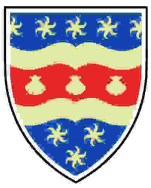
2020

# Honeypot for Wireless Sensor Networks

Markert, Jurgen



UNIVERSITY OF
PLYMOUTH

# Honeypot for Wireless Sensor Networks

by

Jürgen Markert

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of

**DOCTOR OF PHILOSOPHY**

School of Engineering, Computing and Mathematics

April 2020

# Acknowledgements

THESE are arguable the most important points of the thesis. I would like to thank everybody who helped with ideas and comments, as well as my family, my wife, and my children, for their moral support.

Thanks especially to my supervisors Prof. Michael Massoth, Dr Klaus-Peter Fischer-Hellmann and Prof. Steven Furnell for valuable comments. Thanks also to the Graduate School Darmstadt and especially Dr Janina Fengel for further advice and guidance.

Thanks finally to the University of Plymouth and especially the Centre for Security, Communications and Network Research (CSCAN) for making this work possible.

# Abstract

**Honeypot for Wireless Sensor Networks**
Jürgen Markert

PEOPLE have understood that computer systems need safeguarding and require knowledge of security principles for their protection. While this has led to solutions for system components such as malware-protection, firewalls and intrusion detection systems, the ubiquitous usage of tiny microcomputers appeared at the same time. A new interconnectivity is on the rise in our lives. Things become “smart” and increasingly build new networks of devices.

In this context the wireless sensor networks here interact with users and also, vice versa as well; unprivileged users able to interact with the wireless sensor network may harm the privileged user as a result. The problem that needs to be solved consists of possible harm that may be caused by an unprivileged user interacting with the wireless sensor network of a privileged user and may come via an attack vector targeting a vulnerability that may take as long as it is needed and the detection of such mal-behaviour can only be done if a sensing component is implemented as a kind of tool detecting the status of the attacked wireless sensor network component and monitors this problem happening as an event that needs to be researched further on. Innovation in attack detection comprehension is the key aspect of this work, because it was found to be a set of hitherto not combined aspects, mechanisms, drafts and sketches, lacking a central combined outcome. Therefore the contribution of this thesis consists in a span of topics starting with a summary of attacks, possible countermeasures and a sketch of the outcome to the design and implementation of a viable product, concluding in an outlook at possible further work.

The chosen path for the work in this research was experimental prototype construction following an established research method that first highlights the analysis of attack vectors to the system component and then evaluates the possibilities in order to improve said method. This led to a concept well known in common large-scale computer science systems, called a honeypot. Its common definitions and setups were analysed and the concept translation to the wireless sensor network domain was evaluated. Then the prototype was designed and implemented. This was done by following the approach set by the science of cybersecurity, which states that the results of experiments and prototypes lead to improving knowledge intentionally for re-use.

# Authors declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Doctoral College Quality Sub-Committee. Work submitted for this research degree at the University of Plymouth has not formed part of any other degree either at the University of Plymouth or at another establishment. Relevant scientific seminars and conferences were regularly attended at which work was often presented. Four papers and an additional internal workshop paper have been accepted for publication in refereed proceedings.

Word count for the main body of this thesis: **56865**

**Signed:** _____

**Date:** _____

**Publications:**

**Markert, J. & Massoth, M. & Fischer-Hellmann, K.P. & Furnell, S.M.** *Attacks to ZigBee and Wireless Sensor Networks - Honeypots for Detection and Response.* In Proceedings of the Collaborative European Research Conference (CERC) 2012, 29:8

**Markert, J. & Massoth, M.** *Honeypot Framework for Wireless Sensor Networks.* In Proceedings of International Conference on Advances in Mobile Computing & Multimedia (MoMM) 2013, 217:7

**Markert, J. & Massoth, M.** *Honeypot Effectiveness in Different Categories of Attacks on Wireless Sensor Networks.* In Proceedings of Database and Expert Systems Applications (DEXA) 2014, 331:5

**Markert, J. & Massoth, M.** *Honeypot Wording and Definitions in Wireless Sensor Networks.* In Conference Proceedings Paper - Sensors and Applications (ECSA) 2015, 1:5

# Contents

# List of Figures

# List of Tables

# Glossary

| | |
|---|---|
| AES | Advanced Encryption Standard |
| API | Application programming interface |
| AVR | Microcontroller from Microchip AVR |
| BSI | Bundesamt fur Sicherheit in der Informationstechnik |
| CTR | Abbreviation for counter |
| FFD | Full function device |
| GHz | Gigahertz |
| GNU | GNU is not unix |
| IDE | Integrated development environement |
| IDS | Intrusion detection system |
| IEEE | Institute of Electrical and Electronics Engineers |
| IPS | Intrusion prevention system |
| ISM | Industry, scientific and medical |
| JTAG | Joint test action group |
| LTS | Long term support |
| MAC | Message authentication code |
| MHz | Megahertz |
| MITM | Man in the middle |
| OTAP | Over the air programming |
| RFD | Reduced function device |
| SDK | Software development kit |
| SDR | Software defined radio |
| SPI | Serial peripheral interface |
| SVM | Support vector machine |
| UML | Unified markup language |
| WSN | Wireless sensor network |

# Chapter 1

# Introduction

MOST people have heard of the Industrial Revolution. Where initially the power of humans and animals was supplemented by mechanics, and the steam engine provided necessary power and performance in the second phase of the industrial revolution, which is accompanied by the use of electricity, there was a specialisation and miniaturisation of production. Smaller plants had become possible, therefore pulleys, belts and bearings for the transfer of power in the production halls were no longer necessary. The advent of electronic circuits, coupled with the miniaturisation with transistors and later integrated circuits and then microchips allowed automation and control of the machines, which is generally referred to as the third industrial revolution.

The $4^{th}$ stage of the industrial revolution happens right now, where progressive networking of production is taking place. Private citizens furthermore implement networking of everyday objects in their smart homes.

Furthermore, in this fourth industrial revolution, control and operation by remote access to industrial plants may be accessible on the Internet. Besides that, the so-called smart home has become a reality in a growing number of implementations. A central aspect of this fourth industrial revolution is the networking of systems, which is possible in wired form, but is also increasingly implemented with wireless technology. Direct physical access to a network is no longer necessary when using radio technology, only the vicinity of the systems to each other, thus to be in a sufficient range for the desired interaction.

## 1.1 Background

So, the next industrial revolution will be the “Internet of Things” (IoT) (Dexheimer et al. 2014) with “Machine to Machine” communication (M2M) (Glanz & Jung 2010), “smart homes”, “smart buildings” or even “smart cities” (Sorensen 2009). The concept is deceptively simple: Everything can be connected to anything else. Anything can provide services and be monitored and controlled. This evolution is already laid out and well on its way. IPv6 and other technical prerequisites are already available to cope with the challenge of having more and more systems to be accessed, internationally and worldwide.

Yet, these networks do not necessarily have to be wire-bound. A lot of networking technologies have their focus on wireless communication channels. For example, many companies have worked together in the development of Bluetooth as a wireless standard (IEEE - Institute of Electrical and Electronics Engineers 2005).

ZigBee was likewise introduced and developed to be an ideal de facto standard for wireless sensor and control networks. It has been established over the course of the last ten years to address the expected need for measurement and control solutions using wireless communications in the low and medium range of thirty metres and beyond. ZigBee is an additional feature set to the IEEE 802.15.4 standard and defines additional layers on top of it (IEEE - Institute of Electrical and Electronics Engineers 2006) for wireless sensor networks (WSN). ZigBee offers manufacturers all over the world a reliable standard for the upcoming need for team play functionality. About a dozen manufacturing companies worked together on the ZigBee drafts in order to establish a common operational platform for future technologies using low power communications in star and mesh networks (ZigBee Alliance 2008). The current number of ZigBee standard contributing companies is given as 118 according to the 2017 ZigBee report (ZigBee Resource Guide 2017).

In 2006 the first ZigBee certified products were brought to market (ZigBee Alliance

2006). Such products have been present in home automation, in home entertainment and in buildings for monitoring purposes. These wireless nodes are all intended to communicate among each other directly or relayed over peered communication (ZigBee Alliance 2008).

Such products are for example:

- Control4 is a company selling ZigBee products for home entertainment and home control. The press release from Control4 in August 2009 stated that they have already sold 1 million ZigBee products worldwide (Control4 2009).
- The Eaton Corporation sells a product called Home Heartbeat. Customers control and monitor their home electronic equipment on the road by mobile smartphone (Eaton Corporation 2007). With ValveSense they provide a solution for power metering via ZigBee (Eaton Corporation 2012).
- Salus Controls manufactures radiator thermostats that are controlled remotely. Their products are certified by ZigBee (Salus Controls 2009).
- Siemens APOGEE Building Automation is a range of products for application in office buildings. Their products can also be monitored and controlled remotely via ZigBee (Siemens Industry Inc. 2007).
- Philips offers medical device manufacturers a solution for their patient monitoring systems. They developed the ZigBee Health Care standard. Manufacturers can use the design to develop products using the ZigBee network (Philips Applied Technologies 2010).
- Newcastle in Australia is the first Australian city getting a Smart Grid using ZigBee products. Electric Power consuming machines will be controlled via ZigBee to achieve steady power plant utilization (Smart Grid Australia 2011).

New ways of interaction become available through WSN, offering great advantages and a lot of potential to save energy and make life much more comfortable, but also requires

that underlying structures be robust, reliable, safe, and secure. The key aspects of network technologies are commonly the same; application developers are expected to protect communications in order to attain the information security core principles: confidentiality, integrity, and availability (Elahi & Gschwender 2009). These principles were already mentioned by (Clark & Wilson 1987) for computer security in defence.

## 1.2 Motivation

New communication technologies offer new opportunities for developers. Yet, they also pose new risks through potential attacks by an intruder. The purpose of this work is to present the current state of work in the area concerning the security of wireless sensor networks, furthermore, the improvements, resulting in and from the PhD research, will be illustrated.

This thesis examines safeguarding wireless sensor networks, showing possible solutions. It also covers a comprehensive overview of the threats, vulnerabilities, attacks and countermeasures. Furthermore, it surveys the applicability of techniques found in other wirebound and wireless domains to the domain of wireless sensor networks. Finally, it gives an introduction to the legal aspects of security solution research. The objectives are to find the need, to present the current state of the art in wireless sensor network security implementations and to present as a novel approach a honeypot for wireless sensor networks as a very systematic and scientific method. The contribution is twofold, as on the one side the thesis gives a broad overview on the wireless sensor network security aspects, as shown by the research community, and on the other side it presents possible novel improvements by introducing a honeypot for wireless sensor networks. The course of research showed that components in the wireless sensor networks should exist to monitor for attacks, while one attempt is known as an intrusion detection system, a honeypot system was found to be a feasible and suitable solution for wireless sensor networks.

Wireless sensor networks are a new communication technology and offer opportunities

not only for developers, but also emerge as a possible risk containing flaws for an attacker to focus on. The major findings from research on recent literature on the subject pertain to the improvement of the security features of these networks, based upon formal specification, simulation and tests.

Another tendency seen, pursues the development of methods to detect intrusions or intrusion attempts conducted by an attacker against the wireless sensor network. Thus, a further aspect of this work is the analysis of specific attacks and interceptions on wireless sensor networks with the intention of designing corresponding countermeasures. There is a trend in this field to develop methods for tracking intrusions and unauthorised access by an attacker, and for detecting attempted attacks against the stability of wireless sensor networks with intrusion detection systems. Very little research has been done so far in the development of so-called honeypots which offer analysis opportunities on specific attacks and interceptions of wireless sensor networks. These systems represent the current state of the art in the field of research on detecting attacks on wireless sensor networks.

The introduction of new technologies into daily life takes place progressively, however, this should not be done without a safety-critical review. The assumption that every technology is potentially vulnerable to flaws automatically applies to WSNs as well. They must be considered to be failure-prone. The task and the challenge in risk management in this case is to determine to what extent damage can happen in case of failure, and to project what could unintentionally happen if this technology gets used by a third party. This means exploring the possible failure modes and propagation of appropriate mitigation. It is important to ask what will happen and what results it might have, if this technique fails or for any reason does not behave as you would expect from it.

Such considerations are traditionally carried out in a risk analysis; for a smart home it would be considered, what would happen if a shutter or window, a garage or a door would be remotely configured different than normal. The users want reliability, they

must be able to rely on these sensors and actuators. To think of a situation in which a machine can severely get in conflict with the health of users is not too difficult. So it is a very important topic, which affects people in their private life, likely everyone who wants to use new gadgets from current technology in their own home; and also companies that use the technology already in their production.

At the concluding lines of the motivation an outlook to the state of the art attacks that are possible is shown here as such:

An attack to the light control protocol in wireless sensor networks has been shown to be used, and an example attack was executed via a flying drone containing a transmitter with a self executing attack example (Morgner et al. 2017). The authors used responsible disclosure to contact the companies and fixed firmware was released in due course. However, this is only solved when the components attacked are updated.

A further example as motivation is a self-propagating worm, replicating itself from one smart device to another (Ronen et al. 2018). The attack was not carried out completely by the researchers and was done only in a laboratory environment, where it was ensured that no outside components would be affected and spread this malware. But as a conclusion, such attack vectors do demonstrably exist, and also the danger of not-yet-installed, though available, patches is quite real.

The authors state in this example (Cesare 2014), that the smart home opens new attack vectors for burglars with technology hacking tools. They can subvert a burglar alarm system into not detecting anything and therefore not recording any information of the attack. According to the research above this is like operating in a stealth mode. The doors are openable and access is granted but not a single log that the door was opened is written. So for a summary, the burglar gets in, no alarm, no recording, no logging, just people to find out of the stolen goods by missing something when returning home. Where forensic experts at common homes could find evidence on the surveillance system and the lock cylinder, the situation at a smart home could get severe in

this case for the victim, they simply cannot give proof that a burglar has been there, which will come to be an interesting topic for victims and insurance companies alike.

## 1.3 Problem Definition

A technology is not a problem by itself, it is rendered to be a problem by improper use and faulty implementation. This is the same within WSN technology, the implementation and its use may differ from the intentions. Improper use of a WSN can result in various problems, although the WSN was set up to assist and help people. This also means that a modification of the technology by third parties is a problem, as this can affect the proper function. WSN are not designed to detect improper use, or learn to do it. Rather it is assumed, that once put into long term use a technology works as intended.

But in fact, experience shows, over and over again, with each new technology, that there are people who see it as a sporting challenge, as it were, to take over technology for their own purposes and profit. Furthermore, there are also criminals who want to use the function of the new technology for their own benefit and interests, if only to be enriched financially, for example, by theft.

The authors Islam et al. (2012) presented a paper for the discussion of wireless sensor networks in smart homes. They also focused on security aspects and reviewed possible attacks to the WSN in smart homes. They state, that “Although the existing security paradigms as outlined by the standards (for example, 802.15.4 and ZigBee) seem to be sufficient to counter against major attacks, they leave the design and implementation details to the users.” and further they state, that “traditional complex cryptographic algorithms (e.g., public key cryptography) that seem effective to handle most security issues may not be suitable for sensors because of their lesser power and processing capabilities.”

Although WSNs and ZigBee received little attention as to their security in the past years; they were always highly present as a revolution in everyone’s lives. Meanwhile

such applications and devices are available in many areas of life and therefore it has become part of many communications devices as shown in section 1.1. This means for now, among other things, that sometimes this standard is distributed throughout a whole city, as for instance all electricity meters are exchanged with smart meters and can be controlled via ZigBee.

## 1.4 Research Aims & Objectives

The intention and aim of the research is the generation of knowledge of security in the domain of wireless sensor networks, which is planned to be achieved by showing a relationship between wireless sensor networks and well-known WiFi networks in the sense of existing and theoretical attacks and countermeasures.

In the beginning of this work these pointsare there in the researchers statements.

- Review and summary of the usual strategies for detecting attacks on wireless sensor networks and presenting existing solutions, as well as highlighting promising strategies. This will be done in a way to construct a connection and comparison to the WiFi networks security. The outcome is an overview in the form of a table showing the relevant data.
- Perform literature review to understand trade-offs in wireless sensor networks security. Therefore, similarities and differences between wireless network types are to be collected for classification.
- For honeypots' common definitions and setups, perform an analysis and concept translation to the wireless sensor network domain evaluation, then design and implement a prototype. This includes the process to produce a roadmap, starting with the research methodology, the translation between domains, the development of strong connections between the idea of knowledge generation and the synthesis of a result.
- Prototype development to detect the proposed attacks as a model of complete-

ness by using determined scenarios. For this objective, the attacks are performed against a test-bed setup and the security mechanisms are evaluated by their performance in collecting and discerning between normal network traffic and security relevant events.

These are the main objectives that are going to be answered in this work. Outcomes on these points are going to be discussed in the following chapters and concluded in chapter 7. The outcome of the research is a contribution to the security knowledge for the domain of wireless sensor networks, namely in theoretical and also applied science; the solution is meant to be a proof that the way taken is the correct one, and the result is also showing a correlation between the research domains.

## 1.5 Thesis Outline

Within chapter two the topic is about security and reliability in new technologies, focussing upon the use of wireless sensor networks in these new technologies and popular use of such devices, and then showing in the view of the computer scientist the need to protect such devices from misuse. Further into this chapter, the mechanisms currently used on WSN nodes to ensure security objectives are explained, and important facts shown about the shortcoming of the approaches taken. At the end of that chapter results from an investigation into users' statements about the security of their wireless sensor network installation are discussed; this is done to understand the average users' assumptions of the security measures and their use of WSN devices with focus on their need of WSN monitoring mechanism.

Chapter three then is undertaking a review of security in general and especially with regard to its necessity in WSNs. After this discussion of security basics, single aspects and also the big picture, a look to the importance of threats that exist and that directly can happen to WSNs. The chapter then continues with attack detections in WSN; which leads to the objective of having continuous monitoring. The use of honeypots is supported by the findings from the investigations. The practical uses of honeypots are

discussed.

Also in chapter three, an exploration of the the usefulness of honeypots and monitoring analysis on a WSN node is undertaken. These research sections are on the identification of legitimate installation usage, with the goal to secure the user through looking at how users normally interact with their WSN, use the feature set of their system and control their environment. Therefore a particular network packet characteristic for categorisation needs to be present to determine malicious users based upon the network packets received and monitored.

In chapter four, a discussion is performed to achieve a deeper understanding whether the monitoring of potential malicious users and the type of networking packets they send in a WSN is sufficient to generate a reliable metric, purpose of the metric is to measure the performance and coefficient of attack identification.

With the previously achieved results, chapter five illustrates the monitoring methods and procedures. The combination of more than one technique allows for an enhanced detection of the presence of a malicious user; shortcomings of a single technique are compensated by a combination of several ones. This combination enables a pervasive kind of monitoring of malicious users during certain network interactions such as a network joining request, but not others. In the occurrence of diverse attacks happening, technique combination also provides a wide set of knowledge on the attacker's behaviour and skills. This combination of a set of techniques is described in a prototype design as flexible as possible to cover most of the hardware and network setups frequently used. The chapter ends with a conclusion about the framework necessary to perform as described.

Chapter five also develops a deeper insight into the design of the monitoring system. Previously discussed results from chapters 2 and later are included. The subsequent goal is to create a monitoring system that is capable to combine the results from single nodes. Therefore the WSN honeypot system combines the gathered data using a state

of the art procedures within a data analysis system to achieve its results. The development of honeypot analysis and a monitoring system is the core of the demonstration. In this chapter the creation of the prototype is explained in all its four parts.

The following chapter six contains the evaluation of the WSN honeypot system prototype, especially with regard to the detection of attacks. Exploring the system's performance and whether it is possible to figure out a comparison and an improvement to other solutions exists as the logical consequence of the usage.

And at the end, chapter seven consists of the conclusion. It is discussed whether the research goal has been reached, what has been achieved, which open research questions arise, what new research questions were found and consequently, possible future projects are detailed.

In the appendices, the previous papers published in this research work are listed.

# Chapter 2

# Related Work

THE related work is split into sections. Of course there is a literature review necessary, but outcome of other researchers is not only in papers, it is also in presentations at conferences and prototypical implementation of tools and frameworks for numerous purposes. Therefore the related work is regarded in addition to a common literature review to show the state of applied research as well. This chapter will begin with a literature review to lay the foundation on which this work is based; first of all the terminology of the wireless sensor networks is clarified before it is time to describe the attacks on wireless sensor networks in the published works of the research community. Similarly, a look at categorisation of attacks is being taken; these categories will play a central role in the further course of this work. Since this work is not just about theory but also about practical application, the remainder of this chapter is about specific tools, solutions, frameworks and the like that have to be dealt with in this work.

## 2.1 Literature Review

A Wireless Sensor Network (WSN) uses radio frequency (RF) communication at a relatively short range of lower than 100m. The parts of a WSN are often referred to as nodes, those devices are commonly battery or harvested energy powered with ultra-low power standby of mostly less than 100 $\mu$A on average. An overview on common powering practices for WSNs can be found in the work by Roundy et al. (2004). They use small packets of less than hundred bytes at a low data rate, often of less than hundred bits per second. When WSN are implemented, the trade-off's between the

cost of the nodes, the security, radio range, compatibility and flexibility have to be taken into account. In this work the focus is on WSN using IEEE standard 802.15.4, which is a simple packet data protocol for lightweight wireless networks, the network channel access is via Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and optional time slotting (Farahani 2008).

### 2.1.1 Wireless Sensor Networks - Terms and Definitions

IEEE 802.15.4 defines physical radio and a MAC[1] layer, providing a simple packet data protocol for lightweight wireless networks. ZigBee adds layers for logical network, security and application software. ZigBee determines the API, and the ZigBee Alliance certifies ZigBee-compatible devices guaranteeing their interoperability (Elahi & Gschwender 2009).

WSN can make use of 16 channels in the 2.4GHz ISM (for industrial, scientific and medical purpose) band, 10 channels in the 915MHz ISM band, 1 channel in the European 868MHz band at data rates of 250 kbps, 20 kbps and 40kpbs.

According to IEEE 802.15.4 there are two different device types, called FFD (full function device) and RFD (reduced function device). An FFD can operate in three modes, serving as

- Device
- Coordinator
- PAN coordinator (the principal controller of a personal area network)

An RFD can only serve as:

- Device

[1] MAC: media access control, a part of the data link layer in the seven-layer OSI model of computer networking

Some benefits of ZigBee are the reliability, network self healing options, support of a large number of nodes (more than sixty-five thousand). Although security features were added to ZigBee a long battery life is targeted. The global alliance is the basis for global usage, deployment without trouble and supports low cost maintenance and low cost hardware.

These are the ZigBee device types:

### 2.1.2 Coordinator

The coordinator is a FFD, each network has exactly one (Faludi 2010). It manages the whole network, as such it selects the channel to use for the communication and establishes the network communication. It assigns addresses to the end-devices and to the routing devices. It grants devices access to the network and acknowledges them the request of leaving a network. Finally, it is most often the sink of a packet transmission, meaning, that it receives the messages and makes the data usable via a gateway.

Address allocation – Tree: Coordinator has information about all routers and end devices, assigns addresses to routers, routers assign addresses to end devices.

Mesh: Coordinator has information about all routers, assigns addresses to routers, but routers assign random addresses to their children.

### 2.1.3 Router

The router is a FFD, too (Eady 2007). A Router does in principle the same as a coordinator, except: establishing the network. It is used in networks of mesh and tree structures. Its optional use case is the extension of the networks' width. Environmental measurements and control functions can be done as well at a router device.

### 2.1.4 End-device

An end device can be an RFD (Gislason 2008). This concludes, that it is allowed to sleep for energy saving purposes. An RFD uses limited 802.15.4 MAC layer set, needs

less power. As the name suggests it is not involved in routing messages. It can be connected to router or coordinator. End device addressing: When a node wants to join a network, the coordinator assigns an ID. The ID has 16bit length. Each device has additionally an IEEE 64bit address, this address is unique. Though devices in different ZigBee Networks might have the same 16bit address. The shorter 16bit address is used for energy saving purposes.

### 2.1.5 Gateway

The gateway is defined as follows (Farahani 2008). It connects ZigBee network to other networks using a different standard, e.g. LAN. The gateway is part of protocol transcoding. Information from wireless networks are in this case translated from ZigBee packets to the IP packet format by the gateway, and on their answer received the other way around.

### 2.1.6 ZigBee Trust Centre

The trust centre is a single node in the ZigBee network defined by the coordinator (Farahani 2008). It manages the link key and the network key for usage in the ZigBee network. The link key's purpose is a secure communication between two nodes of the network, the network key's purpose it the secure communication in the whole network. When devices want to make use of a link key, it can be pre-installed on the device by the manufacturer, another way is using the key-transport method with the trust centre, a third way is a key-establishment process that makes use of a master-key.

### 2.1.7 ZigBee Protocol Architecture

The ZigBee packet architecture is build of PHY packet, MAC frame, NWK frame and APS frame nested in each other as following.

At the physical layer the modulation/demodulation of signals 802.15.4 is taking place. Besides the radio communication synchronisation components on the channel and frequency at the PHY layer the PHY packet consists next to the synchronisation header

(SHR) of the PHY header (PHR) and the actual PHY payload. At the PHY payload of the PHY frame the following is nested.

The MAC layer (Medium Access Control) is the second part of the IEEE802.15.4 defined data structure, consisting of a MAC header (MHR) a MAC payload and a MAC footer (MFR) with a checksum called Frame Check Sequence (FCS). In the MAC payload the followind resides:

The Network layer (NWK) is built of a header (NHR) and the NWK payload. The payload nests the application support sublayer (APS).

The Application Support Sublayer (APS) sits between network and application, defines set of details to control the function of network, passes information from network to application.

The ZigBee device object (ZDO) is handling device management, its tasks are: it initialises APS, network layer, security service provider, it determines type of device, device and service discovery, as well as network and binding management.

ZigBee and ZigBee PRO since October 2007:

- Final release of ZigBee (few improvements and fixes)
- Additionally ZigBee PRO was published, it has
  - Only mesh networking,
  - Self-forming,
  - Self-healing,
  - End devices make first decision which route to take in mesh networks,
  - Interference are detected, transmission problems are reported,
  - Group-addressing (multicast),
  - Asymmetric link possible (A$\rightarrow$D via B and C, D$\rightarrow$A via E).

Possible topologies differ in WSN and ZigBee networks. Where the WSN definition as in IEEE 802.15.4:2003 spoke of star, peer-to-peer and cluster tree topologies, ZigBee only allowed to support star, tree, and the mesh topology (ZigBee Alliance 2012).

Network sizes differ, the transceivers are equipped with various antenna, transmission power and antenna amplification, depending on the ground, walls, obstacles. Generally, for the best connections good line of sight is needed due to the high frequency used in radio communication, where a Fresnel-zone free of obstacles is vital for connectivity (Kraußе & Konrad 2014).

## 2.2 Attacks on Wireless Sensor Networks

When the topic comes to smart home security, critical voices show up and warn to use wireless technologies. Even more, in an ACM journal (Denning et al. 2013) this approach is pursued, and also in this case spoken of new challenges. This raises the issue of whether the smart home will remain just a pipe dream when threats are not reconsidered, and if this requires to support new technologies. Is the smart home at risk?

The ACM report mentioned (Denning et al. 2013) even states that all these new devices entering consumer homes give reason to think about these new technologies: how to define the smart home by its security requirements and evaluate its installations to discuss improvements. The report introduces a framework by which the current threat can be ascertained, together with the impact on residents.

The concerns around privacy and security are increasingly present, the more network connectivity smart devices have and the more data they possibly collect and publish, by intention or even unintentionally, a study on consumers behaviour shows (Emami-Naeini et al. 2019).

More on smart home security research is presented in a comprehensive security analysis of (Fernandes et al. 2016). They present components of smart homes to be used

as possibly surveillance systems used by burglars to find out whether the owners are at home. Other use is presented for a device called Wink Relay of which the microphone can be turned active whenever this is wanted. They further on continue to analyse a set of several hundred smartphone apps that make use of the Samsung SmartThings framework which is used to read and control smart home components easily. During the course of this research they found more than the half of them requesting more access rights to the smart home devices than needed, which leverage the attack possibilities to steal codes and disable security features or control features such as generating an alarm event or others.

The insecurity of smart home devices not only poses a threat to the owners, it additionally enables malicious users to elevate attack strengths, as described in a paper by Benson & Chandrasekaran (2017). They show an overview and a categorisation of security flaws and weaknesses and their possible outcome in different categories, providing a guideline and discussion on the challenges of security.

Researchers have shown, that so called “google-hacking” nowadays happens for the smart home gateways as they are equipped with standard user-name and password combinations and are reachable via the Internet, therefore they presented issues with smart home gateway implementation (Eilers 2014).

A group of researchers presented a honeypot that presents itself and emulates a WSN at the wire-bound LAN part connected to the Internet (Dowling et al. 2017).

The next pages will give an overview on the threats resulting from attack possibilities to WSNs. This overview consists of possible attack scenarios produced by the fact that wireless transmissions take place on a shared medium, making the listening and eavesdropping to a channel simple, also leading to the possibility of not only reading information of the sensors and actuator, but also possibly reading the data from a key exchange if this is done in plaintext. Moreover, destabilising attacks on the transmission medium will be regarded, starting by simply sending something on the frequency,

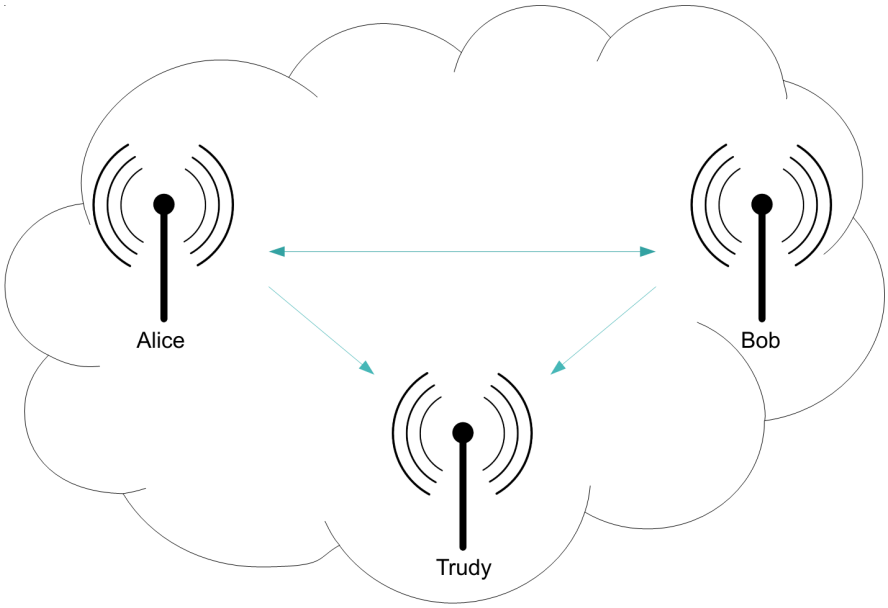



*Figure 2.1:* Passive eavesdropping

but also what is possible by sending real data packets, that are not originating in the established wireless sensor network. Even more, crafted data packets are discussed, that are designed to interfere with the established network communication with different intent, one of them is to cause a re-send of packets to let a result appear again, other packet transmissions are presented that cause changing the structure of the network and establishing active malicious components in the network. The by far most elaborated attacks are these that eventually change the software on the wireless sensor network nodes, doing code or firmware modifications.

### 2.2.1 Eavesdropping

Whenever a wireless communication is established, there is a way to eavesdrop and listen to the transmitted content (Cunha 2007). Cole et al. (2005) distinguish between passive and active eavesdropping. The first is according to their definition the 'Unauthorized, covert monitoring of transmission' (see Figure 2.1). Active eavesdropping is described more like poking into the transmissions until valuable information can be gathered by 'Probing, scanning, or tampering with a transmission channel to access the transmitted information'.

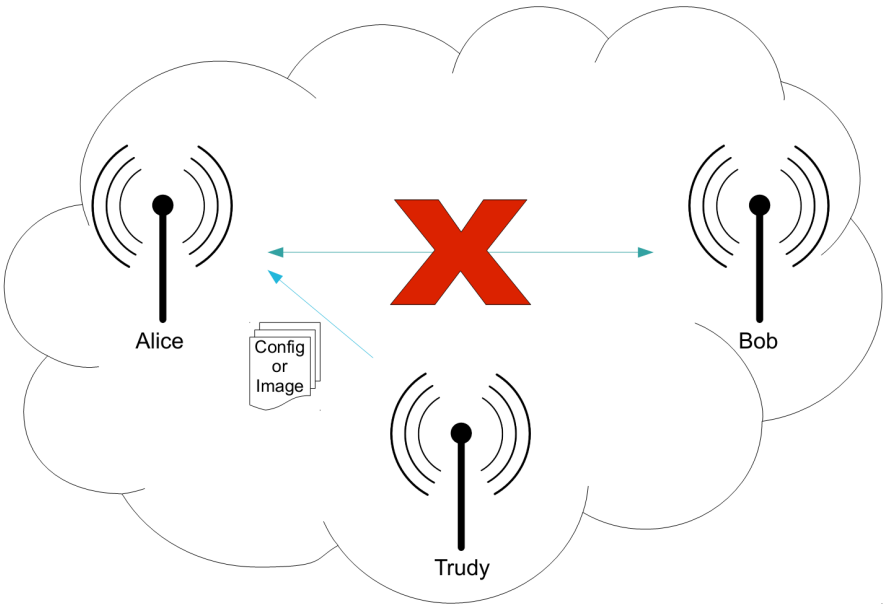



*Figure 2.2:* Over-the-air programming

### 2.2.2 Obtaining Keys

The initial assignment and distribution of encryption keys in a wireless sensor network is very peculiar. The keys in ZigBee networks, for instance, can be assigned before placing the nodes. The key can be read in plaintext when the nodes are captured and wires can be attached for instance to the Serial Peripheral Interface (SPI) lines, this was shown to be effective by Goodspeed (2009). The keys can also be modified over the air (so called "Over-the-air programming" OTAP, see Figure 2.2) (Das 2009). Actual research states that it is possible to eavesdrop the key distribution in plain text, depending on the chosen implementation. Valuable information about the reliability of OTAP can be taken from Aschenbruck et al. (2011). They state, that they refused from using OTAP due to its failure prone implementation and the needed resources.

