UNIVERSITY OF TURKU

A methodology for assessing cyber security in Zigbee-based IoT

Cyber Security

Master's Degree Programme in Information and Communication Technology

Department of Computing, Faculty of Technology

Master of Science in Technology Thesis

Author:

Taneli Vuori

Supervisors:

Saku Lindroos

Seppo Virtanen

June 2024

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Cyber Security
Programme: Master's Degree Programme in Information and Communication Technology
Author: Taneli Vuori
Title: A methodology for assessing cyber security in Zigbee-based IoT
Number of pages: 60 pages
Date: June 2024

More and more devices are connected to the internet every year. The different communication protocols involved in this transition are numerous, and the rapidly evolving internet landscape forces these protocols to evolve with it or fade into obscurity.

Because of this rapid evolution and the exponential growth of internet-connected devices, the cyber security features of these devices must also evolve in tandem. In this thesis a Zigbee scanning system was created for passively scanning wireless Zigbee communications via the wardriving method to gain an in-depth look into the security features of Zigbee devices from real world examples.

This thesis was conducted in two parts, the first part being the theoretical background into the subject and the second part as empirical research. A system for passively scanning Zigbee devices was built and tested, and the results were analyzed.

Two of the wardriving test results were dissected and analyzed in this thesis, where the latter test drive focused almost solely on the cyber security features of the scanned communications. The results were not particularly alarming, but a few deficiencies were detected.

Both legislative and informative measures can and should be taken to combat the issues related to cyber security on the Internet of Things.

Keywords: Cyber security, IoT, wardriving, Zigbee

Table of contents

1 Introduction

Internet of Things (IoT) can be explained in multitude of ways as the concept is quite multifaceted. A way to describe is that it is a wide system in where interconnected physical devices collect, exchange and process data to enable better decision-making by both humans and machines alike [1]. Terms like “Smart Infrastructure” and “Industry 4.0” are a central part of IoT and are concepts that are enabled by IoT. Smart infrastructures are entities that use the data gathered by the IoT devices in order to make higher quality decisions, and Industry 4.0 is a term coined to represent the fourth industrial revolution in the last few decades as the digital transformation of the business world [2]. As a result of this booming industry new challenges in terms of cyber security are growing.

The amount of IoT devices is continuously on the rise. As of Q4 in 2021 there were approximately 12.2 billion IoT devices when counting active nodes, devices, or gateways that concentrate the end-sensors [3]. The number increased 14.3 billion in 2022, and 16.7 billion in 2023 [3]. Most used connectivity type in IoT devices is Wireless Personal Area Network (WPAN), with Wireless Local Area Network (WLAN) shortly behind. WPAN includes Bluetooth, Z-Wave, ZigBee, and others. WLAN includes the all-familiar Wi-Fi and its different versions. Other connectivity types include “legacy” cellular (2G/3G/4G), Low-Power Wide-Area (LPWA) network, wired IoT, such as Ethernet, and as a newcomer the 5G IoT. The number of devices connected is estimated to be near 30 billion by the year 2027 [3].

As mentioned, one of the challenges regarding this fast growth is the cyber security of the IoT devices. The nature of the devices themselves offer some technical challenges in implementing security measures. Limited power, low computational capabilities, and small storage make it hard to implement advanced security solutions, and companies trying to answer the overall demand for these devices may consider overlooking security features for different reasons and therefore design products that are vulnerable [4]. Legislation regarding this subject is also dragging behind which is not encouraging the manufacturers to change the way they build and produce IoT devices. The lack of IoT access controls by users and poor programming practices in unison with the lack of software updates paint a rather unnerving picture about the state of IoT security. Although efforts have been made to “tighten the screw”, so to speak, with an EU-wide Cyber Resilience Act [5], the devices that are already in use will most likely not be affected by any regulations that come into effect. Therefore, it would be beneficial for the field of cyber security to know more about the devices that are already in use, so that it is possible to understand the current situation and the potential problems arising in order to understand what steps to take in the future.

One of the technologies that is a part of the IoT family is the Zigbee communication protocol. It is a low-power and low bandwidth technology standardized in 2003. The protocol standard is upkept by the

Connectivity Standards Alliance (CSA), and currently there are over a billion chipsets sold that use this communication protocol [6]. Zigbee is part of the Institute of Electrical and Electronics Engineers (IEEE) 802.15.4 standard that defines the operations in low-data-rate wireless connections that have either no battery or very limited battery consumption requirements [7]. The 802.15.4 standard also introduces the physical (PHY) layer and Medium Access Control (MAC) sublayer specifications for Zigbee [7].

In this thesis a solution for passively scanning Zigbee communications via the wardriving method is built and tested and the results are analyzed. The specific wireless network scanning methodology in question that inspired this thesis is the one presented by Lindroos et al. in "A systematic methodology for continuous WLAN abundance and security analysis" [8]. The existing scanning methodology has already been in use for a few years, and analysis has been done from the data acquired regarding the effects of Covid-19 pandemic to WLAN security, as well as frequency band and channel distribution in WLAN networks [9], [10]. As the current system designed by Lindroos is very functional, although limited to scanning only WLAN networks, the idea is to increase the efficiency of the scanning methodology via passively scanning multiple communication protocols simultaneously. Zigbee was chosen as it is one of the bigger communication protocols used in IoT.

This thesis is constructed as follows: The first chapter is this introduction. The second chapter is the theoretical framework for this thesis, and the concepts talked about later on will be explained and discussed there. The third chapter will consider wardriving in general, as well as the existing wardriving system that will be built upon. The fourth chapter will consist of designing and building the Zigbee scanning system. The fifth chapter consists of testing and analysis of results produced by the Zigbee scanning system, and the sixth chapter will be a conclusive chapter.

2 Theoretical framework

2.1 Definition of IoT

The term IoT was first coined by Kevin Ashton in 1999 to demonstrate the potential of connecting Radio Frequency Identification Tags (RFIDs) to the internet [11]. The term has since evolved, and many different entities have proposed their own meanings to it, but the general idea behind is always somewhat similar. However, there does not seem to exist a singular correct answer to the question "what is IoT?", so rather than to find the "correct" answer, some definitions by well-known entities are introduced.

Internet Engineering Task Force (IETF) defines IoT on their website as follows: "*The Internet of Things (IoT) is the network of physical objects or "things" embedded with electronics, software, sensors, actuators, and connectivity to enable objects to exchange data with the manufacturer, operator, and/or other connected devices*." [12]. Their definition of IoT is overarching, covering every physical object that has the capacity to connect with other devices and share data. This is an understandable definition, as IoT devices are part of the internet, and thus communicate with other devices. This definition focuses on viewing IoT as a network rather than as singular devices.

Internet Architecture Board (IAB) defines IoT as follows: "- - *large number of embedded devices employ communication services offered by Internet protocols*." [13]. They call these devices "smart objects" and consider that most of these objects work independently in different environments, following the theme of connecting every "thing". This definition is not as extensive but has the same idea as the one by IETF.

Finally, the third definition presented here is the definition by Oxford dictionary, which tracks the development of the English language and keeps a record of words, terms, abbreviations, and their use-cases. The dictionary defines IoT as follows: "*The connection of devices within everyday objects via the internet, enabling them to share data.*" [14]. This is a simple way to put it to a single sentence and is a very approachable and easy-to-understand, although it could be argued that a lot of the devices that are part of IoT are not "everyday objects".

2.2 IoT security landscape

Industry 4.0 has blurred the lines between physical objects and the virtual world. The ubiquity of IoT devices can be seen everywhere, and with this increasingly rapid rise in the amount of IoT devices new cyber security concerns emerge. All of the IoT devices have some sort of purpose, and quite a many of them are made to make life easier for humans, to automate mundane tasks or to improve the quality of life. The idea behind the devices is noble, but the flipside of the coin is that the same devices that improve

our quality of life can be used in malicious activities. The devices themselves might be working as expected, but a malicious actor with knowledge and correct toolset may be able to use these devices for malicious purposes.

These considerations for this type of nefarious activity are not only theoretical. An example of this is the Mirai malware. Mirai malware is used to scan networks for IoT devices that run on a specific type of Linux [15]. The malware tests the default username and password combination on the devices and if it gets access, it can use the device in, for example, Distributed Denial of Service (DDoS) attacks, in which multiple devices try to overload a targeted system with network traffic in order to render it unusable. Collectives of devices infected by this malware are called “botnets”, and the collective computing power of these devices can be utilized in order to launch attacks, such as the famous 2016 attack on the Domain Name System (DNS) provider Dyn, which rendered their services unusable for multiple hours, and thus made some internet platforms unavailable [15].

IoT malware is a major concern. According to European Network and Information Security Agency’s (ENISA) 2022 threat landscape [16], the attack volume towards IoT in the first six months of 2022 is more than the previous four years combined before that. The above-mentioned Mirai malware continues to be responsible for more than seven million attacks in the first few months of 2022, with Mozi, another botnet, behind, with five million attacks. Mobile networks and IoT devices are increasingly both the target and facilitator for DDoS attacks [16]. Weak or nonexistent security measures and weak configurations in IoT devices, such as default or weak passwords, make them easy to corrupt. Some of the responsibility for this is also on users, as a lot of the compromised devices are a result of not updating the IoT devices. In this instance, the Mozi botnet still uses vulnerabilities discovered almost a decade ago to great success, although some legacy systems simply do not have continued support [16]. The lack of cyber security skills of an average user make these problems even more prevalent.

As another example the IoT devices can be used as E-mail spammers. Phishing is the most common vector for gaining initial access to a device or a service by a malicious actor in 2022 as per ENISA’s 2022 threat landscape [16]. Phishing attacks are most commonly made by targeting a large audience and sending them a large number of messages, mostly E-mail. These attacks can be made via SMS as well, or any other messaging platform. IoT devices are ideal for sending a large amount of Simple Mail Transfer Protocol (SMTP) messages as the devices can access the port number 25 which is a Transmission Control Protocol (TCP) port used to transfer SMTP messages. With a large number of IoT devices, a large number of malicious messages can be sent.

ENISA’s 2023 threat landscape [17] shows a similar story. According to this threat landscape, DDoS attacks are largely moving towards IoT and mobile networks. In the case of IoT, the limited hardware

resources in the devices often results in poor security measures. The increasing complexity in both the mobile systems and IoT landscape also has had an effect regarding the users' security skills. Fast development in this industry has made it hard for consumers to adapt to the more complex ecosystems, which can lead to user ineptitude playing a factor in the security landscape. A few new botnets have also emerged that are mentioned in the 2023 threat landscape, namely Zerobot and MCCrash, that utilize known vulnerabilities in order to get control of devices and utilize the devices further in, for example, DDoS attacks.

The Microsoft Digital Defense Report 2023 [18] paints a somewhat similar picture. The report separates IoT and Operational Technology (OT) security from one another. In short, OT security involves safeguarding systems controlling physical processes in, for example, industrial and critical infrastructure, whereas IoT security focuses on devices in larger networks and is mainly concerned with the large attack surface and integrity and confidentiality of data. For the purposes of this thesis the term OT is used as a subsidiary term for IoT. According to the Microsoft report [18], 78% of devices in industrial control networks have known Common Vulnerabilities and Exposures (CVE). Some of the devices have been known to use unsupported operating systems, such as Windows 2000. Approximately 25% of the devices are using unsupported operating systems that are no longer getting security patches. Furthermore, if the IoT devices' Programmable Logic Controllers (PLC) were installed with the latest firmware, the percentage of devices with no known CVE's would increase from 4% to 40%. Security patches therefore are not being done at regular intervals, and although there are legitimate reasons for not updating to the latest firmware, the positive effects of doing so could outweigh the negatives. The final relevant percentage taken from the report is 57%. This percentage represents the percentage of devices that use legacy firmware versions, that have had available updated firmware for more than 10 years that would reduce the exploitable CVE's greatly [18]. Therefore, this is not only an issue of patch intervals, but a tell from more of a systematic issue.

Concerns have also been raised in regard to unsafe IoT devices and the impact they will have on the digital world in the long term. Some, such as Hyppönen, who is a Finnish information security expert and author, compare the modern IoT landscape to that of 1960s and 1970s asbestos usage in an interview [19]. Asbestos was an extremely prominent material used in construction business and was used in many ways and many places, and the consequences of that still linger now, over 50 years later. The comparison here is that the IoT devices do not have an expiration date, and they can be used for as long as the machine still continues to work, and as long as it continues to work it can still be connected to the internet. The effects these unsafe legacy IoT devices will have in the long term are still not very well understood, and the hope here is that most of these low quality IoT devices will have a short lifespan. In any case, the current billions of active IoT devices might prove to be more problematic and the effects might linger longer than anticipated.

A few IoT specific challenges arise when implementing security measures in IoT devices. The first challenge is the computational power limitations of IoT devices [20]. IoT devices generally have as low computational power capability as possible for the task they are made for in order to mass produce them at reasonable costs. IoT devices also have diminished memory, storage, networking, and energy capabilities [20], which results in these devices being harder to secure via heavier and costlier methods that require more capacity in the overhead. As the devices need to operate mostly somewhat autonomously without human intervention, the energy consumption must be low so that there is either no need to swap batteries or it can be done very infrequently, which means the operations of the devices will be kept to the minimum in order to save energy. Similar resource saving can be seen in the memory, storage, and processors of the devices. Often the devices can be quite small, which means there is no space for a bigger battery or for extra memory or storage capabilities, nor the desire or need for them. These resource limitations mean that the devices might not be able to use traditional security measures, such as heavy-duty encryption or signing, for example. The second challenge is the scalability and availability [20]. IoT devices have the possibility to be connected to a vast number of other devices, and thus be exposed to more threats. The availability of the IoT devices at all times also make them extremely reliable targets for attackers, and the large amount of devices connected to the internet make it so that there is no shortage of targets. Concerns regarding weak default credentials can also be observed in a lot of IoT devices.

The consequences of IoT security negligence are sometimes not solely contained in the digital world. As IoT devices include sensors that, for example, monitor pressure or heat in critical areas of infrastructure, disabling these functionalities via vulnerabilities or other deficiencies in the devices can potentially lead to disastrous results. This is especially true for IoT devices, as the devices themselves are usually connected to the internet close to 24/7, meaning they make an extremely reliable target for malicious actors.

Many IoT devices indeed seem to be made for a certain purpose, and security only seems like an afterthought. Users mostly want easy-to-use, accessible products that have a fast initial setup and be pleasing to the eye, which means manufacturers prefer to focus on these aspects even if it contradicts the security features of the devices. The IoT industry is a big industry, and tech companies need to keep up to date with the competition in order to sell their products. This can quite often lead to trying to make affordable solutions, so the product they are manufacturing can compete with others cost-wise, and if the focus is completely on the user experience, not a lot is left in the budget for security measures. Cheap, easy-to-use, and accessible usually ends up winning when customer is looking for a product on the shelf.

Legislative measures have been taken already to combat the IoT landscape in Europe. The EU-wide regulation called the “Cybersecurity Act” [21] introduced in June of 2019 provides a common framework for European cyber security certificates. These certifications lay rules, standards, technical requirements, and procedures on Information and Communication Technology (ICT) products, ICT processes, ICT services, and devices. ENISA is the driving force behind the cyber security act and also acts as the enforcer of the act. These certifications are currently voluntary, so vendors and providers can themselves decide whether their products get certified. Speculatively speaking, this voluntary state can possibly turn into a mandatory one. The certification is recognized in all the EU states, so this gives vendors and providers monetary reasons to certify their products, assuming the certification is understood and appreciated by the end customer. These certification schemes can be used to issue certificates to member states, to have an ICT product certified, or to use and recognize certificates as a demonstration of the products’ trustworthiness [22]. Although regulations are not technically laws, they are considered to be extensions of the law and the will of the authorizing body on what they think is appropriate behavior, and the authorizing body in this case is ENISA. As this act is relatively new, the actual implementation level currently seems to be quite low, but from the point-of-view of IoT cyber security it should have effects on the security landscape in the long run, especially if some of the certificates produced by this framework are implemented as mandatory.

A more specifically targeted regulation, called the “Cyber Resilience Act” [23], is a 2019 proposal to strengthen the cyber security of hardware and software products with horizontal cyber security requirements. According to the European commission, the global cost for cybercrime related attacks on software and hardware was 5.5€ trillion by the year 2021 [24]. The major reasons for this are the insufficient cyber security measures in hardware and software products, and the negligence and inconsistency of the product manufacturers’ updates on these products, as well as the insufficient understanding of such products by users, preventing them for making educated choices in order to use the products in a safe manner. Major points of the Cyber Resilience Act are as follows: Manufacturers must ensure security for the products whole lifecycle; to ensure that both hardware and software products placed in the market have fewer vulnerabilities; and to create conditions for users to make educated choices when using products with digital elements [24]. The act is supposed to also enhance transparency and to provide a consistent cyber security framework, which is used to deploy compliant products to the market [24]. According to the impact assessment of the Cyber Resilience Act [25], an approach of horizontal regulatory intervention would be most impactful, but also the most resource intensive. However, the impact assessment of this horizontal approach indicates that it would heavily reduce the amount of monetary loss caused by weak cyber security measures currently in software and hardware for businesses. According to the same impact assessment [25], the end devices would be more expensive, but that would have to be the trade-off. Although the act covers all software and hardware, the major focus point is IoT devices.

On a national level the EU Cybersecurity Act has prompted the Finnish National Cyber Security Centre, NCSC-FI, to release the "Cybersecurity label", which grants a manufacturer a certified label on their product if the product matches or exceeds the necessary qualifications. The aim for this label is to inform customers, to guarantee the security of the products, and support competitiveness among manufactures in a manner that promotes cyber security. Although applying for this label is currently voluntary, it pushes the IoT landscape in Finland towards a more secure future. On the cyber security label website [26] there is a list of devices with cyber security labels. Currently there seems to be 23 devices that have the label, heavily concentrated on two manufacturers, one that has the label for some of their smartwatches, and another that has the label for some of their routers that are designed for end users. The earliest cyber security label is from late 2020, which means slightly over three years of operations by NSCS-FI in this regard [26]. The label seems to be still quite unpopular, and it remains to be seen whether the label has or will have an effect on customers' behavior or will applying for the cyber security label be more hassle than it is worth. It also seems from the devices that have the cyber security label that it also needs to be renewed from time to time. In the case of the routers that have the label, the renew period seems to be three years as per a "statement of compliance" report from the cyber security label website [26].

Another noteworthy project regarding IoT safety comes from IETF. To reduce the attack surface of IoT devices, IETF developed the Manufacturer Usage Description (MUD) as specified in RFC 8520 [27]. The purpose of MUD is to create access control policies so that the IoT devices only communicate with network objects they are meant to communicate with. An example of this is a light bulb that can be remotely controlled via an application on a smartphone through the network. The MUD then defines that the light bulb may only communicate with that application and nothing else. This way the device gets an extra layer of protection. MUD consists of three blocks: a Uniform Resource Locator (URL) to locate the description, the description itself and how the description is supposed to be interpreted, and the means for the local network management systems to attain the description [20]. The MUD URL is sent from the device to the router or switch of the network, which passes the URL to a MUD manager, which in turn downloads the MUD file from the manufacturer's MUD file server [20]. The MUD specification does require the manufacturer of the IoT device to comply with the specification. The switches and routers in the network also need to implement the MUD protocol, and at the very least one MUD manager service has to be operational [20]. References to the use-rate of MUD seems to be hard to find, so determining how widespread the usage of MUD is difficult. However, entities like Cisco and National Institute of Standards and Technology (NIST) have seemingly adopted the usage of MUD and the protocol seems to be developed quite regularly [28]. Although the MUD protocol needs to be implemented by the IoT devices and supported by the other network elements, the computational resources needed for this are not that high, as the protocol is mainly concerned with creating access lists for the IoT devices. This means that the costs for implementing the protocol are not that high, although

a dedicated MUD file server needs to be hosted by the manufacturer. Although not perhaps a comprehensive security feature, MUD definitely seems to serve as a good addition to the security landscape of IoT. More options for manufacturers to choose from certainly improves the odds of manufacturers implementing some, or any security features to their products.

Other efforts are being made by entities such as NIST and Global System for Mobile Association (GSMA) to combat IoT security deficiencies. NIST offers the "Cybersecurity for IoT" program [29] that provides guidance and education in IoT via their publications and events. The publications include IoT security baselines, privacy in IoT, the future of IoT, and others. The latest publications are from late 2024. GSMA has the IoT security guidelines and assessment documents that give detailed recommendations for designing, developing, and deploying IoT devices and services. The GSMA documents date to 2020, although GSMA mentions on their documentation website that an update is imminent due to interest from their members and the rapidly evolving 5G world [30].

