale up the operations in the future, bigger quantities of the tools will have to be purchased to enable pentesters to work on multiple projects at the same time.

Benefits

The newly designed ZigBee workprogram brings tremendous benefits to the organization, because it cuts down the operational costs. These costs usually take up the most of the profit of an engagement. If they can be reduced, organization’s profit margin will increase, while at the same time they will be able to offer competitive prices.

Currently, when a contract for the IoT penetration test is signed, the preparation takes about 15 man-days. This includes researching about the ZigBee protocol, coming up with the approach and developing a rough version of a workprogram to test it. Then it takes another 7 man-days to set up the test environment, purchase and/or gather appropriate hardware and software tools. This gives in total 22 man-days, which equal to 176 hours.

By introducing the ZigBee workprogram to the team, this time will be reduced from days to hours. The workprogram lists necessary tools and points to the location of specially created VM’s for testing ZigBee. This way the creation of the environment takes as little time as possible as it is limited to several hours, instead of days.

The workprogram also guides the researchers and points out areas of the ZigBee implementation, which need to be tested. No research or ZigBee training beforehand is needed. Pentesters can start with pentesting right away, without spending hours on understanding the logics of the protocol and the ways of exploiting it.

At present, there are maximum 2 or 3 people at Deloitte Risk Services in the Netherlands who are able to perform a test on a ZigBee device. The small amount of trained people, makes it difficult to scale up in the future.

With the ZigBee workprogram however, any one of the 60 pentesters working in the team, will be able to conduct a pentest on a ZigBee enabled device.

The workprogram makes the process of conducting IoT pentesting more efficient and scalable, which saves the amount of time and hence money spend on a pentest.

7.4 Future work

There is still a lot of research to be done on the topic of penetration testing of the IoT devices and the ZigBee workprogram is only the first step in the standardization of the IoT penetration testing process. In the future, the workprogram should be expanded with other attacks from the KillerBee framework, which were not included in this research due to time constrains. Additionally, attacks from SecBee and Z3sec, which are ZigBee testing frameworks based on KillerBee, could be added to the workprogram.

Furthermore, there are various software and hardware tools available for testing the ZigBee protocol and only a few were used during this research. As part of the future work, the ZigBee WP could be enhanced with these tools and the information on how to obtain, configure and use them.

In this research, we looked into the smart light bulb system and assessed the bridge during hardware hacking. As part of the future research, the hardware of the light bulb could also be tested and checked for sensitive information.
Looking further in the future, the IoT pentesting process could be improved by creating alternative IoT workprograms containing information about other IoT protocols, such as LoRaWAN, Bluetooth, Z-Wave or NFC. Determining in which order protocols should be investigated, will happen in the consultation with the pentesters and based on the client requests. Another step could be the automatization of penetration tests, by creating scripts to automatically perform tests based on KillerBee commands. Additional enhancement could be automatization of report writing by sending the output of a test to an official report format. This would make IoT pentesting more scalable and manageable during large client engagements.
8 Bibliography


Li, H., Jia, Z., & Xue, X. (2010). Application and analysis of ZigBee security services specification. NSWCTC 2010 - The 2nd International Conference on Networks Security, Wireless Communications and
The workprogram for penetration testing of ZigBee enabled devices | Report

Trusted Computing (Vol. 2). https://doi.org/10.1109/NSWCTC.2010.261


52


Texas Instruments Inc. (2010). A USB-Enabled System-On-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications.


ZigBee Alliance. (2016). *Securing the Wireless IoT*.


Appendix A: Final version of the ZigBee WP

Disclaimer: Company confidential information (internal websites, documents and emails) was removed from the workprogram.

Workprogram for testing ZigBee enabled devices

WP author: Dominika Rusek, email address xxx
WP owner: Dominika Rusek, email address xxx
If you have any comments or requests for addition, send them to the author.

Version control:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>January 2018</td>
<td>Final version of the workprogram</td>
</tr>
</tbody>
</table>
Prerequisites

1. Prepare hardware

List of hardware:

- 2 x RZ RAVEN USB sticks flashed with KillerBee firmware
- CC2531 dongle for sniffing traffic
- Multimeter
- Saleae Logic Analyzer

You can get the hardware from the asset shop xxx.