Some possible research for obtaining network encryption keys has already been done. These attack scenarios focus on the challenge of extracting the encryption keys from the hardware by using off-the-shelf hardware or homebrewn hard- and software (De-Petrillo 2009).

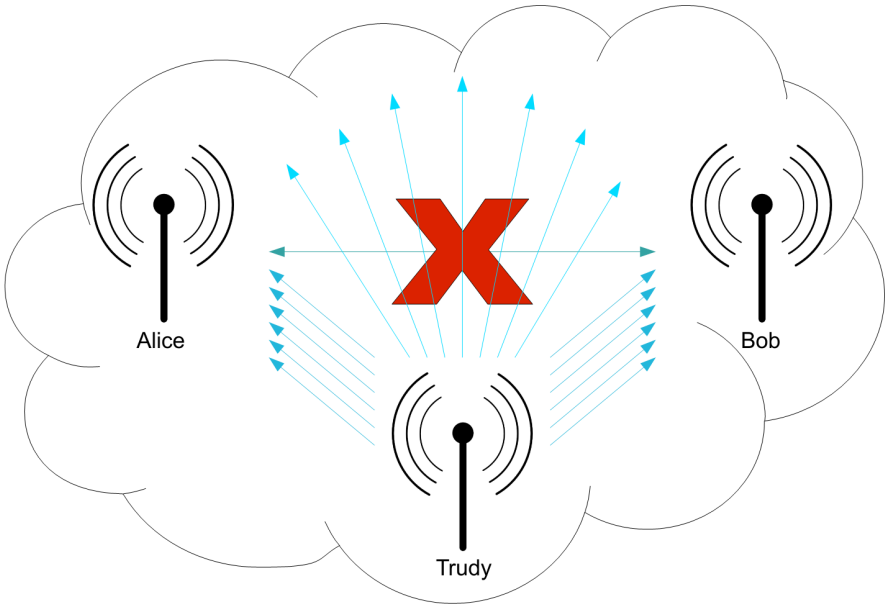



*Figure 2.3:* Jamming

### 2.2.3 Jamming

For every wireless communication, there is a way to interfere with the communication by concurrently sending data on the same medium (DePetrillo 2009). This transmission does not have to have information in it at all (see Figure 2.3). A variation is called the Garbage-Attack and it sends packets that may contain valid data (Raj & Thilagavathy 2012).

### 2.2.4 Redirecting Communication

It is possible to redirect streams of data in a local area network (LAN) in order to eavesdrop as an attacker, these techniques can also be used to launch a man-in-the-middle attack[2] with the intent of changing the transmitted data. One of the possibilities is the redirecting a data stream in a wireless sensor network over a node operated by a malicious intruder. A variation is selective forwarding (see Figure 2.4), or the sinkhole attack. In selective forwarding the attacker claims to be the best route and then drops most of the packets that ought to be redirected via the attackers node. The sinkhole

[2]MitM: a form of active eavesdropping, where the attacker routes the information between the communicating network nodes

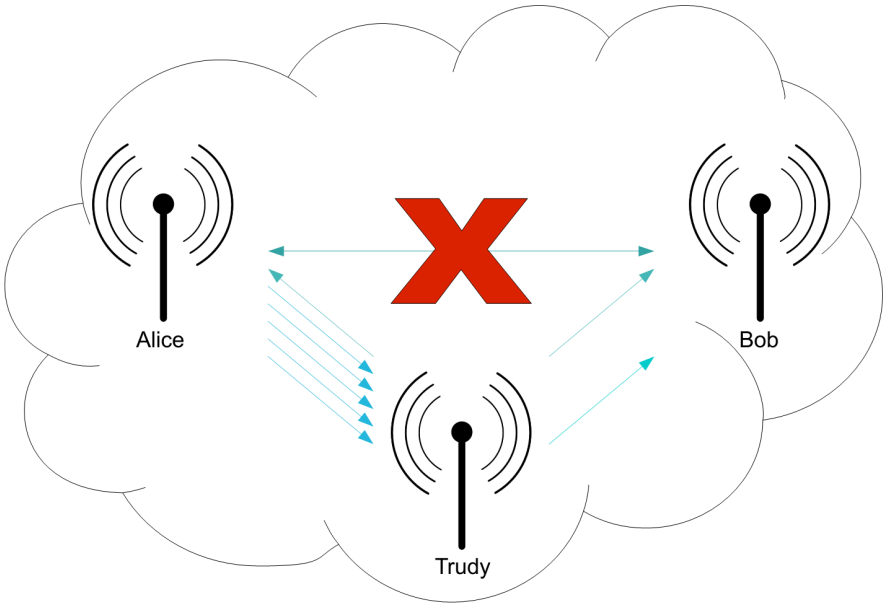



*Figure 2.4:* Selective forwarding

(see Figure 2.5) is the attack where the throughput gets even worse until none of the packets is routed any more.

### 2.2.5 Network Key Distribution

A man-in-the-middle attack is a supplemental method to that of eavesdropping on the channel for obtaining a network key. The intended purpose of this approach is the acquisition or modification of a key. Further attacks will become possible with the possession of such a key. This offers new ways of effectively exploiting the network. Furthermore the modification of a valid network key of one node would probably lead to the loss of this node for the operator, rendering it unusable (Wright 2009).

### 2.2.6 Packet Injection

Generating packets with the purpose to send them to a WSN has different goals. Such packets could be either meaningful, or just nonsense. Stages in between can be used to test WSN behaviour.

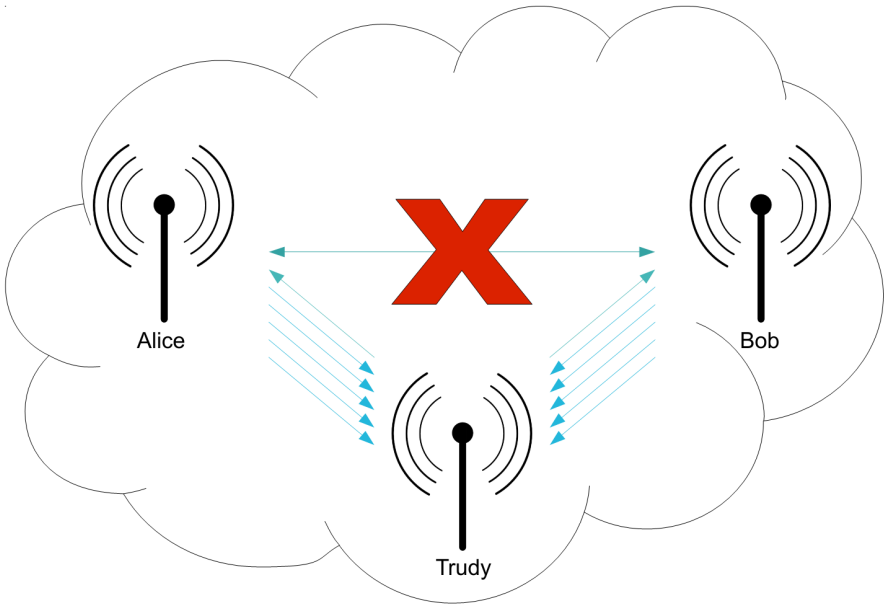



*Figure 2.5:* Sinkhole attack

#### 2.2.6.1 Garbage-Attack

A variation of the packet injection attack is the so called “Garbage-Attack”, where instead of replaying a valid trafffic from a network as in the replay attack a network is flooded with garbage data, which could be randomly originating from a packet generator. A flood of packets leads to a denial of service of devices if they cannot handle the amount of data they receive. This concept was further on considered to be worth doing research on by Goodspeed et al. (2012) where the KillerBee framework was refined to do practical packet injection using their dot15d4 layer extension for scapy.

#### 2.2.6.2 Replay Attacks

It was partly researched how wireless sensor networks react to replay attacks (Wright 2009) and (Wright & Cache 2015). This is an attack on a data transmission with the goal to circumvent security mechanisms or in the case of wireless networks it can also be performed with the goal to extensively drain battery power. In such a scenario, an outside party re-sends captured packets which have been previously transmitted in a network.

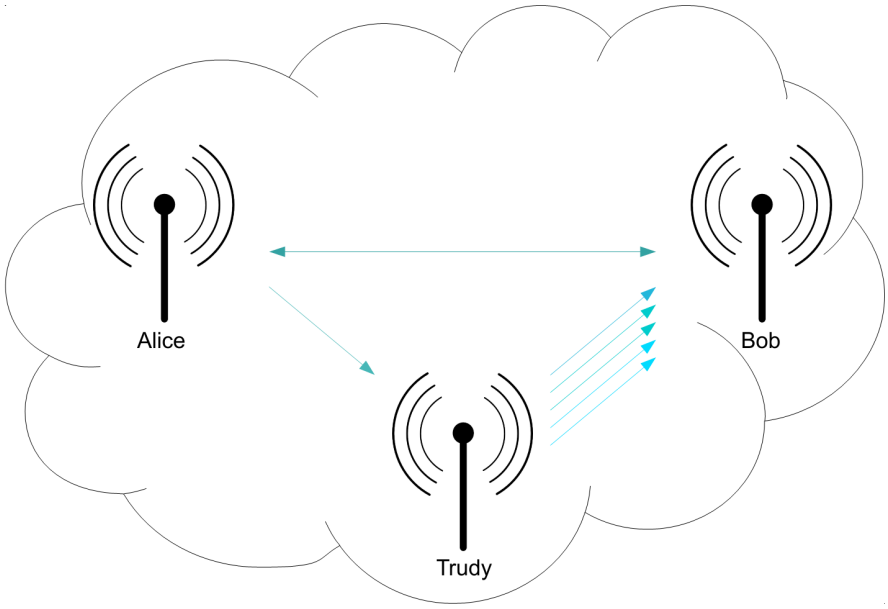



*Figure 2.6:* Replay attack

#### 2.2.6.3 Code Injection

The most interesting generated type of packet is the one carrying exploit-code and and additional payload. Such attacks were shown to be successful by (Goodspeed 2007*b*) and (Goodspeed 2007*a*). Such exploitability leads to the theory and prototypes of WSN worms as shown by (Gu & Noorani 2008) and (Yang et al. 2008).

There are studies on resistance to viruses and worms (Skrzewski 2012), which show clearly that attacks by a worm will stop almost never. Even 10 years after a new worm attack, it is still the case that if you have an unpatched personal computer with an outdated operating system like Microsoft Windows 98, Windows 2000 or Windows XP connected to the Internet without a firewall for a very short time, then this computer will be infected. In principle, all the infected computers eventually would be updated, or go off the grid, at least that is the theory. In reality, these devices are presented and vulnerable for a very long time on the Internet, then infected with a worm with the first waves and have not been updated since (Skrzewski 2012).

### 2.2.7 Attack Categorisation

Attacks can be divided into categories. Such categories were previously presented in various papers and their categorisation can be done in different manners. A common approach is to separate them by security goals as CIA, namely Confidentiality, Integrity and Availability in targeting attacks. One can also sort by the defence mechanisms or even the network layer they target. It is necessary to go more into detail as follows.

#### 2.2.7.1 Categories Confidentiality Integrity and Availability

As seen in this chapter there are possible attacks on WSNs. It is also possible to put them in categories. Common attacks on wireless sensor networks (WSN) as described in current publications and literature are part of a range of working and well-known attacks on these networks: Eavesdropping, replay attack, sinkhole attack, selective forwarding, network flooding, firmware modification and code injection (see section 2.2.3 to 2.2.6.3). They can be sorted by the kind of information security principles they compromise, this will be shown in detail in the section regarding cloning 3.3.1.3.

Unencrypted wireless networks can be attacked with the intent of compromising their confidentiality by sniffing 2.2.1 and man-in-the-middle attacks 2.2.4.

The integrity in wireless sensor networks is targeted for example by: man-in-the-middle attacks and replay attacks, but the network nodes could also be compromised by a firmware modification/replacement (see section 2.2.6.3).

Availability is a problem in unencrypted wireless networks through replay attacks as presented in 2.2.6.2, jamming 2.2.3 and flooding, denial of service attacks 2.2.6.1 and selective forwarding 2.2.4. Modifications to the network nodes and their firmware can also interfere with their availability.

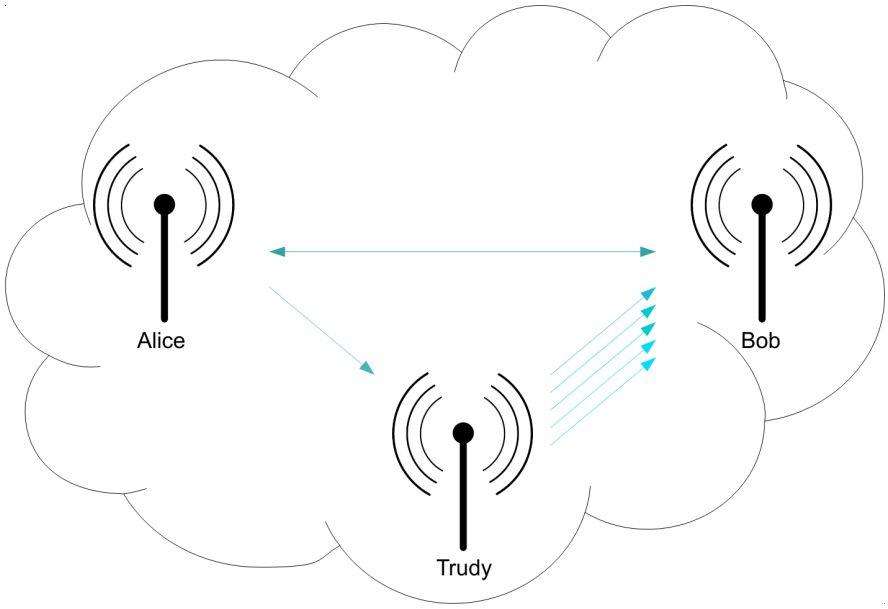



*Figure 2.7:* Power drain through replay attack

#### 2.2.7.2 Component Based Categorisation

These attacks can be subdivided into three categories of components which they target:

1. resource drain attacks
2. network attacks
3. hardware-specific attacks

The first category shall be termed “resource drain attacks”. These attacks target the power of network devices as shown in Figure 2.7 (replay attack). In a battery-driven system, the main goal of an attacker is to exploit the system’s power consumption against the system itself. An attacker might render wireless sensor network unusable by making parts of it run out of battery power through the modification of networking schematics, with the propagation of new routes, and by incessantly sending and requesting data. Batteries are a limited resource, once used up they need to be replaced. Nevertheless, commensurable the network capacity is likewise limited. An increasing

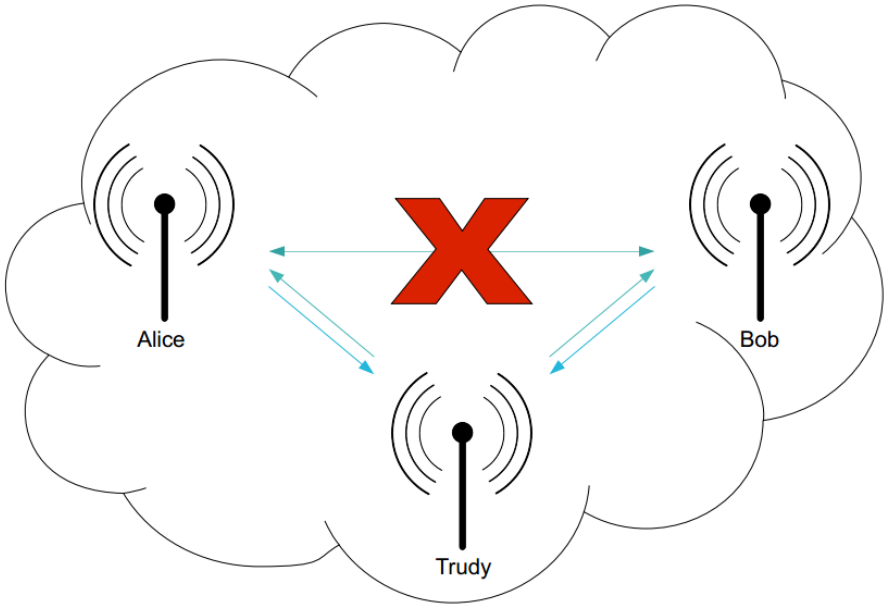



*Figure 2.8:* Network redirection for man-in-the-middle scenario

network load also cripples the already limited throughput of the network as seen in section 3.1.

The second category is called “network attacks”, describing the attacker’s goal of modifying the network’s routing structure. By this, the attacker aims to gain valuable information through redirection. He becomes therefore a “man-in-the-middle” listening to any traffic as seen in 2.2.1. An example is shown in Figure 2.8.

The third category is called “hardware-specific attacks”, standing for the goal to bring networking nodes under complete control of the attacker (e.g. firmware modification as shown in Figure 2.2 or code injection, see Figure 2.9). This could be achieved with physical attacks like attaching wires in order to extract encryption keys from a network node. Attacks on hardware can also be an attack on firmware, resulting in firmware modification. This is possible through security flaws in software components of the nodes.

All of the three attack types mentioned above exist, numerous attacks already were shown on the exhaustive usage of “resources” (see section 2.2.7.2), others on the

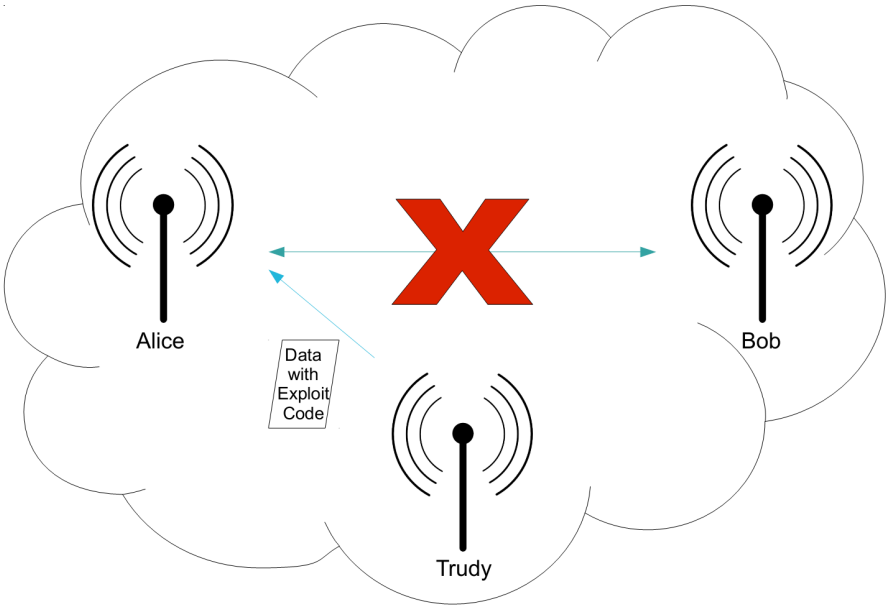



*Figure 2.9:* Code injection attack

attacks to the “networking” (see section 2.2.7.2) and to the nodes “hardware” (see section 2.9). This development should make abundantly clear, that the security threats for WSN are real and serious, which is conclusive proof that solutions to detect attacks are of crucial necessity.

An operators goal in normal network operation is to run everything as secure as needed and keep the bad guys outside. In a honeypot the network operator wants the opposite. In practise a honeypot is usually a specially prepared and deliberately insecure configured system with one or more security vulnerabilities. These monitoring, attack and defence mechanisms were proven and shown to work in classical network architecture, meaning wire-bound local area networks, WAN or MAN with a hierarchical network structure in tree or clustered form and shape.

The main difference between the traditional approach to honeypots and the proposed solution for a WSN honeypot has to be discussed. This will be explained in the following section 4.3 in a very detailed description and design of the features for the honeypot in wireless sensor networks. As mentioned above, the goal of a honeypot is traditionally to lure attackers in to get into contact with the system. In wire-based networks first

there is a host discovery done by an attacker, a scan through a network segment to find reachable hosts, then by service discovery the ones with services, these are interesting hosts. When a host with services is found, an attacker typically tries to find out whether there are logins with weak user-name and password combinations, or services that are not up-to-date with existing security patches and therefore have an exposed vulnerability which can be exploited to gain access to this host. In this "traditional" approach a honeypot is a service that is designed to be vulnerable, for instance with guessing the user/password combination as i.e. root/root or admin/admin.

If the attacker noticed this after having tried the plentiful attacks from his stockpile, then everything is fine. The honeypot has done its work, by having neatly recorded everything and the corresponding samples of possible new exploits are kept safe.

It could even be that an IDS / IPS provides a heuristic to detect an attack on its own, but in the case of wireless sensor network devices with very low capacity this has to be re-evaluated. It follows, that, as written in chapter 2.2.7.2, it is simply not possible to run a lot of additional code to perform calculations or to store very many rules and assign patterns to them. In an example of a WSN implementation, for instance the "Atmel BitCloud" on "Atmel ZigBit" nodes, of 128KB of memory, less than 10KB (Atmel Corporation 2012) are available. In the area of WSN, devices are not resource hungry, WSN are designed to use a very low amount of resources. This results in the conclusion that an increase of the device resource specifications is gratuitous or impossible.

## 2.3 Related Tools for Wireless Sensor Networks Security Research

In the WSN security research area are numerous tools developed for monitoring and programming, others also focus on attacks and WSN penetration. To conclude this chapter an overview is presented on categories of tools, also to show the attacking tools, giving an overview and point out the most important ones used. This tool classification is meant as an overview and introduction classifying the tools on the type of WSN layers they work with:

- Hardware
- Software tools
  - Monitoring tools
  - Programming tools
  - Penetration tools

In the first point hardware, a set of hardware tools that were built by researchers is presented.

The tools section is the largest one, although only the most important and available ones are presented. Only these tools are shown that are relevant and available.

### 2.3.1 Hardware Category

The latest research in literature showed again that there are only a few players in ZigBee security research from the academic point of view. Following are a set of some of the DIY Hardware development for security research, existed and results can be found.

#### 2.3.1.1 GoodFET

In the hardware-development for WSN there is Travis Goodspeed with the GoodFET framework. Travis Goodspeed is a security researcher doing research in the field of WSN, but also in other hardware centric hacking tools. He presented GoodFET as an open-source JTAG adapter which is a variation of the original TI MSP430 FET UIF (Texas Instruments Mixed Signal Processor 430 Flash Emulation Tool with USB Interface) as well as the EZ430U boards, as described in the documentation on the Website (Goodspeed 2009). There, he discussed the Flash Emulation Tool built of an MSP430F1612. It is widely available and is in its current version v4.2 capable of supporting even more devices than the initial design for just TI processors. The development of this tool initially led to the development of tools for this MSP type of

microcontroller, a first set of attacks and security testing results. More on these results is presented in the software section, a few pages later. It can be said, that the development of this tool was necessary for the whole wireless sensor network security research community, as it was delivering results that were till then estimated, but not proven.

#### 2.3.1.2 Api-Mote

Ryan Speers created the Api-Mote, which is a newer and better version of the TelosB Mote[3]. TelosB is a wireless sensor network node that was designed and manufactured by Crossbow, the design is currently belonging by Memsic. The original TelosB consists of a MSP430 microcontroller and a CC2420 transceiver component. The Api-Mote is designed to be a research device, that is more capable of reading and sending crafted messages. It is by design a research tool, not to be meant to be put into production. The development was done with core criteria in mind such as freedom in choosing interfaces, external antenna, changeable power sources and more, as these were details that were not found on the earlier TelosB. The importance of the development of this hardware is obvious, as lots of research results were generated through the use of this device.

#### 2.3.1.3 Freakduino

There is an Arduino project, that developers new hardware on the basis of Arduino, a free and open source platform so that every researcher interested in this topic can work with it. It consists of the hardware and this is enhanced with additional boards that can be used to enhance the platform (Freakduino 2011). As the name already suggests, the Freakduino is based on the Open Source Arduino platform, done by a group of researchers hosting the Freaklabs website. Besides the hardware development, they are also involved in software development as Open Source. As they host a webshop, the results of their work can be purchased. Although being an open platform, the hardware

[3] http://goodfet.sourceforge.net/hardware/apimote/

is rarely seen security research, therefore it was apparently not hotly favoured.

### 2.3.2 Software Tools

Regarding other Researchers and references, this section contains an overview on available tools for WSN security research. It is divided in the following structure:

- Monitoring and network traffic sniffing and network load display
- Programming tools
- Attack toolsets

Tools exist for WSN, in using, programming, analysing and for research. Regarding these tools, between good, of the one part, and bad, on the other part, it is not easy to differentiate. They are needed for security research and development, but could also be used as tools to attack a WSN; there is even a “proof-of-concept” worm that exploits network node communications and propagates itself over the network (Goodspeed 2007*b*), also in smart meters (Davis 2009).

It should be noted that there are several new attacks and ideas by developers regarding security flaws in the WSN field. The KillerBee API (Wright 2009) is a simple but powerful tool, and it is easy to build add-ons on top of it. The KillerBee API and the tools for it provide a basis for further development. Results may be given back to the community.

And new results in the area of WSN hacking were presented. Improvements to the KillerBee API with extensions to enable “war driving” scenarios on wireless sensor networks were developed (Goodspeed et al. 2012). They extended existing tools until all 16 ZigBee channels could be monitored in parallel. The authors then combined this with a GPS signal logger for “war driving” around the campus of Dartmouth. They also wrote semi-automatic tools to attack the networks with a number of different attack types.

15dot4 is a tool to listen to ZigBee and 802.15.4 communication (O'Flynn 2012), it provides an input device for the well known network monitoring tool Wireshark (Combs 2012) to monitor and capture the network traffic in realtime. Both tools feature usage with state of the art development kits, for example the AVR Raven (AVR 2008) and they might be ported to other platforms as well.

Free projects like the Freakduino (Freakduino 2011) make it easy to build WSN components based on the cheap Arduino development kits. These kits have powerful tools available to work with, bringing WSN technology to everybody interested in working with it and doing research on the technology.

The authors Parbat et al. (2010) presented an overview on data visualisation tools for WSNs. These tools are also used for monitoring the WSN and fit to the first category. The authors Dwivedi & Vyas (2011) additionally also presented a very comprehensive survey on tools for WSN experimental research in 2011. They present 63 simulators/simulation frameworks, 14 emulators, 19 data visualisation tools, 46 testbeds, 26 debugging tools/services/concepts, 10 code-updating/reprogramming tools and 8 network monitors (Dwivedi & Vyas 2011).

#### 2.3.2.1 Monitoring Tools

The tools for monitoring the WSNs have different purposes. Some of them are used purely to gather data, others have displaying functionality, and there are tools building a bridge between already existing network analysers and the WSN domain. So, they all appear to have some of these functionality:

- Monitoring the network and its structure,
- monitoring the data gathered,
- a combination of these both categories
- displaying raw data material.

- showing graphs of the data material
- showing graphs of signal strength
- showing WSN channel frequency occupation and usage in packets or data per second.

A sniffer is a piece of software to obtain the information sent wireless in data packets. This information is then further on processed and can be analysed. The basic functionality can be seen in the packet dump and storage of data to logging. The sniffers are equipped with other features occasionally, including filtering, display, and further analysis tools.

The Freakduino sniffer makes use of the tool Wireshark for packet analysis.[4]

Wireshark Zigbee Utility is a tool developed by Jakob Thomasen, currently in Alpha version 0.4[5] The Wireshark ZigBee Utility is enabling easy capture of Zigbee network frames. Then, the monitoring within Wireshark is made possible. It implements a named pipe between Wireshark and the hardware, in the current development stage it is a USB dongle running on a Freescale MC1322x. The author states, that support for other devices can be added easily (Thomsen 2010).

Kisbee is more than a sniffer, it is a complete toolset consisting of:

- A hardware platform
- sniffing firmware for this platform
- bluetooth or serial interface to android device
- android app for analysis of sniffed data, monitoring, storing, putting GPS coordinates to it and displaying it in maps.

[4] https://archive.freaklabs.org/index.php/tutorials/software/feeding-the-shark-turning-the-freakduino-into-a-realtime-wireless-protocol-analyzer-with-wireshark.html

[5] https://github.com/jbthomsen/WiresharkZigbeeUtility/

The hardware is available for purchase from Mike Kershaw, the author of the Wireless tool Kismet[6]. The application is downloadable in the app store.

15dot4[7] is a tool by Collin O'Flynn, working as a 802.15.4 sniffer and capturing framework. This is a group of useful tools for working with 802.15.4 networks. It currently includes a sniffer which interfaces with Wireshark in Windows/Linux. It allows to capture 802.15.4, ZigBee, and 6LoWPAN. Additionally graph is showing the signal strength on the different ZigBee channels.

In 2013 the Atmel AVR Company released a documentation for developers with the intent to have a reliable sniffing tool for their hardware. The documentation is available for download at their website: "Atmel AT02597: ZigBee PRO Packet Analysis with Sniffer". It suggests hardware and software components; as hardware the AVR Raven USB stick or the RZ600-231 as well as deRFusb23E06 can be used to sniff at 2.4GHz, the deRFusb13E06 can be used to sniff for the 900MHz ZigBee. As software component three products were evaluated: Wireshark, Perytons and BitCatcher.

Luxoft BitCatcher is a tool for wireless analysis in ZigBee networks[8]. It can make use of AVR Raven stick and others for data packet input.

Wireshark was mentioned beforehand[9]. It is a packet monitor and analysis software. It receives the data from packet capturing sources, be they wireless or wire-bound. It is an open-source software that displays all the packets it receives and shows their network layers. For debugging networks it is one of the standard tools.

Perytons is more than a packet displaying tool, it contains features to display and debug complete networks. It is designed to be specialised in the wireless networks[10].

Sniffing does not only has to take place on a single channel, as there are more than

[6] http://www.kismetwireless.net/kisbee/
[7] http://sourceforge.net/projects/dot4-tools/
[8] http://www.luxoft.com/embedded-systems-development/bitcatcher/# and https://www.dropbox.com/s/rj4x9s1f0z9iiax/Sniffer_1.0.1_public.zip
[9] https://www.wireshark.org/
[10] http://www.perytons.com/index.php/protocol-analyzers/zigbee/

one channel for ZigBee communication it might be useful to sniff on more than a single one. One possibility is to use channel hopping and when a channel is busy to sniff on it. This is only useful if only a single communication is taking place at one time. If more than one ZigBee communication takes place and they are on different channels, the channel hopping is not sufficient. Instead sniffing on more than one channel is the better approach, it is also called multi-channel Sniffing. The following tools are available for sniffing on all of the channels simultaneously:

1. Peryton Company since 2008
2. Travis Goodspeed in 2010
3. EECS Berkeley, Thomas Watteyne and Boyang Zhang 2012
4. Dartmouth University, Ryan Speers, Sergej Bratus et al. 2012

The research project was started with the aim to build a 16 channel sniffer for 802.15.4 networks that could also be used for other protocols than ZigBee namely IEEE802.15.4e, WirelessHart and ISA100a. These were mentioned before (see section 3.1)[11].

The researchers from Dartmouth also have established a setup to sniff on all 16 channels at once, without doing a channel hopping. It is part of the Api-do toolset and more details on it is found in the attack tools section 2.3.2.3.

#### 2.3.2.2 Programming Tools

This involves programming languages and integrated development environmental. They are available for free with no cost but there are also some of the open source tools available.

AVaRICE is a tool by Scott Finneran for debugging, it offers the possibility to use the GDB (GNU debugger) with the JTAG in circuit emulator to debug Atmel chipsets[12].

[11] https://www-bsac.eecs.berkeley.edu/project/zoom.php?urlmyprojectID=BPN558
[12] http://avarice.sourceforge.net/

AVRDUDE is a tool by Brian Dean to download and upload to the Atmel AVR microcontrollers using the ISP way of programming these chips.

WinAVR is a software bundle consisting of the GCC compiler for the Atmel AVR microcontrollers and other tools together for development[13].

#### 2.3.2.3 Penetration Tools

In this category all the hacking and network penetration tools can be found.

One of the first steps in penetration testing is information gathering and fingerprinting. As seen in research work, fingerprinting as well as quasi "stealth" communication can be achieved. Jenkins, Shapiro, Bratus, Speers & Goodspeed (2014) conducted research on 802.15.4 receivers and tested the comparability of their networking. Although he first assumed an outcome of minor differences he presented the results very proud to be able to not only find out, that fingerprinting is possible, also he discovered the possibility to do data transmissions that could only be read by a set of WSN chip-sets, for other WSN chip-sets the transmission was not read.

Killerbee is one of the first freely available toolsets to conduct research with. Other tools can be divided into the following categories. For example, there are tools simply being single scripts, that automatically does a single task. But there are also tool-set consisting of a set of tools.