One of the driving factors behind this thesis is the worrying state of cyber security in IoT devices. Currently there are driving forces trying to make devices with these IoT characteristics a bit better and safer. However, there are already billions of devices out there in use, and rather than ignoring the possible problems these devices might cause it is better to know the potential problems and counteract if possible, or at the very least acknowledge that those problems exist and, in the future, make more informed decisions as both manufacturers and customers. The knowledge gathered from the start of the "IoT" time-period to this day should be used to improve the state of cyber security in these devices.

2.3 Scanning and IoT

2.3.1 Scanning IP networks

Network scanning traditionally is a procedure where a certain IP address range is sent network communications and the sending machine receives responses from the active devices in the targeted IP network. This is called active scanning, where the machine used to scan the network has access to the targeted network. This same scanning method can be used by malicious actors from either inside or outside of the network, depending on whether they have privileged access to the network. If the malicious actor has privileged access to the network, they can launch a scan from within the network. This scanning can be done via Local Area Network (LAN) or Wide Area Network (WAN), as the actor doing the scanning does not have to be physically close to the network as they only need access to the network either locally via, for example, ethernet cable or WLAN, or via the public internet. Quite often the targeted networks are private networks in a certain IP address range that are separated from the global internet via a firewall. The difference between authorized network scanning and unauthorized

network scanning is usually only who is doing it and the reasons for doing it, the actual act of scanning a network is usually done with similar tools.

There are multiple legitimate reasons for scanning a network and professionals in IT and cyber security field may use this option regularly in order to understand the network they are responsible for. The types of scans used both in authorized and unauthorized can be but are not limited to network scans, port scans, vulnerability scans, TCP scans, User Datagram Protocol (UDP) scans, Synchronized Sequence Number (SYN) scans and Internet Control Message Protocol (ICMP) scans [31].

Companies and other organizations can utilize scanning in their own private networks in order to understand what network elements they have and what they are responsible for. Depending on the network size, scanning can be beneficial in keeping track on which kind of devices are connected to the network and how many new devices are found since the previous scan. Keeping an up-to-date catalogue of the devices in a network makes it easier to spot anomalies in the network and potential threat factors. Keeping the results of the scans can be used to build a history of the networks devices, and therefore the administrators of the network can be more familiar with the fluidity of the network and the potential risk factors. Some networks will have more devices coming in and out, for example public networks in universities, so it is naturally harder to keep track of devices in such networks via scanning, although scanning can still be an excellent tool in the right hands.

Scanning is often used to gather information on outdated hardware and software in the network. As an example, scanning workplace computers that run a certain operating system, let us say Windows, would let the administrators responsible for keeping the devices up to date get an understanding on what Windows version and patch is running on each device, and then to act accordingly, for example force updates on older versions.

On the other hand, malicious actors can use scanning in order to gather information on the targeted network and to find open network service ports or other vulnerabilities. As an example, the abovementioned SYN scan can be used to determine open network ports. Certain network ports are more vulnerable when exposed to an attacker. Depending on the network port, an attacker could potentially gain access to the target network and expand the attack from just scanning to other malicious activities, such as further reconnaissance and consequently privilege escalation.

Defending the network from malicious scanning can be done via different measures. Firewall solutions can be used to create rulesets on what types of connections can be created to the network. Firewalls can be simultaneously bolstered by an Intrusion Detection System (IDS) or an Intrusion Prevention System (IPS), which are designed to detect malicious attempts to the network and in the case of IPS, prevent

them. The network administrators can also utilize the port scanning themselves and find vulnerable or unnecessarily open network ports themselves and shut them down.

2.3.2 Scanning radio frequencies in IoT

The IoT market is large, and it is still exponentially growing. The communication protocols used in IoT devices are developed in somewhat of an isolation from each other, which means that scanning just for "IoT devices" is not so simple and represents an overall challenge from network security perspective. Scanning IoT devices is challenging for a few reasons. First is the IoT fragmentation, which means that the competing communication protocols in IoT market cannot communicate with each other as the standards and platforms do not allow for it [32]. Secondly, traditional network scanning tools might not be useful for scanning IoT devices, as the lack of IP addressing in IoT devices renders some of the tools obsolete [32]. Thirdly, in many IoT devices some of the parameters in the PHY layer are hard coded on the network card, which means that devices that could communicate with them must use that network identifier [32].

The reasons for scanning IoT devices can be quite similar to scanning any other network device. Authorized scanning can be done in order to gather information on the devices, their firmware versions, and potential anomalies in the network that has IoT devices in it. Similarly unauthorized, or malicious, scanning can be used to find open ports and other vulnerabilities in the IoT devices. However, this thesis focuses on the wardriving method of wireless passive scanning in radio frequency networks, so we will discuss the differences between active and passive scanning first in the radio frequency network, and the considerations for Zigbee communication protocol scanning in particular.

IoT devices themselves use wireless communication protocols to communicate with other IoT devices, as most of the IoT devices are wireless. These protocols can transmit data over longer distances, such as LoRaWAN, or locally, such as Zigbee. Long range IoT typically does not communicate directly from one IoT device to another, but rather are connected to the internet and subsequently to an intermediary network, which can be used to communicate with other IoT devices from geographically far away. As we are interested in scanning Zigbee, which is a short-range communication protocol that is used only in local networks, the scanner itself needs to be in close proximity to the devices. Gathering information on short-range devices can be quite challenging, as the scanner obviously needs to be nearby. In order to scan short-range communications, including, for example Wi-Fi, Bluetooth, and Zigbee, the scanning entity needs a tool that can get into the same radio frequency as to what these protocols use and to have support for the communication protocol.

There are two ways to do short-range radio frequency scanning when it comes to the style of scanning: active and passive. Generally speaking, active scanning means that the device you are using to scan the communication protocol, for example Wi-Fi or Zigbee, sends a probe request to the access point, and listens for a response, hence initiating the contact in an active manner. In the case of Wi-Fi, the access point is the router, and in the case of Zigbee it is the controller or the router of that network. In a passive scan the device that is used to listen to these radio communication protocols listens for periodically sent transmissions from the devices. Depending on how often the devices sent messages it might take a bit longer to gather device information as opposed to active scanning. Passive scanning might also miss the transmissions sent as the scanner can often only be on a certain channel for a duration. However, as far as wardriving goes, actively scanning networks is definitely something a person should not attempt without taking extra care about lawfulness and consent. Passive scanning, for example Wi-Fi networks, is something most smartphones are capable of doing and are doing so continuously as long as the feature is enabled on the smartphone, as the nature of connecting to Wi-Fi access points requires pre-connection traffic between the smartphone and the access point. The information you can gather from passively scanning communications in Wi-Fi and Zigbee, as examples, is somewhat superficial, and not illegal to do, although this might differ from country to country. The contents of any of the messages delivered will never be visible unless sent unencrypted.

The scanning method in this thesis will be passive scanning of Zigbee transmissions via the wardriving method. There does not seem to exist a lot of studies or other literature on the topic of Zigbee scanning. However, in their work Gvozdenovic et al. [32] introduce a tool called “IoT scan” and utilize it for both passive and active scans of the Zigbee communication protocol. The main software components in their implementation are GNU Radio 3.8 and Scapy-radio 2.4.5, which are tools that can be used in penetration testing as well. According to one the results of their study, using active scanning in Zigbee generates superior results in discovery speed of the Zigbee devices, as finding up to twelve devices took less than a minute in active scanning, and roughly six minutes in passive scanning. The results also indicate that finding the short network address of Zigbee devices, which is the 16-bit short address, is easier than finding the longer, 64-bit, address.

As the literature around this topic is not very extensive, the work done in this thesis could further help to have better tools available for short-range scanning in the context of IoT. Although this thesis focuses on Zigbee, the fundamentals of scanning apply to other IoT protocols as well.

2.4 Zigbee communication protocol

Zigbee is one of many IoT network types. Other low-power, short-range networks include Bluetooth, NFC, Wi-Fi, and Z-Wave as the most used ones. For low-power, wide-area IoT networks there are

4G/5G IoT, Cat-0, Cat-1, LTE Cat-M1, narrowband, LoRaWAN, and Sigfox [33]. This thesis however will concentrate only on Zigbee communication protocol which is partially defined in the IEEE 802.15.4 standard [7].

The 802.15.4 standard is a standard for low-data-rate wireless networks, and it defines the PHY and MAC layers of the devices [7]. It focuses on low data rate, low cost, low power consumption with or without a battery, and low complexity of the devices and the communications between those devices. The standard considers the existence of IoT, meaning that they target the communication requirements for the current IoT landscape. The standard itself is published by IEEE, and it is developed by the LAN/MAN Standards Committee. The current standard was updated in 2020, and it is the fourth iteration of the standard.

The Zigbee Alliance was born in 2002, and the protocol was standardized in 2003. The reason the Zigbee Alliance was born was because there was a need for an organization that would define an open and global standard for cost-effective and low-power wireless network products [34]. At the time there were other standards, but they mostly focused on higher data rates and battery-powered networks for a lower number of devices [34]. The need for large networks that could operate independently over a long period of time without intervention existed, so the Alliance was created. The Zigbee Alliance rebranded to CSA in 2021, as they have broadened their area of expertise.

Zigbee communication protocol operates on two frequency ranges: 868/915 MHz and 2.4 GHz [35]. The 2.4 GHz band of Zigbee ranges from 2400 MHz to 2480 MHz and has 16 channels from 11 to 26 that are approximately 5 MHz apart from each other. The higher frequency is used worldwide, but the lower frequencies are specific to Europe, United States, and Australia [35]. The higher frequency band is capable of delivering higher data rates than the lower band. The 2.4 GHz is more widely used and is theoretically capable of having a maximum data rate of 250 kbps [36], but in reality, the data rate is probably quite a bit lower than that due to environmental interferences. The data rate for 868 MHz channel used in Europe can reach 100 kbps speed, and the 915 MHz used in Americas can reach 500 kbps speed [36].

The transmission range of Zigbee devices range from 10 to 100 meters depending on environmental factors and power output of the device [36]. This range can be extended by using intermediary networks to transmit the data the Zigbee devices are sending. This range is quite similar to other short-range networks but can prove to be challenging when scanning for the devices, especially in rural areas, because of the distance. Areas with high population density have their own challenges as well as the signal will have problems when trying to penetrate buildings. In such areas a lower range of transmission range is to be expected.

Also worth mentioning is the Zigbee Direct feature [37], which enables the users to control their Zigbee devices via Bluetooth devices, such as smartphones. This means that it is possible to directly, hence the name, control the Zigbee end devices via a Bluetooth device. The devices are separated into Zigbee Virtual Devices (ZVD) and Zigbee Direct Devices (ZDD). ZVDs are Bluetooth devices with the Zigbee direct feature, such as a smartphone, that play the role of a Zigbee device in the network, e.g. a trust center, coordinator, router, or end device. ZVDs are usually used to control the rest of the devices in the network, which means ZVDs usually take the role of a trust center. ZDDs are the devices in this Zigbee Direct environment that are running Zigbee and are capable of running Bluetooth stacks, that enable the data transfers between ZDDs and ZVDs. The communication works by connecting ZVD to a ZDD via Bluetooth, and this communication channel also carries the application messages in the Zigbee network to other Zigbee devices in the network through the ZDD, meaning that all of the devices in the Zigbee network do not need to have the Zigbee Direct feature in order to receive messages through Bluetooth, only one ZDD is needed to "translate" the messages to the network.

The Zigbee infrastructure has the following elements: the coordinator, which also acts as the trust center, router, and end device [38]. The coordinator is responsible for managing the overall network. Coordinator acts as the trust center and provides configurations and end-to-end security between the devices in the network, and also chooses which devices to let in the network [38]. Cyber security wise coordinator is the most central point of the network, as it also stores and distributes the network keys. Routers act as facilitators between the coordinator and the end devices. Routers are not necessary for Zigbee networks, but they help in scaling the network [38]. The end devices are often the characteristic Zigbee devices that have low battery usage and low computational powers, and are therefore energy-constrained, which means when the devices do not need to be used, they are usually idling [38]. These can be motion sensors, light bulbs, or other smart devices. The end devices themselves do not route any traffic, but instead communicate with their router or the coordinator. The end devices do not communicate between each other.

The three ways to build a Zigbee network topology are star network, tree network, and mesh network [39]. Other network topology types are not supported, because the network runs on the singular coordinator, and there is no more than one coordinator per network. Also, as mentioned above, the end devices do not send communications to each other, as they are not designed to route traffic. In the star topology the end devices are directly connected to the coordinator and are called the children of the coordinator. This is a simple and cheap way of building a network, as there are no alternative routes for the end devices to reach the coordinator.

Tree topology consists of the coordinator, routers that extend the network, and the end devices [39]. In this topology the end devices that are connected to a router or directly to the coordinator are called

children, and the router and/or coordinator is called the parent. Tree topologies use hierarchical routing strategies. Tree topology allows for a larger network, but if one parent in the network is disabled, the children are too.

Mesh topology can be used to build a more resilient network, as there are usually multiple routes for the end devices to reach the coordinator, which is necessary for the network to function [39]. This topology also allows for easier expansion of the network. Mesh network allows a full peer-to-peer communication, excluding end device to end device communication. The three different topologies are presented in Figure 1.

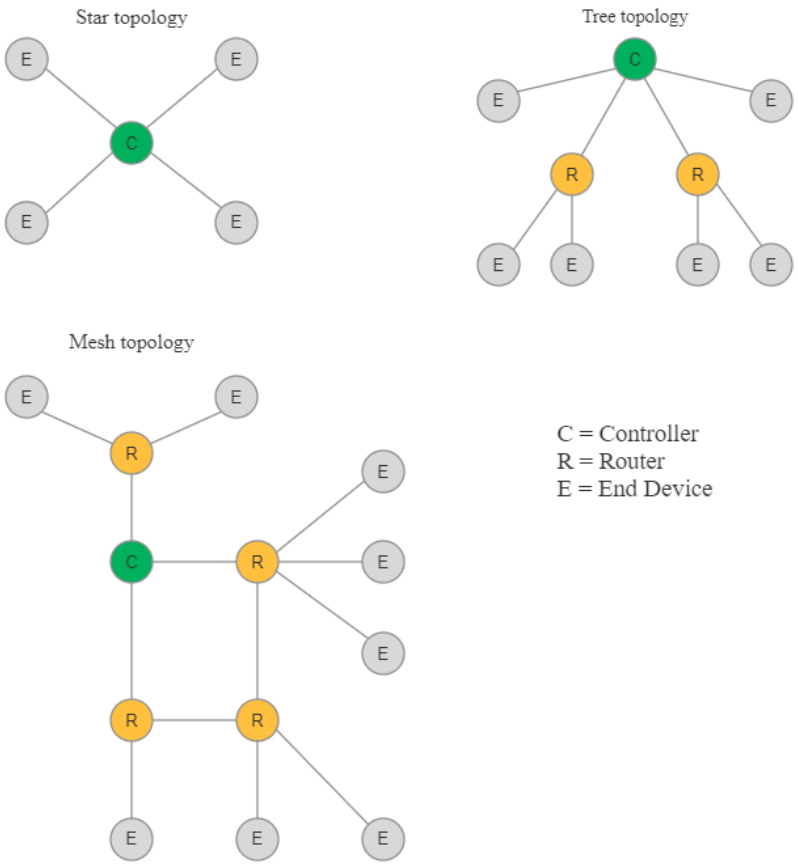

Figure 1. Star, tree, and mesh Zigbee network topologies.

Zigbee architectural stack (Figure 2) consists of the following: The PHY and MAC layer, that are specified by the 802.15.4 standard, the Network layer (NWK), the Security Service Provider layer (SSP), the Application Support Sublayer (APS), the Zigbee Device Object (ZDO), and the Application Framework (APF). ZDO, APS, and application framework are inside the Application layer (APL).

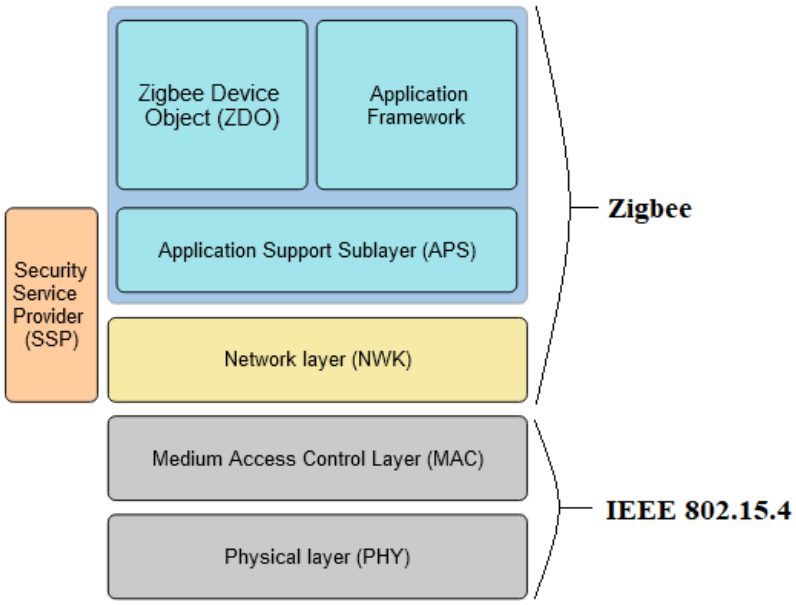

Figure 2. Zigbee architectural stack.

From bottom to top, the PHY layer is defined by the 802.15.4 standard. The PHY layer provides an interface between the physical radio channel and the MAC layer via radio frequency hardware and firmware [7]. PHY layer is responsible for activation and deactivation of the radio transceiver, link quality indication, channel selection, energy detection, as well as transmitting packets via the physical medium. Basically, the job of PHY layer is to translate bytes to radio frequencies and back to bytes again. For a more detailed description of the PHY layer can be found in the standard itself [7].

MAC layer is similarly defined in the 802.15.4 standard [7]. This layer provides the MAC data service and the MAC management service. MAC layer provides a space for appropriate security mechanisms, as well as is responsible for frame validation, channel access, and beacon management. The main purpose is to avoid collisions while transmitting frames as well as to provide an abstraction of the PHY layer.

NWK layer is responsible for finding networks and joining them [34, p. 43]. It is the first Zigbee specified layer and it routes the different Zigbee network topologies, namely mesh, star, and tree topologies. More importantly, the NWK layer is responsible for providing secured transmission between the different devices in the network. The payload of the NWK frame is encrypted.

The APS layer sits on top of the network layer, and is responsible for maintaining local binding tables, group tables, and address maps [34, p. 43]. The local binding table consists of nodes that this node is willing to speak to. APS is also responsible for filtering endpoints that do not exist in the node, that do

not match profile IDs, or are duplicate packets. APS performs automatic retries to maximize the chance of a successful transmission and to inform the sender about the packet delivery situation. APS is a part of the application layer, which is specified by the CSA.

SSP layer offers the NWK and APS layers their security services, key establishment, transport, and frame protection [40]. ZDO layer is responsible for the overall management of the Zigbee device [40]. It is the facilitator for local and over-the-air management of the network, which means that it allows device discovery, managing of binding requests, and initialization of both the NWK and APS layers. ZDO layer is also responsible for managing in which state it is in, meaning whether it is a coordinator, a router, or an end device. The APF layer provides a framework on which applications can run and is also responsible for data exchange between those applications [40]. APF layer also provides an endpoint for each device in order to distinguish one application from another [34, p. 43].

The general Zigbee frame has a few components that can be divided into the 802.15.4 MAC header, the Zigbee NWK header, the Zigbee APS header, and the Zigbee payload [35, p. 567]. The maximum overall packet size of Zigbee packet is 128 bytes, of which the Zigbee payload takes up the most space. The theoretical maximum for the payload is from 102 to 118 bytes depending on the implementation and when using short addressing [35, p. 515]. Depending on whether the frame originates from NWK or APS layer, different security measures are used as can be found in the Zigbee specification [35]. For example, frames originating from NWK layer use Advanced Encryption Standard (AES) if the feature has been enabled, which takes up space in the frame [35, p. 410]. As the overall frame length is quite short the network addressing, for example, should not use too much of the space, meaning that short addressing should be preferred. The exact specifications in different implementations of the Zigbee frame is not that relevant in the scope of this thesis.