Tip: If asset shop doesn’t have the hardware you need, ask xxx: email address xxx.

2. Prepare software

During testing, use 2x RZUSB sticks, one for sniffing the traffic (documentation) and one for attacking the ZigBee device. You can either use 2x Linux ISO on two machines with one RZUSB each or use 1x Linux ISO and specify the COM port used for every KillerBee command.

- Download ZigBee Windows ISO and ZigBee Linux ISO from the network drive, path: xxx.
  Windows ISO contains Texas Instruments SmartRF Packet Sniffer and Peryton ZigBee analyzer. Linux ISO has KillerBee installed.
- Alternatively, install software on your machine:
  On Linux, KillerBee: https://github.com/riverloopsec/killerbee

3. Testing location

To avoid confusion, perform testing in no noise location, such as a Faraday cage or self-made version of a Faraday cage (plastic box & aluminium foil). Alternatively use safe noise location, where you know which traffic comes from which device.
Project Name: <insert name of the assignment>
Scope: <number and type of devices>
Executed by: <insert name of pen-testers executing the test>
Timeframe: <insert timeframe when the test was executed>

## 0. INITIAL STEPS

<table>
<thead>
<tr>
<th>Test synopsis</th>
<th>Results</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 Preparation</td>
<td>&lt;Activities performed&gt;</td>
<td>&lt;Observations&gt;</td>
</tr>
</tbody>
</table>

### Objective

1. Verify with Sharepoint that the latest version of the workpaper is used.
2. Verify if signed engagement letter is in the system xxx.
3. Complete scoping form with generic information: client contact person, planning.
4. Upload this workprogram to system xxx.

#### How

Internal website xxx.

### 0.2 Information gathering

#### Objective

1. Search publicly available information about the device and the vendor. Look for passwords for serial port, encryption keys, e.g. FILETYPE: PDF manual default passwords “[DEVICENAME&VERSION]”.

2. Verify ZigBee Application Profile of the device (security measures can differ per application profile).

3. Look for the device on ZigBee Alliance certified products website.

Refer to the glossary for explanation of the terms.

#### How

- Google Hacking
- News groups
- Pastebin
- ZigBee Alliance website

Look for your device on the list of certified products, open Compliance Document and check section Security (at the bottom of the document).


Report if you find passwords or encryption keys publicly available. Report if sensitive information is available. Report Application Profile of the device. Report which security measures are applied according to the Compliance Document.

### 1. ZIGBEE ANALYSIS

<table>
<thead>
<tr>
<th>Test synopsis</th>
<th>Results</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1 Identify channel</strong>&lt;br&gt;&lt;br&gt;&lt;strong&gt;Objective&lt;/strong&gt;&lt;br&gt;Identify channel on which ZigBee devices are communicating, which is needed for further testing.&lt;br&gt;&lt;br&gt;To install KillerBee check: <a href="https://github.com/riverloopsec/killerbee">https://github.com/riverloopsec/killerbee</a>&lt;br&gt;&lt;br&gt;&lt;strong&gt;How&lt;/strong&gt;&lt;br&gt;Power on the devices. Plug in RZUSB stick into USB slot of the machine. Open Terminal and run <code>zbstumbler</code> command.&lt;br&gt;&lt;br&gt;The command will display information about the devices present on the network.&lt;br&gt;&lt;br&gt;Remember to note down:&lt;br&gt;PANID, Ext PANID, Channel, Source, ZigBee version</td>
<td>&lt;Activities performed&gt;</td>
<td>&lt;Observations&gt;</td>
</tr>
<tr>
<td><strong>1.2 Sniff network traffic</strong>&lt;br&gt;&lt;br&gt;&lt;strong&gt;Objective&lt;/strong&gt;&lt;br&gt;Use ZigBee sniffer to sniff ZigBee communication during pairing for total of 15 minutes. In order to sniff pairing and obtain information, sniff from the clean state – reset devices to default factory settings (press reset button on the device).&lt;br&gt;&lt;br&gt;&lt;strong&gt;How&lt;/strong&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1. Plug in RZUSB stick into USB slot of Linux machine. Run `zbdump` command, to capture pairing and save the output to `.pcap` file. Example: `zbdump –c 11 –w test.pcap`

   Alternatively, use CC2531 dongle with Texas Instrument SmartRF packet Sniffer or Perytons software.