It was further on refined and more tools were built on top or around Killerbee, namely the Api-do tools (2.3.2.3)

Another attack tool set was developed calles Sensys (Giannetsos et al. 2010) and Spy-sense (Giannetsos & Dimitriou 2013).

Joshua Wright was writing an attack tool for ZigBee and presented his work in 2009[14]. His tools were one of the first widely available open source platforms to build attacks

[13] http://winavr.sourceforge.net/
[14] https://code.google.com/p/killerbee/

on (Cache et al. 2010).

One of the other research projects with available code for attacks or defences in WSN / 802.15.4 /ZigBee is Api-do. Ryan Speers and Travis Goodspeed, Sergey Bratus, Ricky Melgares, Sean W. Smith and other researchers were working on this attack framework[15] of tools.

OpenEar is a tool that captures data on one or many up to 16 channels in parallel, waiting for current coordinates provided by a GPS device to combine them in the logfile. Another part of the toolset is the zbWarDrive that assists in the ZigBee wardriving purpose of the attack tool by selecting the best channels to do packet capturing.

The SenSys tool consists of a sniffing component and also has a component for visualisation of the network topology, traffic, status of the network nodes and gives an overview whether an attack was successful; it has attack tools that can compromise the network. Sensys was the first tool that was developed by the authors of (Giannetsos et al. 2010).

Spy-Sense is a tool that is also called spyware[16]. It is a tool as a prove of concept showing that it is possible to have code running stealthy in the background, that was originally placed there by code injection (Giannetsos & Dimitriou 2013).

### 2.3.3 Conclusion of the Related Research Tools

The development of a honeypot for wireless sensor networks required data to work on. It was hence necessary to evaluate all available tools before deciding on a toolset to use during the development of a defensive tool.

The argument in favour of using these tools is obvious: KillerBee is used to attack a network. The gathered data is then used to gain knowledge on the detection.

The KillerBee API soon turned out to be the first choice to achieve knowledge about

[15] https://code.google.com/p/zigbee-security/
[16] malicious code that collects data from the target system and provides it to the attacker

attacks on a WSN in practice and in theory. It is the perfect toolset for information gathering in the form of packet captures of attacks on a WSN. The setup for the capturing is presented in chapter 5.

## 2.4 Conclusion of the Literature Review

The previous sections featured the WSN basics, the standards as well as possible attacks on WSN.

All the previously shown information, considered as a whole, shows that a lot of research has already been done in the field of attacking wireless sensor networks. Encrypting the communication and hardening a system to resist an attack will not stop the attack itself from taking place. It is difficult, if not impossible to prevent a novel attack methods against a network, by just hardening.

The tools section presented in this chapter consisted of the structure shown initially in section 2.3. A focus was set on the non commercially available hardware and software to show the research undertaken by research groups and not by vendors. The hardware presented is as follows:

| Hardware Tools | Type | Sourcemodel | Available |
|---|---|---|---|
| GoodFET | Programmer and debugger | Opensource | yes |
| Api-mote | WSN platform | Opensource | request |
| Freakduino | WSN platform | Opensource | yes |

*Table 2.1:* Hardware for wireless sensor networks

The software tools presented are summarised as:

| SW-tool | type | Release | Version | Maintained | License |
|---|---|---|---|---|---|
| Freakduino | sniffer | 2016 | - | - | BSD |
| WiresharkZigBee | sniffer | 2011 | 0.10a | no | free |
| Kisbee | sniffer | 2019 | R1 | yes | GPL |
| 15dot4 | sniffer | 2010 | 0.5 | yes | BSD |
| Luxoft Bitcatcher | sniffer | 2013 | 1.0.1 | no | Proprietary |
| Perytons | sniffer | 2017 | - | no | Proprietary |
| AVaRICE | programming | 2012 | 2.13 | - | GNU |
| AVRDUDE | programming | 2016 | 6.3 | yes | GNU |
| WinAVR | programming | 2014 | - | yes | open source |
| Killerbee | pentesting | 2019 | - | yes | open source |
| Api-do | pentesting | 2011 | - | - | BSD |
| Sensys | pentesting | 2010 | - | no | - |
| Spy-sense | pentesting | 2013 | - | no | - |

*Table 2.2:* Software for wireless sensor networks

A '-' in the description means, that no information is given, the release date refers to the latest available version online. All of the different types of tools are present: tools to program wireless sensor node devices, tools to analyse the network by sniffing the traffic for analysis, and also tools to find out weaknesses of the wireless sensor network setup. While the commercial tools by Perytons and Luxoft became unmaintained by the end of the thesis in 2018, the open-source and community-driven projects are alive and maintained.

The chapter started with a literature review to create the foundation on which this work is based; first of all the terminology of the wireless sensor networks was clarified before it was time to describe the attacks on wireless sensor networks in the published works of the research community. The immensely important look at the categorisation of the attacks on wireless sensor networks has been made, because these categories will continue to play a central role in the course of this thesis as announced. Since this work is not just about theory but also about practical application, the remainder of this chapter was about specific tools, solutions, frameworks and the like that have been dealt with in this work.

Summing up what has been seen in the last chapters, a lot of research and development has been done on honeypots in wired networks, and security research in WSN is ongoing as researchers see further need in exploring practical and usable intrusion detection for WSN. The further reading is exactly about that, the practical security approaches, the detection and defence approaches.

The related work in the next chapter is a literature review on the security mechanisms and defence possibilities as well as their resource usage and conceptual limitations. This chapter continues the fundamental review of literature and comparable works. A topical separation was nevertheless sensible, because the following should also collate the known and partly standardised safety measures in wireless sensor networks, the reliability and the defence measures, up to a knowledge of missing components. Furthermore, a discussion of the honeypot topic is presented, as well as maintenance measures and resource applications, which the presented defence strategy in general demands. With a look at the legal framework, the chapter will be rounded off and a conclusion will be presented.

# Chapter 3

# Related Security Mechanisms and their Limitation

REFFERING to the introduction 1.1 WSNs needs protection of the network communication. The WSN do not have security features, but the ZigBee specification on top defines these for confidentiality, integrity and partially also availability. As a cryptographic standard implementation the ZigBee Alliance mandates the use of encryption with AES with 128-bit keys and counter mode (CTR) for messages. The message integrity code (MIC) shall be generated by AES with 128-bit key and cipher block chaining (CBC) (Elahi & Gschwender 2009).

Beside the security in WSN, other wireless networks were viewed by Neil Peterson and Jeff Potter (2011), who wrote a summary about the security in WirelessHART and Wi-Fi networks. They say, that in the usage of wireless technologies, it is crucial to have security features enabled and implemented. Their three key areas for security include access control, network protection and device integrity. Therefore, they give the conclusion, that clients should have authentication to the wireless network, the transmissions to be encrypted and the devices protected. When there is a gateway, a proper configured firewall is necessary.

As could already be observed in the design and implementation of the common wireless networks IEEE 802.11a/b/g/n, there is a need for providing security and reliability, preventing data loss. The following paper contains design principles for implementing a secure WSN. It also gives best practice for operating the network. Target readers

are especially in industry as if focusses on WSN in productive environments such as manufacturers or in process automation. The recommendation is basically to activate all security mechanisms that are given by the WSN and to refrain from unencrypted communication or key exchange procedures. The authentication and preservation of data integrity is a requirement in wireless data communication (Masica 2007).

Several threats arise whenever such a network is established. Developers are asked to protect communication in order to attain the classic information security requirements: confidentiality, integrity and availability (Elahi & Gschwender 2009).

## 3.1 Standards for Wireless Sensor Networks

ZigBee is a well-known example of a wireless sensor network feature set among others like Z-Link, SmartLink or Z-Wave. ZigBee PRO 2007 defines mechanisms for authentication, encryption and trust centre functions on selected main components of the wireless networks (ZigBee Alliance 2008).

Network communication, be it wired or wireless, always bears security flaws, which must be dealt with. The network traffic can be listened to, also interruption of the networking is possible, be it fully or partially by different types of interference. When transmissions are forwarded and not just point to point, routing takes place. Routes can be changed and therefore communication be redirected, also information can be intercepted and changed during the routing and forwarding (Cole et al. 2005). In wireless communication this gets even easier, because the information is sent also into the surrounding environment providing access to all parties within signal range.

The analysis of the security and reliability of ZigBee networks has shown that these aspects have not yet been resolved satisfactorily, depending on the chosen implementation. But not all of the above requirements are present in the various configurations. The background to this is as easy as might be: the intended purposes of wireless sensor networks vary widely. One of them is low power consumption, running independently for the longest possible time, another one is for these devices to be as

interoperable, compatible and easy to use as possible (Faludi 2010).

The IEC standard 100.11a (ISA 2011) describes another wireless standard based on 802.15.4, comparable to ZigBee, WIA-PA (Wireless Networks for Industrial Automation - Process Automation is an IEC international standard) or WirelessHART (A wireless version of the Highway Addressable Remote Transducer protocol).

The authors Song et al. (2008) have a look at the WirelessHART standard. They describe it as the premier industrial wireless sensor network standard. They provide solutions for several challenges to a WirelessHART network and present their first protocol stack. Their main challenges were security as well as time and mesh-network management. For security they propose to always use hardware accelerated encryption and decryption routines. For the timing they decided to use an extra timer module because the microcontroller itself provided no satisfactory results and implemented a scheduler with priority for time critical execution and the possibility to assign to other tasks less priority. The challenge for network management was the implementation, the standard sets the goal, the concrete algorithms had to be found and implemented.

## 3.2 Defences to Attacks on Wireless Sensor Networks

The previous section listed known possible attacks on wireless sensor networks and classified them into three categories. This section gives an overview of the consideration of appropriate countermeasures to the presented attack scenarios.

### 3.2.1 Standardised Improvement with Encryption

IEEE 802.15.4 contains nothing about security at all, but additional standardised layers like ZigBee enhance the standard with essential features (ZigBee Alliance 2008). Several concerns about security were targeted by the final release of ZigBee PRO in 2007 (ZigBee Alliance 2008). Improvements were made regarding encryption and authentication and were added to the standard to address threats (Yüksel et al. 2008). The authors' opinion on the key issue of WSN is about the cryptographic key update.

Devices obtained by an attacker still have the key material. Changing the keys every time a node gets unresponsive and the key could have been compromised is a resource drain. Therefore if has to be chosen to have either secure or long-living WSNs. However, key management is suggested to be hierarchical or distributed.

The ZigBee standard defines three key types: master key, network key and link key. The master key is optional and may be used for link key generation set by the device manufacturer, it might also get installed by a trust centre. The network key is a key shared among all devices in a network, therefore it has to be installed by a device manufacturer, it might also get installed by a trust centre. The link keys' use is two-fold, for application usage the key is individually generated and installed to the devices that establish a network, for communication with the trust centre this key is a trust centre link key (Elahi & Gschwender 2009). The standard requires: "A default global trust center link key must be supported by the device if no other link key is specified by the application at the time of joining. This default link key shall have a value of 5A 69 67 42 65 65 41 6C 6C 69 61 6E 63 65 30 39 (ZigBeeAlliance09)" (ZigBee Alliance 2012).

The current ZigBee PRO 2007 standard requires that AES 128 bit encryption (NIST 2001) is provided with any new hardware in order to comply with the ZigBee specification. The strength of the implementations of vendors may differ, especially regarding the protection of shared secrets. AES-128 is discussed in the security community, which considers it to be strong cryptography[1], but the security implementation in hardware may still contain flaws (Böck et al. 2016). The authors were able to identify an implementation flaw, based on using a weak initialisation vector and furthermore found a set of load balancers from a manufacturer, which was relying on another vendor for these routines. Simply spoken, all the devices used the same nonce[2]. Therefore the key material was weak and could be broken. For these devices a patch was released.[3]

[1] https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[2] Arbitrary number to be used just once

[3] Security Advisory Explicit Initialization Vector for AES-GCM Cipher https://support.radware.com/app/answers/answer_view/a_id/18456

In a survey by Chen et al. (2009) about a hundred references to security related research is presented. The survey starts at threats and vulnerabilities. It continues by grouping defences in order to discuss promising solutions and weak ones. For the latter, open research questions are posed. One of the authors' statements is considered helpful in the formulation of the research questions; they propose more research in the area of detection mechanisms, together with research in the area of corresponding countermeasures with the detection of an attack (Chen et al. 2009). For improvement of cryptography they suggest to perform research on the use of public key cryptography, although it is considered resource costly. They propose doing research on optimisation of efficiency. But also in symmetric cryptography they see work to be done on key management procedures. Especially for mobile nodes they suggest to focus on further research. Attack detection is seen as a further research topic and also attack prevention. As a suggestion of a further research topic they see a focus on finding out nodes that have a high probability of being attacked and have intrusion detection mechanisms safeguard those more than others to reduce the overall workload of the intrusion detection system in the wireless sensor network. Also secure routing algorithms research is suggested, with a focus on protecting the routing information in such a way that the routing information is not distributed more than necessary as well as broadcast encryption. Also about secure routing, the suggestion tends to research on mobile WSN. Another proposed further research topic is seen in the physical security of the main controller of the WSN.

### 3.2.2 Intrusion Detection System

The authors Wei & Kim (2012) propose an IDS (intrusion detection system) scheme in a WIA-PA network, which is sensing changes in information transmission. WIA-PA networks use a superframe, which is a kind of scheduling system that determines transmission slots. This timings follow a pattern that can be used to generate a prediction model on when the transmissions will be done. The prediction model can then decide whether the transmissions seen are normal network traffic, and the model can

then also identify attack traffic, which is not following the model. The authors found this to be effective whilst also being resource friendly.

The most important fact apart from the improvement of security features is the detection of attacks to the wireless sensor network. Intrusion detection systems (IDS) are well known in wired networks. IDS analyse network traffic by scanning patterns for anomalies. Due to the two-fold use of the term IDS, this necessitates a concise definition: An intrusion detection system (IDS) is, within the scope of information technology, an application with the purpose of monitoring network traffic for malicious activity. The application of policies leads to the reporting of detected violations to the operator of the system. The paper gives an outlook on intrusion detection systems in WSN, which the authors state, can be used to help identify attacks on the network. They suggest that the coordinator of the network should analyse the network traffic and detect the intrusion. This could be a very spontaneous raise in the network usage, or a marked increase of network load over time, or even a flooding with packets to many different network coordinators. They suggest that the analysis should take place on less mobile nodes, that have higher computational possibilities and power supply. Additionally, a kind of heartbeat message is proposed, to make the nodes periodically transmit their liveliness. The authors close the paper with the comment that even if the attack is continuing, the network is at least aware of the attack happening. (Misic et al. 2005).

Surveillance of an area with the aid of wireless sensors, for example for troop movement observation or for perimeter breach detection is also sometimes called an intrusion detection system (Klues et al. 2005), (Wang et al. 2008). In this given example, the first paper uses the term to describe a network intrusion, the second paper discusses the detection of an intruder in a battlefield by the sensing components of a WSN. They define an intrusion detection system as a component to detect attackers that behave different in a battlefield by analysing their movement anomaly. Two different models are possible, the homogeneous and the heterogeneous detection in WSN. Therefore, the quality of detection is depending on the distribution density of

the sensing nodes as well as their sensor accuracy. The authors state, that there is a difference between a just well-aimed sensor and collaborative sensors to detect an intruder. In their work they use network simulation to determine the detection probability, and end with a discussion on the connection strength. They conclude with the result, that both network models are possible to make a thorough detection.

Intrusion attempts are reported in classical network security solutions by IDS. They use only passive inspection to monitor all network activities for violation of regular use, and have no other function in the network. The exceptions from regular use are put into detection rules, also called signatures. Whenever such a signature is found in the traffic, then an alert to the administrator can be generated. The IDS rely on predefined policies describing normal network activity and normal behaviour (Meier 2007). The signature use is known to be less error-prone, meaning that a low number of false reports are generated; these are called false positives. A disadvantage is seen in the overlooking of an intrusion, because a pattern does not match enough. This circumstance is called a false negative. In contrast to the use of well defined patterns to search for to come to the alerting, there exists another method, called heuristic or anomaly based detection. The use of heuristics is possible to detect unknown attacks, in general an IDS is known to log false positives. However, the use of heuristics may find more new and unknown intrusions, so has a less false-negative rate, but is known to have a high false-positive number, in comparison to the signature based attempt.

An IDS is generally a single system in the network, this is also called NIDS (network-based IDS), but it might also be distributed on several hosts of the network like it is done in an HIDS (host-based IDS). An IDS as an active part of the network that might also stop intrusion or filter or modify malicious traffic and is in this case most often called an intrusion prevention system (IPS) (Cole et al. 2005).

In an IPS, the controlling aspect of Figure 3.1 is much more elaborate than in an IDS. The IPS is seen as an addition to an IDS, meaning that the detection of an intrusion is resulting in activating additional safeguarding methods, of which filtering can be a part

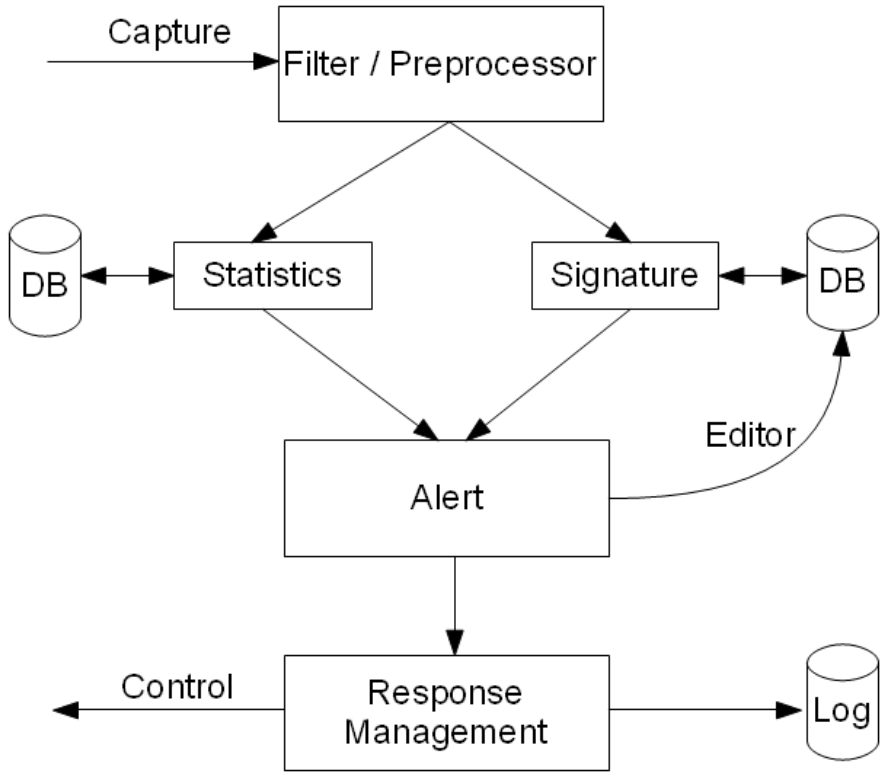



*Figure 3.1:* Intrusion detection system (IDS) working scheme

of, by simply putting a new rule up such that every similar traffic is not to traverse the network any more. All of these setups provide logging and reporting.

In an IDS, every data packet has to be regarded and analysed. Its purpose is to spot the packets of an attack within a large amount of regular and normal network traffic (Meier 2007).

The following gives an overview on the detections, that are possible by pattern analysis, some of the attacks are already detectable by pattern analysis:

Kaplantzis et al. (2007) did a network simulation and had success in detecting selective forwarding attacks in wireless sensor networks by support vector machines (SVM).[4] They also concluded that these attacks are the most difficult to accurately detect, thus verifying assumptions made in previous research. In the paper the authors use a sin-

[4] A support vector machine is a mathematical model to classify and distinguish between patterns, it belongs to machine based learning.

gle detecting node to be able to detect selective forwarding and black hole attacks. The IDS is built with the constraint to have a low impact on the long lasting live of the network nodes, therefore not draining too much energy when running. The single detection means, that all the detection is running solely on the coordinator of the network. The emphasis on two attack patterns is because they consider these to be the most important issues to solve. The results are promising, especially the black hole attack could be always detected, not a single alarm was missed. However, the sink-hole attack was found to be very subtle and the accuracy was deteriorating, the less interaction the attacking node had. However, when the attacker was not wise enough to be sneaky, then the attacks were found with high detection ratio. Therefore the authors conclude, that the sink-hole attack is indeed hard to detect, as it was described already before by (Karlof & Wagner 2003)The absolute numbers and measurements are found in tables at the paper and are not copied here.

Sedjelmaci & Feham (2011) presented a novel IDS for the use in clustered wireless sensor networks. Using SVM like Kaplantzis et al. (2007) the essential point is to teach the detector nodes about normal network behaviour, such that they achieve a detection ratio of over 98% regarding unusual network traffic. The paper proposes an intrusion detection system in a dual-split attempt, meaning that the anomaly based detection as well as a signature based detection are executed. This combination was researched, having combined the befits of both attempts, the signature based detection and the anomaly based detection. The results were shown to provide a good detection with low false-positive alerts. Also they conclude the paper, that the calculation costs at a WSN are not that resource draining and refer to a paper by Stetsko et al. (2010). There, they calculated the ratio of a single bit transmission over the air to an equivalent of one thousand microprocessor cycles. The energy saving component of this approach is the component most worth mentioning.

A lot of research on this topic has been done before, starting from the detection of intrusion on the lowest, at the physical layer (Bhuse & Gupta 2006). Bhuse and Gupta

propose a detection based on anomalies in signal strength. The research was conducted using already available data sources and aiming at an easy solution. It was showing results to the different layers of a wireless sensor network, namely at the PHY, MAC RTG and APP layer (physical, message-authentication-code, routing and application layer) of a WSN; on PHY layer they made use of the RSSI value (received signal strength indicator), on the MAC they used scheduling mechanism for detection. On the RTG layer a technique called forwarding tables were measured. On the APP layer safeguarding mechanisms were introduced by the researchers. The researchers are concluding the paper with the statement, that for the PHY layer the detection with RSSI has a high false-positive ratio. The MAC detection has a little overhead in the workload of the nodes, because they have to keep track of the TDMA (time division multiple access) for their neighbours, however, the detection ratio is good and the overhead was discussed to be fair. For the other proposed technique at the MAC layer the S-MAC is seen, it means, that the use of fixed time slots is negotiated, intruders are not aware of the details and their transmissions at other timeslots than the ones negotiated are easily spotted. At the RTG layer they present the details of their proposed algorithm called IASN (information authentication for sensor networks). The approach is to authenticate the information sent, but not the sender. The neighbouring nodes keep track of the information routed, and can therefore find out whether it is changed. At the APP layer a set of possibilities are presented, but the results are not promising as they suffer from problems in overlooking as well as a large number of false positives. Additionally, using resource saving sleep mode of nodes is not possible, as they have to be listening the whole time. The authors conclude at the end of the paper, that their approach to use a set of several lightweight detection mechanisms is promising, as the results show, except at PHY layer, a low false-positive ratio and due to the low resource usage they are low on energy consumption.

Iwendi & Allen (2011) wrote a comprehensive paper on attacks on wireless sensor networks and demonstrated a way to simulate attacks with the intention of developing appropriate countermeasures. It contains a summary of attack schemes at the WSN

network layers and connects them to appropriate defence mechanisms in a table. In their conclusion they propose the development of a security protocol for a defensible WSN. The authors see especially challenges due to the limited communication range, calculation cycles of the microcontrollers, signal noise on the frequency and hardware attacks. The issues that are arising with regard to CIA (confidentiality, integrity and availability) are limiting the widespread of WSN in the authors' point of view.

The detection of unusual traffic has been extensively researched for wireless LANs. The use of wavelets for fingerprinting normal behaviour compared to unusual traffic has been discussed by Hamdi et al. (2008). The authors propose a multilayer framework for intrusion detection, this novel multilayer statistical approach is introduced as a set of detecting system components and uses a heterogeneous setup. The authors build this setup at multiple network layers and choose different pre-processing methods to gather the data. The pre-processing features are all based on statistics. The results from simulation determine the performance as good in total and propose for further improvement research on the use of additional detector systems to improve the efficiency when the region size to be monitored is growing.

Another proposed method employs a separate IDS which is not part of the network at all. This IDS will only detect a smaller amount of different attacks to the network in this scenario (Bhuse & Gupta 2006). Other research using IDS nodes as part of the network has proven to achieve a higher detection ratio (Hai et al. 2010). The paper proposes a lightweight intrusion detection framework where an IDS component is within every node. Also the authors show algorithms to be used for the detection. In this attempt the nodes all have intrusion detection capabilities and work together for detection, however, the determining whether an attack has happened is done at a central system. If a threshold of alerts is received at this central component, then the node is seen as not trustworthy and is excluded from the routing tables at the WSN. To achieve results network simulation was conducted. During simulation the algorithms resulted in lower energy consumption than other researchers have found. The effectiveness was

shown to be high, in low and high density network structure where collision occur more often. The low energy need was reached by reducing the transmission of alert packets. This means, that if an IDS alert is generated, but has not been sent because of the channel being busy and the same node reads messages, that other components are reporting the same alert, then the alert is not sent. They conclude with their opinion that further research is needed on the performance details.

### 3.2.3 Research Gap at Attack Detection for Wireless Sensor Network

In the paper of Ghosal & Halder (2013) interesting results and conclusions are presented.They propose further research to be done in the areas of:

- cooperative intrusion detection,
- intrusion detection for advanced metering infrastructure,
- hierarchical intrusion detection for wireless industrial sensor networks,
- energy efficient learning solution for intrusion detection, and
- intrusion detection systems in mobile WSNs.

Intrusion detection for WSN could also be achieved by a honeypot for WSN, which would be a solution for the above-stated tasks.

There is currently no known sophisticated implementation of an IDS available for a WSN. The known wireless networks do not have much in common with WSNs, they have to be distinguished according to (Mitchell & Chen 2014). This difference in structure and accompanying problems to other networks is also found in other articles of research publication such as (Kim et al. 2013). Wireless sensor networks do not have the structure and the associated problems of traditional wired networks (Zhang et al. 2003). They are sometimes in tree structures, but also can be done in other and different networking structures, where some of the nodes cannot see each other directly

although being on the same network, in structures like mesh or ring they use forwarding to extend the ranges. In the industry this is seen as a challenge because the differences between wireless networks and wired networks are described as fundamental due to resource limitation and unattended deployed large scale communication devices (Wei & Kim 2012). An attack on the network is however theoretically possible according to current knowledge, resulting in long-term loss of information. It will remain unknown whether or what information has been compromised in the time before the discovery of an attack (Lee et al. 2008). The authors describe an attack that makes use of a malicious node that behaves like being more than just one node with the aim to change routing or interfere with the exchanged information. This is called Sybil-attack. This type of attack can be detected in WSNs using a challenge-response approach. The authors found acceptable detection rate of the approach and suggest to use it in WSN and ZigBee networks. The method mitigates the Sybil-attack. Also the resource usage is highlighted by the authors as being reasonable fair.

There is a definite need for security in WSN communications, and possible improvements should include IDS. The emerging outline can be briefly described as a sensor working on different network layers. Its purpose is to detect anomalies in the net and to report these attacks. This is supposed to provide a new technical instrument for reacting to this new attack scheme. This feature-set forms the common core of a honeypot to support an intrusion detection and prevention system, for example prompting the generation of new signatures.

## 3.3 Introduction of the Honeypot Concept

A honeypot can be described with a single phrase: It is a trap. It is in general an active system in a network, monitoring every connection to this part of the network. The results are written to log files and reported accordingly (see Figure 3.2). The honeypot looks like a normal node of the network to every communication passing by. To an attacker, however, it shall appear as a valuable target (Gupta et al. 2012). A collection of nodes acting as a honeypot is also called Honeynet. The authors show possibilities

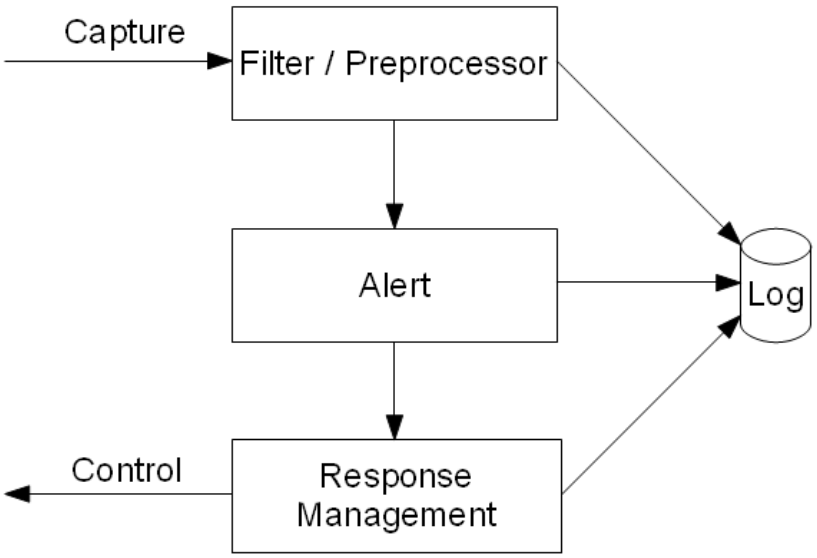



*Figure 3.2:* Honeypot working scheme

to secure a wireless mesh network with different honeypot approaches. They close with the summary of the proposed work.

The advantages of a honeypot might be summarised as follows: instead of inspecting and analysing every packet of the network like an IDS, and then deciding whether this has been an attack or not, a honeypot simply regards every connection as a likely attack (Cole et al. 2005). This behaviour saves a lot of resources because not every packet needs to be processed, inspected and filtered for attacks. Even in the context of WSN and therefore battery driven networks, the honeypot system could make do with very limited resources.

Research in the field of wireless LAN has emphasised the need of a honeypot technique. The introduction of wireless networking was accompanied by security problems. Wireless as a shared medium is prone to interception by common network sniffing tools that are already available. They were existing for wired environments and enhancing them to use wireless devices was obvious. The sniffing of wireless network traffic remains unknown to a targeted organisation, as there is no physical connection necessary at all. An attacker simply needs to be within the vicinity of the network. The author states, that deception is essential in wired networks, when attackers want to be

understood. Her paper focusses on the use of deception in a wireless environment. The common wireless attacks are analysed to find out whether a honeypot technique can help such as honeypots do in wired networks. This concludes the first paper in the row of three with the conclusion, that the wireless honeypot needs to be implemented and tested to get the results (Yek 2003). In the next paper the research setup follows a deception in depth approach. This means, that the honeypot core that is needed to be productive to keep the honeypot in total running even though attacks occur is seen in the middle, and other wireless honeypot devices are grouped around it. On the outside a set of deception wireless devices are placed, they do not have a specific functionality, are just deception devices. The research was done with network scanning tools to find out whether the honeypot routines deployed with wireless connectivity. The authors' conclusion is, that the results achieved were important to know how wireless networks security can be improved with honeypots. The wireless honeypot provided information on the deception effectiveness in a wireless setup. The outer ring was found to be running accurate, the honeypot devices instead were suffering from the load and were shown to have possible improvement. In total it is considered, that the deception effectiveness needs further research. Also running the wireless honeypot in a real-live environment is proposed, also to get results from real-world attackers (Yek 2004). This was also confirmed at later research results, stating, that the wireless honeypot was reported other than the wired honeypot at attack tools. This paper further investigates the inconsistency and tries to find out the root cause to later improve to more reliable results in comparison of wired and wireless networks. The result was showing the difficulties of accurate system fingerprinting in wireless networks in comparison to wired networks. The author concluded, that the use of a shared medium is considered to be the main issue and suggested to eliminate outside introduced effects by building a Faraday cage to find out more on this aspect. As a further result the fingerprinting was more accurate. For further work the invitation of penetration-tester is proposed to have an expert opinion on the honeypot in total (Yek 2005). The chosen reports on honeypots for wireless networks describe tools in a draft and prototype status; those

results shall be the basis for a honeypot for ZigBee networks (Siles 2007). The Honeyspot is a project that is belonging to the Honeynet project, in this case providing a comprehensive discussion on the parts and features of a wireless honeypot. The Honeyspot is going to be discussed and analysed thoroughly in the next chapter, as the design suggestions are fruitful. By now, many tools have been created that make possible to detect an attack for beginners or novices. There are now a number of honeypot implementations for wireless LAN (Siles 2007). Not all of the Wireless Honeypots work "out-of-the-box", meaning, that they do not come such as other tools as an installation file, a live-CD or a firmware for an access-point. They are even more a set of tools and configurations, that need to be brought together to have a running example. This explains the following result: A study by the European Union Agency for Network and Information Security (ENISA) instead came to the conclusion, that for "wireless honeypots" they "were unable to find any working example" (Grudziecki et al. 2012). The authors choose the set of honeypots to discuss by the ones they were able to download and afterwards install them. Discontinued and obsolete wireless honeypots were not tested.

Prathapani et al. (2009) propose a honeypot system for the wireless mesh network standard 802.11s, then still in draft status.[5] They set up the network simulator ns-2 and showed in their environment a very significant and successful result in network throughput with a ratio of high detection to low false positives during a blackhole attack[6]. A transfer of these results to real networks is estimated to be possible and similar results might be achieved in wireless sensor networks based on IEEE 802.15.4 standard communication, this assumption supports further research on this topic. For further work they state that honeypot detection is planned to be done on other types of attacks.

Gupta et al. (2012) propose a honeypot technique for wireless networks. They discuss

[5] IEEE 802.11s was incorporated into IEEE 802.11 standard in 2012, the new standard is since then referred to as IEEE 802.11-2012.