2.4.1 Cyber security in Zigbee

There are not that many works in literature that review Zigbee security in-depth. Because of this, the few existing research on the matter will be used as a source in this section. This section discusses Zigbee security from a general standpoint, as well as with examples in section 2.4.2.

Currently Zigbee is implemented in both home appliances, such as lighting or other smart appliances, as well as in larger and more critical operations, such as smart grids, and in the health industry. A lot of the major players in the industry have adopted some usage of Zigbee in some their products, companies like Amazon and Samsung. According to the CSA [6], there are a lot of use cases for using Zigbee in health industry, retail, vehicles, energy, and city infrastructure. Considering the security challenges in

IoT devices overall and the rising popularity of Zigbee devices, this begs the question whether the existing security features in Zigbee are good enough.

Zigbee utilizes the AES algorithm for encryption. This encryption model is however simplified in Zigbee devices [40]. In this simplification Zigbee security maintains an implicit trust between different layers of the protocol stack and all the programs running on a singular device. This means also that there is reduced storage space needed in network packets, and each of the layers can re-use the security keys. End-to-end security is also offered, which means that the communication source and destination are secured via a shared key, resulting in trustworthy communications [40]. For data integrity Zigbee uses Message Integrity Check (MIC), which also can ensure that the data originated from a source with the correct cryptographic key [40].

The NWK and APS layers, as discussed in previous section 2.4, are responsible for secure transportation of frames [40]. Secure transportation of outbound and inbound frames is handled by the NWK layer. NWK layer initiates the necessary keys and establishes the required security level of the transportation for the keys. APS layer on the other hand secures the frames originating from the APL. This consists of securely receiving and sending packets and creating and maintaining keys for cryptographic calculations. In Figure 3 the packet formation of secure NWK and APS packets are shown.

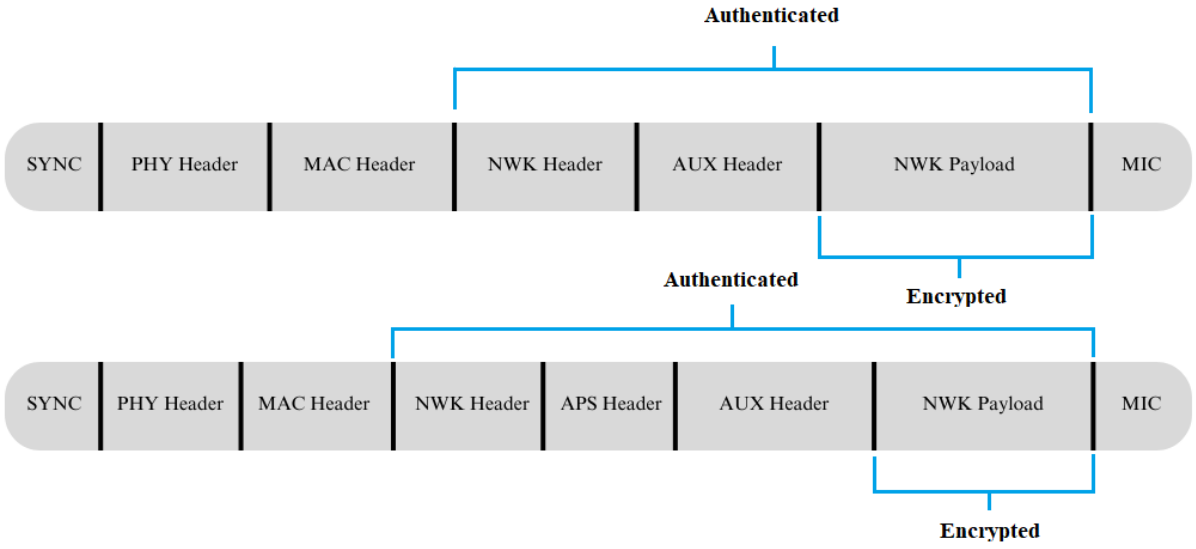

Figure 3. NWK packet security (above) and APS packet security (below) - packet formation.

Zigbee network also always has a trust center, which manages other Zigbee entities in the network and is trusted by other devices to manage cryptographic keys and to disperse them to the devices [40]. Zigbee networks must only have one trust center, and that trust center must be seen as the legitimate trust center by all other devices in the network. Security policies are established, maintained, and updated, if needed, by the trust center.

The security keys used in Zigbee networks are divided into link keys and network keys [40]. A 128-bit network key is shared between all of the network devices and is used to safeguard broadcasts and NWK layer communications. This network key is generated by the trust center, and each device has to attain the network key in order to join the network. This network key should be periodically updated by the trust center in order to minimize the risks of the network key being compromised. The 128-bit link key is used between two devices to secure communication between those two devices. There are two types of link keys: trust center link keys and application link keys. Trust center link keys are used between a device and a trust center, and this trust center link key is also used to send and receive APS command messages between the device and the trust center. Application link keys can be established between any two devices in the network, without the trust center. Application link key can be either preconfigured or requested via trust center with the trust center link key, where the trust center acts as a trusted third party for both devices. A master key is also used in the Zigbee standard, and it is usually pre-installed during manufacturing in each device or alternatively installed by the trust center [40]. This so-called master key is not used in encryption but used as a shared secret in the key establishment process of link keys.

Many of the security deficiencies are very similar in characteristics across different protocols used in IoT communications. Similarly to IoT communication protocols, the computing power is an issue in Zigbee devices, as well as the battery life. The hardware limitations discussed in chapter 2.2 are prevalent in Zigbee devices. The protocol itself does not lend itself to security measures that would use a lot of computing power or require a lot of space in the packet frame. The security measures of Zigbee devices have improved since 2004, when the first version of the protocol was introduced. However, certain limitations, as well as manufacturers' reluctance to improve their products security-wise, are causing vulnerabilities in Zigbee networks and devices.

2.4.2 Vulnerabilities and attack types

The Open Worldwide Application Security Project (OWASP) is a nonprofit foundation that provides users knowledge about security of different hardware and software products, and their vulnerabilities. OWASP has top 10 lists of cyber security vulnerabilities for different areas, and they have one for IoT as well. The list is a broad consensus of the current threat landscape for IoT systems. The latest version of the IoT top 10 list is from 2018 [41], and due to the rapid nature of development in this industry the positions in the list might vary, but the concepts stay the same. The list is as follows: 1. Weak, guessable, or hardcoded passwords; 2. Insecure network services; 3. Insecure ecosystem interfaces; 4. Lack of secure update mechanism; 5. Use of insecure or outdated components; 6. Insufficient privacy protection; 7. Insecure data transfer and storage; 8. Lack of device management; 9. Insecure default settings; 10. Lack of physical hardening. The beforementioned list is not meant to be an exhaustive outlook on threats to Zigbee, as the list is around five years old and is not specifically targeted to Zigbee. However, it does

serve as a good appetizer. Some of the vulnerabilities and attack methods are discussed in this next section. The below discussion is only a scratch on Zigbee security and is not meant to be an exhaustive overview of Zigbee security, but rather a brief look into some potential threats and countermeasures.

In research by Zohourian et al. [42] a review of Zigbee device security was discussed. The research included a table of attacks, which vulnerability they would target, if exploitable, and the countermeasures for that attack. They called it the Vulnerability – Attack – Countermeasure (VAC) sequence. They categorized the attacks into reconnaissance, device manipulation, network control, and Denial of Service (DoS). They did not specify the exact Zigbee or IEEE 802.15.4 layer which the attack would target, but they specified whether it was aimed at the Zigbee or IEEE 802.15.4 part of the protocol stack. In their research they found 16 different VAC's that did not require attack chaining, and 45 VAC's that could be achieved via attack chains. Attack chaining means exploiting an initial vulnerability that opens up other attack options, potentially leading to more severe outcomes as the attacker can exploit multiple vulnerabilities in a row. A lot of the VAC's were not unique to Zigbee, for example man-in-the-middle attacks during initialization, signal eavesdropping, node identification or DoS via overloading the targeted network, among others, are quite normal attack types in other network types as well. However, some of the security challenges were a bit more specific to Zigbee.

One example of a security challenge that is more specific to Zigbee is the key management [42]. In-built weakness in this communication protocol allows for a malicious actor to be able to get the network key via eavesdropping communications between devices in a targeted network. When a new device joins a Zigbee network they initiate the key exchange process between the trust center and the new device. When this happens, the network key of that Zigbee network is encrypted with the trust centers link key or a global trust centers link key before the network key is transmitted to the new device. A malicious actor with knowledge of this default key can intercept the encrypted network key and use the default key to decrypt the network key, therefore gaining unauthorized access to the targeted network. When the malicious actor has gained this type of a foothold in the targeted network, it is theoretically possible to eavesdrop on all the communications on the network.

Another utilized attack type regarding Zigbee are DoS attacks [42]. However, these attacks are not quite similar to traditional DoS attacks, but rather target the weaknesses in Zigbee protocol. The DoS attack can be targeted against Zigbee devices to deplete their batteries, as Zigbee devices are usually not connected to a power current. The attack targets the battery life by continuously sending data packets to force the targeted devices to use energy in order to process the packets. Not only does this have the effect of disrupting the normal flow of the network traffic to and from that device, but it also makes the devices battery life shorter. When the targeted devices battery is depleted, it obviously cannot perform the tasks required of it, which may in turn lead to other problems. Another example of a DoS attack is a

Personal Area Network Identifier (PANID) attack, in where attacker imitates the targeted networks network coordinator and sends beacon frames with the same PANID to the targeted network but uses a different extended PANID [42]. This makes the devices in the targeted network either join the fake network or to continuously search for the legitimate network coordinator without finding it, leading to battery depletion as well. This second type of DoS attack disrupts the flow of the network operations in the targeted network.

The work by Zohourian et al. [42] also mentions the role of the manufacturer and the already existing legacy devices. Many manufacturers do indeed opt for cheaper manufacturing costs, which leads to minimized security features in the devices, which in turn leads to more vulnerable Zigbee devices, which has nothing to do with the actual Zigbee protocol and its security features. The legacy devices also pose a problem in Zigbee. Although the security measures have increased in the protocol, the existing legacy devices in use do not always benefit from them.

Concerns regarding Zigbee safety are very similar to that of IoT in general, although with a few caveats. Zigbee devices can be made safer via mandatory security updates as well as network-wide security policies. As the update mechanisms themselves can be exploited it would be essential for the future to implement such measures to prevent this from happening as well as to increase the level of security in the devices themselves.

3 Wardriving

Despite the ominous name, wardriving is mostly used by information security experts to get statistics and understanding on wireless networks in a certain area, or by people doing it as a hobby. The origin of the term “Wardriving” comes from the 1983 movie “WarGames”, where the term “WarDialing” was coined to represent the act of dialing telephone numbers from a computer in order to gather information on what kind of devices were connected to phone numbers, for example fax machines, computers, or phones [43]. The movie also exposed this practice to the public, which popularized the phenomena. Back in the 1980s the connotation of doing this so-called wardialing was not necessarily all that malicious, but more of an activity for like-minded people to be in the “know” and use this practice to satisfy their own interests. Along with the internet during late 1980s to the whole of 1990s came new ways to do similar activities to wardialing. The same concept was applied to port scanning in the TCP/IP world. There was a lot more to explore in this world, and quite a few enthusiastic hackers had in mind to mass scan areas of the internet in order to explore, or perhaps even to exploit. This in turn led to system operators and administrators building defensive measures for their network, such as limiting connections to come only from certain IP addresses, or perhaps requiring a VPN to connect to their network. From the late 1990s to the early 2000s wireless connections, mostly Wi-Fi, sprung into existence. This made it possible for hackers to connect over-the-air, which was never done before. It was a lot easier to do reconnaissance undetected in this manner, as the device did not need to be physically connected to the targeted network. Getting into a WLAN network made it possible for a skillful hacker to listen to the communications inside that network.

Quite often wardriving is used to gather intelligence on WLAN networks. It can be used for protocols such as Wi-Fi or any other wireless communication protocol, given the correct equipment. In our case, however, wardriving will be used to gather information on Zigbee networks, which is not quite as common, most probably because end users are usually not participants in Zigbee networks themselves, thus this protocol is not as interesting nor popular. Wardriving requires you to have a receiver that can pick up the type of wireless communication the person is looking to gather information on, and software or a command line tool to visually represent the information gathered. In practice the person doing the scanning will physically visit the area they want to scan as they have to be close enough for the wireless communications to reach their receiver. In the case of Wi-Fi, this can vary from somewhere around ten meters and up to a hundred meters usually depending on the environment and the frequency used. As mentioned in section 2.4, the range for Zigbee communications is very similar to that of Wi-Fi. Wardriving is quite often done from a vehicle, as the name would suggest, when the goal is to gather statistical data, as the areas scanned can be quite large geographically, but other modes of transportation are not excluded, although they might be called differently depending on the type of transportation, i.e. warwalking, warcycling etc.

Wardriving can be used in order to gain a better understanding of a how a network works. Wardriving can also be used by malicious actors to reconnaissance a network before attempting to exploit potential vulnerabilities in the network. Some legitimate examples of reasons for wardriving are WLAN coverage analysis, civil planning, and pinpointing security vulnerabilities. The last reason is a double-edged sword, as the exact same thing can be used to find vulnerabilities and exploit them, instead of finding vulnerabilities and patching vulnerable devices. However, finding vulnerabilities often requires active scanning as discussed in section 2.3.2, which is not the way the implementation part of this thesis is built. Active scanning and using this information gathered from this type of scanning in order to gain unauthorized access to the network is illegal, but passively scanning surrounding wireless transmissions being sent by devices and not interacting with them in any other way is perhaps morally questionable, but not illegal. If passive scanning were to be illegal, it would have the potential to only affect people that abide by the law anyways who might be using wardriving as a tool for implementing stronger security measures in their own networks. Malicious actors who use wardriving methods to find vulnerabilities and use those vulnerabilities for unauthorized access would not stop using wardriving method just because it is illegal. The intent and purpose is very important when distinguishing between legal and illegal activities, as well as the style of scanning. It should be noted that laws and regulations differ between regions so it is important to understand that are from the perspective of Finland and Finnish criminal law, and therefore may not apply to the reader's specific country and its legislation.

As Finland is part of the EU, considerations for EU-wide regulation must be also examined. Using the wardriving method means that the scanner is able to gather data on the scanned devices, which means the EU General Data Protection Regulation (GDPR) must be adhered to. As personal data is defined by article 4 in the GDPR to be "any information relating to an identified or identifiable person" [44]. We must examine what kind of data we will be collecting in order to gain an understanding of Zigbee devices and their security features. The data collected via wardriving is quite superficial, but in this thesis it constitutes of at least the device's short or long network address. Although this network address is unique for the device in that network, the same network address in Zigbee addressing can be used in multitude of different Zigbee networks, as the short addresses are only 2-bytes long. As 2-bytes can only amount to 65536 different combinations for the short addresses, the combinations are somewhat limited. The Zigbee devices have a 64-bit MAC address, but the MAC address information is not sent out in every packet as it is only sent in the initial configuration part of the network. With further communications the Zigbee network uses lightweight short addressing in its packets to find the correct partners, as the Zigbee communication protocol is extremely lightweight, see section 4.2.2 for further discussion on this matter, and does not want to send a long address in every packet as sending the longer MAC address would take a substantial amount of space in said packet. These short addresses cannot constitute to be "personal data" as defined in the GDPR, as the short addresses cannot identify a singular device in any way. There is a possibility while scanning that the scanner does come across a long MAC

address, which would be more identifiable. However, there is no link between a MAC address and the owner of the device, as there are no databases for linking the owner to a MAC address, which means that determining the owner from MAC address alone is impossible. The only way to determine the person from a MAC address is to get physical access to the device and confirm the MAC address to be the one that was scanned, but this would potentially require criminal activity to take place. Also, as the devices themselves in Zigbee networks can be a lot of different things from smart lights to other home appliances, it is impossible to determine from the short addressing only what kind of device is being scanned. Scanning does not result in personal data being collected unless the scanner happens to be nearby when the initialization part of the Zigbee network is taking place or the Zigbee devices just happen to use the longer MAC address when communicating with other Zigbee devices in the network, which can happen as can be seen in section 4.3.

There are several ways to wardrive in terms of how its executed and what kind of devices are used. When it comes to the way of how to move around in a targeted area, that is for everyone to choose. The type of transport does not matter, and some may choose to use a bike as their mode of transport, for example. When it comes to WLAN networks, the device that is used to scan the networks can be any device that has a wireless interface card embedded or attached to it. These days almost every smartphone or laptop has the ability to connect to a WLAN network, usually via Wi-Fi, so the scanning entity does not need any extra hardware aside from their laptop or smartphone. However, in order to gather the data in a useful format a software to display the scanned data should be used.

The scanner might choose to bolster the scanning device with an external antenna in order to get higher yield and quality of results, and they may also choose to attach a Global Positioning System (GPS) system to the scanner in order to pinpoint the locations where certain access points were scanned. However, the external antenna and GPS are not required for wardriving, and the goal of what is to be achieved determines the tools used. In this thesis the goal is to build a Zigbee scanning system and to find out about Zigbee security features via the wardriving method, which does not require a GPS, as the location of the devices is not important, and therefore GPS data is not collected.

In the case of Zigbee specifically, scanning is not quite as simple, as the regular laptop or smartphone does not generally have the capability to receive communications done that are defined by the 802.15.4 standard, which Zigbee is part of. As it is required to be able to receive packets in Zigbee communication protocol, a device able to listen to Zigbee communications needs to be attained. One way to listen to Zigbee communications and the one that will be implemented in this thesis, is a USB powered Zigbee interface card that is capable of listening to Zigbee communications. This way the attachment can receive Zigbee packets and be able to determine activity in an area and transmit the data to the laptop its attached to. The rest of it will work similarly as with WLAN scanning, meaning the laptop will have

a software that is able to turn the data into a readable format. This does mean that some equipment is required to be obtained before moving on to the actual implementation phase of this thesis.

3.1 Existing system

This thesis is done to build a system to complement the wireless network scanning methodology presented by Lindroos et al. from the University of Turku. Lindroos et al. have used a wardriving method to scan WLANs in a medium-sized city located in Southwest Finland in order to find out the encryption protocol they use, and alongside gathering information on the manufacturer of the devices as well as the popularity of the two bands used in Wi-Fi, 2.4 GHz, and 5 GHz [8], [9], [10].

The wireless network survey system presented by Lindroos et al. uses a combination of hardware and software to achieve the results wanted. The hardware used in the system are a laptop computer, a dual-band Wireless Network Interface Controller (WNIC) and a GPS receiver [8]. The operating system of the laptop is Windows 10, but the actual WLAN survey platform is a virtualized instance of Kali Linux. The scanning software used is the Kismet software, so that software will also be used to scan Zigbee devices in this thesis, so that the potential outcome could have both of these scanning systems working simultaneously on the same software.

The two articles published by Lindroos et al. that are called “A systematic methodology for continuous WLAN abundance and security analysis” [8] and “The COVID-19 pandemic and remote working did not improve WLAN security” [9] discuss the abundance and security of WLANs as well as the development of the security landscape in the short term. The third article discusses the frequency band and channel distribution in WLANs, but for the sake of this thesis that work is not as major [10]. The goal for their work was to build a system that is efficient, scalable, and easily accessible for analysing and storing WLAN survey data via the wardriving method. One of the motivations for their work is the current IoT landscape. More and more IoT devices are introduced to WLAN environments, so the importance of security aspects of WLANs also increases.

The results of their tests are quite interesting. In short, in their first article “A systematic methodology for continuous WLAN abundance and security analysis” [8] they discuss the findings of their first test drive. They found 720 WLAN networks in the surveyed city in Southwestern Finland. They chose three different areas in the city to survey, and they surveyed those three areas each three times in a consecutive manner. Of the 720 WLAN networks 12,8% were unecrypted, 1% used WEP encryption, 5% used WPA-TKIP encryption and 81.3% used some form of the WPA2 encryption. At the point of their first testing the WPA3 standard was not yet in use, as it became widely available in consumer market only after the year 2020. Consequently, their second article called “The COVID-19 pandemic and remote

working did not improve WLAN security" [9] discussed the effect the COVID-19 pandemic had on WLAN abundance as well as security. However, this study found that between February 2020 and October 2021 the WLAN deployment rate increased by 50,2% in the survey area, but the amount of unencrypted or obsolete encryption methods did not decrease. WPA3 did also not notably increase in use during that time, with only few devices adopting the standard.