2. Reset ZigBee devices to their factory state.

3. After 15 minutes stop the capture and open `.pcap` file in Wireshark. Example: `wireshark test.pcap`

4. Identify information from the test.pcap file such as PANID and verify with the output from step 1.1 whether correct traffic is captured.

   **Note:** Remember to store the evidence in system xxx.

1.3 **Verify if communication is encrypted**

   **Objective**

   Check whether devices use encryption or not.

   **How**

   Open test.pcap obtained in step 1.2 in Wireshark or tshark.

   Encryption is enabled if:

   ZigBee Application Support Layer Command; Security: True

   ZigBee Security Header
   - Expert Info: Encrypted Payload
   - Security Control Field: Network Key, Key-Transport Key, Extended Nonce

   Report if encryption is not enabled.

   **Note:** Remember to store the evidence in system xxx.

1.4 **Key sniffing**

   **Objective**

   Find the key during first pairing of ZigBee devices.
## How

1. Open test.pcap obtained in step 1.2 in Wireshark. After Association handshake, an APS:Command will be send to ZigBee device, which contains the Network Key (NK) encrypted with the Master Key, also called Default Link Key or in Wireshark called Key-Transport Key (KTK).

ZigBee uses AES 128 bit encryption. For Home Automation profile, Default Link Key is hex encoded of string “ZigbeeAlliance09”. It can be used to decrypt random Network Key to obtain plaintext ZigBee Network Key. For other Application Profiles, try finding Default Link Keys online.

2. Alternatively, run `zbdsniff test.pcap`. It will attempt to decode plaintext key from capture file.

Report if:
- KTK is default or easy to guess.
- NK is easy to guess.
- You are able to obtain NK by decrypting it with Master Key.

If the key is not found, move on to: **Step 4: Hardware analysis** or **Step 5: Firmware analysis** to verify if the key can be obtained by means of hardware or firmware hacking.

### 1.5 Reusing keys (optional)

**Objective**

In case multiple devices are tested, verify if key obtained from key sniffing on one device is the same as key obtained on another device.

**How**

Compare hex encoding of obtained cleartext keys. If DeviceA key is the same as DeviceB key, it is a security finding.

Report if:
- KTK = KTK
- NK = NK

### 1.6 Replay attack

**Objective**
Check whether it is possible to perform replay attack on the device.

**How**

1. Run `zbdump` document the attack e.g. `zbdump -c 11 -w test.pcap`
2. Send command to the device e.g. turn light bulb off and on.
3. Stop `zbdump`.
4. Run `zbreplay` e.g. `zbreplay -c 25 -r test.pcap`

Report if the attack is successful and message is transmitted. You can verify it in `.pcap` capture.

### 2. FINALIZE TESTING

<table>
<thead>
<tr>
<th>Test synopsis</th>
<th>Results</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>2.1 Save evidence</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Objective:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save output of automated tools.</td>
<td>&lt;Activities performed&gt;</td>
<td>&lt;Observations&gt;</td>
</tr>
<tr>
<td>Save Wireshark logs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Save screenshots from testing.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3. REPORTING

<table>
<thead>
<tr>
<th>Test synopsis</th>
<th>Results</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3.1 Create the report</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Objective:</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>List observations. Describe meaning, potential impact, and likelihood of each observation.</td>
<td>&lt;Activities performed&gt;</td>
<td>&lt;Observations&gt;</td>
</tr>
</tbody>
</table>

### 4. HARDWARE ANALYSIS (optional)

<table>
<thead>
<tr>
<th>Test synopsis</th>
<th>Results</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>4.1 Understand architecture of the device</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Objective</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | | |
Obtain schematics of PCB. If there is limited time, request technical documentation during kick-off meeting.

How

Search for terms: teardown, datasheet, PCB.

For more information on hardware hacking, refer to:

Internal website XXX.

Report if you find passwords online for serial port.