[6] blackhole attack: different name for a packet drop attack; instead of relaying the packets like it was propagated, the data is discarded

a group of honeypots, which are then called a honeynet. The authors differentiate between different types of honeypots such as honeypots for research and honeypots for production environments. In detail they propose using a concept called honeyPHARM described by Hassan & Al Ali (2011), their solution has specified for the 802.11s wireless mesh networks. The additional use of wireless technology at the nepenthes system together with the pharm component to build the so called honeyPHARM is proposed in both these papers. The authors Hassan & Al Ali (2011) propose the use of the available product called a honeyPHARM, a distributed honeynet for the collection of malware in networks. Collected data will be used to discover new attack methods. This will be an addition to the nephentesPHARM system. The authors were looking forward to combine the PHARM with Dionaea, a honeypot system that was not yet compatible by the time of writing the papers.

The authors Raj & Thilagavathy (2012) propose a solution to a specific type of network load caused by so called jamming attacks in wireless sensor networks. Due to the amount of commonly available radio channels in these networks, shifting to another network channel is suggested each time a jamming attack is detected. The authors claim that a residual network activity should keep the channel busy for a limited time during the short interval in which the jamming is not taking place as to make the jamming attacker believe that the jamming attack is successfully disturbing the WSN communication. As this could even also attract an attacker and act as a decoy for the attacker, this behaviour could also be described as deception, or, simply put, a honeypot or honeynet.The use of honeypot works together in this case with the channel switching proposal to have a positive effect on the network recovery. The paper discusses finding an optimal channel switching algorithm to enhance connectivity in the presence of jamming. a monitoring node considers a channel switch and sends a notification message to the network components, this means that the jammed channel will still be used and the jamming is likely to continue. However, the real packet transmission is done at the new channel. The authors see further research at nodes missing the channel switch command as well as node schedule synchronisation.

The authors Muraleedharan & Osadciw (2009) propose a framework using a honeypot technique combined with swarm intelligence for a "battlefield monitoring" application in a wireless sensor network. They state that traditional security schemes can not be applied in the field of WSN due to resource constraints. They planned to build a simulation to prove this proposal, and validate the results. The authors of the paper propose an efficient way of detecting an intruder using Honeypot as well as swarm intelligence, hereby the honeypot enables the tracking of intruders. The authors found this to be an effective way to reduce denial of service attacks. With the knowledge of existing attacks the further attack patterns can be estimated by the use of pattern recognition. The sooner an attack is detected the more reliably the network is to be defended. The honeypot network is using game-theory, meaning that there is a value that is traded, and if the value is good this increases the likelihood of a node being on the route for a message. Also the trading process makes the honeypot network look like a usually productive WSN. For future work the authors want to use network simulation to produce results. As a conclusion the authors state that the honeypot renders the detection and tracking of intruders ideal with regard to cost, time and availability while also having a good sensor threshold.

As a first introduction to honeypots, a look at the relevant literature should be undertaken here.

The book from Spitzner (2003*b*) called "Honeypots: Tracking Hackers" can be seen as one of the primary literature sources on honeypots. He is also the founder of the Honeypot project (Spitzner 2003*a*)[7]. Spitzner defines a couple of very fundamental statements for the community of honeypot researchers, developers and users. His honeypot definition is:

> "A honeypot is a security resource whose value lies in being probed, attacked or compromised".

[7] http://www.honeynet.org

The subject of a honeypot is to lure the hacker in, to let him interact with it and let the attack happen; enduring as long as it takes while everything is recorded in the background.

He also mentions one of the first publicly known honeypots from the mid-1980s, where the IT administrator and author Clifford Stoll from the book “The Cuckoo’s Egg” found out that an attacker had gotten access to his infrastructure. But instead of wiping the compromised accounts he decided to investigate the attackers behaviour (Stoll 1989). He further differentiates between production honeypots in the sense of increasing the security of a network and to protect an organisation, and a second type of research honeypots with the purpose of gathering information about vulnerabilities, threats and attackers with their tools, motives and tactics.

One example of an SSH shell honeypot is called Kippo and was developed by Tamminen (2009). In this case, a prepared honeypot seems to be a valuable target to the attacker, showing a seemingly standard full featured shell, which can be used to configure the system. Changing passwords or downloading packages from other servers is possible. But in all these cases the installation of the packages will fail, and also the execution of downloaded scripts is not working properly and presents the usual error messages. The way that the attacker in the honeypot interacts with it gives a lot of information for further analysis. There are numerous log files available for Kippo that can be watched in a playback mode and can be analysed further. Kippo was developed further on and this lead to a new project called Cowrie, that has the Kippo sources as a base and builds further features on top of it. One enhancement is the simulation of insecure telnet protocol available besides ssh (Oosterhof 2018).

The honeypot’s purpose is that it is there to deceive and entrap an attacker and exists as long as possible to hold and investigate him. With this attraction to a seemingly easy target, casual attacker may be distracted from the really important systems. It is not wanted that the attacker remains there for a long period of time, but that he suspects under no circumstances that he’s fallen into a trap. Although there is also a

paper by Wagener et al. (2011) that describes the behaviour of attackers when they recognise a system as a honeypot. Their honeypot is an SSH emulation that after a while of interaction insults the attacker, some of them leave instantly, some insult back and some start to do research on the honeypot and its logic, trying to understand it. The honeypot called Heliza contains a self learning component, which is configured to keep the attackers as long as possible inside the decoy. To achieve this the commands that can be executed at a shell are processed such as wget is always possible, so that in the logs will be presented from which IP data was requested to load onto the system. Unlike an unpacking algorithm, which will not be executed to do extraction of software from the archive just downloaded. Machine learning was found to be an effective way to configure the honeypot by itself to an optimum. The behaviour of Heliza changes over time and is depending on the history of the commands already executed. The authors conclude, that Heliza is an ideal source for studies on the attacker itself. Is there something to learn from the timing of the keystrokes, or from something happening if Heliza print an insult message in a language that matches the connecting IP range and so on. The authors also state, that the honeypot Heliza has its limits, it is likely possible, that experienced hackers find possible ways to still execute commands even though Heliza should prevent this. Therefore the authors propose on further research on a test setup where an experienced hacker is trying to circumvent Heliza. In penetration testing terms this would be called a white box testing, where all information is given to the tester before the penetration test is executed.

In the paper by Qiang & Yuqiang (2013) the use is made clear once again like previously described. The authors also state that a honeypot is a playground for an attacker and that the results can then be used to increase the security in networks. The aforementioned final step with the use of samples from the authors looks similar, because they focus on wire-networks to generate patterns that can be deployed again to an IDS in the network. This is presented as a cooperative system of honeypots together with intrusion detection system. The authors call it CSHIDS. It is a chain of three worksteps, called concealment, data-capture and security, that is beginning at the honeypot

to collect the attack data, another system uses the information from the logs to generate patterns out of the labelled attacks and generates attack rules with the intention to react to a new attack pattern. It was shown that new attack detection is effective in the experiments using simulation.

In Honeynet Project (2002), the authors state thathoneypots are systems designed to be compromised by an attacker. Using activity monitoring with a honeypot makes reasonable sure which way an attacker took to compromise systems. By having a honeypot next to the production systems the intruder will probably leave traces there. The authors define a set of honeypots in a network as a honeynet and two critical elements: data control and data capture. Data control is defined by the authors as being stealthy to the attacker, who should not be able to find out that the traffic is controlled. Data capture means that the logged traffic is not allowed to be detected or destroyed. Therefore reasonable storage and access control has to be in place.

This was refined by Spitzner (2003*b*) where the author states two more requirements for a honeynet, namely collection and analysis. Collection means, that the gathered information off honeypots forming a honeynet need to be collected and centralised to have the highest possible value. This data collection then is used for the fourth requirement, the data analysis, which is the opportunity to analyse gathered data to glean information from the honeynet of honeypots.

So the basic elements of using honeypots forming a honeynet are:

1. Data capture
2. Data control
3. Data collection
4. Data analysis

The design of a honeypot for wireless sensor networks needs to fulfil the needs of these four aspects.

Another valuable resource on getting honeypot background information and a decision list on how to handle the tasks that come with a honeypot is the work by Grimes (2005), which contains a list of basic steps to understand what a honeypot is not and what it might be, and also a list of points what a honeypot definitely is. As stated before in the honeypot basics, this is a resource that is destined to be attacked. The author states, that there exist several generations of honeypot types. The generation "I" honeypot is a single host that acts as a honeypot, whereas the generation "II" honeypot is a combination of tools that safeguard the honeypot once it gets compromised by an attack. The further generation "III" honeypots differ from "II" in the way of their analysis features according to Provos & Holz (2007). However, the first requirements of honeypots are still always:

1. data control
2. data capture

According to Grimes (2005) "Data control is making sure the compromised honeypot is not used to attack other legitimate resources. Data capture is recording everything the hacker does.".

He continues and generates these requirements from the first ones:

1. Store collected data remotely. You want to store as much evidence as you can remotely. If it is stored locally, the hacker can find and erase it.
2. Don't let the hackers discover your monitoring devices. If your monitoring tools are discovered, the hackers could disable them, delete the collected data, or just avoid the honeypot.
3. A honeypot should strive to look like a production asset.
4. Your honeypot system should be designed to prevent a compromise to the production network. This means, that the hacker should never have access to legiti-

mate data, systems, or user accounts.

### 3.3.1 Honeypot Categories by Level of Interaction

Honeypots are divided in literature into categories depending on their level of interaction.

#### 3.3.1.1 High Interaction Honeypot

A high interaction honeypot is explained in the statement from Bechtold & Heinlein (2004), where the authors make clear, that a high interaction honeypot can be a non-virtual system, once set up and not well maintained but with no real functionality on the network. It is constantly monitored and interactions with it captured. Regarding advantages and disadvantages of this approach it can be said, that it has both: a definitive advantage is the fact, that it can not be detected as being a trap, because of its high interaction level and mostly also the installed vulnerable services are not emulated, but real. If this system still gets effectively exploited without this event being monitored and detected, this leads to a clear disadvantage, because obviously, this compromised system is one in a possible row of such in the network, that can be used for further attacks. This possible drawback must be circumvented! In one of the first tryouts of Lance Spitzner to make a honeypot the attacker simply wiped the whole harddrive of the honeypot after detecting that all of the attacks to the Internet were blocked by a firewall (Spitzner 2003*b*).

#### 3.3.1.2 Low Interaction Honeypot

A low interaction honeypot also has its advantages, for instance it is created with the intent to not and hopefully never get hacked. All the services present are only simulated and do not exist. But due to all these simulations and most often also the fact that the operating system itself runs on a virtual machine and not a standalone one, this can be a clear disadvantage, because an attacker can figure it out when exploring a honeypot system. Research on this topic is documented by Holz & Raynal (2005) and Wenda

& Ning (2012). So the results might therefore be somehow not as accurate as on a physical machine. Nevertheless, this is the more secure way to build a honeypot in the context of traditional wire-bound network hosts according to Bechtold & Heinlein (2004) and Cole et al. (2005). The statement by Holz & Raynal (2005) is, that the high-interaction honeypots are set up for so called script kiddies solely as a logging platform for the administrator, they are efficient, but prone to be found out automatically as honeypots. Further work was presented as every attacker to be found needs its own kind of attraction. This means, that a low level script kiddie needs another bait than a high level expert and a honeypot administrator has to consider this circumstance. The statement by Wenda & Ning (2012) is, that for the attacking party it is crucial to detect a honeypot. The authors collect information for a determination whether a system is a honeypot. This information can further on be used to find out how to improve honeypots to not so easily being fingerprinted as such. At the end of the paper the authors collect all relevant facts on the detection methods and are elevating the result of the experimentation to the further research on this topic.

The survey by Bringer et al. (2012) gives a good overview over 60 papers since 2005 regarding honeypots. They have put the recent development into five categories:

1. new types of honeypots to cope with emergent new security threats,
2. utilizing honeypot output data to improve the accuracy in threat detections,
3. configuring honeypots to reduce the cost of maintaining honeypots as well as to improve the accuracy in threat detections,
4. counteracting honeypot detections by attackers, and
5. legal and ethical issues in using honeypots. Bringer et al. (2012)

They state, that their “literature reviews indicate that the advances in the first four areas reflect the recent changes in” their “networking environments, such as those in user demography and the ways those diverse users use new applications.” For the ethical

ascpects they state that their “literature reviews on legal and ethical issues in using honeypots reveals that there has not been widely accepted agreement on the legal and ethical issues about honeypots, which must be an important agenda in future honeypot research.” Bringer et al. (2012). Also very essential is the fact, that Bringer et al. (2012) state that there is only one approach discussed by Prathapani et al. (2009), that gives a possible solution to a black hole attack in wireless networks by a set of message requests and responses that when found compromised lead to the result of a currently ongoing attack happening during simulation runtime.

Observed attacks on a network should also be analysed later on to devise countermeasures. This could lead to an enhancement in network security. The reactions may be passive and not interact with the attacker at all. Active reactions on the other hand could include countermeasures such as alerting the rest of the network to the detection of an attack and flag a specific part of the network as not trustworthy, which should then be ignored for routing purposes. The protecting nodes defend the working network through directed responses upon detection of potential intruders (Das 2009). In their paper a list of honeypots is provided by the agents to the clients at the time of network establishment. The roles in this network will change over time following a predefined schedule in order to attain a lower false positive intrusion detection rate and to render these networks harder to attack. Furthermore, the clients are forbidden from using the honeypots for networking purposes since connections to these honeypots would be regarded as an attack.

A paper by Mostarda & Navarra (2008) suggests assigning specific roles in wireless IDS. The roles shall then change over the lifetime of the established network. Some of the nodes are normal routers and forward the traffic, others have sensing functionality and are part of a distributed IDS . This approach seems to fit well in the context of wireless sensor network honeypots. This work uses another technique to specify intrusion detection patterns and attack signatures, it is a specification-based IDS. It is done by an IDS specification that is transformed by the detection in the sensors. The local

detection then forwards the information over the network to the other sensor nodes. In this setup the possibility to put the nodes into sleep mode is still available, issues arising with the sleep mode are discussed by the authors as well. The evaluation took place as experiments in sensor networks without attacks in comparison to attack and the authors point out a good performance at an overhead of around twenty to fourty percent of energy consumption increase.

Kaplantzis (2006) introduces and describes a conceptual honeypot system without discussing an implementation under real-life conditions. Further development remains to be done. There is still a gap in the research of honeypots for wireless sensor networks. The idea of a honeypot as alternative approach to IDS and its proposed features should also be implemented in WSN. Such a detection entity should span different layers, starting from the physical and MAC layer and reaching up to the high layers such as ZigBee has (Kaplantzis 2006). The process of pattern generation for an intrusion detection system is described by the author as follows. The signature generation is considered to be automatically by using the advantages of packet capture of a honeypot. Her approach follows the work of (Kreibich & Crowcroft 2004). The authors describe the so called Honeycomb. It is a honeypot that features signature generation that then can be used in the intrusion detection system. They made use of a tool called honeyd, which is a low interaction honeypot, honeycomb is a plugin for honeyd. The signature generation can itself be a problem, because with an internet worm, the worm payload itself can be part of the detection string.

In (Qassrawi & Hongli 2010) regular clients are also forbidden (like in (Das 2009)) from using the honeypots for networking purposes, since connections to these honeypots would be regarded as an attack. Since the roles of nodes in this network should change over time between honeypot nodes and regular nodes, a predefined schedule will be used to guarantee a low false positive intrusion detection rate. This method renders the network harder to attack. By using this method detecting new attack methods is definitely possible with honeypots. The rest of the network will be left in a functional

state while the honeypots are under attack. As an additional benefit, new countermeasures can be prepared while the attack is still ongoing. All these presented uses of honeypots are suitable to increase the reliability and availability of ZigBee networks (Qassrawi & Hongli 2010).

#### 3.3.1.3 Single Honeypot Node Comparison to Honeynet of Honeypot Nodes

The discussion by Hai et al. (2010) on the use of a single networking node (see 3.2.2) did not yield satisfactory results in comparison to a single intrusion detection sensor in a wired network. This is due to the possible configuration of WSN nodes to form different topologies (Krontiris et al. 2009). These could be a star topology, or a meshed network. In meshed networks, each node in the network has only a few peers to communicate with. The authors of the paper describe a collaborative identification of an attacker. They made use of simulation and at the end of the experiments also described the identification of a malicious node in a real setup. In this setup they made measurements to find out how much effort is needed for the identification. The future work is described as concentrating on the identification of more than a single attacking node. A single node can therefore only detect attackers within its direct communication range (Shin et al. 2010). This prevents the detection of attacks in other parts of the network which are not routed through this part of the meshed network. The proposed schemes in the hierarchical detection were done with experiments. The importance of clustering is emphasised. The authors state that a hierarchical approach is useful to secure industrial production and WSN. Islam et al. (2010) discuss a solution to this problem in their paper which focuses on hierarchical intrusion detection systems. They assumed a routed tree topology, and implemented the detection mechanisms in the main routing nodes. The authors make clear, that the use of a four layer architecture to identify an attack is successful in the specification based detection. The topology is divided into the sensors, then two layers of monitoring devices and then at the the top the base station of the WSN. Further research includes the setup and experiments in a simulator.

The topologies a WSN or a ZigBee network can make use of were described previously in section 2.1.7.

Several technical details are worthy of discussion. One of the proposed features of a honeypot should be that it might be temporarily part of the net at various different times (Huang & Lu 2012). It should not always be next to the same node in the network, because these nodes in the net might not be the target of the attacker. One of the proposed features named here involves cloning. Those resources of special appeal to the intruder might duplicate themselves virtually during the course of an attack, expanding the target range and thus reducing the likelihood of an attack on the real network (Huang & Lu 2012). The paper gives details on the cloning to be dynamic. As described in the introduction of the network coverage model section, the covered area is to be maximised to have the best effect. The cloning algorithm chooses the variation as well as the range of particles. Also the amount of clones and their position is determined by the algorithm. The paper closes with checking the algorithm validity and analyses the experimental results to be an effective way to determine ideal sensor node placement. Further research is proposed on energy consumption centric optimisation of network coverage.

### 3.3.2 Robustness

This section is about robustness of networks. The network robustness is a summary of different approaches. It is shown that when in a meshed network a significant removal of nodes happens, then the network is split. As a result of the split a set of clusters arise. In wireless sensor networks the network should stay stable, therefore it is of interest what other researchers do and find out on this topic on how to keep a network in a stable status. Whenever such a WSN is to be established, it is customary to deploy a larger amount of nodes than minimally required as a best practice method to ensure higher reliability of communication and interaction through redundancy (Cai et al. 2011). Such redundancies might be needed for several reasons: some of the nodes might run out of battery power sooner than others, or the signal strength of neighbou-

ring nodes may deteriorate, or other wireless communications are taking place on the channel and frequency used, or interfering with the mesh network, or natural radio activity has increased within a period of time (higher ambient noise), or just plain stability of the network. The authors state, that they can always find a critical number for sensor density. This means, that when this threshold value is undershot, then the network connectivity is falling into pieces, but and just a bit above this value the network reliability is high enough. Throughout the paper they have proven this to be true for a set of simulations for different sensor network deployment schemes, whether they were circularly distributed or hexagonally distributed. The result is that they can determine the point of density of sensors when there is a sharp decrease of connectivity when nodes fail to work.

The authors Min et al. (2010) designed and implemented the security communication services of WIA-PA, a wireless sensor network standard. This standard is comparable to ZigBee, but features improved robustness when compared with ZigBee due to channel-switching. According to the authors the security in a WIA-PA network has two levels. The security manager is located at the first level and the security manger agents are at the next level which is providing these policies to the end devices. Different encryption strategies are present in WIA-PA by design, this also includes the key management procedures for these strategies. This also includes end to end encryption as well as point to point encryption, located at the application layer and at the data link layer.

The authors Radmand et al. (2010) did a comparison of WSN standards that are are available for industrial application. They focused especially on improvements with regard to the weaknesses that ZigBee has in the domain of the oil and gas industry. The authors present an overview of existing vulnerabilities in ZigBee. They found that even though lots of vulnerabilities were found and closed with the new ZigBee standard released, some of the vulnerabilities were still present in the standard. This means, that even with the implementation of the new standard, and the encryption methods that

were enhanced with it, attackers have the possibility to execute attacks even without having compromised the key material beforehand.

Another paper focusses on robustness and fault-tolerance in wireless sensor networks (Ghosh et al. 2011). An interesting fact is that the authors use the problem with regard to biological information flow to map it to a wireless sensor network, the robustness of a genome is the background to construct similarities. They propose the use of the attractor theory in the gene regulatory network to create fault- tolerance in WSN routing, a comparison between genetic networks to wireless sensor networks is proposed. The authors conclude, that the models are proven and the future work will concentrate on using this knowledge in a simulation.

The article by Zand et al. (2012) describes communication standards for sensor networks and has a look on the "Wireless interface to Sensors and Actuators" WISA, based on physical layer of IEEE 802.15.1, and 802.15.4 based ZigBee, WirelessHART, WIA-PA and ISA 100.11a (developed by the "ISA" International Society of Automation) for the use of wireless technologies in industrial monitoring and control networks. They state that the wireless communication have "the potential to provide fine-grained, flexible, robust, low-cost and low-maintenance monitoring and control." They state, that "they still have severe limitations, especially when real-time, reliable distributed control operations are concerned." Future research topic suggestions are seen but not concluded in details.

Alcaraz & Lopez (2010) have reviewed the standards ZigBee PRO, WirelessHART, and ISA100.11a. They considered their coexistence, reliability, and security in their communications. They have also identified a set of threats and potential attacks in their routing protocols. In conclusion they provided recommendations and countermeasures for each one of the presented potential attacks to the three wireless standards. In their paper the authors have split the threats on the standards into three groups regarding the attack surface of confidentiality, integrity and availability concerning attacks. This is called the security model for the classification of attacks. They state, that from their

research most attacks on WSN are performed by insiders that are authorised to do interaction with the networks. From their research the three standards observed can be attacked the same way on confidentiality. ZigBee PRO and WirelessHART have the same attack surface for integrity concerning attacks. WirelessHART and ISA100.11a seem to be a bit more vulnerable to denial of services attacks, which is due to the fact that their routing can be manipulated by a malicious node, which then generates false routing tables which makes them prone to worm- and sinkhole attacks, which attacks are not present in ZigBee PRO by design if the ACL possibility is used and configured properly, rendering only trustworthy nodes to be able to become part of the network. Furthermore, they suggest hardening the key renegotiation process, as this is possible to be done without encryption. For a set of attacks they suggest the use of a separate intrusion detection, due to the fact that this cannot be solved otherwise by improving the standards. For ZigBee PRO this is seen at:

- Routing Falsification
- Sybil
- Overloading
- Flooding
- Selective Forwarding
- Black Hole

And for WirelessHART and ISA100a as written before, the worm- and sinkhole attack can also be detected by an IDS.

### 3.3.3 Hardening

The goal of protecting a network from a successful attack is supported by hardening. This term is defined as the improvement of a technology for the reduction of attack vectors (Barnett 2006). This can for instance also be achieved with the application of

encryption methods. The intention is an improvement of reliability and availability (Cole et al. 2005) of network nodes. The author presents a guidance on how to configure Apache properly, as this is seen as a challenge, The author explains the way to harden web servers as well as a view on web proxy, a software program that executes as both server and client to do requests for clients. The book gives an insight into web server security misconceptions commonly seen. Hardening the web-server means for the author, that ports available are reduced to the minimum. Additionally the operating system needs to be hardened in the way that vendor patches available have to be put in place in a timely manner, unnecessary daemons should be disabled, access management has to be implemented and access control has to be put in place. Configuration hardening is a very important topic, but the author also gives insight into web application scanners that are available for free.

There are already guidelines for manufacturers regarding the many possibilities in the configuration and in the design of ZigBee networks. These have been written in part by governmental research departments. One example is the security guide by (Masica 2007) for the U.S. CERT. A document (Bundesamt für Sicherheit in der Informationstechnik 2009) by the German BSI (German governmental institute for standards and best practises in network security) discusses possible problems in ZigBee installations and provides advice with minimal requirements for the development of new products using ZigBee networks. Some of the examples are:

- refraining from over the air programming and plain-text key exchange.
- using MAC address filtering
- planning the channel usage to minimise interference,
- planning of redundancy for robust network,
- protecting the ZigBee Coordinator/Gateway.

A honeypot system should be a hardened system. Singh & Verma (2011) list a number

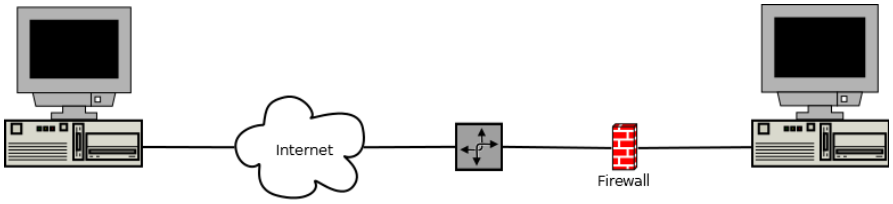



*Figure 3.3:* Honeypot host being attacked via Internet

of possibilities for hardening a WSN in the paper “Security For Wireless Sensor Network”. They also show a list of unresolved problems, and state that these problems are difficult to solve. The authors have pointed to issues related to security, in which attacks occur simultaneously on the WSN. The attacks can happen at different protocol layers of the network architecture.

### 3.3.4 Summary of Defence Strategies

Therefore, to clarify this once again unequivocally, a honeypot can not be portrayed as active line of defence, which it is not. But by the use of honeypots, it is possible to perform a detection of attacks, which is a first basic part of a line of defence, which was previously not possible. A honeypot in a traditional setup can be either set up standalone, or set up with a corresponding production network. As a standalone solution it is solely present at a network connection. For illustration purposes the following honeypot diagram is shown, showing a possible setup of a honeypot. The first diagram 3.3 shows an attacker over the Internet reaching the honeypot host on a single host computer, a firewall has to be present to safeguard the Internet should the honeypot node been taken over.

The next diagram shows for illustration the following honeypots’ purpose in a set of attack decoys, different setups on different machine types, the variety offers a greater “playground” for an attack to happen. The diagram shown is a possible setup of a honeypot combination at figure 3.4. It shows an attacker over the Internet with the honeypot hosts on a number of host computers, again a firewall is present to safeguard the backwards possible attack after taking over a honeypot component. The advantage

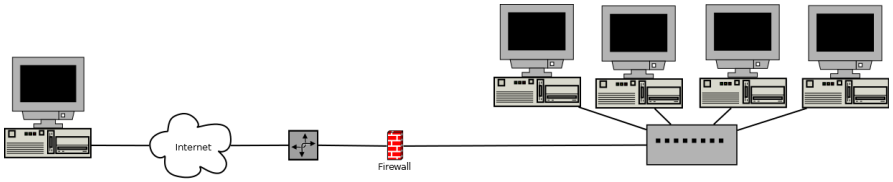



*Figure 3.4:* Several honeypots are reachable via Internet

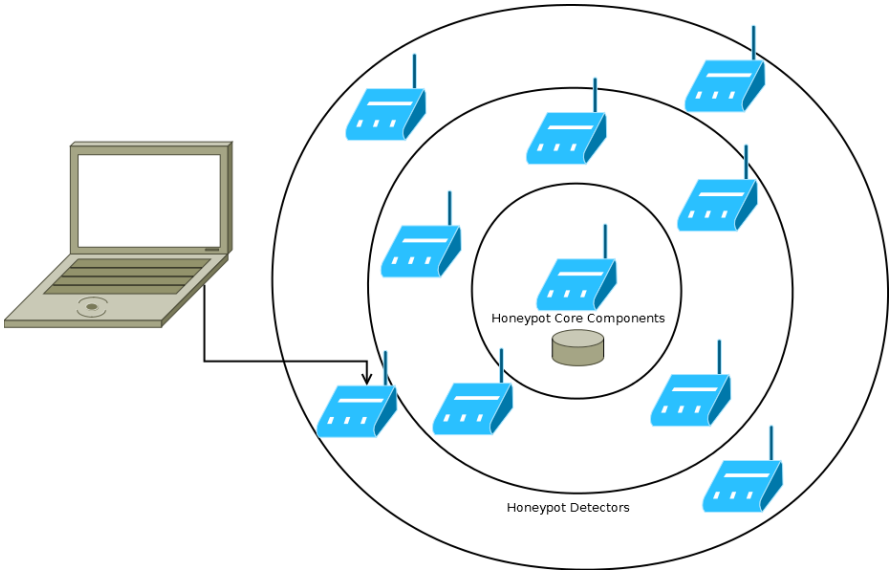



*Figure 3.5:* WSN honeypot illustration

is, that attacks to a variety of system components running under different operating systems can be detected, in principle it is possible to have just a single host device, that appears as a variety of virtual hosts as described in the literature by Provos & Holz (2007).

As a summary, administrators can gain additional knowledge about the course of an attack. Afterwards they can improve their own response and safety procedures. There are plenty of reasons that really speak for the use of honeypots, and those reasons are justified, supported and conclusive. For wireless networks, literature has shown, that the concept in wireless networks is found to be in a circle structure, due to the wireless signals being stronger at the center. The most important parts of the honeypot are located there, for instance the devices that log every information to disk, see section 3.3. In the paper discussed there, the deception in depth approach with a honeypot core

needed to be productive and other parts of the honeypot grouped around it was presented. This approach is proposed for the wireless sensor network honeypot as well, in later parts of the thesis this is described. However, for illustrative purpose, an overview is presented as follows, see figure 3.5.

## 3.4 Feasibility of the Solutions

This chapter takes a look at the solutions and discusses whether they seem to be useful for a wireless sensor network setup or not. During a previously undertaken literature review it was made clear, that with WSN setups the costs differ from other installations. Some solutions are easily implemented or have been input to other standards, other solutions need additional network components or in some other way additional resources, an overview is presented.

### 3.4.1 Costs in the Various Standards

The authors Jin-Shyan Lee & Shen (2007) compare the wireless standards IEEE 802.15.1, 802.15.3, 802.15.4 and 802.11a/b/g based on their technical specifications, and refer to their throughput, cost, and specific throughput as to cost. They emphasise, that they do not give recommendations, as every wireless network has its own purpose and the scope varies from use case to another.

The design of wireless sensor networks relies on low power consumption through low computational need as well as low transmission power. Any security mechanism in these networks should therefore be lightweight. Researchers have focused on lightweight security mechanisms to address this fundamental need for security together with power saving features.

The security features of ZigBee also limit their ease of use and render interoperability problematic: Encryption details have to be distributed and managed, requiring a bigger feature set as well as more network traffic (Faludi 2010).

The paper by Fabbricatore et al. (2011) describes these problems in the area of smart

homes. The authors see several problems for existing home automation solutions involving ZigBee. They propose using additional cryptographic extensions on top of ZigBee, or to change the protocol, should ZigBee remain too insecure for their purposes.

A defence mechanism such as intrusion detection should react early on, due to the fact, that an attacker will continue the work on the system until an attack is successful. Ignoring an attack completely is in general not advisable (Cole et al. 2005). Countermeasures like ignoring a part of the wireless sensor network for the time a detected attack takes place, helps to save some of the limited amount of battery power (Wood & Stankovic 2002).

### 3.4.2 Cost of Encryption/Improvements

In ZigBee, several security features are implemented by design. But due to the increased computational effort and a higher amount of data transmitted, these improvements result in a higher power consumption.

This is because security needs computational power and thus results in a higher energy consumption (Faludi 2010), therefore this is in direct conflict of nearly every WSN task and feature.

Successful attacks might last as long as the attacker wants, even when encryption is used to protect the communication. These attacks can not even be analysed, because usually, there is no log of network activities and hence, attackers will leave no traces. There is no “packet capture” of attacks, and the probed exploit code is not available in the wake of an intrusion attempt. An attacker will thus not get noticed and therefore, will remain unidentified. It is hardly possible to gather the leftover information for forensic analysis (Cole et al. 2005).

There is hence a need for an intrusion detection system, and also for a system reporting the break-in attempts.

### 3.4.3 Cost of Hardening

Having blind faith and trust in the existing countermeasures is greatly unjustified. Radmand et al. (2010) showed in their comprehensive paper, that the cryptography used in ZigBee is not encompassing enough to cover all possible attacks. Some of the challenges, like confidentiality, are solved encrypting the data stream. Yet, issues on the network layer such as replay attacks still remain. They also state, that the manufacturers should provide a minimum of security, due to challenges with encryption keys stored in plain-text in the nodes. These could be extracted by an attacker tampering with physically captured wireless sensor network nodes.

Several issues were solved according to research conducted since the release of ZigBee PRO 2007, but a few issues still remain. Also, a flaw in the key distribution algorithm of ZigBee 2007 was formally proven in a publication by Yüksel et al. (2011). Conceptual issues cannot be solved through hardening, when staying compatible with the standard is necessary.

Hardening a system to resist an attack will not stop the attack itself from taking place. There are clearly still challenges left in the process of improving the security features of wireless sensor networks (Yüksel et al. 2011).

### 3.4.4 Cost of Intrusion Detection Systems

Processing every packet, its inspection, and the detection of attacks by matching patterns all require a lot of resources. In the context of WSN networks, and therefore battery driven network nodes with very limited resources, this is a clear drawback of an intrusion detection system approach in wireless sensor networks.