3.1.1 Potential improvements to the existing system

The current system designed by Lindroos is quite good at what it does, which is to gather and store information regarding WLANs and to use this information to draw conclusions in the cyber security sphere. The analysis and other work done by Lindroos et al. provides us with insight into the WLAN world and gives us an understanding of the situation in the actual world regarding WLAN security. However, as their work only considers communications defined by IEEE 802.11 standard, the work could be improved in scope. The goal of this thesis is to introduce another wireless communication standard mainly used in IoT devices, Zigbee, that is defined by the IEEE 802.15.4 standard. The reason for increasing the scope is multifaceted. Having a wider look, and therefore more data, into the devices in use and their potential security features only gives us more information to analyze. The more data we have, the easier it is to understand the current state of the overall IoT security landscape, and the easier it is to predict where it might be going in the future. Secondly, the massive increase in IoT devices in the last few decades forces the question for cyber security experts on how to deal with this massive influx of devices in the global and local networks, and what is the actual situation they have to prepare for. Having a better insight into the IoT world from both Wi-Fi and Zigbee can lead to interesting results.

As the existing system is already a proven concept to work, the work done in this thesis will be quite similar in nature. The time which it will take to actually survey the area should not increase, although the two communication protocols do have different styles of transmission, where Zigbee devices transmit a lot more infrequently than WLAN access points. Therefore, the speed of the car is a consideration, as WLAN devices and access points, in theory, are easier to detect than Zigbee devices. The methods for driving will be discussed further in section 4.3.

The improvement for the existing system is to basically add a way to detect another type of wireless communication. How this will be done in the end is yet to be determined. It could just be to include the configured Zigbee sniffer dongle to the existing laptop system capable of scanning WLANs, or it could require two different laptops each running Kismet on a Linux system. Whatever the case, the added functionality will provide a more in-depth look into the network landscape in Finland and could be reflected into other parts of the world as well.

4 Specification and design of a Zigbee scanning system

The plan in this chapter is to build a wireless Zigbee scanning system capable of passively scanning surrounding Zigbee communications. There are several steps to consider when building such a system. The required hardware and software need to be mapped out and obtained and the hardware and software need to be configured in a way that best support the intended purpose of scanning Zigbee communications. The system needs to be thoroughly tested and troubleshot if necessary, and the test results need to be logged so that the cyber security features in the scanned Zigbee devices can be found and analyzed. The proposed design includes a laptop running on a Linux distribution of Ubuntu, a chipset capable of listening to Zigbee communications connected to the USB port of the laptop, and the Kismet software running on the laptop to use the data source and log the data. This design is illustrated in Figure 4.

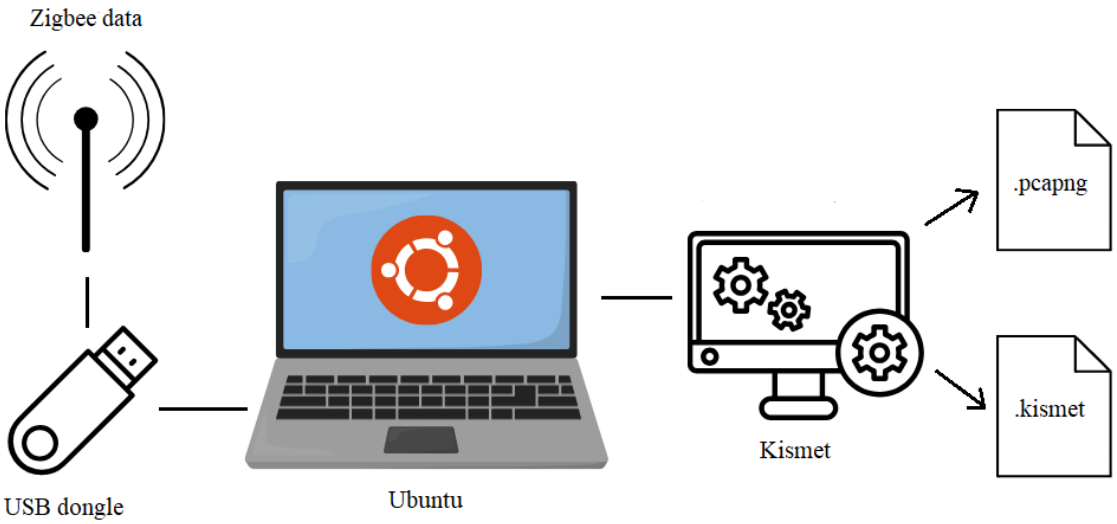

Figure 4. The proposed design of the Zigbee scanning system.

4.1 Selecting the devices for the system

There are several requirements and things to consider for the devices that are used to scan Zigbee communications. Firstly, the chipset used for scanning needs to be able to detect 802.15.4 communications. Secondly, the chipset needs to be compatible with the Kismet sniffer software and the chipset needs to be able to be connected to a USB port, as the host device is a laptop. Thirdly, scanning Zigbee communications can be challenging, as the communication protocol does not transmit the signal far with the typical maximum being around a hundred meters [36]. For this reason, the USB device used for scanning Zigbee communications needs to have a good reception, which can potentially be achieved by the USB device having an external antenna.

The software tool that is used by the existing wardriving system is Kismet, which is an open-source sniffer software developed to capture Wi-Fi, Bluetooth, and Zigbee, among others. Kismet can be operated via Linux, macOS and Windows, although you do need a Windows Subsystem for Linux (WSL) in order to run it on Windows [45]. The choice was made to use Linux as the operating system for this thesis. More specifically, a long-term support release version 22.04.2 of Ubuntu was chosen. This Linux distribution was chosen because of the ease-of-use and familiarity as well as because of the existing documentation that it has in regard to Kismet. The host device itself will be a laptop for convenience.

The developers of Kismet have listed a list of compatible data sources for the software, and there are six different chipset options for gathering Zigbee communications that are compatible with the Kismet software. Due to the availability of different types of chips and also the existing experiences of other users, the chipset of choice for this thesis is the CC2531 chipset manufactured by Texas Instruments. The documentation for this particular chipset as a data source for the Kismet software can be found on their website [46]. A secondary chipset was also chosen, and again due to availability the nRF 52840 chipset was chosen. Both of these chipsets can be used to listen to 802.15.4 communications by default, although they both need to be flashed with the sniffer firmware. In the case of CC2531 it can be done via the CC-Debugger, which is a small programmer and debugger for low power chipsets developed by Texas Instruments.

Because of the low range of Zigbee communication, an external antenna is also preferred in order to gain better results. Luckily in the market there are CC2531 chipsets with an external antenna already attached to them. Therefore the list of hardware to be used in order to be able to passively scan surrounding Zigbee communications is as follows: A host device with Linux operating system, a chipset that can listen to Zigbee communications, which in this case is the CC2531 as the primary choice and nRF 52840 as secondary choice, a software to initiate the scanning and to output the communications into human readable form, which is the Kismet sniffer software, and an antenna to amplify the range from which Zigbee communications can be detected, which is included with the CC2531 chipset.

A lot of the information that this section contains is from the Kismet software website documentation page [45], which has extensive documentation about the Kismet software and associated tools. For clarity purposes references will refer to the documentation on the website as a whole, not to a specific web page, as the information is gathered from dozens of different web pages, which are all part of the same documentation. Only if there is a specific need to reference a certain web page in the documentation, will it be done.

The first step was to obtain a host device for this project. As Linux distributions are quite lightweight, they can be run on a large array of devices. A laptop was chosen, as the project requires the author to move around with the device. As the author does not have a suitable device already in hand, one needs to be obtained. The aim was to meet the minimum system requirements for Ubuntu 22.04.2, to have a processor that is compatible with Linux and be nice to use. The laptop of choice for this project is HP Pavilion model 15 with 8 GB of RAM, Intel Pentium Gold 7505 processor, Mesa Intel UHD (TGL GT2) graphics card, and disk capacity of 512 GB (SSD). It runs Ubuntu 22.04.2 Long Term Support (LTS) 64-bit. Figure 5 shows the obtained devices apart from the laptop and the secondary chipset nRF 52840.

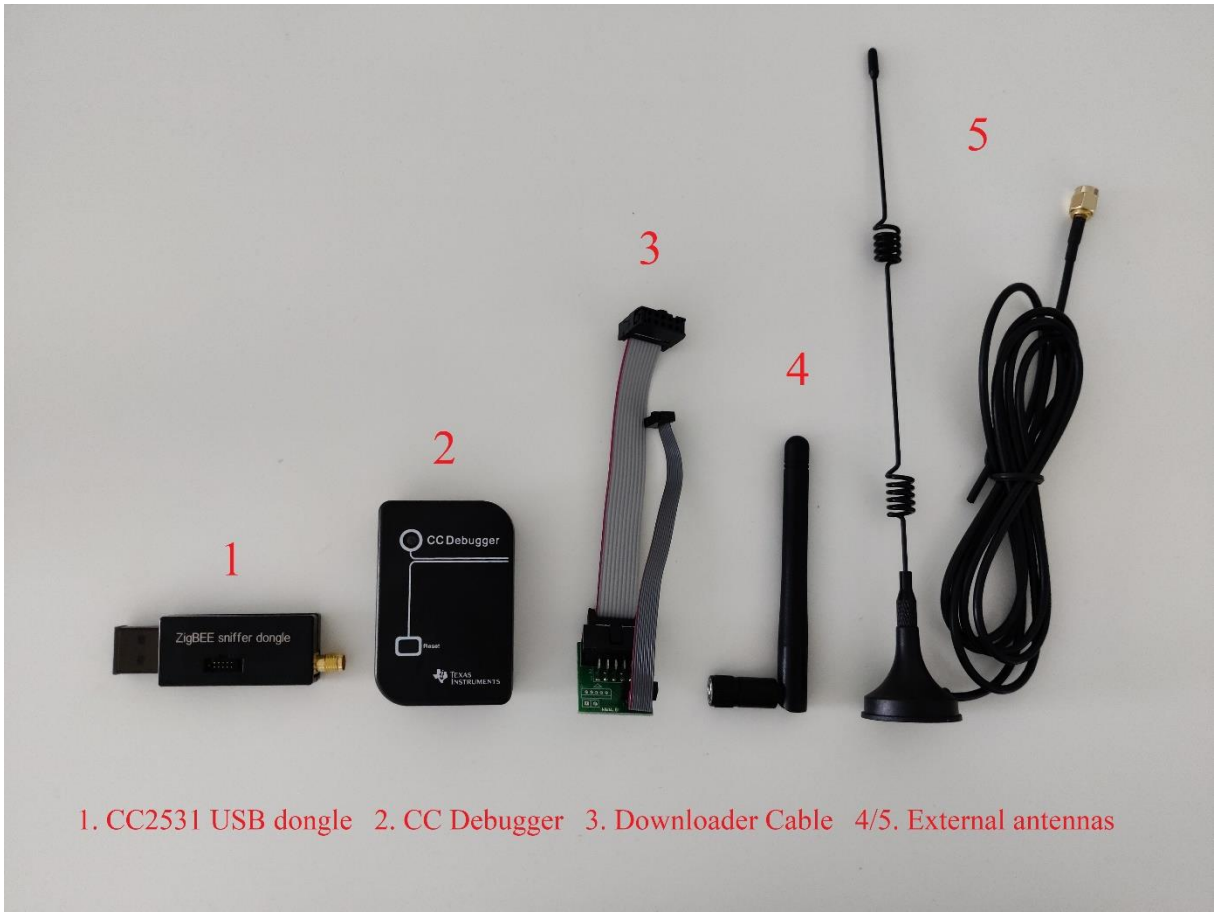

Figure 5. Devices obtained.

4.2 Preparing CC2531 USB dongle

Next step was to flash the CC2531 USB dongle with a sniffer firmware, as the default firmware does not support using it as a sniffer. The flashing can be done with Texas Instruments' flash programmer software [47]. This software can be used to flash CC-type chipsets, for example the CC2531 USB dongle. This step was done in Windows 10. Steps for flashing the chipset are as follows (also illustrated in Figures 6-8):

1. Install Texas Instruments' SmartRF Flash programmer version 1 [48].

2. Install CC debugger driver [49].
3. Connect the CC debugger to the CC2531 USB dongle via the downloader cable and connect both the USB dongle and the CC debugger to USB ports on a Windows machine (Figure 6).
4. If the light on CC debugger is green, continue to step 5, if not, troubleshoot.
5. Get the CC2531 sniffer firmware .hex file (Figure 7). This file can be found on the Texas Instruments' website under the SmartRF protocol packet sniffer download file [50]. Version 1 was used.
6. Flash the CC2531 USB dongle with the correct sniffer firmware via the SmartRF Flash programmer (Figure 8).

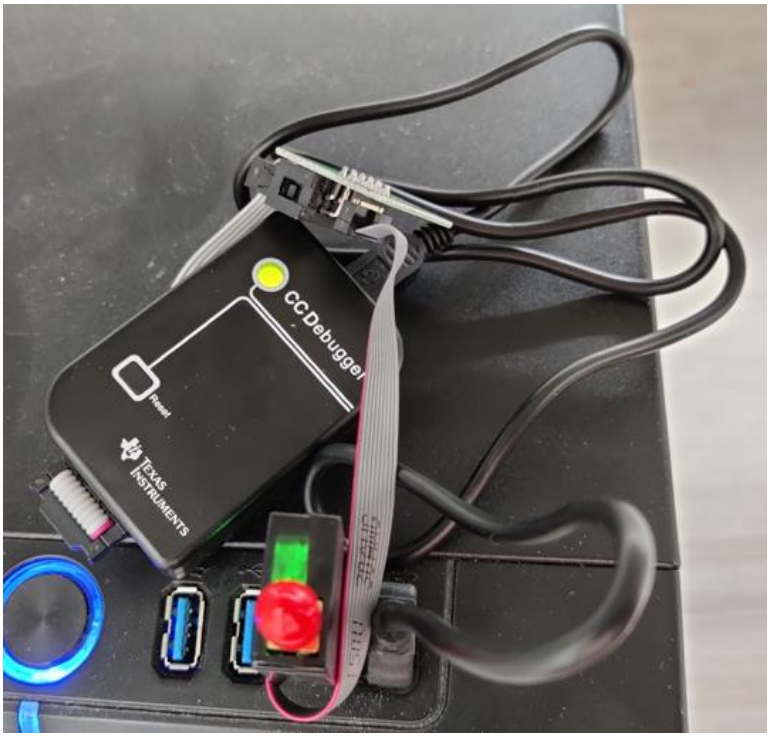

Figure 6. Connecting the CC debugger and USB dongle to a Windows 10 desktop computer (CC debugger light is green).

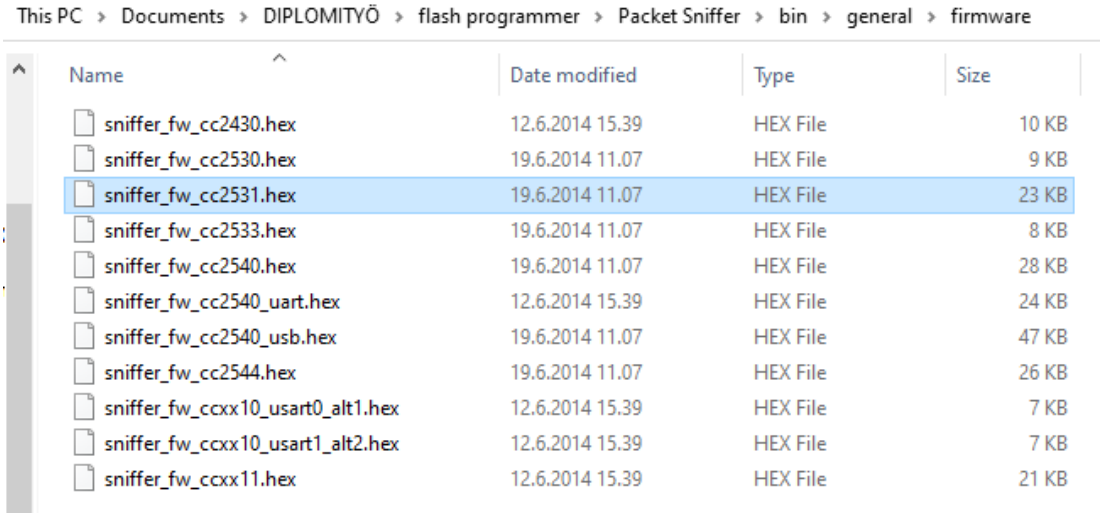

Figure 7. Finding the correct .hex file.

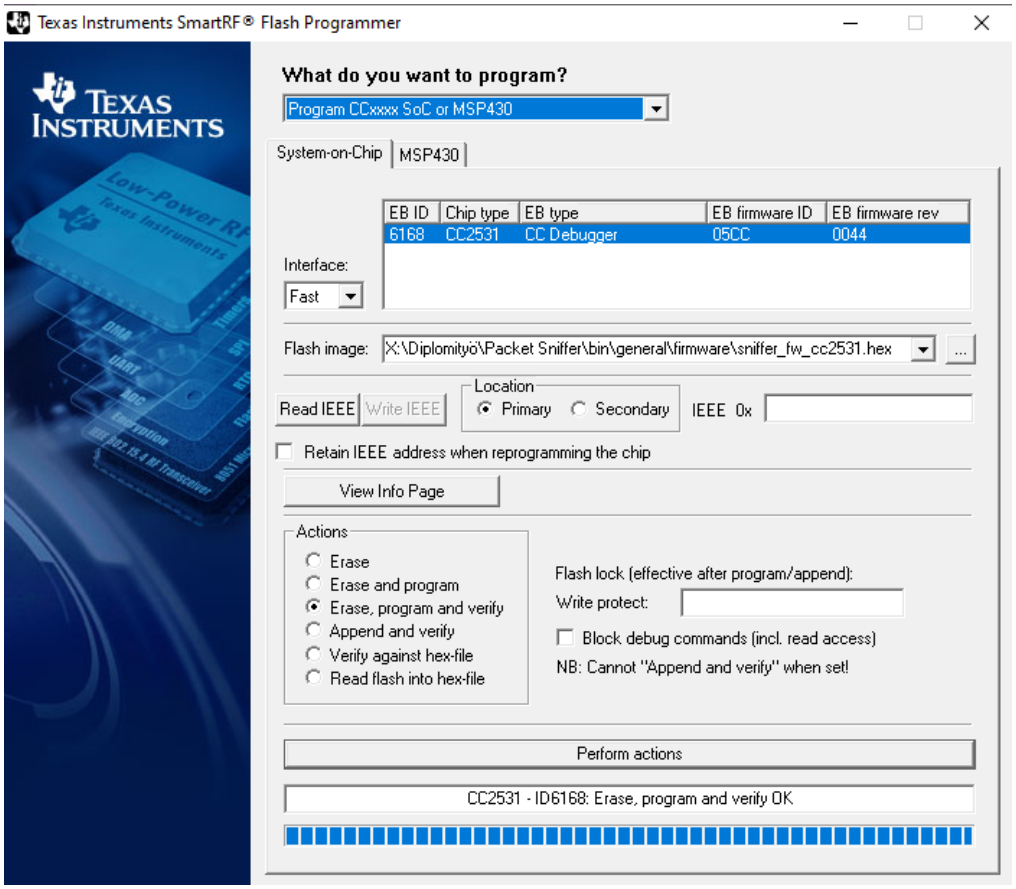

Figure 8. Flashing the CC2531 chipset with SmartRF Flash programmer.

4.3 Installing Kismet software and testing CC2531

As we now have the CC2531 chipset flashed, we can proceed to the next step, which is to install Ubuntu 22.04.2 LTS on a laptop, and then install the Kismet sniffer software in it. The installation of Linux distributions is quite well documented, so there will be no documentation about it here. Installation of the Kismet software can be done in a few ways, but Kismet happens to have ready-to-install packages for several Linux distributions, and the Ubuntu 22.04.2 is one of them. Other option would be to manually download the files from a GitHub repository. The installation is illustrated in Figure 9.

```
tansku@tansku-HP-Pavilion-Laptop-15-eg0xxx:~$ wget -O - https://www.kismetwirele
ss.net/repos/kismet-release.gpg.key --quiet | gpg --dearmor | sudo tee /usr/shar
e/keyrings/kismet-archive-keyring.gpg >/dev/null
echo 'deb [signed-by=/usr/share/keyrings/kismet-archive-keyring.gpg] https://www
.kismetwireless.net/repos/apt/release/jammy jammy main' | sudo tee /etc/apt/sour
ces.list.d/kismet.list >/dev/null
sudo apt update
sudo apt install kismet
```

Figure 9. Using GNU Wget software to retrieve the correct files for Ubuntu 22.04.2.