<table>
<thead>
<tr>
<th>4.2 Identify sensitive components</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
</tr>
<tr>
<td>Find out where on the hardware, sensitive components are located.</td>
</tr>
<tr>
<td><strong>How</strong></td>
</tr>
<tr>
<td>Open up the device, take pictures for evidence, be sure to note the steps to assemble it.</td>
</tr>
<tr>
<td>Find out where microcontroller and memory are located, and where could the firmware be stored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.3 Identify interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objectives</strong></td>
</tr>
<tr>
<td>Try to find interfaces accessible by an end user, which can be potential entry points/attack vectors.</td>
</tr>
<tr>
<td><strong>How</strong></td>
</tr>
<tr>
<td>Look for UART or JTAG interfaces on the device.</td>
</tr>
<tr>
<td>Check if there is ZigBee or WiFi antenna.</td>
</tr>
<tr>
<td>Use a multimeter to verify if there is data flowing in the pins.</td>
</tr>
<tr>
<td>Note: Take pictures and notes of your findings.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>4.4 Try to connect to identified interfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objective</strong></td>
</tr>
<tr>
<td>Get sensitive information from identified interfaces. Look for UART or other services.</td>
</tr>
<tr>
<td><strong>How</strong></td>
</tr>
</tbody>
</table>


Connect to identified ports to access serial port, debug information, shell.
Report if there are default or easy to use passwords.

### 4.5 Identify sensitive data in memory

**Objective**
Try to get access to the memory of the device.
Try to find encryption keys in the memory of the device.

**How**
Acquire volatile memory
Example: `dd if=/dev/kmem of=/root/kmem`
Use volatility framework to search through the memory or strings `| grep`.
More advanced version, dump memory directly, by taking the chip out and using EPROM to read the memory. Be careful with unsoldering and soldering the device, as it can destroy the hardware. Refer to hardware hacking on KB or hardware workprogram for more information.
Report if you can extract sensitive information from the memory, like passwords, encryption keys.

---

### 5. Firmware Analysis (Optional)

<table>
<thead>
<tr>
<th>Test synopsis</th>
<th>Results</th>
<th>Evidence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5.1 Obtain firmware</strong></td>
<td>&lt;Activities performed&gt;</td>
<td>&lt;Observations&gt;</td>
</tr>
</tbody>
</table>

**Objective**
Sensitive data can be found in the firmware, such as encryption keys or passwords.

**How**
1. Request firmware during kick-off meeting.
2. If step 1 is not possible, search for firmware images that are available to download from the Internet. Look at manufacturer site. Check for file extensions such as: .bin, .fw, .chk, .stk, .sys, .rar, .pkg, .rmt, .zip or .tar.
3. If step 2 doesn’t work, try to extract firmware from the hardware from debugging port or from ZigBee SoC (System on a chip). Refer to Workprogram Hardware Hacking for more information. Alternatively, put the device in update mode (look for available updates for the device and request them), sniff network traffic and extract the image.

5.2 Perform forensic memory analysis

Objective
Access sensitive data which is the memory, such as encryption keys, passwords, configuration files.

How
Use scalpel, foremost, binwalk –e image, JFFS2 or strings | grep
For detailed information consult with Workprogram Firmware Reversing:
Internal website xxx.
Report if you find sensitive information in the firmware.

Glossary

Ext PANID Extended PANID, 64-bit value set
JTAG Joint Test Action Group
KTK Key-Transport Key, also called Master Key or Default Link Key
NK Network Key
PAN Personal Area Network
PANID Personal Area Network Identifier
RZUSB Atmel RZ Raven USB Stick
SoC System on a chip
UART Universal Asynchronous Receiver/Transmitter
<table>
<thead>
<tr>
<th>ZHA</th>
<th>ZigBee Home Automation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZLL</td>
<td>ZigBee Light Link</td>
</tr>
</tbody>
</table>
Appendix B: Interview questions

1. How does the process of generic pentest look like?
   Please describe actions taken.
2. Is there any documentation/guidance/checklist available for the pentesters?
   If yes, please elaborate.
3. How does the IoT pentesting look like at the moment?
4. What would you improve in the process of IoT pentesting?