### 3.4.5 Cost of Honeypots

One of the drawbacks of a honeypot system approach in wireless sensor networks could be that it exposes additional targets which might get corrupted and then be taken over by an intruder. But since any other part of the net could equally get compromised,

this is a small drawback. The advantages of a honeypot far outweigh its hypothetical disadvantages.

The installation and maintenance of a single honeypot or of a distributed honeypot network clearly requires additional effort. Yet, this effort should be made on top of common improvements to the wireless sensor network like configuration hardening and regular firmware updates. Some implementations that offer over-the-air programming features for updating nodes already exist (Unterschütz & Turau 2012).

### 3.4.6 Legal Aspects of Attack and Defence

The legal aspect of security is worth of discussion, but will not be done very comprehensively in this thesis, as it references jurisdiction, ethics and morale, all location-dependent. This topic is not the focus of this work, nevertheless some of the aspects are shown and presented here that can be found to be written by computer science researchers. When an attack is happening it is allowed to defend, which is true for our real world scenario as well as for a “cyber”-scenario (Koch 2008). This means, that it is also allowed to prepare for an attack. But the usage of hacking-tools for defence is only found to be legal during the time-frame of an attack. In the time-frame right next to the attack, before that and afterwards it tends to be not allowed, and if an attack is not happening, if is not allowed at all. This means, the usage of such hacking-tools and methods is very limited (Koch 2008). Another way of a so called “hackback” is the shutdown of malware-networks, which are established with criminal intention, devices in a network are taken over and various methods are used to create a benefit for the attacker. To shut down such networks, specialists work together to analyse such malware-components and to find ways to disable them from spreading, in the best case to shut down the whole malware. Analysis f.e. for the Zeus botnet can be found from (Andriesse et al. 2013). This Zeus derivative was taken down in 2014 and further reading is found at Sandee (2015).

### 3.4.7 Legal Aspects of a Honeypot

The work of Radcliffe (2006) and Radcliffe (2007) describes the legal aspects on honeypots, with a focus on US laws. Those readings are proposing the use of honeypots, but every step has to be documented. Also, a recommendation to get a management approval is given. With this documentation and an approval the legal system in the US is satisfied, should it be necessary to defend the action taken to set up a honeypot.

Legal issues in using honeypots were also described by Rubin & Cheung (2006). They focus also on the ethical aspects of putting honeypots into production. The author poses a set of questions besides the general recommendations and legal discussion that is seen in other papers, too - they discuss the subsequent steps, which have to be done once an attack was monitored by a honeypot. There is no guidance by them on what to do then they even argue, that this does not have necessarily to be reported to other parties such as law enforcement, especially if nothing of value is in conflict.

Further reading can be done in the paper by Scottberg et al. (2002). It presents a number of legal issues and concludes with:

> "The sociological and legal issues behind the use of honeypots in cyberspace present new challenges that do not have direct precedents in the physical world."

Another paper describes the fact in US law (Mokube & Adams 2007). The authors state that there are three core concerns when establishing honeypots with regard to legal operation. The first one is so called entrapment, which is present when the honeypot is set up and running for some kind of government-owned institution. So, entrapment can not be used when the honeypot is running for a non-governmental administrator. This is also true for non-governmental organisations with regard to privacy concerns. As long as a normal user is not monitored, it is not considered privacy relevant. With regard to liability there is a potential issue when the honeypot could be used to carry out

further attacks. The authors state, that this can be solved by documenting everything the honeypot does beforehand and then going into development and implementation. They conclude, that when due diligence is conducted properly, then the law is obeyed and there is just 'a minor issue' remaining (Mokube & Adams 2007).

Additionally the diploma thesis of Baumann (2002) contains a chapter about legal aspects of honeypots in Switzerland. It states, that

> "Running a honeypot in a legally safe way is nearly impossible."

Continuing reading other authors are focussing on legal aspects of honeypots (Dornseif et al. 2004). They state that administrators setting up a honeypot have a special responsibility that the honeypot is not used as a stepping stone to attack others. In criminal law they argue that if a honeypot is used to attack a third party, then the administrator might be punished if this help was performed willingly. So if the honeypot is set up very precise and accurate, then there is no intention for it to be a stepping stone. Therefore, they deem this not intentionally and argue that tort law is not applicable.

Further reading, another author is focussing on legal aspects of honeypots and honeynets (Sokol 2015). This states that there are two main concerns when using honeypots. The first is privacy and the second one is liability. The paper then continues with a lot of details in EU law on these topics and suggests to use a banner when a connection is established, that every connection by an authorised user is recorded. It is also recommended to make sure the EU territory is not exited by data traffic, so that there is unambiguous certainty which laws have to be applied. Regarding privacy law they consider an attack in the knowledge that every step performed on any system is logged. Every attacker therefore is already accepting that each and every step is monitored. Liability of the administrator remains an issue, when the honeypot is used to carry out other attacks without being noticed. When being detected in a timely manner, this is deemed unproblematic.

The next paper discusses civil and criminal liability for honeypots. It also presents a

section of attackers 'liability as well as administrators' liability(Sokol & Andrejko 2015). On the topic of civil and criminal liability the paper discusses issues of liability and how they can be targeted.

A further paper is regarding privacy topics in honeypots. It supports a basis for the processing and collecting the data that is captured at the honeypot with regards to EU law (Sokol et al. 2015). It focusses on data processing as well as data collection. With regard to EU law the topic of privacy is discussed for honeypots. A special interest is put on data retention, as the requirements of data retention differ for EU members due to the EU Data Retention Directive.

As a summary, the paper by Sokol & Host (2016) can be seen as a survey on the legal aspects of honeypot in the EU. It describes the evolution of honeypots, and connected to it, the changes in the legal aspects with the honeypot changes. It is the first paper by the authors which has also a forensic focus on digital evidence retention. It emphasises to anonymise evidence before they are published to the public. A further paper focusses additionally on the privacy issues that can be generated by the use of honeypots and honeynets, with regard to the GDPR (Sokol et al. 2017). Especially with the need to anonymise IP addresses that can be logged by the honeypot, there is further research suggested. Also sharing the honeypot data across borders need further research for a thorough position and recommendations.

With regard to local law an article by Fraunholz et al. (2018) gives a deeper insight into german law with an outlook to EU legislation. They are the first to bring up the topics of copyright and self-defense into the legal discussion on honeypots. Their conclusion is following afore-mentioned recommendations, namely to make sure for a honeypot administrator that it is documented accordingly and that applicable local laws have been determined and addressed accordingly in the honeypot setup and configuration.

However, wireless honeypots are still an under-represented topic, also in the legal issues.

## 3.5 Summary of Findings in the Literature Review

The previous chapter featured the wireless sensor network standards and the security and defence solutions if available to possible attacks on WSN and their countermeasures as well as the costs for the security.

All the previously shown information, considered as a whole, shows that a lot of research has already been done in the field of securing ZigBee networks, regarding integrity and confidentiality. Defending on attacks regarding availability still remain a topic of further research.

Summing up what has been seen in the last chapters, a lot of research and development has been done on honeypots in wired networks, and security research in WSN is ongoing as researchers see further need in developing practical and usable intrusion detection for WSN.

This chapter continued the basics review of literature and similar works. This division in theory in the previous chapter and practice in this chapter appeared and still appears as sensible, so that the amount of information could be presented adequately. The following sections were intended to collate the known and partly standardised security functions in wireless sensor networks, the reliability and the defensive possibilities, in order to assess the still missing components. Furthermore, an outlook on the topic of honeypots was presented with a section on maintenance. A first discussion on the honeypot subject has taken place, with the maintenance measures and the elements of the necessary resources of the defence strategies presented, and general insights were given. The subject of legal regulations rounded off the chapter and finally a summary was presented.

A first look into the research method is needed here: this it is a research method that was postulated by Schneider (2011) in his paper called “Blueprint for a Science of Cybersecurity”. As an expert in this field of research he recommends using legitimate principles to improve the security of computer systems. He encourages system develo-

pers to use these principles as tools for creating systems. The approach is that security can be compared to this types: multiple assets, multiple attacks, and multiple identities and countermeasures, so that it is possible to create data sets in which researchers have a knowledge base on the principles listed above. This means, that he collected a set of work-steps to be performed and gave guidance on how to continue working on them, with examples for further clarification. A system developer should first be clear about what the system needs to be capable of, what it should do, and what avoid. It is necessary to be aware of what should not be done, but at the same time also how this could be determined to be an attack, so that countermeasures can be taken. The whole method is further on given details on and shown in the following chapter.

# Chapter 4

# Research and Methodology

As a summary of what is presented in the previous chapter so far, a lot of research and development has already been done for honeypots in wired networks, also security research in WSN is going on. Other researchers found that more has to be done in practical and usable intrusion detection for WSN. The further reading is exactly about that, the practical approaches and the tools and techniques that are further developed than theory or simulation.This chapter is dedicated especially to the topic of honeypots in wireless environments. The deployment is of interest, various details of the attacks and their analyses are explained, as well as a series of reflections on assessability presented. The chapter will then continue with the first drafts of a honeypot for wireless sensor networks, followed by a discussion of the methodology, inspired by Fred Schneider, who, with his publications, provides a basis for viewing the science of cyber security.

## 4.1 Introducing the Research

The general objective of this research project is to create an advanced monitoring architecture assisting in detection of security incidents in wireless sensor networks (WSN). This, as a first goal, is intended to be an additional protecting element that is far beyond a simple “works for me”-approach, which will improve the security of WSN users through continuous monitoring.

The second goal is that of generating undetectable or even transparent or hidden detection of attacks, which is deemed to be the appropriate way in order to prevent mali-

cious users from learning of their detection and also of their attack being monitored for further analysis.

Such a system only needs a limited amount of administrative tasks; as it is running autonomously, it only needs interaction when malicious activities occur. Operational status would indeed not need permanent administrative interaction.

This system is intended to be useful for many usage scenarios. For a start, it could be an interesting approach to have perimeter security with this system. This means, that the production systems are still in the centre of the network, but have a safeguarding monitoring system spun around them. In the example of a building, a picture can be to have the production network inside the building and the monitoring systems attached to the building. Therefore an attack has to pass those to reach the production network. The primary user of the system is not defined in total, he is seen as an example of an owner of a smart home. Another user could be the security manager of a production plant. By safeguarding the perimeter, an outside attacker is spotted in a timely manner. Also insider attacks could be localised.

Honeypot techniques have never been adopted in a WSN yet. Many details of the research depend on the findings collected during all stages of development. Yet, the main intention and scope of the hypothesis is clear. The conduction of research means to explore hitherto unconsidered and unregarded aspects, in this case of security threats against wireless sensor networks and new defence mechanisms with the use of honeypots, because no one has ever applied honeypot methods in a wireless sensor network up to now.

The challenge presented to such a novel defence lies in the detection of the attack process. Only then will it be possible to develop new countermeasures and defence mechanisms, possibly also leading to a new hardening concept of the whole WSN and its system. The detection is the job of dedicated network sensors. A honeypot is a part of the network which appears to an attacker as a valuable point of interest. Yet,

the honeypot has only one goal: to attract an attacker with the purpose of collecting information about attacks on the network and to report the attack. This also includes new attacks and their attack procedure.

A major aspect of the research is to determine the need for such systems, including the capabilities of software using wireless sensor networks as a carrier. The concluding part of the research makes use of the collected results of research and the research-findings data for the consideration of an appropriate design of such honeypot for wireless sensor networks. This shall be done not in the traditional meaning of an IDS in a LAN or a honeypot in a LAN, but rather in the context of wireless sensor networks, featuring new detection mechanisms and new countermeasures.

The use of honeypots is described in the primary literature as an attraction for the attacker (See 3.3). They are set up to be attractive to an attacker, but are also capable for the purpose to monitor and detect an log the events. Classically there is no immediate defence mechanism in a honeypot, no defensive reaction to the intruder. By the honeypot, there will be no automated response; there will be no new information generated by the honeypot; no update of the system by its own, or any hardening mechanism developed and sent to other participants in the network.

Recalling the case of Wardriving in Dartmouth from section 2.3.2, where in almost every home ZigBee-enabled devices could be found because of most electricity meters having been upgraded to smart meters with wireless option, that were reachable by default. Right now, a critical point in time has come, and a bridge should be built by comparing WSN security with a related topic: There are studies on resistance to viruses and worms as seen in 2.2.6.3, which show clearly that attacks by a worm will stop almost never.

This is also the statement from above in section 3.4.2, where it was shown that an attack can take place and continue as long as no one will notice it and consequently nothing is done about it. So this is the main point of the whole story about IT security:

the threat must be recognised and the vulnerabilities need to be analysed.

With the ensuing chain of inferences it has to be proposed that a honeypot is suitable, in fact, to improve the security of ZigBee based networks. There are host-based honeypots in classic literature. These are in a network, wait for an attacker; they pretend to be a service, which can be exploited 3.3.1.1.Such a host-based honeypot, as it can be established for the wireless sensor networks, would even so be a way to learn about attacks. However, there is a big difference in practise from the host-based honeypots for wired networks to WSN honeypots: the distance and the linking or its connection. In the former, it will not matter where in the world the attackers themselves are; via the Internet, they can access any reachable host. If it's a honeypot on a corporate Intranet, then an attacker who has gained access to this Intranet probably wants to examine all of the hosts one by one and thus also the honeypot.

In a WSN, it is assumed, that there will be just the radio link distance between attacker and honeypot. By using this wireless connection only a limited distance is possible. The physical distance is thus lower, reduced to several metres or a few kilometres. If an attacker would target now at all ZigBee devices in the city of Plymouth, would anyone notice it later? One could examine this only if there would be at least one honeypot for ZigBee in Plymouth. Similarly, one can also speak of the Dartmouth case of wardriving mentioned above.

Some researchers say, as cited in the sections 1.3 and 3.4.1 that in the context of smart homes this is absolutely serious, because the dwelling thereby becomes an additional point of attack which must be considered for its security. As already posted previously in this thesis 1.3 there are critical voices that reject the use of ZigBee and even warn about ZigBee to use as a communication medium in smart homes. In the previous literature research in section 3.4.1 it was mentioned that the use of ZigBee technology appears to many researchers as no suitable medium for the smart home. In one of the quotations in this thesis in section 3.4.1 it was already referred directly to these regarded issues in the smart home. There, a change from ZigBee is proposed.

For the smart home, one could be thinking this idea further: If the home, flat or house is exposed to a general threat, such as a burglary with an universal key or a con artist who opens a door and holds it open for others: should one not learn about these attacks as well as the technical side of countermeasures?

Since just the possibility to already control a door or window from a distance is already sufficient to gain access to a building, the consequences when a large part or even all of the devices can be remote-controlled and manipulated are obvious.

In future there is the need to detect such attacks; the attacker is having a clear advantage because the hurdle of security implementation is not hung high enough, yet. Safeguards of the apartment can outsmart an attacker mechanically; for smart homes though, there is also the possibility to technically outwit the safeguards from afar. The hypothesis now is that traditional security measures no longer stand and the question must be asked whether new approaches are needed.

The intention during this research project was to achieve a prototype through definition, then product design; and by this the integration and validation of a monitoring solution capable to safeguard wireless sensor networks. This result is achieved by a clear structure, beginning with the basics of wireless sensor networks, their purpose and use, and the possibilities that are thought to be available for permanent monitoring the security of these wireless sensor networks.

This work is done in the following five steps.

**Analysis of monitoring methods and procedures for wireless sensor networks** The first step consists of looking at the IT security basics, which lead to a discussion of the usefulness of monitoring WSNs. A deep knowledge about the necessity of thorough monitoring for a WSN lead further to a justification of improvement in monitoring methods and procedures in wireless sensor networks. Throughout this step, a look at several methods and procedures is necessary. This thesis focusses on a lot of topics, one analysis of the presented is on monitoring methods. The background research was

introduced in section 3.2.2. It was further elaborated in section 3.3.

**Determining whether these monitoring methods and procedures are suitable for use in wireless sensor networks** With the knowledge from the first step, the next one consists of an extensive research of recent monitoring attempts in literature and available products. This step is then clearly laid out, to dive deeper into the monitoring methods and procedures especially with regard of establishing them in an environment of wireless sensor networks. During this step, multiple methods and procedures are determined to be suitable or not. The ultimate goal of this step is to gather a number of research attempts that can be refined and combined to improve monitoring WSNs. Furthermore, the discussion was concluded in a recent chapter, on the topic of costs in 3.4.4 and 3.4.5 with all of its details.

**The design and evaluation of a new monitoring method and procedure for wireless sensor networks with the goal of improved permanent safeguarding** The objective of permanent safeguarding by a monitoring solution for WSNs is necessary to strengthen the security of those wireless sensor networks. With the background of previous steps that addressed multiple methods and procedures, a resulting new method and procedure is in consideration. The comparison of all the investigated attempts led to the result that a single method and procedure is seen as constructive:

> If it is possible to monitor a malicious user and the underlying networking patterns, which is also known as packet analysis, it would enable for any WSN the opportunity to secure it.

This would be starting at the first detection of malicious users based upon receiving a networking packet or broadcast message, requesting access to a network or generating and broadcasting a new network. Nonetheless, despite the fact that such technology has been proven with a good level of success on standard networks, the accuracy is unknown on wireless sensor networks. In a WSN the networking and

interacting differs as compared to wired networks; thus they are expected to have difficulties to operate at a similar safety and reliability level. As a consequence, this third step, the evaluation of a monitoring analysis technique within a wireless sensor network, is done. The term honeypot will be introduced herein as a type of monitoring. This work is presented at the end of this chapter in section 4.5 as well as section 4.6.

**The architecture design necessary to run the new monitoring method and procedure for wireless sensor networks** Here, the necessary level of meaningful honeypot analysis is presented, which would not be suited to all WSN hardware. Not all of the WSN systems have sufficient resources to handle these requirements, therefore rendering the effort infeasible for those systems. The provided insight from initial studies of honeypot capabilities and their analysis is that the performance would not have proven reliable enough to be deployed in every situation and setup. It has to be chosen wisely and this work wants to give help for the decisions. So the design of a monitoring architecture for a set of wireless sensor networks, using a larger number of monitoring techniques that draw upon the different hardware components of a WSN node, namely a honeypot for WSN, is part of this step. This step is focused on in section 5.1 as well as section 5.2.

**Implementation and evaluation of a prototype including a test in real world environments to showcase the useful application** The final step was to design and evaluate a prototype of the monitoring architecture. This is presented in section 6.1.

## 4.2 Wireless Sensor Network Honeypot

Honeypots are as shown divided into high-interaction honeypots, most often on real hardware with real services, and the ones called low-interaction honeypots running mostly virtually and with simulated services on top. This chapter first of all focusses on the task to find a location for honeypots in wireless sensor networks. As it was stated already before, a WSN is different from other networks in structure and capabilities. With a look to the common categorisation in the scheme of wirebound networks a set

of topics need to be considered. The following questions are arising.:Where will a WSN honeypot be localised in this scheme; where does a WSN honeypot belong to? Under a now initial first assumption that in such as a non firewalled or filtered network in a WSN theoretically all nodes can communicate with each other, there is no difference in a WSN honeypot as to the placement. It should be connected to the network, in WSN speak be part of the personal area network (PAN). When additionally the transmission range is considered to be important, then it should be considered to put a production network in place and a set of detecting honeypot nodes around it.

Will a honeypot for wireless sensor networks work in the scope of high-interaction honeypots, or will it be in the category of low-interaction honeypots? This is an interesting topic to find a comment to. In general, in wire-bound networks, a high-interaction honeypot is a real system, that however is specially safeguarded in the sense of the gathered data to not be deletable and that after a compromise the system cannot be used as a hop to attack others. Whereas a low-interaction honeypot means, that the functionality and the interaction with the system is not really existing as in a given example, the shell-commands issued are not executed on the system, just the responses given to make it look for the attacker that a real interaction with the systems is happening. In this situation, giving a prediction where a WSN will be found, deciding on a high- or low-interaction honeypot the answer can only be: it depends! For the high-interaction honeypot this is considered to be the right decision, because the WSN nodes do not allow a node simulation or virtualisation easily, which would be assumed in a low interaction honeypot. This however is not considered to be true for all scenarios, low-interaction honeypots are considered to be more stable in an attack scenario. A clear either-or statement thus cannot be given.

What techniques lead to the research goal now? This needs to be answered with help of the research method. It is not considered to be quantitative or a qualitative research. Where should large numbers of test users come from in a field where something new is considered to be created first. A qualitative research could possibly be doable, if the

possibility exists to find a test case scenario and a number of experts for a survey. The latter is kept as a possibility in consideration.

Will it be at the end of the creation of an entirely new and innovative own technique? Possibility exists, that at the end of the research the outcome is not the same as foreseen and expected. However, the choice of all the work-steps is considered and written down in the following pages, and will lead the route to the product to have experiments on for evaluation.

Will it be a high interaction honeypot, or a low interaction honeypot? Will it be the combination of the two techniques? Should a hybrid honeypot approach for WSN be examined, incorporating a mixture of high-interaction and low-interaction honeypot techniques? This is an interesting question, or set of questions, leading towards a dicision. As stated three paragraphs above, this is not so easy to answer, but the current assumption is that the line of separation of these two categories is not that easily applicable to wireless sensor networks. As a result, it could be, that a decision between the one and the other is not possible, because, neither of them exist in a WSN in the classic meaning and definition, but in the end the single solution will have aspects of both of the categories incorporated. This topic is also found to be written about in the paper (Markert & Massoth 2015) and to be found in die Appendix of this thesis.

A detailed discussion on the definition and wording of WSN and the use of honeypot functions was put to a conference paper (Markert & Massoth 2015).Several implementation details were already discussed there, like the cloning feature (see also section 3.3.1.3). Virtual cloning of additional devices can of course only be solved “virtually”.

ZigBee offers some security features out of the box. The level of security offered by ZigBee, along with its weaknesses, has been explored only in part to the present day. Merely a few people have been doing research in this area at all.

## 4.3 Honeypot for Security Enhancement

This thesis follows a defined structure with aims and objectives that were presented in the first chapter. Although the structure is given, it is found to be important to give an insight into the questions that are asked during development of the honeypot. From the literature review and all the projects running on wireless sensor networks security from recent time until this thesis research it was evident, that no viable honeypot for a wireless sensor network was found. There was, however, a set of attack tools found, and attack types seen defined.

Furthermore, it is of interest whether the paradigms from a honeypot as seen in 3.3 remain in a WSN honeypot and exist unchanged. If they do not, it is of interest what changes are needed to the paradigms, such as if the WSN honeypots are valid, or what changes in their definition should be.

An estimation of the results needed to be discussed in two steps beforehand. The first discussion gives insights on the possibilities regarding the attacks that can be detected with a honeypot for WSN by definition (Markert & Massoth 2014). The second discussion to be mentioned focusses on the wording and general definitions of a honeypot for a WSN (Markert & Massoth 2015).

There are a number of attack vectors on wireless sensor networks, like they were shown in section 2.2. On wireless sensor networks they have been exemplified and roughly summarised there. But why is this a problem, for whom, or in what context?

The solution to the above stated attacks is clear on one hand: listening. Let the attacker romp around and learn from it. And the second aspect: from the patterns learned, improvements must then be worked out. Honeypots are by definition a playground for attackers, their stored logs can be further on used to develop countermeasures.

With the assumption that a honeypot can be a solution, the advantages and disadvantages of this technology shall be determined.

Another example would be the following: a honeypot is a traditional environment responsible for the attraction of attackers from the environment. So a honeypot for wireless sensor networks should really be of interest to the attacker, who look for this for an intrusion. One question is then, do the assumptions hold for the attractiveness of a honeypot (wired, IP V4, etc.) even with a WSN honeypot? The first thought on the scientific nature of this PhD research project is: there is no honeypot for wireless sensor networks as yet! It has never before been shown or proved by anyone that you can build a honeypot for WSN. There has never before been written in detail how someone established one. There have been vague suggestions that you can think of such a thing existing, but never a result.

According to the research seen at 2.3.2.3 it is very important to have a productive WSN and a WSN with honeypot purpose at the same type of WSN hardware. A very skilled attacker could otherwise make use of the "'stealth mode"' attacks on WSN nodes as described in the literature. The paper cited describes the possibility to generate data packets that are read by one type of WSN transceivers, but not by other types (Jenkins, Shapiro, Bratus, Speers & Goodspeed 2014). To reach this the authors made use of changing the packets sent in the sequence they are constructed. This is done in four variations, namely Preamble change, Franconian Bridge, Franconian notch as well as Cumberland Gap. The first one is is the change of the packet preamble, which means, that the packet initialisation sent to get transmitter and receiver synced is defined by the standard, but it was shown that receivers exist, which receive packets even if the preamble was not standard-conform, containing a shortened set and not eight symbols as defined. The Franconian notch is a variation on the preamble content, there exist chip-sets, that react on packets transmitted even if the preamble is not following the notation specified in the standard. Franconian Bridge means, that between a valid preamble and the per the standard then following start of the packet content a number of randomly chosen symbols is inserted. The start of the packet is defined by the information called SFD (start of frame delimiter), a value set generally to 0xa7h. This then is a mixture of both techniques described before. Also, here the tested devices were

responding differently. The Cumberland Gap is logically connected, it is the variation of having a valid preamble, then a wrong SFD value, followed by a number of random symbols, then another preamble, a valid SFD, then the usual remaining packet structure. The devices tested did provide different responses to this test-case. As a conclusion the authors pointed out, that fingerprinting WSN devices was shown to be possible. For further work the authors pointed out to build a database on test results and refine the software for automated fingerprinting. Additionally they have further fingerprinting techniques in consideration and would like to push forward to intrusion detection evasion. The content was extended for a conference paper on the topic (Jenkins, Shapiro, Bratus, Goodspeed, Speers & Dowd 2014).

The question primarily starting all the following is: “How to track hackers in wireless sensor networks”. So, this is a primary concern, and it is asked at the beginning of the research and further questions are all stated with this in mind. First of all, on the different attack types it is important to know the capabilities of attack detection methods, which approaches exist, which can be used in a WSN? Which attempts are known and which are under research for the detection of new and for unknown attacks? For attack detection, is an IDS suitable for a WSN, and are there limitations? Is there a possible approach to transfer the concept of a honeypot to wireless sensor networks? Would it be possible to establish such a honeypot in a wireless environment of wireless sensor network nodes with typical network structure and under the restriction of limited system resources and have reasonable output for detection? Is the recent development of attacks and tools supporting the research question, is its importance and the need to develop countermeasures such as an attack detection or honeypot given by actual attack surface or vectors? Is a smart home integrity part of this research, should something be said on this topic? Is it measurably secure or safe in a wireless sensor network with a honeypot? Is a honeypot an improvement? Can the recognition behaviour of this system be specified, and how can requirements be formulated for a honeypot which describe the improvement? How can the requirements be tested to be met, can it be tested for completeness? How can consistency be checked? Are there

existing definitions of honeypots sufficient in order to protect the wireless sensor networks? Are there restrictions in the current definition of honeypots, is there something on their definition, wording, and in its way it is implemented somehow limited? Is the transfer from wire-bound honeypot to WSN honeypot possible, does this change the wording or notation?

This work refers to the issues and makes use of methods from the existing technology systems that are inherently related. It is a goal to identify ways to support the requirements for security in wireless sensor networks with the use of honeypots. Such an approach will in the future by increasing deployment of intelligent technology in homes and dwellings and through the addition of more and more interfaces become more important.

There were quite often attempts to improve WSN security in theory, so it is clear that direct comparison is not possible due to the lack of resulting solutions. Since there is currently no implementation, answers to the above questions can not be given as yet. A demonstration that a WSN honeypot supports security is in fact necessary, to validate its usefulness.

## 4.4 Honeypot Proposal

All the previously discussed points about a honeypot for WSN target the initial research question: “How to track hackers in wireless sensor networks”.

And the honeypot for WSN as a solution has been discussed and shown to be worth conducting research on. First of all, a detection of an attack by a hacker is of crucial necessity. And of course, this is clear as well, the collected information is without value if nothing is done with it. Data needs to be processed, the information has to be analysed!

More details on the design of the solution for the data processing will be shown in section 4.4.3.

### 4.4.1 Deployment of the Honeypot

For the deployment of the honeypot sensor, that needs to be laid as allurement to attract and catch hackers during an attack, several details have to be regarded. The references that were shown earlier (3.4.1, 1.1) often came from the topics of the smart home and smart metering. This is one application out of many.

This setup shall be discussed in more depth as well, because it is one of the most common used application scenarios for the use of the new wireless technology ZigBee, and because these ZigBee based network nodes can work together in wireless sensor networks.

When the smart home and as well the security of smart meters are currently under attack, or are assumed to come under attack in the future, then sensors for attack detection should be available. Also, they should be deployed in the smart home, or at least in the same region as the attacked WSN.

A possible solution for the installation of intrusion detecting sensors is to deploy them in several places in every street, to be precise into street lamps. There is a project in Hampshire, where the following picture 4.1 is taken from; they equipped their street lamps with ZigBee compatible devices.

Those are about perfect, due to the fact that they are placed homogeneously about 20 meters apart from each other, so they could additionally create a meshed network for communication. Also, they are grid powered, so the batteries of a node could be charged and they would never run out of energy.

Furthermore the lamps are high enough above ground to get a good connection to each other as well as a good connection from street level up to the attic of a normal house. Skyscrapers in larger cities would of course need additional installation of repeaters. Good line of sight is needed due to the high frequency used in radio communication as seen in 2.1.7.

*Figure 4.1:* The street lamp Mayflower installation

### 4.4.2 Attacks via Internet

A short digression into so-called search-engine driven attacks, sometimes also called google-hacking: it is a known and common method to search with a web search engine for patterns, which are used by for instance web-cams or network printers. Recent development has shown, that this is also the fact for some implementations of smart home gateways. They are equipped with standard user-name and password combinations and are reachable via the Internet. So there is definitely a security issue with these smart home gateway implementation as shown in 2.2.

When it is supposed that it is possible to detect an attack, the obvious next question which is immediately following: what will result due to this detection? In some ways, the attack must ultimately be signalled, and the resulting findings are processed. Several options can be discussed then, which are then divided into categories.

Signalling: For a detected attack, a visual or audible alarm is triggered, though this notification only works in close proximity to the sensor that detected the attack. Hence it is also evident that the attacker can also become aware that the attack was detected. This does not yet explain how a possible benefit may be drawn from knowledge of the

attack. In order not to signal locally, the information about the detected attack has to be transferred over a medium to a remote location. Another possibility here is a wired connection for audio or visual signals or a bus system which transmits the signalling. The signalling could furthermore, take place on a device that the user uses daily. It is conceivable, for example, to signal in the categories via Bluetooth, Wi-Fi or ZigBee from the sensor to the user's smartphone or tablet. Short range communication such as NFC is not seen as useful, due to the limited range of a few centimetres. More on the signalling will be described in a later section called "Signalling and Managing" (see section 5.2.1.4).

All of the above-mentioned circumstances, of course, require that a notification can be displayed on the users device, so an application has to be created for this, which can display the alert in an signal format that is appropriate. From there, one could think of a packet distribution scheme by the smartphone to a worldwide honeynet that conducts research on the attacks on wireless sensor networks.

A key point of further work will therefore be to answer questions arising from the above-mentioned considerations. How urgent is automation for the signalling of an event? Is a flashing light considered sufficient enough or does it have to be a text message to the user's mobile device? Other possibilities like an email or a phone call could be considered, too.

### 4.4.3 Metric to Measure the Results Discussion

The last thing that has to be considered for the research question is how to measure the work done. A metric is needed to show results. What can this metric be, then?

With regard to this metric, the following statement can be given. When an improvement is the targeted goal, then it is necessary to find out the status quo and also what would be an improvement. In this case the following assumption can be done: when an attack happens to a WSN, then it is either not recognised at all because it is an attack type that has no outage as a result, or possibly the WSN is affected in some case, that can

be recognised by an operator. Still, the operator does usualy not know details, whether it is simply interference with other networks. So the capability to detect an intrusion is a must, and when these can be counted, this results in the first possible metric, the number of detected intrusions.

If for test case reasons the attack on the WSN and the honeypot is done in a lab test-case, then the occurrence of the attack is known for sure. If in this case the attack is not detected and reported accordingly, then it is obvious, that it was overlooked. So if the total number of attacks in the lab is known and the number of attacks found is present as well, then a comparison between these two is possible. This means, that a ratio can be determined.

The same tactic can be used for another nice measurement. If the time an attacker is present at the lab is known, then also the time of the measurement of the honeypot can be compared to that. As long as there is interaction with the honeypot, some information should be present, and this is true for the time an attacker is working with a honeypot. This time can therefore be possibly known. And it is also a good way of measuring the distraction of the attacker from other attack goals by the honeypot.

If there is some way of finding out something on new attack types, then this would be a great success. It depends on the honeypot at the end, whether it is capable of storing enough information about a new attack type or not. As the outcome of the research is unknown beforehand, it is assumed to be possible, but unsure that new attack types will be definitely found. These were the possibilities found:

1. Total number of detected intrusions.
2. Ratio of detected intrusions to not detected intrusions.
3. Time the attacker was working on the honeypot nodes.
4. Number of new attack types recorded.

Summarising the above lines, it is stated that the number of detections as well as the

ratio of detections and non detections could be taken as a measure, also the time-frame in which an attack is occurring, until it is reported. This was discussed, as well as, necessarily, the number of unforeseen attack types detected. The metrics were written down as a possible tool to determine accuracy. When looking at the last one then it becomes clear, that this cannot be taken as a metric, because the total number of different attack scenarios is essentially unknown. And if the total number is unknown, the number of new detected attack patterns could be seen as possibly growing to infinity.