Running the Kismet software only requires a simple “kismet” command in the terminal, or with “sudo kismet” to run it with superuser privileges. The tool can be used “off the shelf”, but there are several ways to configure it via the configuration files located in /etc/kismet/ if installed from the packages. If not installed from the packages, the configuration files by default are installed into /usr/local/etc/. The

configuration format is plain text and takes the form of “option=value”. This is the main way to configure the software. [45] The Kismet configuration folder is illustrated in Figure 10.

```
tansku@tansku-HP-Pavilion-Laptop-15-eg0xxx:~$ cd /etc/kismet
tansku@tansku-HP-Pavilion-Laptop-15-eg0xxx:/etc/kismet$ ls
kismet_80211.conf   kismet_filter.conf   kismet_memory.conf
kismet_alerts.conf  kismet_httpd.conf    kismet_uav.conf
kismet.conf         kismet_logging.conf  kismet_wardrive.conf
tansku@tansku-HP-Pavilion-Laptop-15-eg0xxx:/etc/kismet$
```

Figure 10. Kismet configuration files.

Kismet software can be used via the Linux terminal, or with a graphic User Interface (UI) on a browser. Starting the software up with a “kismet” launches the software with a default configuration, assuming no chances are made in the configuration files. To have the network interfaces configured and the capture process started, Kismet requires root privileges, which can be either granted by installing the capture tools as suid-root, or by running the whole Kismet software as root via the “sudo kismet” command. The “sudo kismet” command is used here to run the Kismet software, and then the software web UI is accessed via the browser. By default, Kismet uses the port 2501 to listen for client connections, so accessing the web UI can be done by typing http://localhost:2501 on the search field. The user is prompted for a username and password the first time they visit the web UI. The default UI that can be seen on startup is illustrated in Figure 11.

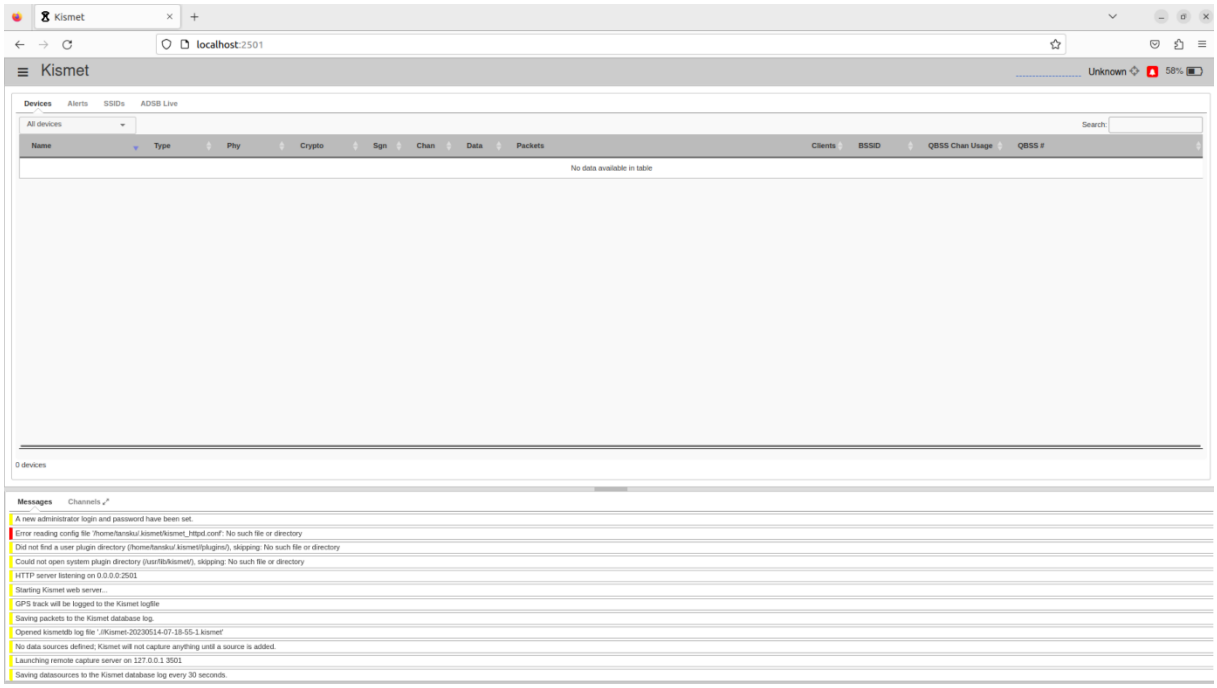

Figure 11. Default web UI for Kismet.

Now that both the CC2531 USB dongle and Kismet software are both up and running, we can add a data source to Kismet from which to capture Zigbee traffic. By inserting the CC2531 USB dongle to the host laptop and searching for data sources on the web UI the available interfaces can be seen. This step is illustrated in Figure 12. Other available network interfaces are also shown, namely a Wi-Fi and

Bluetooth interface, but those are of no interest right now. Now the CC2531 USB dongle is up and running, and in theory it can be used to capture Zigbee traffic given that the Zigbee communication is near enough for the antenna to catch it.

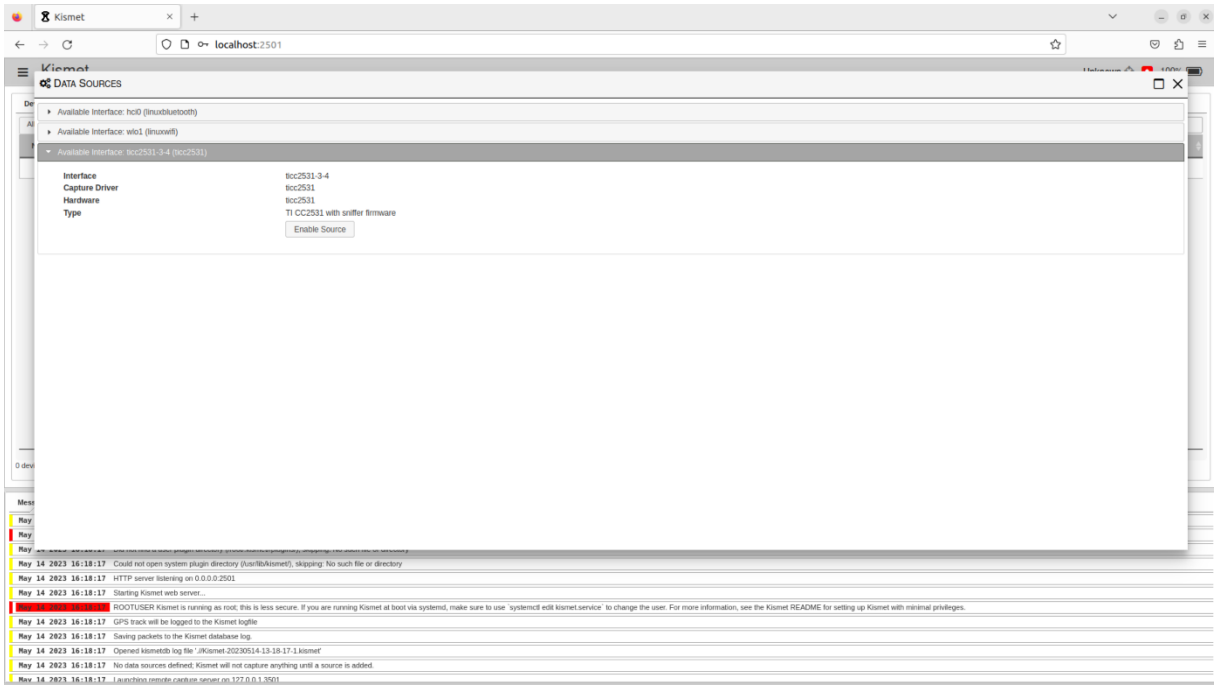

Figure 12. CC2531 as a data source.

To test that the CC2531 chipset USB dongle and the antenna is working a test run needs to be done. By choosing the correct data source, which in Figure 12 can be seen to be “ticc2531” and pressing “Enable source”, the view that is demonstrated in Figure 13 pops up.

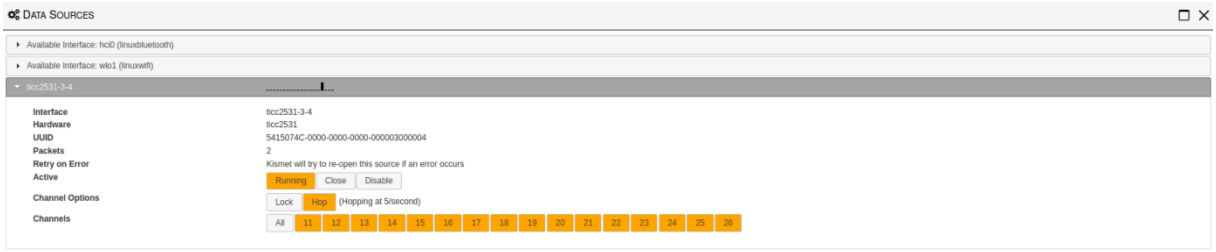

Figure 13. Choosing the data source and channels to detect communication in.

Enabling a source to gather Zigbee communications starts producing results if there are transmitting sources nearby. Although the 2.4 GHz frequency range is shared with Wi-Fi, the 802.15.4 uses a different kind of physical encoding standard, which means that the Wi-Fi network cards cannot see 802.15.4 packets and 802.15.4 devices cannot see Wi-Fi packets.

The output for detecting Zigbee communications are divided into columns in the Kismet web UI. The columns tell information about the following aspects if information is available: Name (MAC address or other identifier), Type (communication protocol type), Phy (physical layer type), Crypto

(cryptographic features), Sgn (signal strength), Chan (channel), Data, Packets (visual view of when packets move to or from a device), Clients, Basic Service Set Identifier (BSSID), QOS enhanced basic service set (QBSS), Chan Usage, (channel usage) and QBSS.

As mentioned in chapter 3 of this thesis, detecting surrounding Zigbee communications is not illegal in any way. Being aware of surrounding over-the-air communications is a basic feature that many devices hold, for example smartphones and laptops are very aware of surrounding WLAN networks, and passively "scan" them at all times when the wireless capability is enabled. This type of passive scanning gathers only basic information, such as device identifier and security features, which are essential for interacting with these devices, for example interacting with WLAN routers via your smartphone. The same principle works with Zigbee as well as we are not actively listening to any communications, but rather passively seeing what devices are communicating their presence near us.

Figure 14 has the output from passively listening to Zigbee communications, with some part of the device identifier censored just in case. MAC addresses are unique identifiers assigned to network interfaces within a network segment

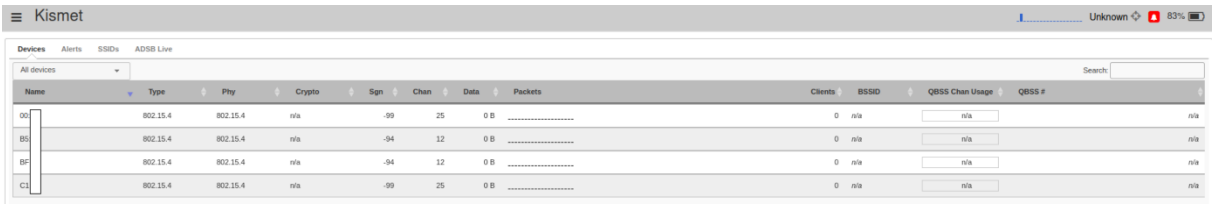

Figure 14. Output of passively detecting nearby Zigbee communications, identifier partly censored.

What is interesting here is that we can clearly see that these identifiers are not your regular MAC addresses, as these are a lot shorter than the regular 6-byte, or 48-bit, MAC addresses. The regular MAC address length in Zigbee devices is 8-bytes, or 64 bits, which means it has sixteen hexadecimals. An example of this could be 01:23:45:67:89:AB:CD:EF. This is a bit different from the traditional 6-byte, or 48-bit, MAC address, but it is specified in the 802.15.4 standard. This MAC address uniquely identifies the device and distinguishes it from all other devices in the world, including Zigbee devices. For the sake of clarity, MAC address, long address, and extended address are used in this thesis interchangeably, but they all mean the same thing, which is the 8-byte address. As said, the identifiers for the devices are only 2-bytes long, containing 16 bits, which means four hexadecimals. These short addresses can also be said to just be short unique identifiers. The reason that the identifiers are this short is most likely to do with the fact that the frame format in Zigbee networks is extremely lightweight, meaning that a regular 8-byte MAC address would take up a substantial amount of the packet capacity. As mentioned in section 2.4, the maximum frame length is 128 bytes. As the MAC address is needed in order to establish connections between devices, the length of the MAC address becomes problematic as

it needs to be sent via the network frame. Because of this the Zigbee networks instead assign short network addresses to devices after the initial communication in order to transmit frames more efficiently. It is quite easy to figure out the manufacturer of the device from traditional 8-byte MAC addresses, but with these short addresses it can be impossible to determine. The short addresses are also more limiting quantity-wise, but LANs most likely do not require many devices, and because the range of Zigbee communications is by default so short, there should not be any overlaps with other Zigbee networks even with just 2-byte identifiers, as there are 65536 possible combinations when using 2 bytes (2^16).

According to Zigbee specification [35, pp. 336-340], there are several ways for a Zigbee device to join a network. Joining a network through association, which uses the MAC layer association procedure can be done when network discovery is enabled either for a child device or a parent device. Parent devices in Zigbee network are coordinators and routers and child devices are end-devices. For a child device that is not already in a network, the request will cause the NWK layer to find a suitable parent device, where the parent device is open for requests and is advertising the correct device type. If there are no suitable parent devices nearby, the child device will not join a network. If the attempt to join a network is successful, the joining device will be assigned a 16-bit logical address by the NLME (Network Layer Management Entity), which is the coordinator, for further transmissions between the parent and the child device.

This parent-child relation will be held in the corresponding neighbor table in the parent device. These 16-bit logical addresses are what we can see in Figure 14, when the child devices are communicating with the parent device. If a parent device wishes to be the one to initiate for a child device to join a network, the parent device must be either a coordinator or a router in order to initiate the procedure via the MAC sub-layer. In this process, the parent device will first determine whether the child device is already on the neighbor table by searching for a 64-bit address and whether its device type is correct. If a corresponding 64-bit address is found and the device type is correct, the NLME will allocate a 16-bit network address for the child device. If the 64-bit address is not found in the neighbor table, the NLME will allocate a 16-bit network address for the child device. If the device type is incorrect or the parent device does not have the capability to accept more child devices, the process will be eliminated.

Another way of joining a network is by rejoining it after losing connection [35, pp. 341-344]. This way of rejoining a network utilizes the NWK rejoin request in the NWK layer. It is otherwise very similar to the abovementioned MAC procedures, but NWK does not require an authentication step as NWK commands utilize the NWK built-in security. This rejoin procedure utilizes rejoin request and response between the joiner and the router, and the update-device commands from router to the trust center. In the communication between the joiner and the router the frames are encrypted with the NWK key, and

from the router to the trust center the frames are secured with both the APS and NWK key. This way of joining enables a device to join back a network that does not otherwise allow new devices to join.

The third way of joining a Zigbee network is to join directly. What this means is that the parent device is preconfigured with the 64-bit address of the child device, and using this data the parent device may initialize a request set to the 64-bit address in order to initiate the joining process. This follows a similar route to the two ways of joining mentioned above, where the NLME assigns a 16-bit network address for the child device, and after this they can communicate with each other.

These short 16-bit network addresses are problematic in the scope of this thesis, as the aim is to get an idea of Zigbee security via the wardriving method, but if the results are always as limited as in Figure 14, it can be near impossible to determine what types of devices we are receiving transmission from. From regular 48 or 64-bit MAC addresses the manufacturer can usually be determined, as the first few characters in a MAC address correspond to the manufacturer. If the address is only four characters long, the manufacturer becomes impossible to determine. However, the devices do have 64-bit MAC addresses, but they are just not sent via the communications between the devices in order to save valuable space in the network packet frame.

Some other interesting features here are the type and PHY columns, which reflect the 802.15.4 standard communications, meaning we are finding correct communications. The next two interesting columns are the signal strength column, as well as the channel column. Signal strength is represented in minus decibel milliwatts (-dBm) format, which goes from 0 to -100. The signal strength in wireless communications is always represented with -dBm and the number indicates how strong the signal is. The higher the number, the more loss of the signal there is, and the lower the number, the stronger the signal. The channel column tells us the dominant channels, which in this case are channel 12, which corresponds to 2.410 GHz, and channel 25, which corresponds to 2.475 GHz. As seen in Figure 13, the channels vary from 11 to 26, so it will be interesting to see whether this trend continues in later attempts, and what the reason for it may be. The crypto column will definitely be an interesting one to follow in future tests, as this one did not produce any results. Whether it is from the fact that no cryptographic measures are taking place, which is not likely, or whether Kismet is not able to translate it correctly, which is somewhat likely, or whether the sniffer is not capable of detecting this information from the communications, which is potentially likely, remains to be seen. The cyber security of these devices is the most important in the scope of this thesis. However, other things are at least good to know in order to cultivate an understanding of the Zigbee communications.

To check if there are differences when using a different antenna, a test needs to be made with the antenna number five in Figure 5. A snapshot of results with both of the antennas are taken one after another, and

each will be given ten minutes to detect nearby 802.15.4 transmissions and see if there are any differences. Ten minutes should be sufficient time for devices in range to transmit signals to one another. This is done in an apartment building, which has a few apartments building near it, but no other residency approximately one hundred meters from the building. For this test, we are solely interested in the range and therefore detecting capabilities of the antennas, so we will be focusing on signal strength and the number of devices found. First, we will be testing the antenna that is numbered as four in Figure 5. This antenna found fifteen different unique identifiers around the test laptop, with signal strengths varying from -87 to -97. The average signal strength was -93. Then we will test the antenna that is numbered as five in Figure 5. This antenna found seven different unique identifiers of which two the first antenna did not find. The signal strengths varied from -94 to -98, with an average signal strength of ~-95,9. The first antenna found significantly more devices, but the second antenna found two devices that the first antenna did not find. The average signal strength of the first antenna was stronger than that of the second antenna, so the first antenna will be used in further tests.

4.4 Preparing nRF 52840 USB dongle

The nRF 52840 USB dongle can also be used to similar effect that the CC2531, although setting up the nRF 52840 is a bit different. nRF 52840 also requires flashing it with a sniffer firmware, as the default firmware does not support Kismet. NordicRF [51] and the Kismet home website [45] has good documentation about doing this, so the steps are as follows:

1. Install the nRF Connect for Desktop (v3.2.0 or later) [52] and the J-Link debug probe (Figure 15).
2. Install the programmer within the nRF Connect software (Figure 16).
3. Download the correct firmware provided by NordicRF [53].
4. Flash the nRF 52840 USB dongle with the sniffer firmware (Figure 18 and Figure 19).

Below can be seen the steps in order to get the nRF 52840 USB dongle to work with kismet. In Figures 14-19 are mostly technical measures that need to take place before using the USB dongle in sniffing activities. Figure 17 shows the default view of the nRF programmer. Figure 20 illustrates the stages from inserting the USB dongle to a computer through installing the firmware on the computer. The documentation in NordicRF site [51] on how to program the USB dongle proved to be extremely helpful, although the colors of the LEDs seem to differ from the instructions, as the blue light was red in the instructions. By pushing the “RESET” button on the USB dongle we make it go into a bootloader mode, which enables us to configure the dongle.

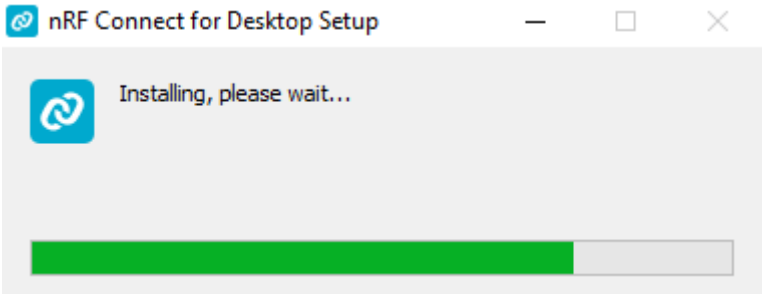

Figure 15. Installing the nRF Connect for Desktop (version 4.1.2).

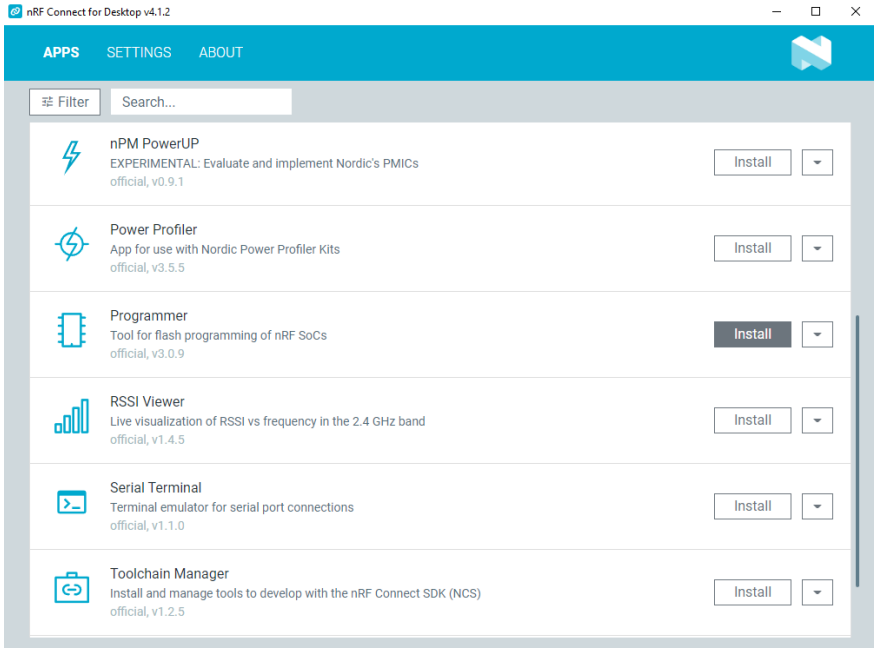

Figure 16. Opening the nRF Connect and installing the programmer function.

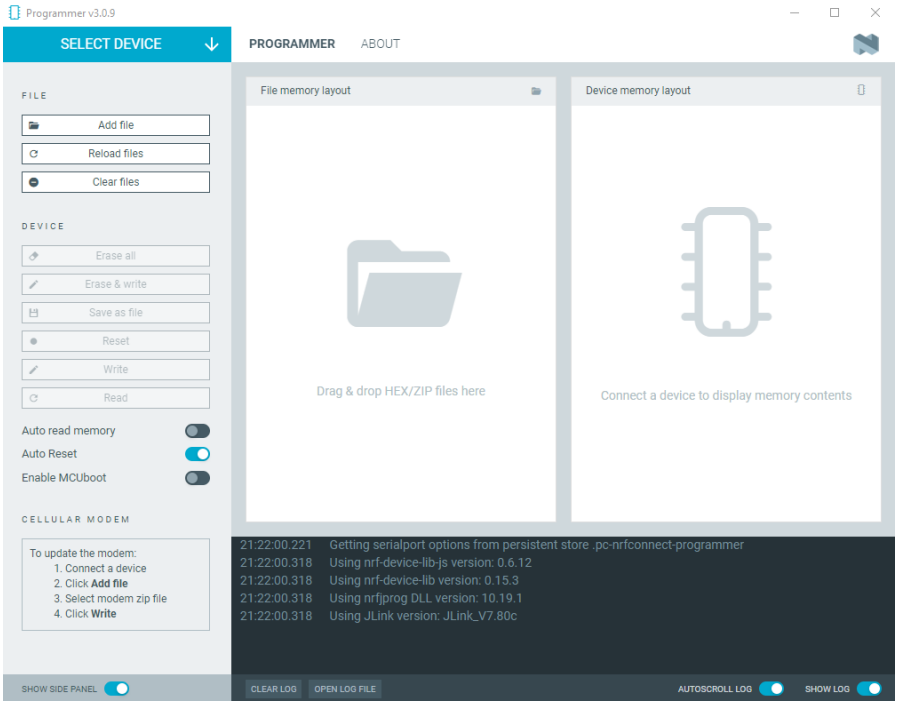

Figure 17. Default view of the nRF programmer.

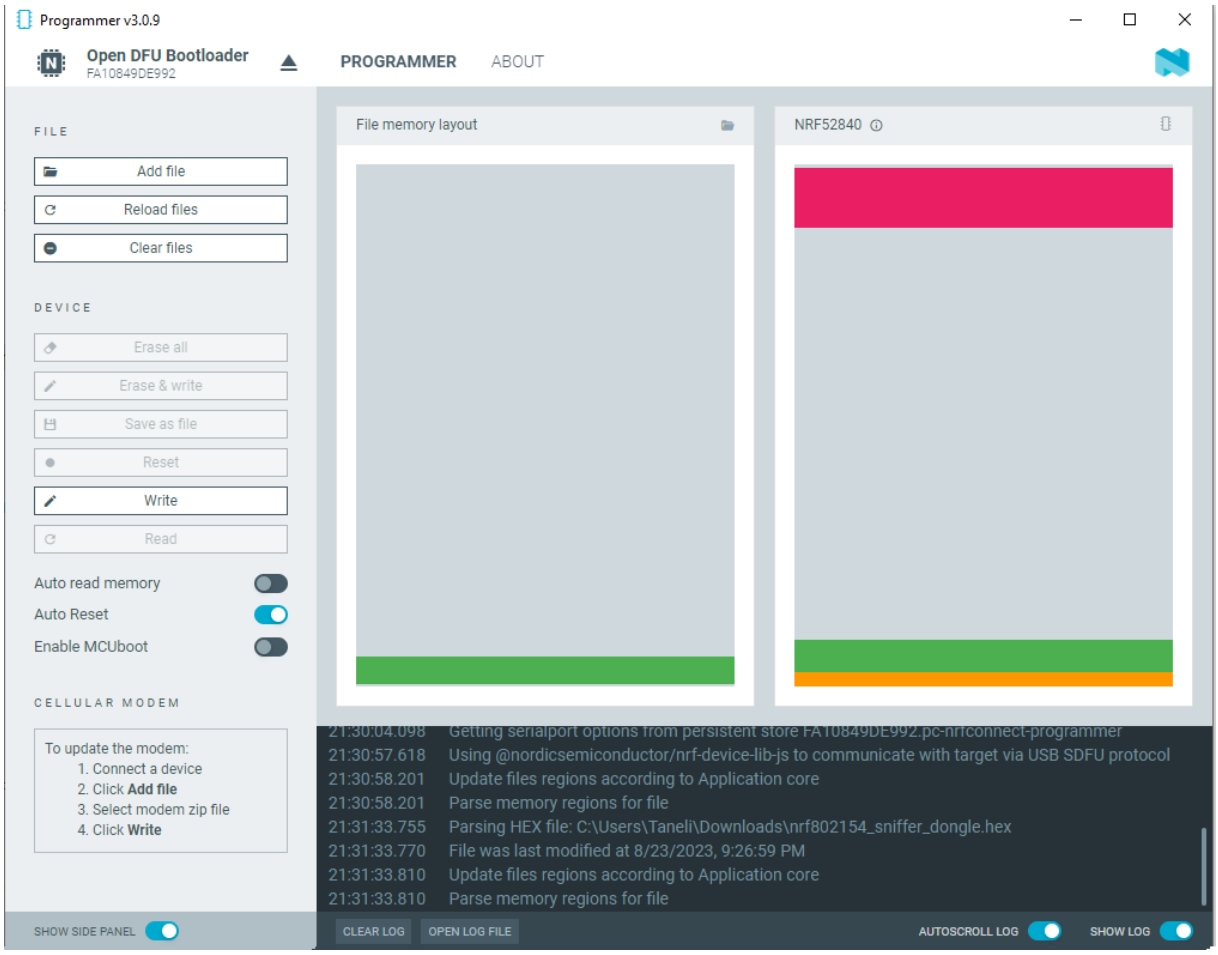

Figure 18. Adding the nRF 52840 USB dongle and the sniffer firmware to the programmer.

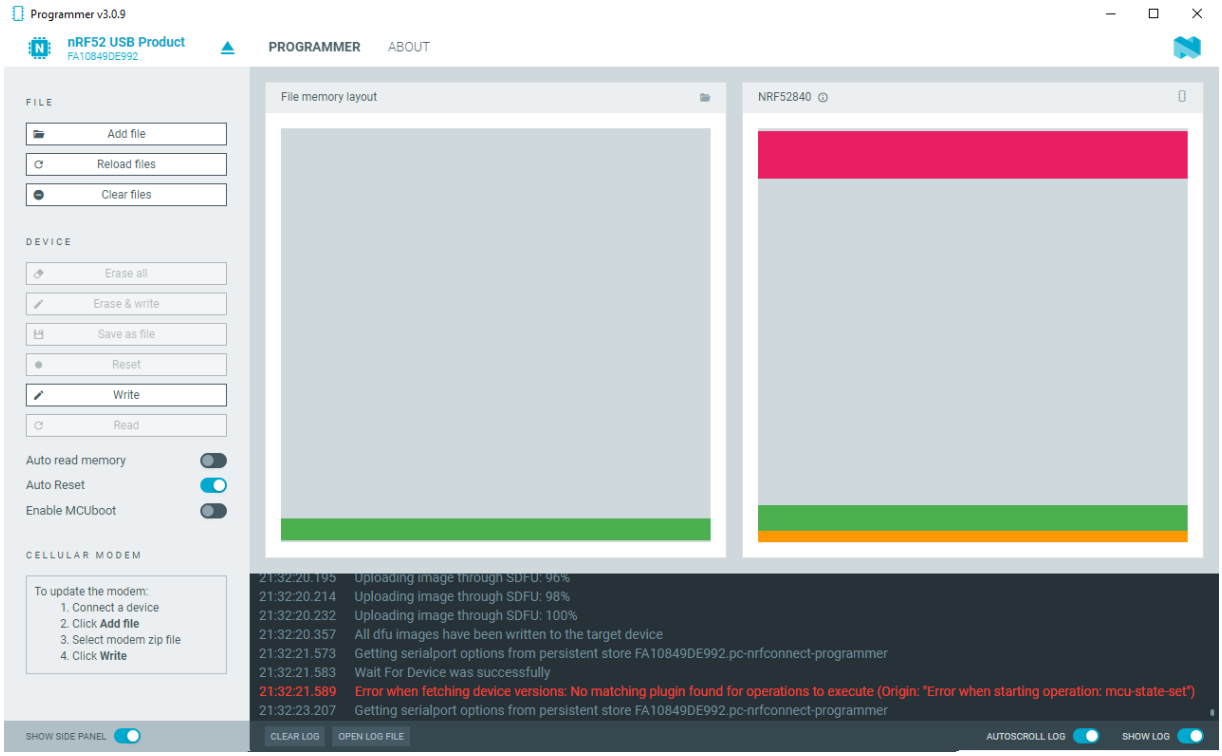

Figure 19. Flashing the nRF 52840 USB dongle with the sniffer firmware.

Figure 20. Inserting the nRF 52840 dongle to Windows 10 machine (no light), pressing the reset button on the USB dongle (blue flashing light), flashing the USB dongle with the sniffer firmware (green flashing light).

4.5 Testing the nRF 52840 USB dongle with Kismet

After flashing the chipset, we can test the nRF 52840 capabilities in detecting nearby 802.15.4 communications via Kismet software. However, the nRF 52840 requires us to manually add it as a datasource as opposed to the TICC 2531 USB dongle that was detected automatically by the Kismet software as soon as you plugged the USB in. This is done by adding the source to the Kismet config files, namely the kismet.conf file. This could also be done by using an override file that has precedence over other files, but we are going to test using the kismet.conf file first. Checking the Kismet documentation on nRF 52840 [54] we need to add the source command "source=nrf52840:device=/dev/ttyUSB0" to the kismet.conf file as shown in Figure 21. After doing this, we launch kismet via "sudo kismet". However, launching the Kismet software causes an error message "nrf52840 failed to open serial device – No such file or directory". This is illustrated in Figure 22.

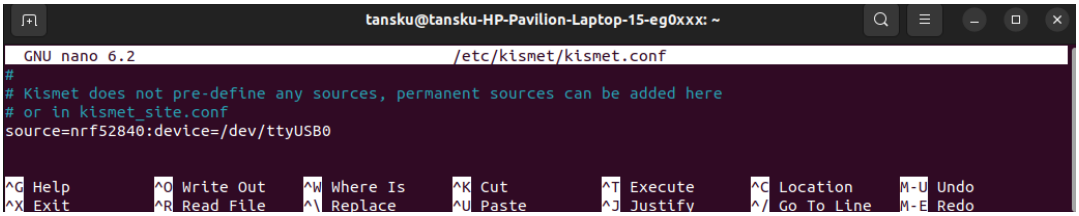

Figure 21. Adding the datasource to the kismet.conf file.

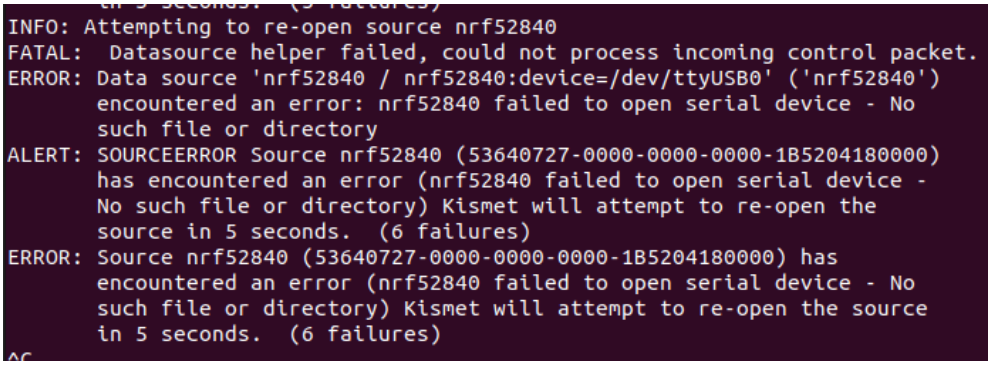

Figure 22. Datasource encountered an error.

Because the /dev/ttyUSB0 folder shown in Figure 22 does not exist, we must create it. Creating a file in /dev directory can be done via the “mknod” command. The whole command we are going to use is “mknod -m 666 /dev/ttyUSB0 c 4 0”. To break the command apart, “mknod” makes a directory entry for a file. “-m 666” stands for the file rights, in this case when a text file has the “666” permissions, which means it has read and write permissions for everyone. “/dev/ttyUSB0” is the location we want the file to be in, as “/dev” holds special and device files. “c 4 0” stands for c = type, 4 = major, and 0 = minor. C-type device sends and receives single characters, such as bytes, as opposed to b-type, which communicates by sending entire blocks of data. The major number can be found in the /proc/devices file, illustrated in Figure 23.

Figure 23. Finding the major number.

What we are looking for is the TTY number, which in Linux means an abstract device. The number for this Linux machine’s TTY is 4, so we are going to be using that number. The minor number cannot be found like this. The minor number however can be found in the Linux kernel archives [55]. The /dev/ttyUSB0 minor number is 0 according to the archives, so we are using that number. Launching the Kismet web Graphical User Interface (GUI), we can see from Figure 24 that the nRF 52840 is added as a data source.

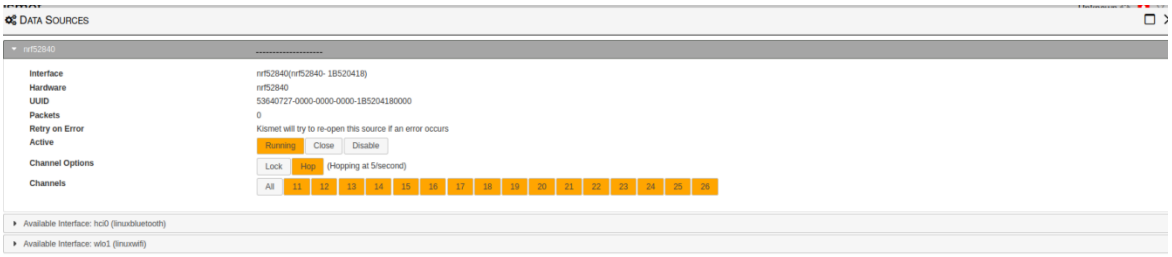

Figure 24. nRF 52840 up and running.

However, it is not producing any results of nearby 802.15.4 communications. The reason for this could be a mistake in configuring, hardware failure, there are no nearby communications, or the range of the nRF 52840 for detecting these types of communications is too short with the integrated Printed Circuit Board (PCB) antenna. Testing quickly with the CC2531 USB dongle, it seems that there are indeed nearby communications currently transmitting 802.15.4 signals, so the reason cannot be that there are no nearby communications.

As we have a working CC2531 USB dongle, the nRF 52840 will not be further tested. This part is left in the thesis to provide the reader an alternative chipset for Zigbee scanning, and to give an idea on which kind of problems might arise during initialization and configuration. We also cannot exclude the possibility that the nRF 52840 chipset has deficiencies that stop it from working, although this seems rather unlikely. This part will also be kept for the reason that if the reader wishes to duplicate this work in their own environment, this thesis could be a starting point. As the nRF 52840 is not the only option available, the CC2531 will be used unless that, too, does not work or does not prove to be a good option otherwise.

4.6 Logging

The log files you get by capturing Zigbee traffic are by default in .kismet file format [45]. This file format is based on sqlite3, which is a database file that contains relevant information about packet and non-packet data, location information, device records and client records, among other things. As with other SQLite databases it can be read with software designed to do so. For ease-of-use a browser application can also be used. As .kismet files are based on sqlite3, they can be read with applications that can read sqlite3. The other built-in options for filetypes in Kismet are .pcapng, .pcapppi, and .wiglecsv. The first, .pcapng, is a packet capture format used to directly feed the data into Wireshark or similar tools, and it contains more details about the data packets. The second, .pcapppi is also a packet capture format, but for legacy systems, and should only be used if the used legacy packet process does not support .pcapng. The third, .wiglecsv is meant for directly uploading to the Wigle project, which is a community data collection site for wardriving. For testing the .kismet file format is going to be used. If necessary, the .kismet files can be converted to other file types, such as the previously mentioned .pcap, .pcapng, .wiglecsv, and others, such as JavaScript Object Notation (JSON) records or Keyhole Markup Language (KML).

Viewing the .kismet files via a sqlite3 viewer application either on desktop or in browser provides quite superficial results. To get more in-depth analysis potential the .kismet file can be turned into a .pcapng file, which is the filetype that contains the most information as an output of the Kismet software. The .pcapng file can be read in Wireshark, for example, to further analyse the communications between Zigbee devices. Kismet has a tool called kismetdb_to_pcap, which can be used to convert the .kismet file into .pcapng file as illustrated in Figure 25. This way of converting a file to its most content rich file format will be used further in the analysis part of this thesis.

```
tansku@tansku-HP-Pavilion-Laptop-15-eg0xxx:~$ kismetdb_to_pcap --in 5kertaa.kism
et --out kismet5kertaa.pcapng
Done...
```

Figure 25. Converting a .kismet file into .pcapng file.

5 Testing and evaluation of the scanning system

In this part we will be conducting test drives using the wardriving method as discussed in chapter 3. We will be choosing the CC2531 USB dongle for this part of the testing. We will be initially driving a route in a municipality in southern Finland three times across three different days and try to gather results of 802.15.4 communications via our Zigbee sniffer device. We will try to find devices that are using Zigbee communications despite the fact that Zigbee devices transmit quite infrequently, and the range is low. As we are using a car, we are moving between points quite fast, so it will be interesting to see the differences in results between different days. The route will remain the same for all of the three days, but the time of day might slightly alter. The route we are driving has quite low speed limits, from 30km/h to 50km/h, which can be beneficial. This might improve the number of devices we can detect transmitting Zigbee signals, as we are longer in one place, therefore increasing the odds of the Zigbee devices transmitting signals as we go past. The route starts and ends at the same location and is roughly eight kilometers long. The route takes place around and inside the city center. The laptop and the CC2531 USB dongle will stay on the front passenger seat for the whole duration of the drive. There exists a possibility that the results from this drive are unsuccessful, in which case there might be a need for doing similar type of testing via foot or bike in case the car method is not successful. Optimally we would get at least some results via the wardriving method. Kismet is used with default configurations in the following tests.