### 4.4.4 Risk Analysis of the Data an Attacker Might Gather

According to 2.2, the authors differentiate the data that has to be secured. On a low-level view of the attack, this information could for example be a kind of interpreted view of the data, for instance a number. On a mid-level view, it could be for instance financial data, and seen from a high-level abstraction, it could be a process of snooping.

## 4.5 Honeypot Sketch

The presented literature research emphasises the development of a wireless sensor network honeypot. The following project sketch is basically a preliminary outline. The description and requirements of the project are provided with goals and ideas. First rough use cases are created and described features can be planned. All this is then put together in a detailed specification.

Whenever a honeypot is attacked it records the attack and causes the attacker to assume that he is actually attacking a worthy network, the targeted node seems to be part of the wireless sensor network. This concept requires the detection of malicious parties as well as malicious network traffic, likewise it is described in the work by (Prathapani et al. 2009) before in 3.3.

The work presented by Mostarda & Navarra (2008) in 3.3.1.2 and (Das 2009) 3.3.1.2 proposes changing roles in the network. This is seen as an supplemental possibility

to enhance a honeypot for WSN in the future, but for the initial work a fixed set of productive nodes and a fixed set of honeypot nodes seems to be more sufficient.

This will help to keep results separated from productive network traffic so the the conclusion will then be that the detection of new attacks and their methods is possible with honeypots. Once attackers engage the honeypots, the functional part of the network will be left with additional time to perform normally while preparing countermeasures. This should increase the reliability and availability, fulfilling the main purpose of honeypots described as seen in section 3.3 by (Gupta et al. 2012).

Unknown attacks can also be found and screened for reporting. The development of new countermeasures will only be possible with the detection of unprecedented attacks. The honeypot concept for wireless sensor networks is designed to feature new detection mechanisms and new countermeasures compared to traditional approaches.

The implementation of a honeypot for wireless sensor networks is a new way to deal with new attack scenarios in the field of WSN communications. It deals with the expected upcoming contributions in the field in the area of WSN regarding security flaws and could also lead to be a part of a new kind of alternative approach to improve security within WSNs.

## 4.6 Honeypot Concept

The proposed honeypot concept is presented in this section, along with a description of its features and design. It has been shown in the previous section that a honeypot could indeed help enhance the overall security in WSNs. Several ideas were presented, and cited. This section will combine these ideas and put them together to a concept for the honeypot in WSNs. The ideas were described for nearly all of the WSN layers up to the ZigBee layers. Some were regarding the hardware, some the basic transmission, others were regarding the improvement of packets and their reliability and some of them were regarding the upper software layer for the users. So the honeypot has to be described on several layers and the proposed features are a set on all of the layers.

Following is a list of features that seem to be suitable for a honeypot proposal and therefore are going to be incorporated into the design of the features to be put into the prototype. Some of the ideas of the literature research are referred to as they seem to be a practical proposal also for a wireless sensor network honeypot. More on the details is then found in the later sections in chapter 5 and 6 then.

1. The first look is to the hardware. For a security related piece of network part it should be clear, that it should be a hardened system that is well sealed and tampering with the hardware should not be made easy with for instance exposing the debug-connectors externally. There exist rules and suggestions like the ones presented before from the that of the BSI in section 3.3.3 can be taken into account. Also the suggestions by Singh & Verma (2011) that were presented in section 3.3.3 can be taken into account.

2. The system additionally features a feedback channel for reporting attacks. This can be probably done via a covert channel or other medium via a gateway. This might be done on other frequencies or channels like for instance GSM, 5GHz Wi-Fi, 868 MHz ZigBee, LAN and others. For a first implementation these hidden feedback possibilities do not need to be implemented. This is considered as not being necessary as a matter of research, as it is a trivial task which only requires to have a feedback channel attached. As an example such features exist as modules to be connected via a serial connection and in an example of GSM be battery powered, thus the data transfer is not a scientific task any more. Modules for reference are commonly available in stock, which have a small form factor, and just need power and control via serial interface to set up a connection with a SIM card inserted to the board.[1]

3. A honeypot must not be recognisable as such to an attacker. An attacked honeypot should therefore react in the same way as any other attacked part of the WSN. This leads to the assumption, that its hardware capabilities should be the

[1] example: www.mouser.co.uk/ProductDetail/DFRobot/TEL0113

same. For safety and reliability reasons, it could be equipped with a larger battery or a wired power supply. It was shown by researchers that it is possible to do a fingerprinting of devices, meaning, that if device has a different transceiver, then a different behaviour could possibly be observed. As the transceiver is most often bound to the processing capabilities, such as increased computational power or more memory, increasing these details could then result a priori to a detection as being another type of device, which an attacker might not want to interact with. (See section 2.3.2.3 and 4.3)

The features listed above are considered useful for a production network and can be seen as a minimal set of features that makes a prototype honeypot into a first field test honeypot. The difference to be shown is that during the research it is irrelevant whether the honeypot is specially shielded to be tamper-resistant or not. In the lab it is considered not useful to improve on this topic, as no real world attack scenario is considered during research. The same is true for a suggested hidden feedback channel. During the tests it is considered to be sufficient that the packet transmission to a central honeypot-component is out of band on a serial connection. Whether this is later on changed to a connection on WiFi, GSM or laser-link is irrelevant, as the data transmission stays the same by the time the connection is established and data transmission is working. The third point is important already in the lab, because the devices' limitation that were discussed to be the same as any other WSN devices have to be considered to be important during the prototype creation. Limiting resources after a product is running is not seen as a good practice.

The requirements for the wireless sensor network honeypot evolve from the above arguments, of which the first one is that the honeypot core components in production should not expose sensitive access mechanisms. This requirement means to be justified by the hardening aspect in computer science. So the question here is what this means on an abstract level and what this should mean in the implementation. For the overview it means, that the components that do the intrusion detection at the honeypot

are seen as important, thus it is necessary they keep their work up and running and they guard the data they gathered. In detail this means that access to the hardware and also to the software should be restricted. The question to address is then the actual details on how restrictive can such a core component be set up. This question will be taken up again in the next chapter where the requirements list is to be found.

The second point that was given above is the requirement that detection is meaningless if it has no effect, in the sense of that no action will ever be done. This requirement is on an abstract level; there must be the possibility to read and understand the gathered data, which for this purpose has to be sent from the node where the intrusion was detected to a system component that can store securely or alert an operator.

The third point given means, that the productive network in all of its parts has the requirement to be set up coherent with similar system components. The single requirement derived from this statement means, that the use of the same or at least comparable hardware for the transceivers is needed, as the hardware tends to react differently. More on this fact and the meaning of it for the further work with a discussion on the chips in the testbed is found in a further chapter.

As a summary, the next three points will not be regarded further, as their implementation is impracticable for an experimental prototype setup. They should therefore be taken into account for further research.

1. Like it was presented in the previous section 3.3.1.2 the authors Mostarda & Navarra (2008) suggest assigning roles in the WSN with changing roles, which seems to be a very good idea for future research, but does not necessarily has to be implemented at early stages. This feature should be part of the WSN honeypot in future development as well.

2. Sharing the opinion of the authors (Huang & Lu 2012) in section 3.3.1.3, the idea of a feature of cloning and replication is not practicable for a WSN, because nodes with identical signal strength and location could reveal that they are there

as a bait. (It seems to be better for Wi-Fi or LAN.)

3. The proposed scheme by (Qassrawi & Hongli 2010) from 3.3.1.2 is not suitable, because as said previously, scheduling and changing roles is not going to be implemented in a first version of the honeypot for WSN, but could be the part of a subsequent research work.

Moreover, the recording of unknown attacks will undoubtedly lead to the development of new countermeasures and hardening. The proposed honeypot concept for WSNs combines new detection mechanisms offering new countermeasures for the stability and reliability of the network.

As said previously in this chapter, the detection of an attack by a honeypot and the insight in the behaviour of an intruder can be used to devise appropriate defence mechanisms or to generate updates for the complete network. Considering the publications over the recent years that were discussing new attack schemes, there is an expected growing market for researchers to do research on security flaws simply due to the also growing opportunities for intruders to have a focus on the WSN 2.3.2.3. Honeypots will help in detecting and resisting these unavoidably upcoming attacks. It should be understood, though, that the security of a WSN network is not static, but rather an ongoing process to be kept up during the WSNs whole lifetime.

## 4.7 Methodology

The research methodology is a very interesting topic, especially in computer science, which is different from other sciences. Other research areas are is commonly differentiated between qualitative and quantitative research method. The first one is based on a large number of measurements. It is assumed that the more data is generated or found during the research, the more precise the result can be. This can be done especially in various types of sciences, where the result that is found in an experiment can be repeated and not random. In a set of experiments an assumption is tested to be valid; if this hypothesis is significantly shown to be correct, then new knowledge can be found.

In the area of computer science when something new is considered to be created at first, this approach is difficult. Especially for this thesis, an experimental setup where a quantitative approach can lead to a result could not be foreseen. Also the use of quantitative research needs a proof to be correct, therefore control groups are needed to see any bias. With regards to qualitative research this is a more user centric way of doing research, which often consists of to interviews and user behaviour observation, it is not aimed at finding an exact number for the research, but to find a correct algorithm for a question. It could be possible to do qualitative research with a honeypot, if there is a group of test users that run a honeypot, and also test attackers to do attacks, and then these groups fill out a survey of their user experience, which results can then be used for improvement. As written, this needs groups of people that are proven experts, which is needed for the quality in qualitative research, and a proper survey. This attempt was taken into consideration throughout the thesis but not done due to the lack of possible test case scenarios and the low number of experts available for filling out a survey. However there was a paper found during literature research on doing cyber security research by a method for cyber security that was found to be promising, and the focus was set to work with this research method.

First, the research method is evaluated here, a summary of the paper “Blueprint for a Science of Cybersecurity” by Schneider (2011) is given.

Essentially, its author is concerned with improving the security of computer systems by using generally valid principles. He wants to give system developers these principles as tools to create such systems.

His point of view is, that security can be compared as follows: there is a number of assets, a number of attacks and a number of detections and countermeasures, therefore it is possible to build a dataset, where researchers can contribute to a knowledge base regarding the principles listed above.

According to his point of view, a system developer must first of all be clear about what

the system to be developed has to do, with what it should do it, and what it should refrain from doing.

In order to ensure that the system does not do what it is not supposed to, it is necessary to be aware of what should not be done, but at the same time to be aware of how this malfunction could be achieved by an attack and to provide appropriate related countermeasures and regulations.

He postulates that these generalisations help to use defences, countermeasures and rules in the case of the always-same attacks by category, and also in combinations of these, the compilation of various measures.

The author states, that laws can be generated. These laws build the basis for the science of cybersecurity, meaning, that it is possible to understand the logical connection between attack, policy and defence in such a way that statements are valid like “defences in class D enforce policies in class P despite attacks from class A” (Schneider 2011).

In detail, the science of cybersecurity according to Schneider consists of three building blocks. First of all, the author suggests that all interfaces of a system must be known and listed. This means that the obvious interfaces have to be thought of, i.e. the ones existing by intention. Furthermore, there are interfaces that are unintentional (so to speak a side-channel) and there are still interfaces that cannot be seen as such at first glance (for instance by freezing the memory of a system to reach a cold boot attack scenario). Those who know the interfaces also understand the possibilities of attack, according to the author. The final step of this analysis is about the roles of the people involved.

In his paper he describes that after the knowledge of interfaces the classes of attacks are going to be determined. Afterwards it is necessary to know the assets to be safeguarded by the three principles Confidentiality, Integrity, Availability (CIA), or if this does not fit, probably a necessary new definition, as an example hyperproperties are

given as introduced by Clarkson & Schneider (2010). When it comes to the protection mechanisms, he structures the defence into the three categories: Isolation, Monitoring and Obfuscation.

For isolation, he does not present an own law but states that these are virtualisation and sandboxing topics.

Moreover he describes two mechanisms as examples of laws for safeguarding systems. The first one is the reference monitor example, where he describes so to say a wrapper, where each and every execution is granted only when the system will not reach an invalid status. His “inline reference monitor” solution needs one instance per executed program.

The law about “Attacks and obfuscators” means, that diversity is important to reach a resilient system setup. If systems are build equally, and programs running on them are the same, then this setup is very likely to be attacked, all in one go. A vulnerability that can be used to propagate a worm by exploiting other systems in the same setup could lead to a loss of security on all systems at once. So the obfuscator law means, that it is good to use tools and setups to guarantee that variations of programs exist to make them resilient to attacks.

According to Schneider the science of cybersecurity can lean on existing knowledge such as formal methods, byzantine fault-tolerance and experimental computer science. Formal methods are a set of mathematical proofs that are used to guarantee a program does exactly what it is supposed to do. While this method sounds perfect its use of resources and computational power increases with the size of the proof. However, it is used for routines that are not allowed to fail, for safety critical reasons as an example. The byzantine fault tolerance is a simple way of generating security by redundancy, very simply spoken. But as vulnerabilities then would also be replicated, the author suggests using the obfuscation law presented above.

Regarding the topic “experimental computer science”, he is of the opinion, that: “So

just as experimentation is useful in the natural sciences, we should expect to find experimentation an integral part of computer science." (Schneider 2011). He further concludes compared to other sciences, that "(..)experimentation for a science of cybersecurity will require test-beds where controlled experiments can be run."

To other researchers he concludes with "Since a science of cybersecurity should lead to new ideas about how to build systems and defences, the validation of those proposals could require building prototypes. This activity is not the same as engineering a secure system. Prototypes are built in support of a science of cybersecurity expressly to allow validation of assumptions and observation of emergent behaviors. So, a science of cybersecurity will involve some amount of experimental computer science as well as some amount of experimentation." (Schneider 2011).

### 4.7.1 Research Methodology by Schneider

So, as a conclusion after reading and comprehending the paper, this research work will be based on top of this science of cybersecurity blueprint written by Schneider (2011), which seems a very good summary to build upon. Thus, the research here will be using his assumptions, work steps and arguments, also to accept his recommendation to include experimentation as a reliable step in the science of cybersecurity.

This includes first to describe the interfaces that are possible. Secondly, it is necessary to understand the policies in wireless sensor networks, what can be achieved and possible ways to do it.

### 4.7.2 Following the Method of Schneider

As the title states, the research method is based on the method Schneider (2011) suggested. Therefore the goal of this section is more on writing about this research methodology, which is thus based on the work of Fred B. Schneider.

First of all the interfaces of the WSN wireless sensor networks will be described. According to Eckert (2009), the possible ways to exchange information are called channels,

and she separates legitimate and hidden channels. Such channels (or interfaces, according to the wording of Schneider) are used by systems and users for interaction with procedures, services and processes. For wireless sensor networks, an analysis is needed to understand the type of channels or interfaces that can be used, upon this the work then concludes. The analysis is done in section 4.6 as well as a paper (Markert & Massoth 2013) that focussed on interfaces at all network layers.

The procedure for gaining knowledge is seen as eventually resorting to experiments. It is a system to be discussed, designed, developed and implemented, it is a prototype to obtain a measurement result from. The research underlying this work is performed on the basis of an exemplary implementation of a method.

This chapter was dedicated to the topic of honeypots and showed immense challenges in their application in wireless networks. Guidance aids were presented for the installation in a realistic scenario. Furthermore, a number of findings on the assessability of the quality of a metric have been identified. The chapter started with a design of a honeypot for wireless sensor networks. Furthermore, the work of Professor Schneider was acknowledged, who, with his work ' blueprint for a science of cyber security ', created an inspiration for the work to be done here. His work emphasises the science through experiments, which in turn lead to a realisation, since they create universally valid knowledge.

# Chapter 5

# Proof of Concept

THE following chapter addresses the concepts of the wireless network and discusses the basic concepts of the experiment. This chapter also deals with the applied construction of the prototype. The chapter starts with a basic design, the analysis, the concept and the specification, before it continues with the planning of implementation and validation, up to a viable prototype. This is followed by a discussion on the treatment of attacks, together with an evaluation of how to deal with the attacks. Next is a series of technical details on the testbed for the prototype, consisting of various wireless sensor network components, all of which are necessary for the test setup, followed by a detailed discussion of the attacks that can be executed with the testbed. The chapter concludes with a discussion of the testbed part used for detection of attacks.

## 5.1 Overview

The research started with a survey of relevant literature to gain solid background knowledge in WSN technology. A summary of the current state of WSN security is provided in chapter 4. This reflects the initial research into the state of the art in WSNs, and currently existing security threats to them. With regard to the classical information security requirements the need in WSNs was formulated. Essential recent improvements in WSN security were composed and remaining tasks identified (see chapter 4). The results found were presented in a talk and revealed the gap in WSN security. One of the essential steps was working out the categorisation of attacks on WSNs (see

chapter 4). This was done with regard to security mechanisms present in the ZigBee PRO published in 2007. The findings were categorised and hints from other authors regarding shortcomings noted to focus on later in the research. A sketch of possible improvements was done by introducing intrusion detection systems (IDS) and the topic of honeypots. Improving security methods to detect misuse of a network can be done in two ways, using an intrusion detection system, or alternatively a honeypot. It is a fact that there are already theoretical intrusion detection mechanisms that are able to detect a distinct intrusion scheme with the use of pattern recognition in an invariable network structure. The potential for IDS with special regard to honeypot architectures was shown in an internal workshop paper at SEIN 2011, titled “Attack Vectors to Wireless ZigBee Network Communications - Analysis and Countermeasures” (Markert et al. 2011), describing the results by then. This presentation of the problem with security and a possibility to improve it leads over to the next work step, the specification.

In the previous chapters emphasis is placed on the need for enhanced security in wireless sensor networks, combined with the demand for some kind of ongoing monitoring, looking for unusual events in the traffic seen. The determined urgency of the approach using honeypots for WSN security is suggested.

The requirement for the network nodes in an IDS approach is high on resources in memory, computational power and energy consumption, this is estimated as counterproductive as other mechanisms during an attack that were described in 3.4.1 and as a result would be lowering the overall performance of the WSN. With this in mind, it is important in the design of a security enhancement to regard the performance and resource needs. The detection of misuse is possible solely with ongoing monitoring of the network. To reach this goal, the system therefore has to make available a detection mechanism, combined with methods for reporting and notification.

For the full function honeypot a complex implementation scenario can be foreseen, with a widely distributed honeynet, collecting information from lots of member honeypots. An early prototype for this honeynet will be rather reduced in the number of available

interacting networking nodes in hardware, and tested in a laboratory environment first.

To be more specific, if misuse of a network is detected by a honeypot node, then an alarm event will be triggered and the current notification routines are activated, may they be local, or signalling to a remote user, as well as forwarded to a central alert system. Indeed, showing an alert is one of the aspects, which will be the focus in this research. In the end, the efficacy of this honeynet will be quantified in testing and evaluating the prototype implementation.

In the course of these studies and research throughout the thesis, the goal has always been to implement and test the honeypot principle in a WSN with a reporting system.

### 5.1.1 Wireless Honeypot Model

Due to the research goal of a process of attack detection, a step by step approach seems feasible to generate an intuitive user experience. The process from the attack happening until the user gets notified is presented in a sequence of steps that is also shown in the next figure. It contains four parts that were determined in the analysis. This model is refined as it is the basis for further development.

For the design of a honeypot for wireless sensor networks the components need to be determined. In previous pages an outlook to some of the details was already given. In section 4.4 the need to have a detecting component was initially described. Then the data of the detection needs to be collected, put together, and be combined by the honeypot. It is assumed, that there could be more than just one detecting component. The third component is to process the data and report the output to a central component that then in a fourth step will signal the alert to the operator. This is a diagram and list of requirements for honeypot that was generated in the previous pages, see 5.1:

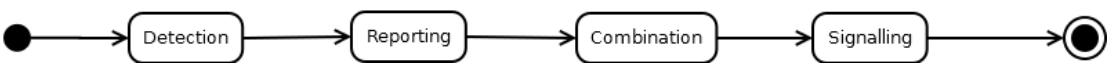



*Figure 5.1:* UML diagram

So as shown above, these are the steps in the process of realising the prototype:

- Detection of the attack and initiating the collection of relevant message content
- Combining relevant information to generate a report to be sent
- Reporting the event to a system that has a user interface.
- Signalling the event reception to the user of the honeypot

The previously noted points are fulfilling the requirements that were presented above. The first requirement definition is beginning in the section 4.3 where it is stated, that it has to be regarded that the recognition is possible, with the understanding of the restrictions, what the flexibility of the prototype is, how to test for completeness and what the need of the prototype is with regard to resource usage.

Furthermore, in a later section 4.6 of chapter 4, there are 6 points listed, that appeared in the discussion as the requirements of a honeypot prototype. They are precise and very short, helping to determine, whether the prototype of the honeypot fulfils this measures.

1. hardened and sealed system
2. additional feedback channel for reporting
3. must not be detectable
4. assigning roles in the honeynet
5. cloning and replication
6. scheduling scheme.

Likewise it was already written in that section about these six specification criteria, the first three are indeed attainable and could be regarded as an ambitious goal.

So, the question is what there is in the pertaining literature so far.

### 5.1.2 Wireless Honeypot Discussion

The Honeynet Project released a set of requirements at their website as a result of their internal discussions on the content and the standards of a honeynet collaboration (Honeynet Project 2004).

As these requirements and others stated before (see section 3.3) are obviously made for a honeypot in a wire-bound environment, they need some slight interpretation to make them fit to wireless sensor networks, which has to be done with the goal not to change the intention of the requirement! This will be followed by a necessary first discussion.

Thus, an interpreted and modified requirements list based on the work of Grimes (2005) in section 3.3 for wireless sensor networks honeypot would be:

1. Store the capture away from the sensor nodes. You want to keep the data even if the node gets damaged or stolen.
2. Don't let the hackers discover your monitoring devices. If your monitoring tools are discovered, the hackers could disable them, delete the collected data, or just avoid the honeypot. (unchanged)
3. A honeypot should strive to look like a production asset. (unchanged)
4. A honeypot system should be designed to prevent a compromise to other networks. This means, that by hacking the honeypot the hacker can not have access to production data, production systems, or production accounts.

The requirements as presented above need a discussion, whether or not these can be fulfilled. This is done in the following lines, and afterwards the requirements summarised for a related applicability.

The first item means, that the honeypot needs to be designed to forward the data of the attack. For this to be in a manner not detectable by the attacker, this obviously needs

to be at least on another frequency or channel, even better done on another medium entirely.

This conclusion of the first item was put into design of a hidden transport channel. If other channels or medium is needed to be used an evaluation of these medium is needed. The remainder of this section will be a discussion about an overview of the possibilities.

Costs occur if the techniques used need special hardware; commonly used hardware is for instance Ethernet, more costly are wireless chip-sets (Wi-Fi, Bluetooth, Zigbee, GSM), most expensive are fibre channels. For the discussion of the throughput, which depends on the technique used, and also on the noise ratio of the medium a dedicated feedback channel has the highest reliability. With a view on the hidden factor, which is higher if a service is attacked, when communication on this channel is made impossible. So an attacker cannot recognise the communication spontaneously. If a transport is used on the same channel or frequency, it is very likely to be detected by an attacker and it has another problem, too: if the attacker performs a denial of service attack, then communication on this channel is not possible any more. So it is clear, that the overall usefulness depends on the hidden factor, also it depends on the cost of the solution.

The result of this discussion is, that the attack detecting component should ideally be wire-bound to the analysis module to deliver best results.

The second item from the list above on non detectability is important and does not interfere with the requirements so far, as it was also shown in section 4.6 as item number three and also found to be important. Also the third item in the list above has the same context as well and fits to the same item number three in section 4.6.

The fourth item in the list above is connected to the first item of the list in section 4.6 and contains the system configuration. However, it gets more specific here to not allow a re-use of the honeypot nodes as a jump host to conduct further attacks on other networks.

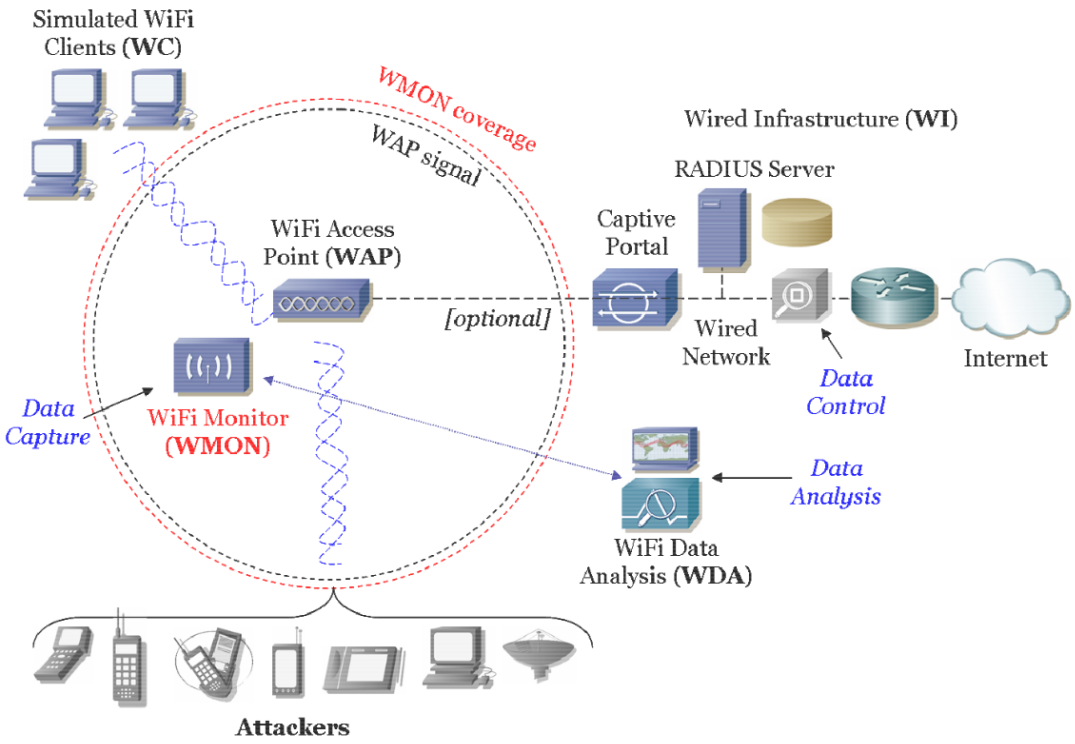



*Figure 5.2:* HoneySpot design by the Honeynet Project

### 5.1.3 The Wireless Honeynet Definition

Now a discussion on wireless sensor networks is necessary. Here, the separation of networks is not possible. One cannot possibly put firewalls on wireless sensor network nodes due to resource and other restrictions. Well, but what if it were made possible, would it help after a compromise? For this, one has to look at the design of a wireless sensor network node, as compared to the picture shown by Siles (2007), see figure 5.2.

The picture makes it clear like it was shown in the design phase, for a wireless honeypot the important aspect is to have a standard out of the box productive network, that, if it is compromised holds the first assumptions of honeypots:

```
Data control
```

So, the productive wireless sensor network is destined to get attacked, but when at-

tacked, this should not lead to the possibility that other parts of the network or even other wireless sensor networks can get attacked. In theory, this could be accomplished, although it seems to be a very hard task. In wireless networks each participant in communication can move and change the network structure when reachability to one or other network nodes improve or deteriorate. It has been shown that there wireless sensor network worms are possible (Gu & Noorani 2008), and one of the latest results was an Internet of Things worm draft for the Philips SmartLink light bulbs and other similarly controlled devices (Ronen et al. 2018).

The image above 5.2 shows the components WAP, WC, WMON, WDA and WI.

- WAP (Wireless Access Point) module: the wireless network infrastructure for clients, decoy for attackers to connect to as the target.
- WC (Wireless Client) module: these are legitimate end user devices that connect to the WAP. They are a requirement to generate network load and make it look just like a normal network to an attacker.
- WMON (Wireless Monitor) module: this device collects all the wireless traffic for analysis.
- WDA (Wireless Data Analysis) module: analyses network traffic from the WMON, necessary to determine the attacks and make sure the details are captured.
- WI (Wired Infrastructure) module: this is the “wire-bound” part of the production network, it is an optional element containing services a network commonly has for authentication and routing to other networks.

The WAP in a wireless sensor network honeypot can be seen as the coordinator of the network (probably routers, too, which will be found out). The WC in a wireless sensor network can be seen as the end-devices. The WMON in a wireless sensor network can be seen as the central honeypot component, reading all the traffic from the wireless

medium and forwarding it for analysis. The WDA in a wireless sensor network is the analysis part that is controlling data storage and triggers an alerting component for the security administrator. The WI in a wireless sensor network is uncommon, as in lots of wireless sensor network installations there are no Internet services used within a wireless sensor network.

With these components of the wireless honeypot Siles (2007) proposes to do data capture, control and analysis.

The Data Capture element from the requirements from Honeynet Project (2004) is done within the WMON, whose challenges are:

1. to be set up similarly to the WAP, meaning that the technical parameters are similar.
2. to forward the data capture as fast as possible due to the limited local storage capabilities.
3. to be on the same channel as the WAP.
4. to cover with the WMON receiver at least the WAP area; the author suggests to deal with the receiver sensitivity and the antenna for the WMON.
5. to reliably discern legitimate traffic of the end devices and exclude these from analysis consideration.

The Data Analysis element from the requirements of Honeynet Project (2004) is implemented within the WDA. The WDA challenges are:

1. to be set up to have a direct communication channel to the WMON.
2. to receive all data in a technical standardised PCAP format.
3. to give an overview of the network status

4. to have storage capability at least as much as the WMON storage.

5. to have storage capability for a suggested period of several weeks.

The Data Control element from the requirements by Honeynet Project (2004) is implemented in the WI module of the wireless honeypot; its challenges are:

1. if designed to be a public wireless network, to be set up to have a channel to the Internet to make it look like a real wireless network or an Internet emulation.

2. if public, to make sure that Data Control holds, it is more important than Data Capture.

3. to provide additional services in an Intranet.

As stated above, the WAP, WC, WMON and WDA components can be added to a wireless sensor network honeypot and there is limited modification needed to the requirements. However, the challenges remain, that now there is a completely other medium, and solutions cannot simply be adopted from 802.11 to 802.15.4 networks. Moreover, the "Data Control" element must be discussed. The WI as described by Siles (2007) is not mandatory, and also, the Internet uplink is not mandatory, therefore the Data Control attribute of the definition of Honeynet Project (2004) is not always present in a wireless network as designed for HoneySpot, and it is possible to argue, that the Data Control element is therefore also not to be required in the wireless sensor network honeypot if there is no Internet uplink.

### 5.1.4 Problem Definition and System Specification

First of all, the problem is defined, that is intended to be solved. This will be presented in a simple, natural language and is intended to be completely independent of the computer science world.

The goal accordingly is: "Therefore a system is going to be developed, that has the

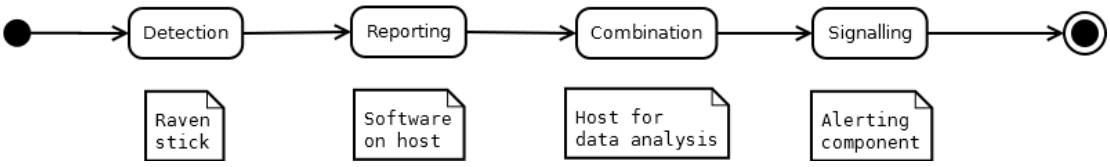



*Figure 5.3:* Honeypot sketch

capabilities to act as a decoy. It is defined as a honeypot[1] for wireless sensor networks. More components are defined up to the signalling of the attack."

A sketch of a honeypot is presented in picture 3.2. This is a very abstract sketch and does not involve a view to the different hardware components, it is more a logic combination of abstract components that form together a honeypot. The system components that were introduced until this point are now put together. Also the components are enhanced with a specification of the hardware modules that were considered, so that there is a chain of system components working together. Further work was done to get a more precise image of how a honeypot for ZigBee should look like, see figure 5.3.

The capture and detection of wireless transmission of an attack is intended to happen with the use of the AVR Raven component as shown on the diagram. The further step consists of a refinement; the system broken into parts.

### 5.1.5 Analysis of the Demands on the System

The system needs to have the following parts. Like it was presented in the system specification, the parts of the honeypot are listed and shown here in detail. The proposed honeypot is a set of components that have to work together in order to achieve a goal of incident detection and handling.

The Honeypot Sensor is the first component that has to be designed, consisting of an analysis of suitable micro-controllers, firmware, radio unit and power supply by battery.

[1] see: 3.3

The requirements analysis is the first approach to precise the system. It has to stay there for a long time, at least it should run one year on battery. It has to be a good reception area for an attack, so it is going to be equipped with an aerial for good reception. The notice that an attack is happening has to be transported to a destination where it can be shown. This has to be in time and reliable, realtime-reporting it not seen as necessary. It might sound strange, but an alarm within less than a minute can in a discussion be seen sufficient.

The better and more detailed the requirements are listed, the less functionality must be changed during programming, which would be time consuming and error prone.

The following is a set of requirements that are hard facts to be implemented.

REQ 1 - the detection has to happen on a node-class device.
REQ 2 - the detection and the production net have to be of the same type of devices.
REQ 3 - the detection is not allowed to be found out by the attacker.