The first test was done in October of 2023, on a Friday evening, approximately at 6 pm. The speed limits in the area were between 30km/h and 50km/h, but at points there was little to no other traffic so the speed could be dropped slightly below that. The number of unique devices we found transmitting signals on the route was 18. All of the 18 unique addresses were 16-bit short network addresses. Before doing a second test, it is quite hard to know how many Zigbee devices there are overall along the route, as the Zigbee devices transmit infrequently and with a rather short range. There are houses and apartments along the route quite close to the road, so if there are Zigbee devices transmitting in those buildings the signal should be strong enough to reach the Zigbee sniffer. As we are continuously driving and not stopping, it might be mostly up to luck whether the devices transmit signals as we go past them. However, further testing needs to be done in order to find out whether there are more devices on the route, and whether the devices are the same as in this first test. If the devices are not the same, we can determine that there are more devices along the route than just the 18 we found on this first test drive. As it was not possible to check the laptop screen while driving, it was hard to determine which areas had the most activity. There were apartment buildings on the route, which house more people than other areas, meaning that there could be more IoT devices in use as well.

The second test drive was done in the same month, the next day, Saturday, at approximately 6 pm as well. The number of unique devices found was 14, which is slightly lower than the number of devices found in the first test drive. Of the 14 devices found, 11 were 16-bit short network addresses and 3 were 64-bit MAC addresses. What is interesting to find out though, is how much overlap there is between the devices found on Friday and the devices found on Saturday. To figure that out, we will be checking the .kismet log files to find out the unique addresses of each device and compare them to each other. For this we are going to assume that the address stayed the same between Friday and Saturday. Checking the .kismet files from Friday and Saturday, only five devices overlap between the tests, meaning that there are 27 unique devices found across the two days. This means that driving the route just once does not give a good grasp of the devices transmitting Zigbee signals. There also exists a possibility that some of the devices were not in use on Friday, that were on use on Saturday, and vice versa, although this most likely would not explain the discrepancy alone. However, a third test on a consecutive day, Sunday, will be done to further get information on how to best utilize the CC2531 Zigbee sniffer.

The third test drive was done the following day, Sunday, at approximately 1 pm. The number of unique devices found was 16, which is similar to previous days. However, of the 16 devices 11 were completely new 16-bit short network addresses, meaning that over the three days a total of 38 unique devices were found assuming the network addresses have not changed during the three-day period. This does mean that driving the same route has always given more unique addresses, which most likely means that even after three test drives there are probably still more devices to be found on the route. We can also determine from this that the most efficient way of gathering Zigbee signals is to drive the same route multiple times back-to-back, so that we maximize the chances of the devices transmitting signals as we are going past them. The number of times to drive the same route on the same day would probably be at the very least three but could potentially be higher. To test this, we are going to drive the same route five times on a singular day to determine the amount of times we should drive the same route in order to maximize the amount of devices we find in a reasonable manner. Although there exists a possibility that even after driving the same route ten, or even twenty times, we would still find new devices. However, driving the same route that many times would not be practical, as the routes are often somewhat long, so we will be driving the same route five times and check the results.

The fourth test drive was executed the following Tuesday between 4.30 pm and 6 pm. The test route was driven five times in order to find an efficient way to gather data. The route was driven similarly all five times so that the results would not be skewed because of a driving style or other factors. The hypothesis was that there would be new devices found every time the route was driven as the Zigbee and/or 802.15.4 devices transmit infrequently, and results do seem to confirm this. 19 unique devices were found on the first lap, 15 new unique devices on the second lap, 12 new unique devices on the third lap, 3 new unique devices on the fourth lap, and 3 new unique devices on the fifth lap. Overall, the

number of unique devices found was 52. As can be seen, after the third lap the number of new devices found dropped drastically. Based on this result, it would be reasonable to deduce that driving a route three times would provide the greatest results when considering time spent versus amount of devices found. The goal is not to find every device, but to get enough data in order to make conclusions on the security of found devices.

5.1 Analysis and results from the first test round

After the initial test drive the collected results were further analyzed. As the fourth test drive was the most comprehensive of the test drives, the result of that test drive will be analyzed first. Out of the 52 devices found, 3 had 64-bit addresses, and 49 devices had 16-bit short network addresses, which means that it is hard to gather information from the networks addresses alone. This does however confirm the result of study by Gvozdenovic et al. [32] mentioned in section 2.3.2, where one of the findings was that finding Zigbee devices by their short address is easier.

Traditionally it is possible to determine the manufacturer from a 48-bit MAC address, as the first three hexadecimal pairs determine the manufacturer, also known as Organizationally Unique Identifier (OUI). However, the addresses that we found on our fourth test drive are 64-bit addresses that are specified by the 802.15.4 standard, so figuring out information from just the MAC address alone can prove to be challenging. Inputting the three 64-bit addresses to Wiresharks' OUI lookup tool [56] does not provide information about these 64-bit addresses, as the tool states that the manufacturer cannot be found. The short 16-bit addresses naturally will not have an OUI included in its unique identifier.

However, by opening the .kismet file in sqlite3 and downloading the individual device information files we can find info on some of the devices manufacturers, although it is at this point unclear on what information it uses to figure out the manufacturer. However, according to the device information files we can determine that of the 52 devices 5 have the manufacturer information, and 47 do not. As the difference between the numbers is so big, it can probably be determined that it will not be easy to figure out the manufacturer of all the devices from the .kismet files. However, we can try to find similarities in the five devices that have the manufacturer information. The first common factor is that they all have short 16-bit network addresses that start with "00:", and the second octet is either "01", "02", "03", and "0C". However, all of these five devices have a different manufacturer. As Zigbee devices mostly just use the short network addresses to communicate with each other and the addresses may change, using these addresses in order to figure out the manufacturer is not possible in most situations. However, it does seem that certain manufactures might use a certain type of a starting octet or ending octet in order to distinguish their devices from others, but these manufactures do not have public information available on this, so confirming the hypothesis is difficult. It would be beneficial to know the manufacturers of

the devices for statistical purposes, but as of now the gathered data does not seem to have that information.

The 2.4 GHz band of Zigbee ranges from 2400 MHz to 2480 MHz and has 16 channels from 11 to 26 that are approximately 5 MHz apart from each other. This range is also used by Wi-Fi, so implementing Zigbee and Wi-Fi devices in same area might require manual handling of channels in order to avoid overlaps. Results of the fourth test drive show the distribution of used channels in Figure 26. As can be seen from the distribution, the devices seem to favor the channels 11, 15, 20, and 25. The most likely reason for this is the fact that Wi-Fi has 14 channels between 2400 MHz and 2495 MHz, with each channel having approximately 20 MHz of space, which means the Wi-Fi channels overlap between each other. However, the most used Wi-Fi channels are channels 1, 6, and 11, which correspond to ranges 2401-2423 MHz, 2426-2448 MHz, and 2451-2473 MHz [57]. The overspill effect of this Wi-Fi channel distribution on Zigbee is that the little gaps between these Wi-Fi channels can be used effectively by Zigbee in order to avoid interference. Zigbee's channel 11 operates around 2400 MHz, 15 operates around 2425 MHz, 20 operates around 2450 MHz, and 25 operates around 2475 MHz. When compared to Wi-Fi channels 1, 6, and 11, we can clearly see from Figure 26 that Zigbee favors using channels that are below, between, or above the frequency ranges of the Wi-Fi channels in order to avoid interference.

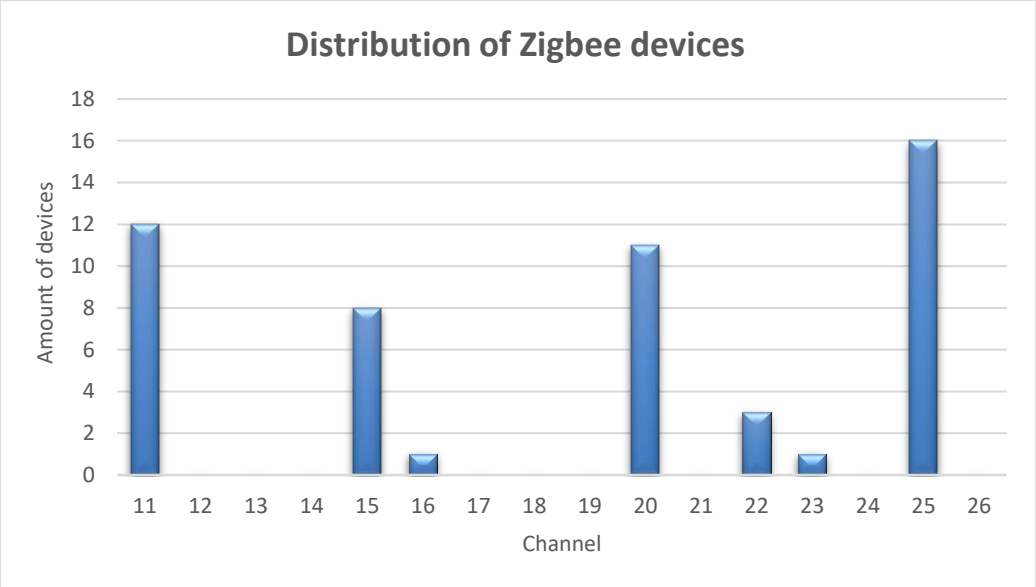

Figure 26. Distribution of Zigbee devices on the fourth test drive.

It is not apparent from the fourth drives .kismet file what kind of features, security or otherwise, these devices might have. However, we can convert the .kismet file of the fourth test drive, as illustrated in Figure 25, into a .pcapng file and analyze it further in Wireshark to get a more in-depth look at the packets.

Here we encounter a challenge. Opening the fourth test drives .pcapng file in Wireshark does not correctly show the packet contents of the Zigbee communications. Every packet is marked with "[Malformed Packet]" -tag, and the packets themselves are missing data. The same repeats in other .pcapng files taken with the same CC2531 USB dongle. The problem could be a multitude of things. According to Wireshark documentation [58] the [Malformed Packet] -tag could be a result of a wrong dissector, a buggy dissector, packet not being reassembled, or packet is malformed. As for the dissector related problems, it is very unlikely that Wireshark would be using the wrong dissector, as the problem is in every single packet. Also, using different dissectors and disabling/enabling protocols has no effect on the outcome. The dissector in Wireshark for Zigbee could potentially be buggy or incomplete, but for a protocol this well-known it would be quite unusual. As for the packet related problems, the packet not being reassembled is related to the packet length, where in some cases the packet length is longer than a single frame and thus is not reassembled. This should not be the problem in this case, as the captured packets length is a maximum of 124 bytes, or 992 bits, which should not be enough to trigger such an issue. Packets being malformed is probably always a possibility, meaning that the packet or part of the packet is not following the protocol specifications, but even this seems unlikely as every single packet in the .pcapng file is malformed, not just a few.

5.2 Analysis and results with the improved system

There are a few things first that we can do to troubleshoot this. If the problem lies with the CC2531 USB dongle, we can try a backup CC2531 chipset in the case that the first CC2531 dongle has a hardware or a software failure that prevents it from working as intended. We could also try logging the data directly into PCAP-NG format in case the problem lies in converting the .kismet files to .pcapng files.

Because both of the abovementioned ways to troubleshoot are quite a low barrier to try, we will be trying them both. We will prepare another CC2531, let us name it CC2531-2, for another test drive, and have it log into .kismet file format, which will be later converted into .pcapng. We will also be using the original CC2531 to directly log into .pcapng format in addition to .kismet file format to resolve whether the in-built converter is working correctly in Kismet. The CC2531-2 will be prepared the same way as in section 4.2 the first CC2531 was prepared. Having the original CC2531 log into .pcapng in addition to .kismet the "kismet_logging.conf" configuration file must be altered to support this as shown in Figure 27.

```
# By default, Kismet only enabled the unified 'kismet' log; the pcapng option is
# provided for special configurations as a legacy fallback mode.
log_types=kismet,pcapng
```

Figure 27. Adding .pcapng filetype to output log types.

First, a test drive with the original CC2531 USB dongle is executed where the Kismet software outputs the .pcapng file directly alongside the .kismet file. This fifth test drive took place in a small city on the

western coast of Finland and 20 different devices were found on the drive. Unfortunately, the results of this fifth test drive were similar when the .pcapng file was viewed in Wireshark. Every packet had the "[Malformed packet]" -tag, and the packets were missing quite a lot of details, including everything related to security of the packets. The packets had one extra feature which the previous .pcapng files did not have after conversion from .kismet files, and that is the "Hash algorithm" value. The hash used to ensure data integrity in the packets is Cyclic Redundancy Check 32 (CRC32).

Logging the results directly into a .pcapng file did not work to solve the issue, so another CC2531 USB dongle is prepared in the case of the original chipset being faulty. If the CC2531-2 dongle has similar issues the problem may be in the firmware. The CC2531-2 dongle will be prepared similarly to the first one, meaning the chipset will be flashed with the same sniffer firmware. The sixth test drive will be executed similarly to the fifth test drive. The city is the same at western coast of Finland as that of the fifth test drive. The freshly flashed CC2531-2 USB dongle was able to find fourteen devices, but the same problem persisted as with the original CC2531, as the .pcapng only showed malformed packets in Wireshark.

The issue was reported to the Kismet developers in order to solve the problem. The contact was made via the developers Discord server. After the problem was explained to the developers of Kismet, they found a bug with the CC2531 USB dongle when it was used with Kismet. The bug was fixed swiftly on 17th of April 2024, and the fix was committed to Kismets' GitHub page [59]. After the fix the current version of Kismet was removed from the laptop used for testing and the newest version was downloaded similarly to how it was done in section 4.3 (Figure 9) from the repository, however this time Kismet was pulled from the current git build in which the fix was in effect, and not the latest release version as it was originally installed in section 4.3.

After this, a seventh test drive took place similarly to the fifth and sixth test drives in the same location and the same route. This time the original CC2531 was used to gather data, although the CC2531-2 was brought alongside just in case. This time the CC2531 was able to find twelve devices, and after the bugfix to the Kismet software the data was finally portrayed in Wireshark correctly from the .pcapng file. This time there were zero malformed packets and Wireshark was able to identify the communications as Zigbee communications, and Wireshark was also able to view the packets in detail. Next up is analyzing the security features of Zigbee devices via this wardriving method, but as the seventh drives' results are very limited, no further analysis is made on the seventh drives' results. The eighth test drive will take place in a larger city so that enough data can be collected and analyzed.

The eighth drive took place in a larger city in southwest Finland. As we now know the results will be fully functional, we are looking to gather a larger dataset to have more to analyze and perhaps to see

patterns in the security features of the devices. Although we have found out that the optimal way to gather data in a route would be to drive the route at least three times to gather most if not all the devices in that routes vicinity, this time a route will be driven just once. The goal is not identifying every Zigbee device in the route, but rather to have a certain amount of data to work with. In the future a more comprehensive wardriving undertaking could be attempted to figure out Zigbee activity and security levels in certain areas, but for the purposes of this thesis the methodology for analyzing the results is of more interest.

The number of devices found in the eighth test drive is 61. However, it was observed during driving that some short network addresses were used in multiple locations. For example, the short network address of “00:01” was observed to be transmitting in at least three different locations. Kismet does not differentiate between the devices with the same short network address in the web GUI, so all devices with the short network address of “00:01” will be output in the same line in the web GUI. However, it is possible to differentiate the devices with the same short network address by looking at the timeline and the Personal Area Network (PAN) addressing. Devices with the same network address but different destination PAN address are in different networks, therefore different devices. The number of different network addresses found were 61 in Zigbee communications, but after analyzing the .pcapng file in Wireshark, it was possible to differentiate 72 devices. The amount of packets sent in in total between these 72 devices was 101, of which, according to Wireshark, 73 are Zigbee data packets, 27 are Zigbee command packets, and 1 is Zigbee APS acknowledgment (ACK) packet.

Because Wireshark displays the extended source address alongside the short network address, it is possible to determine the manufacturer from this, which was not possible from the .kismet files previously in section 5.1 All of the 72 devices displayed the manufacturer information in Wireshark. Four different manufacturers could be found, of which an overwhelming majority, 66 out of 72, devices were from a single manufacturer. The manufacturer in question is a large manufacturer, and the 66 Zigbee devices were their lighting products. This is rather unfortunate in terms of analysis, as the dataset is quite homogenous. This is, however, quite interesting, given that the route driven in the city was nine kilometers long, with only one kilometer of the drive overlapping, and because the city is densely populated. The amount of devices seems rather low for the area driven, but it is hard to determine how popular Zigbee devices are in Finland, and what kind of Zigbee devices users are interested in. The highest activity levels in Zigbee communications on the drive was near apartment buildings, so the end-users are most likely private individuals rather than public entities.

The channels used are quite similar to the fourth test drive’s, meaning the Zigbee devices prefer channels that have less interference with other wireless communications such as Wi-Fi. This means channels 11, 15, 20, and 25, although both the upper and lower end of channels are more prevalent in this test drive,

which could be due to certain manufacturers' devices preferring certain channels. Figure 28 shows the channel distribution of the eighth test drive.

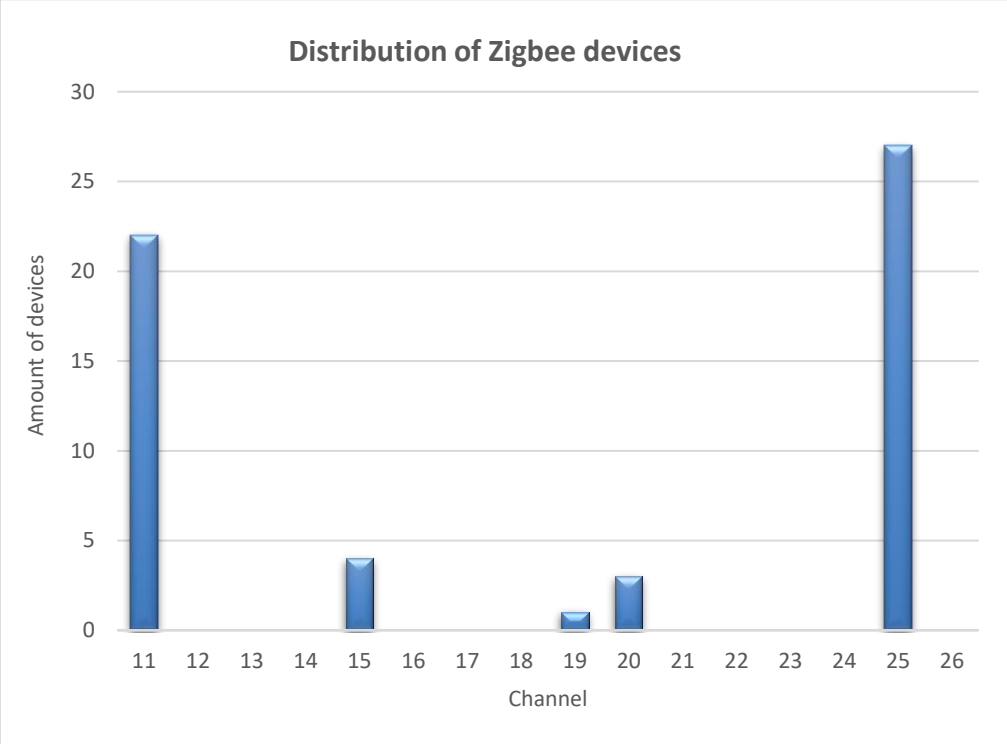

Figure 28. Distribution of Zigbee devices on the eighth test drive.

Next challenge is to figure out what the packet contents mean in terms of security. Some of the security features were already discussed in section 2.4.1, for example the use of network keys and link keys in a Zigbee network and the encryption protocol used. The communications between Zigbee devices are secured via network keys and link keys. As every device in a Zigbee network uses the network key to both encrypt and decrypt network messages, it is essential that the network key is secured. The only exception to a device having a network key is during joining, in which state the joining device will send message through the parent device until the new device is fully authenticated. The link keys are either pre-configured on the devices or transported via a key-transport service in the network. Depending on the manufacturer the pre-configured link key can be quite easily found on the internet and used to initiate the join process to the network. The trust center decides which type of a link key or pre-shared installation code it requires in the initialization process to the network. In the eighth test drive no network keys or link keys were in plain text in the packets, and all of the 72 devices communications were encrypted, so the payloads could not be viewed. Some global default link keys could be used to try and join a certain network, and therefore initiate the key establishment process with the trust center, assuming the trust center accepts global default link keys, and in this way try to attain the network key. However, this will not be tested in this thesis and is only mentioned for academic purposes.