The requirements above exist due to a set of logical conclusions. The first requirement needs explanation: REQ 1 exists because of the possibility of attackers to directly physical interact with devices. In a production network of node-class devices the components are commonly set up in a way that is different from a test setup such as open access to the hardware, debugging functionality enabled and connection to their ports enabled. A production network has to be secured, and the detection components have to be secured as well. The node class device also means, that it has the same capabilities as a regular node with regards to computational power or system memory. This concludes with the second requirement which focusses on the networking. As it was shown by researchers, wireless transmission can be crafted in a way that it is only read by a type of transceivers chipsets, but not by others. Therefore, the detection components have to have the same type of chipset as the production network. Different chipsets would make no sense, as attacks could happen without a detection. The

third requirement might seem obvious, as an attacker may have different reactions to finding out being recognised. It depends, whether the recognition happens before, during or after the an attacker had interaction with the honeypot. If before, the attacker might possibly refrain from attacking. As an alternative possibility, the attacker might also choose to do something that was not planned, such as something that gives nonsense information to the honeypot. However, there might also be the possibility, that nodes get compromised or stolen, if this scenario is possible. This is a situation that should be avoided.

Following is a list of components that are involved in the honeypot in total.
COM 1 - decoy devices of honeypot
COM 2 - detecting component that forwards data for analysis
COM 3 - analysing component
COM 4 - alerting component
The component 1 has the purpose to attract an attacker. Be it a real productive system or just a WSN without a real use, it is important that the decoy is present, that network traffic is existing as a wireless transmission. An attack should be detected by a honeypot node, a so called sensor. This is the component 2. The attack is then preprocessed and of course forwarded to a node where it is going to be post-processed. The data is transferred from the wireless sensor network away to another platform. At this component 3, the analysis takes place. Then it is sent to its destination, that destination can be enriched with a visual interface, a buzzer or light-signal, it can also use the message and forward it. There are enough message gateways possible that can be thought of, messenger, email- or SMS-gateway or what ever the medium for the forwarded message will be. In the event of an attack this has to be told to the operator of the honeypot. Whereas the last step at component 4 is seen as purely optional for the detection and analysis, it is important for a honeypot operator. This last step of signalling is important, but to be defined later, because it is the last and assumed to be the most trivial task.

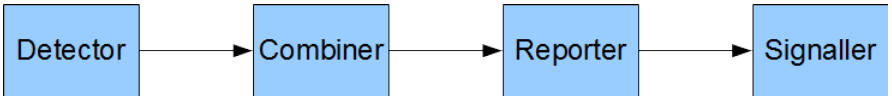



*Figure 5.4:* Component view for illustration

For illustration purposes the components are drawn here. This also includes a description of the components, see figure 5.4. More on the components is found at a later section in this chapter.

The first component, the decoy wireless sensor network has to run on standard WSN components. Also the second component, the sensor, has to run on standard hardware by Atmel, used here on AVR Raven boards and the Atmel ZigBit modules available. The second component is preferably the ZigBit USB stick, because it will be running on a gateway system, which is represented by similar hardware with serial interfaces. The third component is a processing component, this will run on a host computer which will then forward the data to a PC. This is the fourth component.

## 5.2 From Concept to Design Solution

In the section of the problem definition and resulting demands the need of a honeypot for wireless sensor networks was made clear. It is described in the requirement analysis that the detection as well as a graphical user interface is desired. The system could as an extension be designed to have an API interface component to work with other honeypots with the intent to allow communication between multiple honeynets.

The specification of this distributed honeynet architecture was discussed in section 4.4 of this report and will rely on the results named there. The goal is to specify complete details for implementing the system in all of its parts, in the network as well as in the users' node. Section 4.6 focused on concepts and section 4.4.2 looked at the details of the technical prerequisites; the full functional specification will give an explicit overview of the system design as a whole.

To provide details for the next step, drafts of the graphical user interface and other

connections will be determined and drawn (see section 5.6.5). This is going to be done for all of the central elements of the system: the detector, the reporter, the combiner and the signaller. It is expected, that formal development of these central elements will depend (or rely) on these details.

As the previous section described the layers of components to be worked at, this part describes the honeypot. There has to be a destination for the alerts a honeypot generates, and these alerts should also be reacted on and used to develop appropriate countermeasures. Therefore there will be a host based client component with a graphical user interface to display those alerts and reports. This client's hardware will be a standard laptop, connected to the honeypots via a gateway. Alternatively a mobile device could be thought of.

The development of a honeypot prototype will start with implementation of the mechanisms to the design of the proposed product. As development platform the Atmel AVR (and former Meshnetics) products called Raven (see Figure 5.5) and ZigBit (see Figure 5.6) seemed to be equipped with state of the art chipsets and a well established developer community, also regarding ZigBee related topics.

In total a set of five MeshBean2 boards from Meshnetics as well as four AVR Raven USB sticks and eight AVR Raven mobile nodes will be used for this task. The development boards contain the same chipsets and are already code compatible by the microcontroller used: an Atmel ATmega 1281 and Atmel ATmega 1284p, as well as the by chipset used for wireless communication, an Atmel AT86RF230. The Atmel AT90USB1287 features additional on-chip USB capable communication for the Raven USB stick.

As such, it is a classic situation from transmitter to receiver, in which the sink is always the coordinator, the gateway to the host. The question which arises at this point is the amount of detail necessary. The participating modules are shown here schematically.

The Raven (see Figure 5.8 and 5.7) and Meshbean (see Figure 5.9) modules come

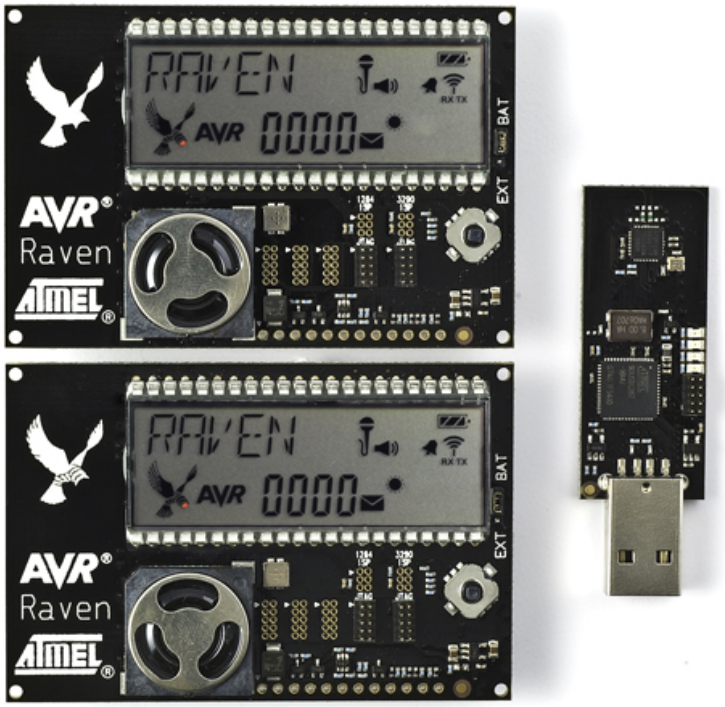



*Figure 5.5:* Atmel AVR Raven development kit

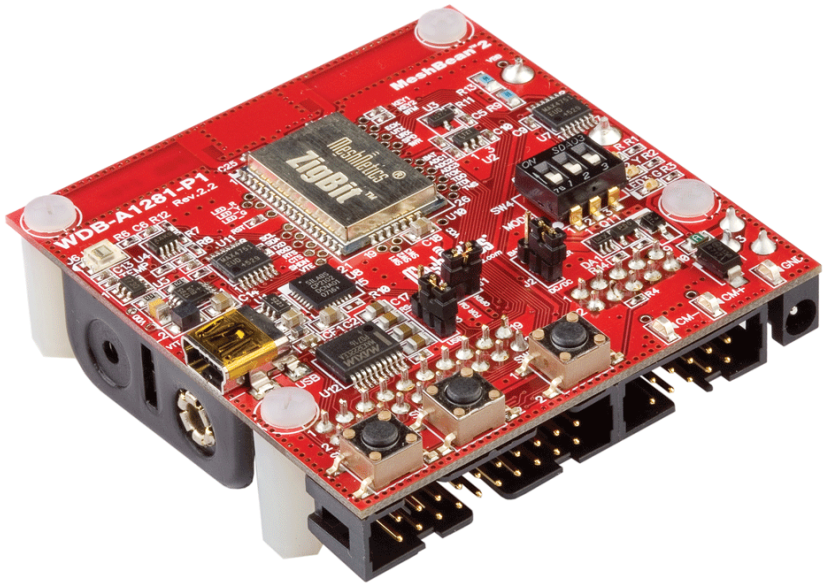



*Figure 5.6:* Meshnetics MeshBean2 board with ZigBit module

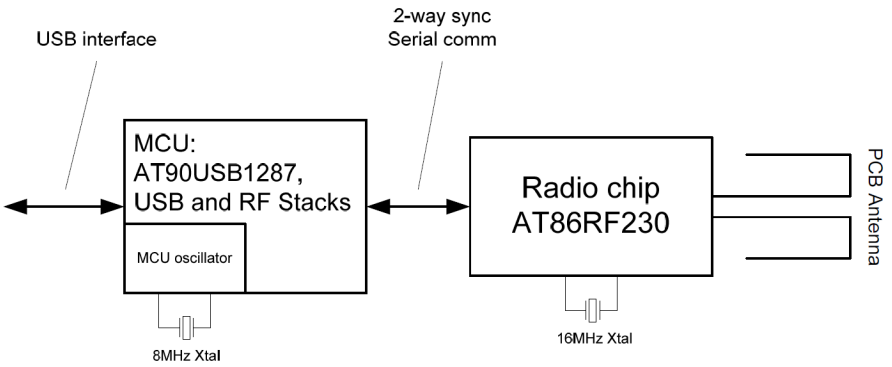



*Figure 5.7:* Diagram of AVR Raven USB stick

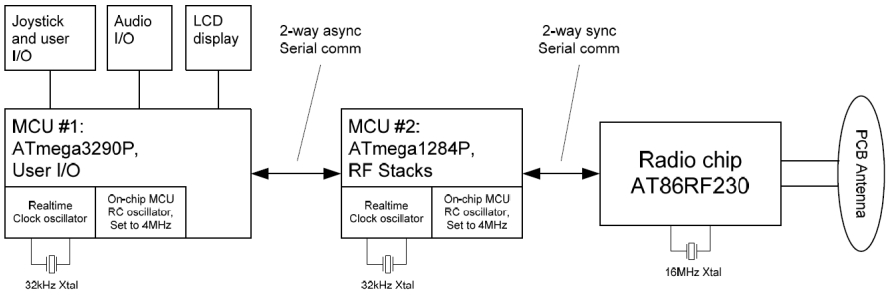



*Figure 5.8:* Diagram of AVR Raven module

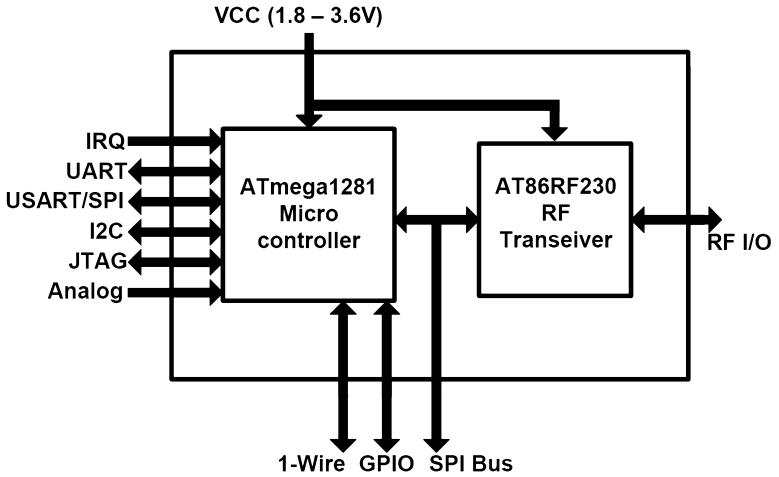



*Figure 5.9:* Diagram of ZigBit module on MeshBean2 board

*Figure 5.10:* Atmel AVR JTAGICE mkII

with details as to their design, this makes the credible statement that these are in principle based on the same hardware and software.

Also the used programs, tools and techniques should be listed at this point. For programming the Atmel AVR Studio 4 with debugging functionality will be used. Its use is free of cost and therefore ideal for development in a low cost research setup. For debugging purposes an Atmel JTAG device was bought (see Figure 5.10).

The AVR JTAGICE mkII is manufactured by Atmel. It is a very powerful development tool for programming and can also be used to debug the programs directly on the microcontroller via JTAG interface. It is very important to mention this, because development without a serious and proper debugging hardware is nearly impossible, using OTAP and just serial programming is really very slow.

From a basic design with the sketch of the honeypot in this section (see Fig. 5.11) where a redirection attack is shown to happen, a more elaborated design can be developed. This will be shown after the introduction of its parts.

As described previously, the nodes have to be configured in a way that an attacker can not distinguish between “normal” nodes and the ones with honeypot functionality.

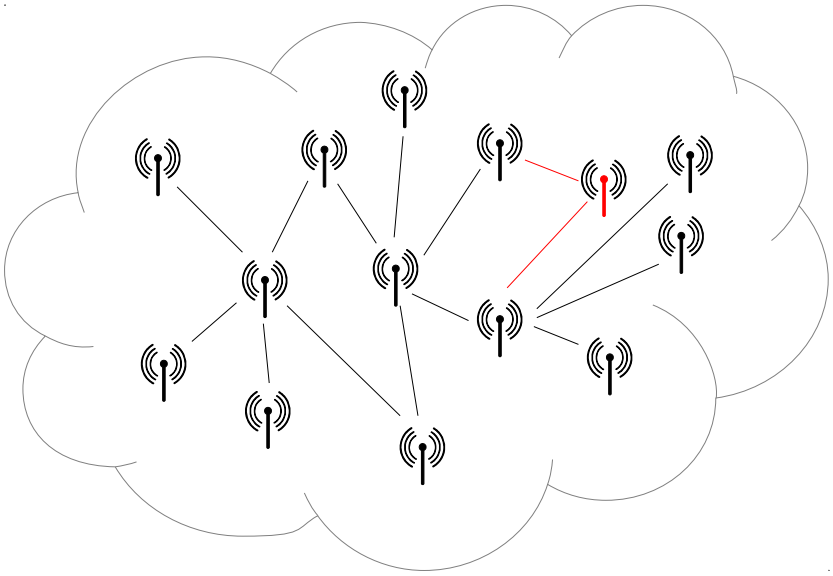


*Figure 5.11:* Basic concept of honeypot showing an attacker in a WSN

### 5.2.1 Architecture of the System Design at a Higher Level

The features on a high level of abstraction are defined in an architecture specification. These parts are an overview of the modules and their interaction. This involves important data structures for data storage. The communication must be documented in detail. For the most crucial key components the decisions are defined. For example this could be an algorithm decision or a threshold value or retry counter definition. The solution principles of the crucial key algorithms are defined.

The system consists of different components, some are hardware-centric programmed with the given C/C++ language, the signalling component is going to be implemented with an object-oriented solution in Java for Android. The host based application is going to be implemented in C++ aswell.

Therefore in the detailed design the discussion is about objects, modules, processes, network components and is much finer and more structured.

In this design the client computer has to be connected with the monitored network. In the development phase this should not be cost-intensive, so it is clear that this is

done ideally via a direct connection, for instance by serial communication. All the transferred data could of course also be provided for example via GSM network, SMS based texting or email communication via the Internet. Due to the increased costs this is not done in the research phase. The capabilities of the MeshBean2 boards make these an ideal network component for this task as well an ideal gateway due to their possible USB connection for serial communications.

#### 5.2.1.1 Detector / Sensor

The detection part is developed within an Atmel microcontroller environment. When the honeypot nodes are up and running and the individual nodes are reporting to the central component of the network, then a basic honeypot functionality is achieved.

This detection will determine any misuse of the detector as a part of the honeypot. This will be done according to different levels of interaction according to the description in section 4.4.2.

The misuse detected will be differentiated and reacted upon correspondingly. The simplest case would be for instance a probing of a honeypot device, which can also be caused by an automatic service by another device. High interaction with the honeypot device, though, would be medium on the scale. The highest level of misuse is the deploying of new firmware or other code, which behaviour should always be regarded as a direct violation of all security rules. The honeypot detector as described in the previous sections 4.6 is responsible for a permanent and ongoing security detection, more precisely a misuse detection and categorisation, so that the security can be kept on a level.

With an ongoing security enforcement, a foundation stone is laid for the benefit of WSNs. The honeypot detector has a reporting scheme that could also easily be adopted for other research groups' projects, so additions to this work are kept open and an exchange of research outcome is possible with these research groups.

#### 5.2.1.2 Reporter

The implementation of the Reporter is a set of functions to report the event that was triggered by the misuse of a honeypot system, and will be reported to the combiner to forward it for signalling. Forwarding the information shall take place in a secure manner, and end to end encryption of the report is to be considered in the prototype state of the reporter. The reporter can be part of the honeypot host, but can be also seen as separated due to the fact that the reporting network traffic could take place on another medium like GSM networking.

#### 5.2.1.3 Combiner / Collector

The Combiner or Collector is the sink of the honeypot network, as it were; the place where all the detected anomalies from the honeypot sensors are collected and will be forwarded for signalling. Being a very important part of the network, it is clear that it has to be secured in a hardened and very fail-proof manner. The Collector shall also get more than enough storage to buffer accumulated data until it can be forwarded to the signaller; when there is no direct connection currently possible, the collector's buffer for security incidents should be designed to handle the information and therefore should not run out of memory.

#### 5.2.1.4 Signalling and Managing

The implementation of the signaller is a prototype, and its purpose is to show the honeypot status to an operator. An evolution to completion can be handed over to project groups for further development. The fundamentals of the signaller are signalling alerts complete with misuse categorisation and the overview of the total honeypot network status. Later development could implement authenticated update mechanisms for the devices and further analysis options for the reported alerts.

### 5.2.2 Design: Detailed Design

In software design, a detailed plan is created, by which the program system should be implemented.

In the design phase, all final decisions are made in the ideal case, which determine the behaviour of the software in every possible operation case, the algorithms for its execution and the data structures of the underlying information.

During the design, a general model of the final honeypot is created that is independent of the programming and its language.

The model must contain all the key elements such as classes, objects, states, operations, and attributes. The model must show all the relationships between the key elements to them, which are responsible for the problem solution or program management.

It is basically divided into problem-specific conditions and needs at the application level.

At the application level, for example, objects may be needed to manage the user input and classes for the graphical, structured output.

All interaction sites and interfaces between functionally separate software modules are designed far-sighted.

With the demands on the system and a first idea of the system in the previous section, now a class design is created. The figure 5.12 shown below was created with the premise that all components can be shown, even if no program or routine exists at all. The design is in this class diagram, the figure 5.12 shows a class diagram of the parts of the honeypot system for creating the WSN honeypot.

This diagram shows on the upper left the honeypot used by an operator via a graphical user interface, where alerts are shown to the operator providing additional details. For this the honeypot with the GUI makes use of the stored attacks in a database, which

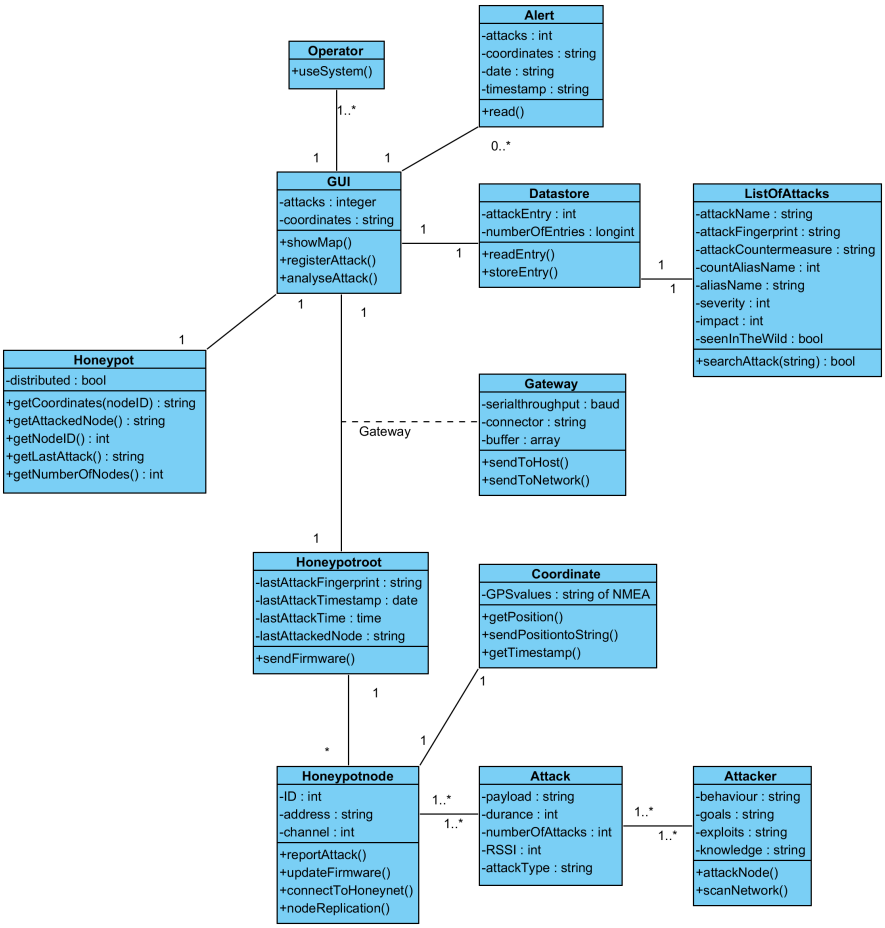



*Figure 5.12:* Honeypot class diagram

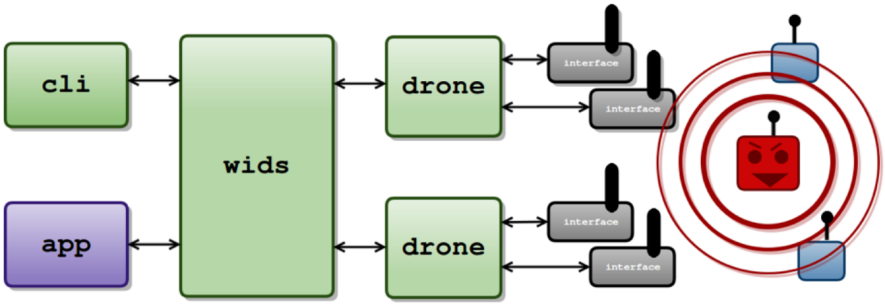



*Figure 5.13:* Beekeeper overview diagram

were stored including type of attack and additional information. Between the operator centric component and the wireless sensor network the data is exchanged over a serial connection. Such connections are often called a gateway, when data from the WSN is transported to a kind of personal computer or similar device. The core component in the diagram is shown as honeypot-root, and it receives the attack information of connected honeypot-nodes. The attacks are initiated by an attacker and result in data of the attacks being monitored and given to the node, then the root, and the application with GUI for storage to the database.

Further information on this design was made irrelevant as less than a year after providing this information a software called beekeeperwids was presented by researchers, which was taken into consideration to be a basis to build further work on. For completeness this software is shown in detail, starting from an overview down to a class diagram view, too.

The graphics of the honeypot have to be regarded as a holistic approach as said before, the following diagrams however are the result of further work, and as development went on further, the graphics had to be revised again. Following is a diagram as presented by the researchers connected to the beekeeperwids, see figure 5.13.

This is an overview, showing the capturing components, the filtering components, the storage and analytics as well as the alert presentation component. Following is an explanation with more details. The interface is a system component such as the AVR

Raven stick with sniffing capabilities. It captures the packets and forwards them to a drone component. The drone has the purpose to do a specific type of work steps according to a type of filter plugin that is loaded. In the case of attack detection there is a plugin running that does filtering of packets. The drone component then gives relevant packets to the central beekeeper daemon component, which does store to a central database. Also, at the central beekeeper component there are modules running, which can check for an attack matching.

But other plugins in the drone could also be implemented, such as automated packet replay, sending random packets and so on, but this is not the way that the drone is intended to be used. The beekeeper daemon is offering services via an API. The drone is sending packets to the destination of packets that follow the filtering rules. Four modules were provided by the authors to detect malbehaviour on the WSN:

- Bandwidth usage detection,
- Beacon request detection,
- Denial of service detection,
- Disassociation detection

An analytics module and a base module class is implemented.

The beekeeper is a tool that lays the foundation for detecting intrusions on a wireless network. This tool is open source and was released under the GPL. Following now is a set of class diagrams to explain the components that were used as a foundation and then to be extended with additional features. The diagrams can be read just as normal UML class diagrams, meaning that at the top is the name of the class, below are the attributes and on the bottom the operations, see figure 5.14.

This picture gives a complete overview on the Beekeeperwids. The overview is intended to be a basis for discussion on the components. On the left-hand side there is the

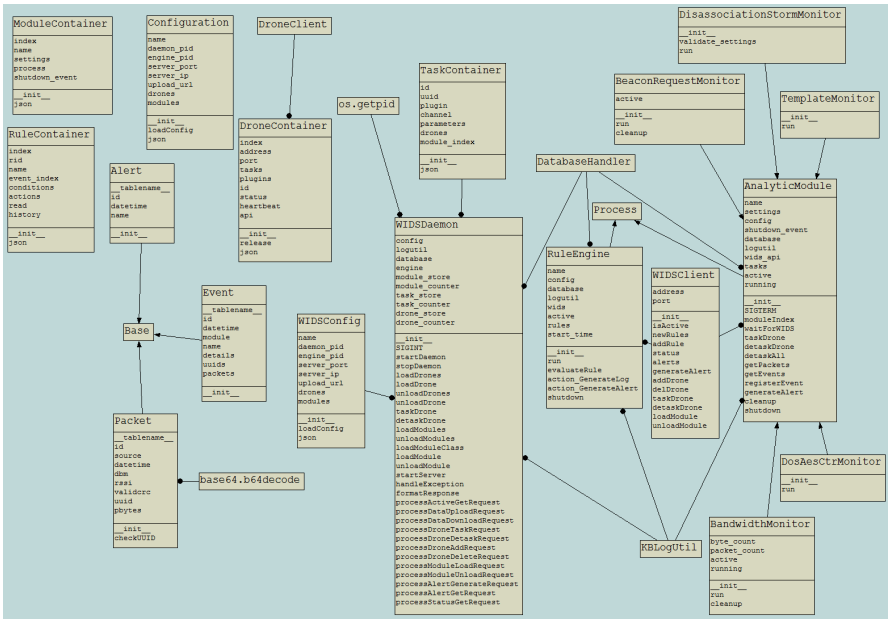



*Figure 5.14:* Beekeeper overview class diagram

description of a database. It is filled by the packets that are received from the monitoring system components, the so-called drones. After pre-processing to refrain from forwarding valid network traffic to the database the data is put into the databases and stored there. It is possible to tell drones to actively search for attacks. It also is possible to tell the Beekeeperwids to repetitively scan the database of captured packets for an attack pattern. Those are written into rules such as for instance: If there is a capturing of network-requests, then the amount of those requests shall be checked for a threshold; if this is exceeded, then an alert must be given. As shown in the diagram above, the packets in the database have all the relevant information with them, such as packet content or the signal strength.

This following diagram 5.15 explains the drone component of the honeypot. Such as it was stated above, a drone is a system component that has direct access to a sniffing component. All network traffic is received by the drone from the sniffer. It is well known what is valid network traffic in the WSN, so the focus is to store only unusual network traffic. By eliminating known traffic from storage, lots of resources are kept available from the limited resources. The class BasicPacketFilter is a component of each drone,

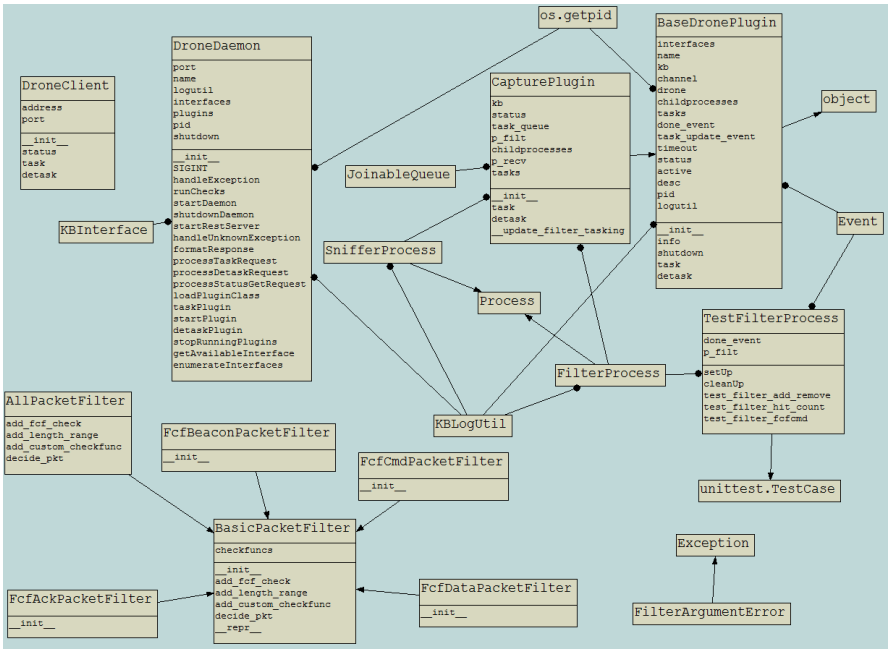



*Figure 5.15:* Drone class diagram

and each of its plugins helps to forward relevant data for analysis. With regard to communication, the drones are part of a network, because each drone sends the new relevant data that was captured to the core component. In a lab environment this is an IP based network, all components have an IP address and a port. The configuration is seen above in the class DroneClient. This setup is a possible implementation; for every possible setup another configuration is plausible. Arguing and discussing network topologies is as told not the preferred domain of this thesis. What remains open for discussion is the connection to Killerbee; the framework has far more possibilities than what is known to the public. For instance, it is possible to give each system component its location coordinates by using the global positioning system (GPS), and this data could be forwarded to the core component of the honeypot. This could be implemented in the future, as the GPS coordinates could be included in the Killerbee frameworks data, they are prepared already in stub-alike codings, such information would then just be forwarded.

This diagram 5.16 shows the function of the analytic module, whose main purpose is the configuration of the monitoring scripts that are intended to run on the data sub-

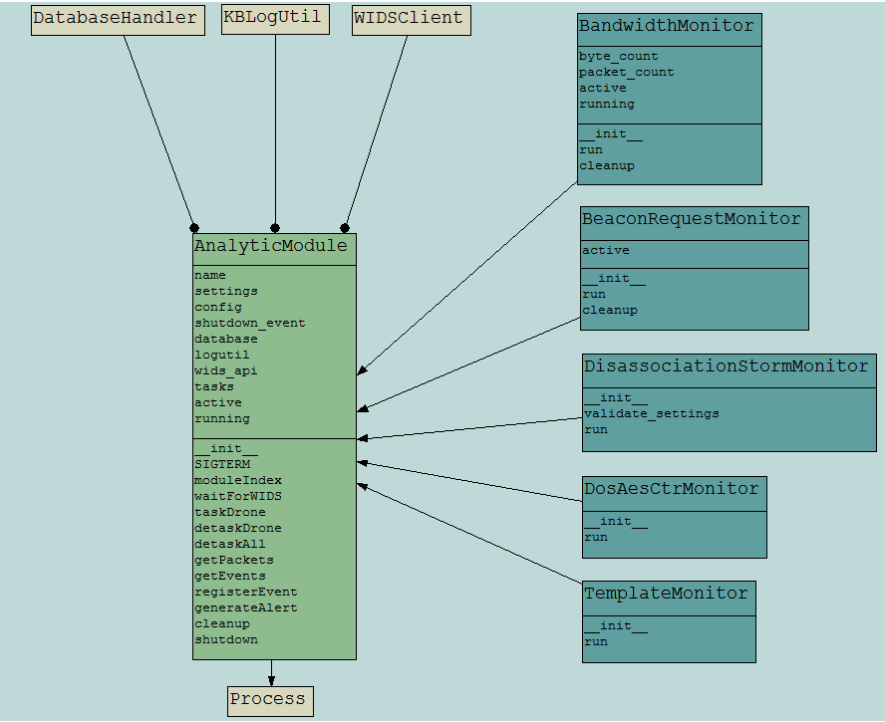



*Figure 5.16:* Analytic module class diagram

mitted from drones. This means the data that was captured by a drone needs a first decision, whether it is relevant to be further processed and if so, be forwarded for a further work step. This is done with the goal to have only relevant packets forwarded to the central component and there to be stored in the database to be the basis of an alert. For clarification, normal network behaviour, such as the reporting of environmental parameters to the WSN gateway is not intended to be seen as a possible attack. However, if the monitoring scripts are configured, they run an analytical task on the captured packets in the database. Whenever an attack pattern matches, there will the operation generateAlert be running.

It is a wish to generate knowledge using the honeypot. This is not only intended to be the knowledge that an attack has happened, Furthermore, there is the wish to learn something about new attacks. Those new attacks can be detected. However, this is resource costly. The explanation is as follow for a honeypot that is set up in a real production system. For unknown attacks a full capture must be made, meaning that all

packets received by the drone must be stored to the database. If the existence of an attack is assumed to have happened, but still is unknown because it did not generate an alert, but a strange behaviour of the wireless sensor network has been observed, then there still is the full packet capture that might give more information to the operator. This means, especially in real production system installation of a honeypot, that if unknown attacks shall be detected, then a well dimensioned data storage must exist. At non-production honeypots, also known as scientific honeypots, it is easier because the amount of data can be configured to a low number of data packets regularly sent, for instance only transmitting environmental parameters every half an hour and be silent in between. So, if the attack has now been written to the database and this attack has then been analysed by the operator, then the likely found underlying attack pattern can be understood and appropriate countermeasures can be developed. Those should be given back to the community as a monitoring module for the beekeeper. An automatic scripted solution for analysis and update can be considered for further research.