In Figure 29 we can see the Zigbee network layer data and the Zigbee security header. Figure 29 represents a typical packet from the eighth test drive. The packets also consisted of the IEEE 802.15.4

TAP and the IEEE 802.15.4 frame control field. From the Zigbee network layer data we can see a field "Security: True". This means that the Zigbee security header is enabled, and has the following fields: MIC, security control, frame counter, source, and key sequence number [35, p. 455].

Figure 29. Zigbee network layer data and Zigbee security header.

Zigbee does not encrypt the network header nor security header, which means they are sent in cleartext as seen from Figure 29. Zigbee does, however, authenticate both of these by using the AES-128 to create a hash of the entire network part of the packet. This hash is called MIC and is used to ensure the network portion of the packet has not been modified in any way, ensuring the integrity of the package. Alterations to the MIC make the receiving device discard the message. The network payload is normally fully authenticated and encrypted [60]. The packets sent by the 72 devices found in the eighth test drive had their network payloads fully encrypted.

The security control field shows the fields security level, key identifier, and extended nonce. The security level sub-field indicates the security level for this security header, which in this case is 0x0, or 0x00, meaning that this header is not encrypted, and the frame integrity check is not on. The security level goes from 0x00 through 0x07 with increased security levels the higher you go in numbers. However, this result is at odds with the fact that the security header in Figure 29 does indeed have a 32-bit MIC. If the security header is not encrypted and the MIC is 32-bit long, the security level should be 0x01, not 0x00 [35, p. 456]. The reason for this is not immediately obvious, but the same repeats in all the Zigbee packets of the eighth test drive. The key identifier sub-field identifies the key-type that is used to encrypt the frame, in this case the network payload. In Figure 29 the key identifier field is 0x1, or 0x01, which means that the key used in encryption is the network key. A value of 0x00 would be a data key, a value of 0x02 would be a key-transport key, link key, and 0x03 would be a key-load key [35, p. 456]. All of the packets in the eighth test drive were encrypted via the network key. This seems to be

the de facto way of encrypting the network payload in Zigbee communications. The extended nonce sub-field is set to 1 if the sender address is present in the security address. This was set to 1 in all of the packets in the eighth test drive.

The frame counter field represents each device's sent frames. A device in the network counts their own sent frames, as well as the other devices' frame counters in the same network. Every time a device in the network sends a packet, the frame counter increases in increments. A receiving device compares the received frame counter values, and if it has not increased since last time, the device discards the whole packet. The frame counter is used to protect against replay attacks in which an attacker would imitate a legitimate party to manipulate the traffic whilst pretending to be the correct recipient or sender [60]. The frame counter was present in all the packets in the eighth test drive.

The source address field, or extended source, is present when the extended nonce sub-field is set to 1. The extended 64-bit address responsible for securing the frame is shown in this field [35, p. 457]. The extended source address was visible in all the packets in the eighth test drive.

The key sequence number field is only present when the frame is secured by the network key, so when the key identifier subfield of the security control field is 0x1. Whenever the network key is updated, the sequence number increases in increments [60]. However, each of the packets in the eighth test drive had the key sequence number of 0, which could mean that the network key has never been updated in those Zigbee networks. This, however, cannot be proven to certainty, but that is how it seems. If the network keys in the Zigbee networks have indeed stayed the same for a longer period of time, the probability of a malicious attacker attaining and abusing the network key increases as time goes on.

Some security features that Zigbee supports but were not visible in the eighth test drives' packets, could be implemented to improve the security of those devices. It seems that none of the Zigbee networks found have changed their network key from the original network key. This could definitely be improved with runtime key updates, where the trust center periodically generates a new network key and distributes it to other devices in the network by encrypting it with the old network key. This would also put the frame counters back to zero, as the frame counter can only hold 32-bits. The frame counter cannot wrap back to zero, so depending on how many packets the device sends, a network key update might be necessary from this angle as well. It should also be mentioned, again, that some Zigbee trust centers will accept default link keys when a device attempts to join the network, thus initializing the key establishment process for a potentially malicious entity.

The key establishment and exchange are essential for Zigbee networks to stay secure. If an attacker is able to get hold of the network key for the Zigbee network, the whole network becomes compromised.

Vulnerabilities to key processes are a top priority for Zigbee networks and further research on that topic is encouraged.

The results of the eighth test drive security-wise seem to be overall quite okay, although some abovementioned things could either be improved upon or implemented. However, the fact that 66 of 72 devices were from the same manufacturer, and even similar type of product, means that the homogeneity of the dataset might skew the overall Zigbee security landscape in this sampling, meaning a more extensive wardriving research would need to be accomplished with preferably hundreds, if not thousands, of devices found in an area with many different types of Zigbee devices.

5.3 Integration to the WLAN scanning system

Including the Zigbee sniffer into the wardriving system of Lindroos is the natural next step for this thesis. The aim is to have both the Wi-Fi sniffer and the Zigbee sniffer working simultaneously on a single device, outputting both results. Alternatively, we could have two different devices separately capturing these two traffic types.

The requirements for running both the Wi-Fi and Zigbee sniffing tools simultaneously in a possible implementation is having a laptop running Linux either via virtual machine or as the main operating system, at least two USB ports to support both the WLAN GPS and the Zigbee USB dongle, and the Kismet software running on the Linux operating system. Depending on availability of devices, it could be easier to separate the two wireless feeds into two different physical systems. Lack of available USB ports on laptops might also be a problem, especially if more than two USB ports are needed for any other external devices. However, it should be possible to separate the two feeds in Kismet and to just use one laptop, given that the laptop has all the pre-requisites mentioned above. This way just one system needs to be used.

If one system is used with multiple data sources, the output can be harder to read, assuming Kismet does not separate the two different feeds into two different logs. It is not possible to test this as of the writing of this thesis, but assuming Kismet cannot separate the logs, there is an option in Wireshark to separate the results in the .pcapng file via protocol type to make it easier to read. It is also possible to use filters in Wireshark to just view one protocols' communication, so having two different feeds outputting results, even if they are output to same .pcapng file, should not be a significant problem.

6 Conclusion

The goal of this thesis was to build a wireless Zigbee scanning system and to use the system to gather data which can be further analyzed. The result was a Zigbee scanning system that is capable of passively listening to nearby wireless Zigbee communications and log the gathered data into a human readable format.

The initial planning for the Zigbee scanning system started with picking suitable hardware and software for the project. The devices were obtained and the hardware and software were configured for the purposes of passively scanning nearby Zigbee communications. After the initial configurations the system was tested and troubleshot, and subsequently improved. After the first test round the improved system was used for further testing and analysis.

The number of devices transmitting Zigbee signals found in these tests were surprisingly low, but the routes driven were also quite short as the longest test drive was only nine kilometers long. Unsurprisingly the density of the population correlated with the amount of devices found. It was also found that Zigbee devices generally prefer to use channels in the 2.4 GHz band that are not preferred by other wireless communication protocols.

Cyber security features of the devices were found and discussed. The eighth, and last, test drive was the most comprehensive of the test drives as the drive took place in a densely populated city in Southwest Finland and the system was fully operational by the eighth drive. From the eighth test drive it was possible to differentiate 72 different Zigbee devices from the log files. The log files were analyzed for their cyber security features. The conclusion from the log files was that all of the devices implemented had at least some security features, although the dataset was quite homogeneous with a heavy skew towards a single manufacturer.

This thesis provides an established way of building a Zigbee scanning system for security professionals to use. The products and devices used in creation of Zigbee scanning system can be changed to other hardware and software and other wireless communication protocols, or be exactly replicated, but the idea behind is the same; to provide a non-theoretical way of gathering and analyzing data, and thus be able to secure IoT networks better and to potentially create higher quality and safer IoT products. Although the results of this thesis were not overly alarming, more research in both Zigbee and other IoT protocols are required and encouraged.

Similar implementations for other wireless communication protocols are definitely wanted in the future, as many of the wireless communication protocols operate independently of each other and therefore the tools and expertise required change from protocol to protocol.

The future of IoT and Zigbee require the security professionals all around the world to be equipped with tools and knowledge on the matter at hand, and this thesis provides at least one tool to the arsenal. The rapidly increasing amount of devices that are connected to the internet can be both exciting and alarming, but with correct tools and skillsets these challenges can be tackled.

7 References

[1] ENISA. “Internet of Things (IoT).” ENISA. Accessed: Jan. 2023. [Online.] Available: https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/iot

[2] M. Ghobakhloo, “Industry 4.0, digitization, and opportunities for sustainability,” *Journal of Cleaner Production,* vol. 252, Apr. 2020, Art. no. 119869.

[3] S. Sinha. “State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally.” IoT Analytics. Accessed: Apr. 2024. [Online.] Available: https://iot-analytics.com/number-connected-iot-devices/

[4] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum and N. Ghani, “Demystifying IoT Security: An Exhaustive Survey on IoT Vulnerabilities and a First Empirical Look on Internet-Scale IoT Exploitations,” *IEEE Communications Surveys & Tutorials,* vol. 21, no. 3, pp. 2702-2733, Apr. 2019.

[5] European Commission. “Cyber Resilience Act.” Policy and legislation. Accessed: Jan. 2023. [Online.] Available: https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act

[6] Connectivity Standards Alliance. “The Full-Stack Solution for All Smart Devices.” CSA. Accessed: Apr. 2024. [Online.] Available: https://csa-iot.org/all-solutions/zigbee/

[7] *IEEE Standard for Low-Rate Wireless Networks*, IEEE Std 802.15.4-2020, IEEE, New York, NY, USA, July 23 2020. [Online]. Available: https://standards.ieee.org/ieee/802.15.4/7029/

[8] S. Lindroos, A. Hakkala and S. Virtanen, “A systematic methodology for continuous WLAN abundance and security analysis,” *Computer Networks,* vol. 197, Oct. 2021, Art. no. 108359.

[9] S. Lindroos, A. Hakkala and S. Virtanen, “The COVID-19 pandemic and remote working did not improve WLAN security,” *Procedia Computer Science,* vol. 201, pp. 158-165, 2022.

[10] S. Lindroos, A. Hakkala and S. Virtanen, “Battle of the bands: a long-term analysis of frequency band and channel distribution development in WLANs,” *IEEE Access,* vol. 10, pp. 61463-61471, Jan. 2022.

[11] N. Sharma, M. Shamkuwar and I. Singh, “The History, Present and Future with IoT,” in *Internet of Things and Big Data Analytics for Smart Generation*. Cham, Switzerland: Springer, 2019, pp. 27-52.

[12] Internet Engineering Task Force. “The Internet of Things.” IETF. Accessed: Jan. 2023. [Online.] Available: https://www.ietf.org/technologies/iot/

[13] *Architectural Considerations in Smart Object Networking*, RFC 7452, Internet Architecture Board, Mar. 2015. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7452

[14] Oxford Learner's Dictionary. “IoT.” Accessed: Jan. 2023. [Online.] Available: https://www.oxfordlearnersdictionaries.com/definition/english/iot?q=iot

[15] O. Buxton. “What Is the Mirai Botnet?” Avast. Accessed: Feb. 2023. [Online.] Available: https://www.avast.com/c-mirai

[16] ENISA. “ENISA Threat Landscape 2022.” ENISA. Accessed: Feb. 2023. [Online.] Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2022

[17] ENISA. “ENISA Threat Landscape 2023.” ENISA. Accessed: Feb. 2024. [Online.] Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

[18] Microsoft Corporation. “Microsoft Digital Defense Report 2023.” Microsoft. Accessed: Feb. 2024. [Online.] Available: https://www.microsoft.com/en-us/security/security-insider/microsoft-digital-defense-report-2023

[19] R. Scammell. “Mikko Hyppönen: Smart devices are “IT asbestos”.” Verdict. Accessed: Sept. 2023. [Online.] Available: https://www.verdict.co.uk/mikko-hypponen-smart-devices-it-asbestos/?cf-view&cf-closed

[20] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen and B. Stiller, “Landscape of IoT security,” *Computer Science Review,* vol. 44, May 2022, Art. no. 100467.

[21] European Parliament, Council of the European Union. “Regulation (EU) 2019/881 of the European Parliament and of the Council.” EUR-Lex. Accessed: Sept. 2023. [Online.] Available: https://eur-lex.europa.eu/eli/reg/2019/881/oj

[22] ENISA. “Learn more about EU Cybersecurity Certification.” ENISA. Accessed: Sept. 2023. [Online.] Available: https://www.enisa.europa.eu/topics/standards/certification/eu-cybersecurity-certification-faq

[23] European Parliament, Council of the European Union. “Regulation (EU) 2019/1020 of the European Parliament and of the Council.” EUR-Lex. Accessed: Sept. 2023. [Online.] Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:32019R1020

[24] European Commission. “Cyber Resilience Act.” Policy and legislation. Accessed: Sept. 2023. [Online.] Available: https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act

[25] European Commission. “Cyber Resilience Act - Impact assessment.” Policy and legislation. Accessed: Sept. 2023. [Online.] Available: https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act-impact-assessment

[26] National Cyber Security Centre of Finland. “Cybersecurity Label.” NCSC-FI. Accessed: Sept. 2023. [Online.] Available: https://www.tietoturvamerkki.fi/en

[27] *Manufacturer Usage Description Specification*, RFC 8520, Internet Engineering Task Force, Mar. 2019. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8520.html

[28] National Institute of Standards and Technology. “MUD Related Resources.” NIST. Accessed: Sept. 2023. [Online.] Available: https://www.nccoe.nist.gov/mud-related-resources

[29] National Institute of Standards and Technology. “NIST Cybersecurity for IoT Program.” NIST. Accessed: Apr. 2024. [Online.] Available: https://www.nist.gov/itl/applied-cybersecurity/nist-cybersecurity-iot-program/publications

[30] Global System for Mobile Communications Association. “GSMA IoT Security Guidelines and Assessment.” GSMA. Accessed: Sept. 2023. [Online.] Available: https://www.gsma.com/iot/iot-security/iot-security-guidelines/

[31] G. Lyon. “Port Scanning Techniques.” Nmap. Accessed: Apr. 2024. [Online.] Available: https://nmap.org/book/man-port-scanning-techniques.html

[32] S. Gvozdenovic, J. K. Becker, J. Mikulskis and D. Starobinski, “IoT-Scan: Network Reconnaissance for Internet of Things,” *IEEE Internet of Things Journal,* vol. 11, no. 8, pp. 13091-13107, Apr. 2023.

[33] Microsoft Corporation. “IoT technologies and protocols.” Microsoft. Accessed: Sept. 2023. [Online.] Available: https://azure.microsoft.com/en-us/solutions/iot/iot-technology-protocols/

[34] D. Gislason, "Zigbee Wireless Networking". Burlington, Massachusetts, MA, USA: Elsevier, 2008.

[35] Zigbee Alliance. “Zigbee Specification Revision 22.” Zigbee specification. Accessed: Sept. 2023. [Online.] Available: https://csa-iot.org/wp-content/uploads/2022/01/docs-05-3474-22-0csg-zigbee-specification-1.pdf

[36] Connectivity Standards Alliance. “Zigbee FAQ.” CSA. Accessed: Oct. 2023. [Online.] Available: https://csa-iot.org/all-solutions/zigbee/zigbee-faq/

[37] Connectivity Standards Alliance. “Zigbee Direct FAQ.” CSA. Accessed: Oct. 2023. [Online.] Available: https://csa-iot.org/all-solutions/zigbee/zigbee-direct-faq/

[38] X. Fan, F. Susan, W. Long and S. Li. “Security Analysis of Zigbee.” Semantic scholar. Accessed: Oct. 2023. [Online.] Available: https://www.semanticscholar.org/paper/Security-Analysis-of-Zigbee-Fan-Susan/3d1d5a51d05cde08b6e52afd5bd7bc325b487a10

[39] J. Li, X. Zhu, N. Tang and J. Sui, “Study on ZigBee network architecture and routing algorithm,” in *2010 2nd International Conference on Signal Processing Systems*, Dalian, China, 2010.

[40] S. Khanji, F. Iqbal and P. Hung, “ZigBee Security Vulnerabilities: Exploration and Evaluating,” in *2019 10th International Conference on Information and Communication Systems (ICICS)*, Irbid, Jordan, 2019.

[41] OWASP Foundation. “OWASP Internet of Things Project.” OWASP wiki. Accessed: Nov. 2023. [Online.] Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project#tab=IoT_Top_10

[42] A. Zohourian, S. Dadkhah, E. C. P. Neto, H. Mahdikhani, P. K. Danso, H. Molyneaux and A. A. Ghorbani, “IoT Zigbee device security: A comprehensive review,” *Internet of Things,* vol. 22, 2023, Art. no. 100791.

[43] C. Hurley, "WarDriving drive, detect, defend: a guide to wireless security", Rockland, Massachusetts, MA, USA: Syngress Publishing Inc., 2004.

[44] European Parliament, Council of the European Union. “Regulation (EU) 2016/679 of the European Parliament and of the Council.” EUR-Lex. Accessed: Jan. 2024. [Online.] Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[45] Kismet. “Docs.” Kismet documentation. Accessed: Apr. 2024. [Online.] Available: https://www.kismetwireless.net/docs/

[46] Kismet. “Zigbee: TICC 2531.” Kismet documentation. Accessed: Apr. 2024. [Online.] Available: https://www.kismetwireless.net/docs/readme/datasources/zigbee-ticc2531/

[47] Zigbee2MQTT. “Flashing the CC2531 USB stick.” Zigbee2MQTT flashing instructions. Accessed: Apr. 2024. [Online.] Available: https://www.zigbee2mqtt.io/guide/adapters/flashing/flashing_the_cc2531.html

[48] Texas Instruments. “FLASH-PROGRAMMER.” TI flash programmer. Accessed: Apr. 2024. [Online.] Available: https://www.ti.com/tool/FLASH-PROGRAMMER

[49] Texas Instruments. “CC-DEBUGGER.” TI CC-debugger. Accessed: Apr. 2024. [Online.] Available: https://www.ti.com/tool/CC-DEBUGGER

[50] Texas Instruments. “PACKET-SNIFFER.” TI packet sniffer. Accessed: Apr. 2024. [Online.] Available: https://www.ti.com/tool/PACKET-SNIFFER

[51] Nordic Semiconductor ASA. “nRF Sniffer for 802.15.4.” Nordic Semiconductor documentation. Accessed: Oct. 2023. [Online.] Available: https://infocenter.nordicsemi.com/index.jsp?topic=%2Fug_sniffer_802154%2FUG%2Fsniffer_802154%2Fintro_802154.html

[52] Nordic Semiconductor ASA. “nRF Connect for Desktop.” nRF software. Accessed: Oct. 2023. [Online.] Available: https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop

[53] Nordic Semiconductor ASA. “nRF Sniffer for 802.15.4.” nRF sniffer. Accessed: Oct. 2023. [Online.] Available: https://github.com/NordicSemiconductor/nRF-Sniffer-for-802.15.4

[54] Kismet. “Zigbee - nRF 52840.” Kismet documentation. Accessed: Oct. 2023. [Online.] Available: https://www.kismetwireless.net/docs/readme/datasources/zigbee-nrf52840/

[55] Linux Kernel Organization. “Documentation.” Linux admin guide. Accessed: Oct. 2023. [Online.] Available: https://www.kernel.org/doc/Documentation/admin-guide/devices.txt

[56] Wireshark. “OUI Lookup Tool.” Wireshark OUI lookup tool. Accessed: Sept. 2023. [Online.] Available: https://www.wireshark.org/tools/oui-lookup.html

[57] *802.11-2020 - IEEE Standard for Information Technology--Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks--Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 802.11.-2020, IEEE, Feb. 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9363693

[58] Wireshark. “Wireshark User’s Guide - Appendix A. Wireshark Messages.” Wireshark documentation. Accessed: Apr. 2024. [Online.] Available: https://www.wireshark.org/docs/wsug_html_chunked/AppMessages.html

[59] Kismet. “Kismetwireless GitHub.” Kismet GitHub page. Accessed: Apr. 2024. [Online.] Available: https://github.com/kismetwireless/kismet

[60] Silicon labs. “Zigbee Security - AN1233.” Knowledge article. Accessed: Apr. 2024. S [Online.] Available: https://community.silabs.com/s/article/zigbee-security?language=en_US