## 5.3 Realisation

In the realisation the above defined structure is described with the syntax of the programming language. As the structure from the beekeeper is already given, the task is to enhance it for the desired purposes. Likewise, it was stated before, a possible attack is seen on the ZigBee networks, and as an example for an application there is the Philips Livingcolors that uses Zigbee Light Link (ZLL). In the upcoming sections there will be shown, what can be done from an attack point of view. However, the beekeeper is by default not capable to handle ZLL. Therefore, here an addition to the beekeeper is needed to be implemented to detect attacks to the ZLL. By gathering information around the attacks it was seen that some ZLL transmission routines are implemented for secbee and z3sec. The attacks on ZLL executed later in this chapter are for a simple example the ZLL identify, where a ZLL device starts blinking for a timespan between 1 and 65534 seconds. During this timespan, the device can be controlled remotely by a user, but then continues the identify action by blinking. Only a hard reset of the device

renders it operational again. The now following implementation shows how the attack detection is done for this example.

### 5.3.1 Implementation: Software Engineering, Programming

To implement the predefined software concepts, the modules and program routines are programmed individually into a sequence of commands according to specification.

The following is part of the implementation and is shown on the following pages as sketch (see figure 5.3) as well as pictures taken of the testbed setup. This is later on followed by screenshots of attack and detection performed.

The beekeeper cannot by default handle ZLL for detection. It is very simply stacked. When the Identify packet is going to be generated, it follows the way through all the stack levels: The ZLLIdentifyRequest is bound to the ZigbeeZLLCommissioningCluster. The ZigbeeZLLCommissioningCluster is bound to ZigbeeAppDataPayloadStub. The ZigbeeAppDataPayloadStub is part of the ZigbeeNWKStub. The ZigbeeNWKStub is part of the Dot15d4Data. The Dot15d4Data is bound to Dot15d4FCS and Dot15d. A Dot15d4FCS packet is data, that is already validated to be actual network traffic, no random signals are expected here. Dot15d4 is the same without check. Thus, an attack pattern is generated by the following code (excerpt from touchlink-lib of z3sec):

```
1     def create_identify_request(self, dest_addr, duration=0xffff
      ):
2         """
3         Create a touchlink identify command frame.
4         Parameters
5         ----------
6         dest_addr : int
7             The IEEE address of the device to which the command
      shall be sent.
8             This address can be obtained form a scan response
      frame.
```

```
9           duration : int, optional
10              The duration of the identify operation. A value of 0
    aborts a
11              previously initiated identify operation, 0xffff lets
    the target
12              identify for a default time. All other values in the
    range of 0 to
13              0xfffe state a identify duration in seconds.
14          Returns
15          -------
16          scapy_pkt
17          """
18          # create IEEE 802.15.4 layer
19          mac = Dot15d4FCS() / Dot15d4Data()
20          mac.fcf_security = 0
21          mac.fcf_ackreq = 1
22          mac.fcf_pending = 0
23          mac.fcf_panidcompress = 0
24          mac.fcf_srcaddrmode = 3  # long
25          mac.fcf_destaddrmode = 3  # long
26          mac.dest_panid = 0xffff
27          mac.dest_addr = dest_addr
28          mac.seqnum = self.seqnum
29          mac.src_panid = self.src_pan_id
30          mac.src_addr = self.src_addr_ext
31
32          # create NWK layer
33          nwk = ZigbeeNWKStub()
34
35          # create AppDataPayload layer
36          app = ZigbeeAppDataPayloadStub()
37          app.delivery_mode = 0  # unicast
38
39          # create Light Link Commissioning Cluster frame
40          com = ZigbeeZLLCommissioningCluster()
41          com.transaction_sequence = self.transaction_sequence
```

```
42
43          # create Identify Request command frame
44          cmd = ZLLIdentifyRequest()
45          cmd.inter_pan_transaction_id = self.
    inter_pan_transaction_id
46          cmd.identify_duration = duration
47
48          self.update_sqn()
49
50          return mac / nwk / app / com / cmd
```

It makes use of the definition written for scapy to generate the following code:

```
1  ## This file is for use with Scapy
2  ## See http://www.secdev.org/projects/scapy for more information
3  ## Copyright (C) Ryan Speers <ryan@rmspeers.com> 2011-2012
4  ## 2012-03-10 Roger Meyer <roger.meyer@csus.edu>: Added frames
5  ## This program is published under a GPLv2 license
6
7  """
8  Wireless MAC according to IEEE 802.15.4 / Zigbee / ZigBee Light
       Link
9  """
10
11 import re, struct
12
13 from scapy.packet import *
14 from scapy.fields import *
15
16 ...snip
17 bind_layers( Dot15d4, Dot15d4Data, fcf_frametype=1)
18 ...snip
19 bind_layers( Dot15d4FCS, Dot15d4Data, fcf_frametype=1)
20 ...snip
21 class Dot15d4Data(Packet):
22     name = "802.15.4 Data"
```

```
23      fields_desc = [
24                       XLEShortField("dest_panid", 0xFFFF),
25                       dot15d4AddressField("dest_addr", 0xFFFF,
    length_of="fcf_destaddrmode"),
26                       ConditionalField(XLEShortField("src_panid",
    0x0), \
27                                                lambda pkt:
    util_srcpanid_present(pkt)),
28                       ConditionalField(dot15d4AddressField("
    src_addr", None, length_of="fcf_srcaddrmode"), \
29                                                lambda pkt:pkt.
    underlayer.getfieldval("fcf_srcaddrmode") != 0),
30                       # Security field present if fcf_security ==
    True
31                       ConditionalField(PacketField("aux_sec_header
    ", Dot15d4AuxSecurityHeader(), Dot15d4AuxSecurityHeader),
32                                                lambda pkt:pkt.
    underlayer.getfieldval("fcf_security") == True),
33                       ]
34      def mysummary(self):
35          return self.sprintf("802.15.4 Data ( %Dot15d4Data.
    src_panid%:%Dot15d4Data.src_addr% -> %Dot15d4Data.dest_panid
    %:%Dot15d4Data.dest_addr% )")
36
37      def guess_payload_class(self, payload):
38          if ord(payload[0]) & 0x01 and ord(payload[0]) & 0x02:  #
     Inter-PAN Frametype
39              return ZigbeeNWKStub
40  ...snip
41  class ZigbeeNWKStub(Packet):
42      name = "Zigbee Network Layer for Inter-PAN Transmission"
43      fields_desc = [
44          # NWK frame control
45          BitField("reserved", 0, 2), # remaining subfields shall
    have a value of 0
46          BitField("proto_version", 2, 4),
```

```
47          BitField("frametype", 0b11, 2), # 0b11 (3) is a reserved
    frame type
48          BitField("reserved", 0, 8), # remaining subfields shall
    have a value of 0
49      ]
50
51      def guess_payload_class(self, payload):
52          if self.frametype == 0b11:
53              return ZigbeeAppDataPayloadStub
54 ...snip
55 bind_layers( ZigbeeAppDataPayloadStub,
    ZigbeeZLLCommissioningCluster,
56          profile=0xc05e, cluster=0x1000)
57 ...snip
58 bind_layers( ZigbeeZLLCommissioningCluster, ZLLIdentifyRequest,
59          command_identifier=0x06, direction=0)
60 ...snip
61 class ZLLIdentifyRequest(Packet):
62      name = "ZLL: Identify Request"
63      fields_desc = [
64          # Inter-PAN transaction identifier (4 octets)
65          XLEIntField("inter_pan_transaction_id", 0x66666666),
66          # Identify duration (1 octet):
67          #   0x0000: Exit identify mode
68          #   0x0001 - 0xfffe: Number of seconds to remain in
    identify mode
69          #   0xffff: Remain in identify mode for a default time
    known by the receiver
70          XLEShortField("identify_duration", 0xffff),
71      ]
72 ...snip
```

To add the detection to the beekeeper, additions had to be made on these files: ine wids-daemon.py the module had to be entered to the module list during startup and also added to the loadModuleClass as follows:

```
...snip
from beekeeperwids.wids.modules.identify_flood import
    IdentifyMonitor
...snip
  def loadModuleClass(self, module):
    ...snip
    if module == 'IdentifyMonitor'      : return IdentifyMonitor
...snip
```

After these prerequisites, the new file for monitoring called identify-flood.py was implemented as follows:

```
import time, logging, signal
from time import sleep
from scapy.all import Dot15d4FCS, Dot15d4CmdDisassociation,
    ZigbeeNWKCommandPayload

from multiprocessing import Process
from datetime import datetime, timedelta

from beekeeperwids.wids.modules import AnalyticModule
from beekeeperwids.utils import dateToMicro

class IdentifyMonitor(AnalyticModule):
    '''
    This plugin attempts to detect the identify-attack, making
    ZLL devices blink for n seconds.
    Tools such as z3sec perform this scan.
    '''
    def __init__(self, settings, config, shutdown_event):
        AnalyticModule.__init__(self, settings, config,
    shutdown_event, "IdentifyMonitor")

    @staticmethod
    def validate_settings(settings):
        '''
```

```
22          performs validation to ensure the neccessary settings
    are present
23          '''
24          required_settings = ['channel', 'module_index']
25          for setting in required_settings:
26              if not setting in settings.keys():
27                  error = ec.ERROR_WIDS_MissingModuleSetting
28                  data = settings
29                  return (error,data)
30          return (None,None)
31
32      def run(self):
33          self.logutil.log('Starting Execution')
34          self.active = True
35          channel = self.settings.get('channel')
36          time.sleep(3)
37          self.logutil.log('Submitting Drone Task Request')
38
39          # Task drones to capture ZLL packets.
40          parameters = {'callback': self.config.upload_url,
41                        'filter'  : {
42                        'fcf': (0x0300, 0x0100),
43                        'byteoffset': (9, 0x03, 0x01)
44                      }}
45
46          uuid_zbnwk = self.taskDrone(droneIndexList=[0],
    task_plugin='CapturePlugin',
47                                      task_channel=channel,
    task_parameters=parameters, module_index=self.moduleIndex())
48          if not uuid_zbnwk == None:
49              self.logutil.log('Successfully tasked drone with
    task: {0}'.format(uuid_zbnwk))
50          else:
51              self.logutil.log('ERROR: Failed to Task Drone')
52
53          # Get packets from database and run statistics
```

```
54          while self.active:
55              # filter the packets back from the database more, by
     time, etc as needed
56              pkts = self.getPackets(uuidFilterList=[uuid_zbnwk],
    new=True)
57              self.logutil.debug("Found {0} packets since last
    check.".format(len(pkts)))
58
59              # perform the analytic
60              for pkt in pkts:
61                  self.logutil.debug("Got pkt from DB: {0}".format
    (pkt))
62                  spkt = Dot15d4FCS(pkt.pbytes)    # scapify the
    packet's bytes
63                  self.logutil.debug("Content of spkt from pkt:
    {0}".format(spkt)) #debug ausgabe
64                  #self.logutil.log("This is an example log
    message.")
65                  #TODO publish events or alerts to the database
    as needed
66      if ZLLIdentifyRequest in spkt:
67          if spkt.identify_duration > 0x000F:
68                      self.logutil.log('Generating Alert:
    IdentifyAttackDetection')
69                      self.generateAlert('
    IdentifyAttackDetection')
70                      self.registerEvent(name='
    IndentifyAttackDetection',
71                                         details={'channel':channel, '
    pkt':pkt})
72
73              # This is the (approx) interval at which the
    database is queried:
74              time.sleep(30)
75
76          self.shutdown()
```

The same sequence of work steps at the identify detection is performed at the join request. With this request, a device is asked to leave the network and join another network. After this step, it is configurable by the new network. This means, that this ZLL device will loose the pairing and can be remotely controlled by the new network. This attack follows the way through for join through these stack positions: The ZLLNetworkJoinRouterRequest is bound to the ZigbeeZLLCommissioningCluster. The ZigbeeZLLCommissioningCluster is bound to ZigbeeAppDataPayloadStub. The ZigbeeAppDataPayloadStub is part of the ZigbeeNWKStub. The ZigbeeNWKStub is part of the Dot15d4Data. The Dot15d4Data is bound to Dot15d4FCS and Dot15d. The attack pattern looks like this:

```
1     def create_join_router_request(self, dest_addr, channel,
2             encrypted_network_key, network_address):
3         """Create a touchlink join router command frame.
4
5         Request the target device to leave its current network
    and join an
6         other network.
7
8         Parameters
9         ----------
10        dest_addr : int
11            The IEEE address of the device to which the command
    shall be sent.
12            This address can be obtained form a scan response
    frame.
13        channel : int
14            The channel to which the targed device switches
    after receiving and
15            accepting this command.
16        cnrypted_network_key : byte string (16 bytes)
17            The encrypted network key of the new network. Pass
    random bytes if
18            you are not interested in controlling the target
```

```
    device afterwards,
19              but want to dissconnect the target from its current
    network.
20          network_address : int
21              The short network address the target should use
    after joining the
22              new network. However, the target might ignore this
    value and assign
23              a different address to itself.
24
25          Returns
26          -------
27          scapy_pkt
28          """
29          # create IEEE 802.15.4 layer
30          mac = Dot15d4FCS() / Dot15d4Data()
31          mac.fcf_security = 0
32          mac.fcf_ackreq = 1
33          mac.fcf_pending = 0
34          mac.fcf_panidcompress = 0
35          mac.fcf_srcaddrmode = 3  # long
36          mac.fcf_destaddrmode = 3  # long
37          mac.dest_panid = 0xffff
38          mac.dest_addr = dest_addr
39          mac.seqnum = self.seqnum
40          mac.src_panid = self.src_pan_id
41          mac.src_addr = self.src_addr_ext
42
43          # create NWK layer
44          nwk = ZigbeeNWKStub()
45
46          # create AppDataPayload layer
47          app = ZigbeeAppDataPayloadStub()
48          app.delivery_mode = 0  # unicast
49
50          # create Light Link Commissioning Cluster frame
```

```
51          com = ZigbeeZLLCommissioningCluster()
52          com.transaction_sequence = self.transaction_sequence
53
54          # create Network Join Request command frame
55          cmd = ZLLNetworkJoinRouterRequest()
56          cmd.inter_pan_transaction_id = self.
    inter_pan_transaction_id
57          cmd.pan_id_ext = self.src_pan_id_ext
58          cmd.key_index = 4
59          cmd.encrypted_network_key = encrypted_network_key
60          cmd.network_update_id = 1
61          cmd.channel = channel
62          cmd.pan_id = self.src_pan_id
63          cmd.network_address = network_address
64          cmd.group_id_begin = 0
65          cmd.group_id_end = 0
66          cmd.free_network_address_range_begin = 0
67          cmd.free_network_address_range_end = 0
68          cmd.free_group_address_range_begin = 0
69          cmd.free_group_address_range_end = 0
70
71          self.update_sqn()
72
73          return mac / nwk / app / com / cmd
```

For detection of a network join command by the attacker, the following code was written:

```
1 import time, logging, signal
2 from time import sleep
3 from scapy.all import Dot15d4FCS, Dot15d4CmdDisassociation,
    ZigbeeNWKCommandPayload
4
5 from multiprocessing import Process
6 from datetime import datetime, timedelta
7
```

```
8  from beekeeperwids.wids.modules import AnalyticModule
9  from beekeeperwids.utils import dateToMicro
10
11 class JoinAttackMonitor(AnalyticModule):
12     '''
13     This plugin attempts to detect the join-attack, making ZLL
   devices belong to a new controlling device
14     Tools such as z3sec perform this attack.
15     '''
16     def __init__(self, settings, config, shutdown_event):
17         AnalyticModule.__init__(self, settings, config,
   shutdown_event, "JoinAttackMonitor")
18
19     @staticmethod
20     def validate_settings(settings):
21         '''
22         performs validation to ensure the neccessary settings
   are present
23         '''
24         required_settings = ['channel', 'module_index']
25         for setting in required_settings:
26             if not setting in settings.keys():
27                 error = ec.ERROR_WIDS_MissingModuleSetting
28                 data = settings
29                 return (error,data)
30         return (None,None)
31
32     def run(self):
33         self.logutil.log('Starting Execution')
34         self.active = True
35         channel = self.settings.get('channel')
36         time.sleep(3)
37         self.logutil.log('Submitting Drone Task Request')
38
39         # Task drones to capture ZLL packets.
40         parameters = {'callback': self.config.upload_url,
```

```
41                      'filter'   : {
42                  }}
43          uuid_zbnwk = self.taskDrone(droneIndexList=[0],
    task_plugin='CapturePlugin',
44                                      task_channel=channel,
    task_parameters=parameters, module_index=self.moduleIndex())
45          if not uuid_zbnwk == None:
46              self.logutil.log('Successfully tasked drone with
    task: {0}'.format(uuid_zbnwk))
47          else:
48              self.logutil.log('ERROR: Failed to Task Drone')
49
50          # Get packets from database and run statistics
51          while self.active:
52              # filter the packets back from the database more, by
     time, etc as needed
53              pkts = self.getPackets(uuidFilterList=[uuid_zbnwk],
    new=True)
54              self.logutil.debug("Found {0} packets since last
    check.".format(len(pkts)))
55
56              # perform the analytic
57              for pkt in pkts:
58                  self.logutil.debug("Got pkt from DB: {0}".format
    (pkt))
59                  spkt = Dot15d4FCS(pkt.pbytes)    # scapify the
    packet's bytes
60                  self.logutil.debug("Content of spkt from pkt:
    {0}".format(spkt)) #debug message deact
61                  # publish events or alerts to the database as
    needed
62           if ZLLNetworkJoinRouterRequest in spkt:
63          if spkt.network_update_id != None:
64                      self.logutil.log('Generating Alert:
    JoinAttackDetection')
65                      self.generateAlert('JoinAttackDetection')
```

```
66                          self.registerEvent(name='
    JoinAttackDetection',
67                                      details={'channel':channel, '
    pkt':pkt})
68
69              # This is the (approx) interval at which the
    database is queried:
70              time.sleep(30)
71
72          self.shutdown()
```

A detection of network interference was developed as a module, the following lines perform the data packet comparison of the last minutes at the channel.

```
1  import time
2  import signal
3  import logging
4  from multiprocessing import Process
5  from time import sleep
6  from datetime import datetime, timedelta
7
8  from beekeeperwids.wids.modules import AnalyticModule
9  from beekeeperwids.utils import dateToMicro
10
11 class InterferenceMonitor(AnalyticModule):
12     '''
13     This plugin attempts to detect missing frames, which could
    be an indicator of jamming attack by
14     attempting to make the network not functioning any more.
15   Tools such as scapy packet generator and other transmitting
    tools perform this attack,
16   also random transmission on the channel can be the cause.
17     '''
18     def __init__(self, settings, config, shutdown_event):
19         AnalyticModule.__init__(self, settings, config,
    shutdown_event,"InterferenceMonitor")
```

```
20
21      def run(self):
22          self.logutil.log('Starting Execution')
23          self.active = True
24          channel = self.settings.get('channel')
25
26          time.sleep(4)
27          self.logutil.log('Submitting Drone Task Request')
28
29          # Task drones to capture data packets.
30          parameters = {'callback': self.config.upload_url,
31                        'filter'  : {}
32              }
33
34          uuid_task1 = self.taskDrone(droneIndexList=[0],
    task_plugin='CapturePlugin',
35                                      task_channel=channel,
    task_parameters=parameters, module_index=self.moduleIndex())
36
37          if uuid_task1 == False:
38              self.logutil.log('Failed to Task Drone')
39          else:
40              self.logutil.log('Successfully tasked Drone with
    UUID: {0}'.format(uuid_task1))
41
42          # Get packets from database and run statistics
43          while self.active:
44              datetime_now  = datetime.utcnow()
45              datetime_t60  = datetime_now - timedelta(seconds=60)
46              datetime_t120 = datetime_now - timedelta(seconds
    =120)
47
48              n60  = self.getPackets(valueFilterList=[('datetime',
    '>',dateToMicro(datetime_t60))],
49                                     uuidFilterList=[uuid_task1],
    count=True)
```

```
50            n120 = self.getPackets(valueFilterList=[('datetime',
    '<',dateToMicro(datetime_t60 )),
51                                                    ('datetime','>',
    dateToMicro(datetime_t120))],
52                                      uuidFilterList=[uuid_task1],
    count=True)
53
54            mean60 = n120/2.0 #60-120 seconds is a 60 second
    range so 2 * 60sec intervals
55            self.logutil.log("debug: Found {0} data packets in
    last 60 seconds, and {1} per 60 secs average over the prior
    60 seconds (absolute {2}).".format(n60, mean60, n120))
56            # Calculate average of number of packets typically
    seen in a given timespan,
57       # and if new number is significantly lower (than 20%
    before) give alert message.
58            if n60 < (mean60*0.1):
59                self.logutil.log(">>>>>ALERT: Noticed decreased
    data packets. (n60={0}, mean60={1})".format(n60, mean60))
60                self.registerEvent(name='InterferenceDetection',
     details={'channel':channel, 'n60':n60, 'n120':n120, 'mean60'
    :mean60})
61                self.generateAlert('InterferenceDetection')
62
63                          time.sleep(10)
64
65     def cleanup(self):
66         pass
```

## 5.4 Validation

Thorough testing is necessary on the algorithms, the functionality and the collaboration of all components at the end, too. This happens in the algorithms, the modules and the system as a whole. The setup for validation is found as a lab picture taken, see figure 5.17. For a further clarification whether an algorithm works as expected, a packet cap-

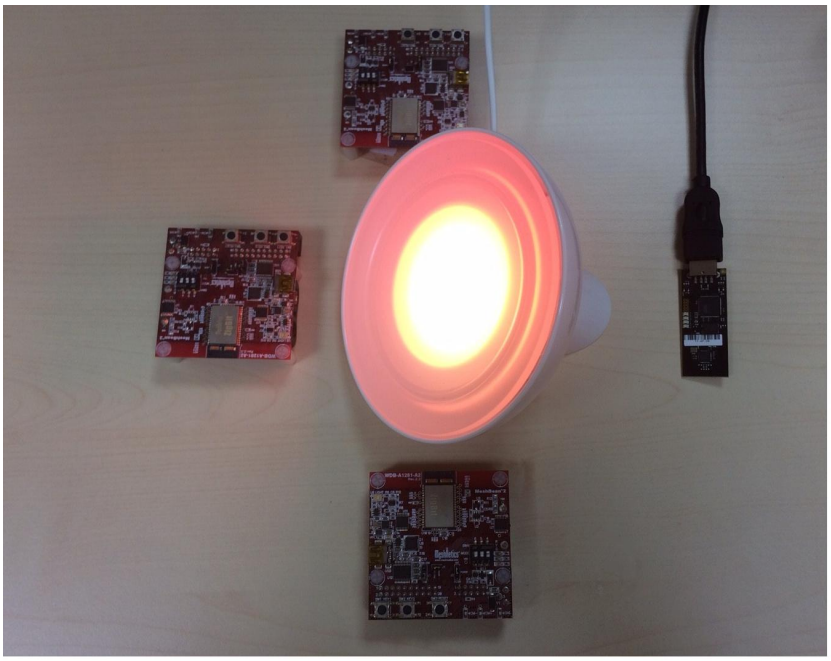


*Figure 5.17:* Productive part example for honeypot decoy

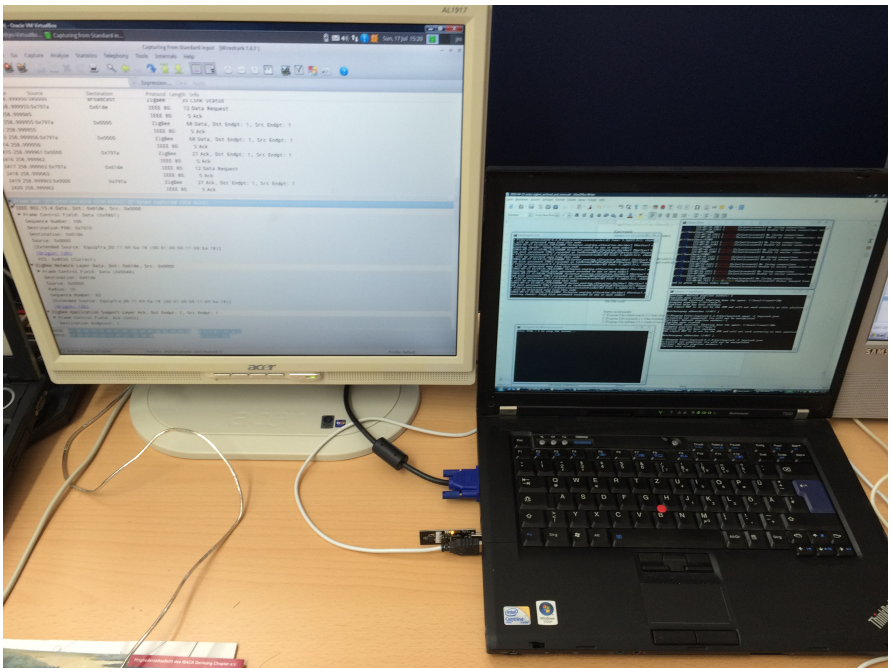


*Figure 5.18:* Setup of the detector showing packet capture with Raven stick

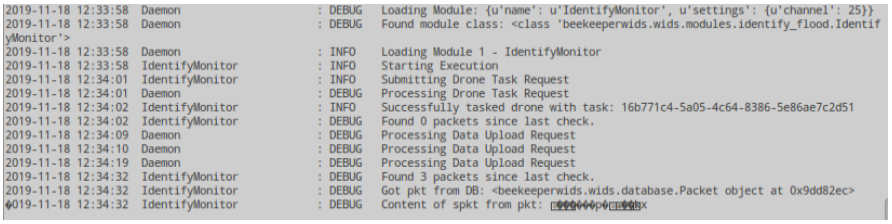



*Figure 5.19:* Identify attack detected at debug log

turing device with the same transceiver specification is suggested in parallel, such as it is shown in figure 5.18. More on validation steps and results is found at the end of this section.

### 5.4.1 Testing and Evaluation of Algorithms Used

During the verification, the algorithms used in the implementation, as well as algorithms established in the design phase must be checked to see whether they work, are correct and error-free. It should also be at least still considered here, whether other, similar algorithms can partly solve the problem better. For illustration purpose this is discussed at an example of an attack that is expected to be detected by a drone. For the algorithm testing and evaluation the debug output of the drone was considered. When the attacker in this example is causing the sending of an attack pattern for ZLL Identify, then the beekeeper is wanted to perform the capturing of the attack, too. If this algorithm of the drone is working as intended is then proven by the drone; as it captures correct, a system log is written. The prove is then, that this debug log is expected to contain the detection. To show an example, in the screenshot there is the result of the detection as an example of identify patterns sent from the attacker, see figure 5.19.

### 5.4.2 Testing of Each Module

Each module and each practical related software component still must be tested for proper functionality.

This includes checking for input and output parameters and the robustness with faulty handover.

During the testing the modules were inspected for parameter defaults.

### 5.4.3 Test of the Entire System

Then, the collaboration of the whole system is tested. The description in this section provides an outlook to a preferred testing scenario of the implementation. Running tests with attacks to the network will provide necessary results to be interpreted and used to improve and refine the prototype. One of the key aspects next to functionality is the system's effectiveness. The testing of functionality under previously considered attacks and duration is essential. When the assumption that use of the honeypot yields efficiency of detection of attacks this will be a success of the proposed solution. The result will be a well tested honeypot for wireless sensor networks prototype. A very challenging aspect is due to the fact that no one ever succeeded in presenting a working honeypot system for WSN. Papers and reports on a vague idea of this system were presented and also cited in this chapter, but the challenge here is to provide a running prototype and to present the tests and their results that were based on this new honeypot system.

### 5.4.4 Validation description

The validation description is following, describing the setup and the test that has to be done to show the function that now can be performed. First of all, for the setup:

1. there is an attacker environment set up,
2. also there is a productive wireless sensor network and
3. the honeypot detection is running.

First, the validation test routines:

1. In a first test the honeypot is loaded with the existing detection schemes.
2. Then an attack is undertaken.

3. This is validated by inspecting the attack result as it can be seen on the attacked devices.
4. The expected result at the detecting component is that the attack was not detected and
5. The devices have reacted to the network commands of the attacker.

This result should be run repeatedly several times and also with all the available detection modules, i.e. with three modules and five tests each, this would be a set of 15 test results for the attacks on the setup without the detection scheme. The results should be put into a table to document what has been done and what the result was. In a subsequent step the new attack detection scheme should be run, and the attack should be undertaken once more. The setup remains the same, but the results are expected to be different, as the attack should be detected, captured and reported accordingly.

1. In the validation test the honeypot is loaded with the newly written detection schemes.
2. Then an attack is undertaken.
3. This is validated by inspecting the attack result as it can be seen on the attacked devices.
4. The expected result that is that the attack was detected and captured and
5. the devices have reacted to the network commands of the attacker.

However, in the second time, there the alert is generated and there is a packet capture for analysis.

### 5.4.5 Validation results

This is to present the validation results. It was estimated, that 15 test results for the attacks on the setup without the detection scheme should be found in a table. The second row of test results led to a packet capture for analysis for the second row of tests, therefore, the validation is seen completely with the result as follows.

| Launched Attack | Loaded monitor | Detected |
|---|---|---|
| Identify | BeaconRequest | 1 alerts of 5 |
| Rejoin | BeaconRequest | 2 alerts of 5 |
| DosAesCtr | BeaconRequest | 0 alerts of 5 |

*Table 5.1:* Validation results with attack detection

See table 5.1 for the test results in the lab at the first validation testcases described and the next table 5.2 for the test results with another run as explained. The false positives in the first overview were produced by the complexity of the attack pattern, the rejoin and the identify do send out a beacon request during the scanning for the device by default. However, as this is a low number of scans it just sometimes triggers the threshold of the BeaconRequestMonitor. The next screenshot shows a packet capture, there it can be seen, that the identify in this case first sent three broadcast requests before an answer was sent by the ZLL device. And these broadcasts triggered the false positive reaction of the beekeeper, see figure 5.20.

| Launched Attack | Loaded monitor | Detected |
|---|---|---|
| Identify | Identify | 5 alerts of 5 |
| Rejoin | Rejoint | 5 alerts of 5 |
| DosAesCtr | DosAesCtr | 5 alerts of 5 |

*Table 5.2:* Validation results with expected attack detection

## 5.5 Operational

Also operational systems need to be maintained. The maintenance includes operational and functional controls as well as running the necessary equipment at the installed system components.

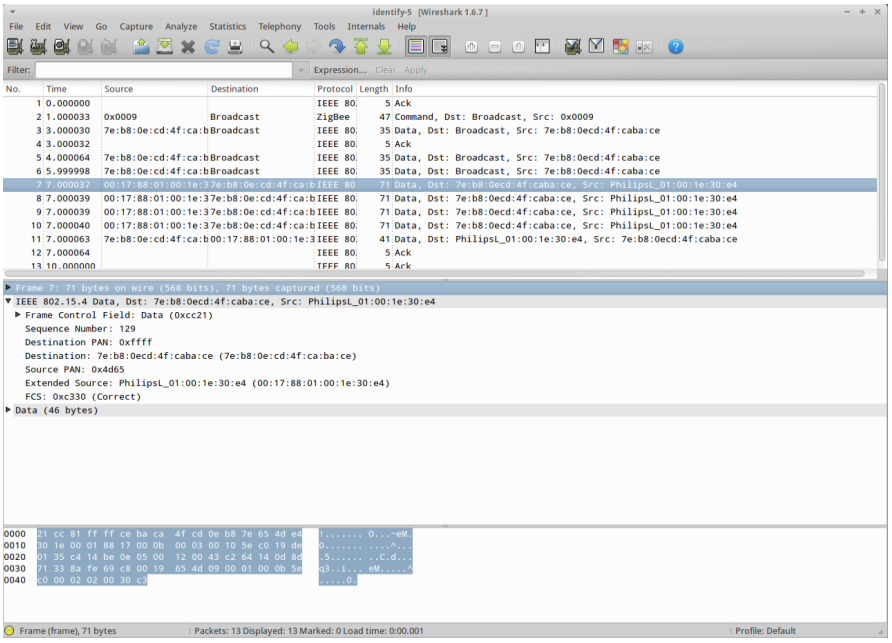



*Figure 5.20:* Identify packet capture with broadcasts

### 5.5.1 Care and Maintenance: Error Correction

Even a flawless functioning software system requires maintenance. New errors may occur after a while. It is necessary to add some more lines to the current version of the honeypot prototype, because it deviates a little from the original idea in some places.

### 5.5.2 Improvements: Revise System

There are often changes and improvements to any system, even in embedded systems.

From the latest sections, it is found that the setup needs changes.

## 5.6 Testbed Setup

The lab setup is as follows. A description is necessary for all of its parts. This section is describing the parts of the working place, it is not one working place, there are three, one for each component of the honeypot, because the different WSN components have their own development environment each. The AVR Ravens are set up with the libra-

ries for Ravens. The Meshnetics Meshbeans have their Meshbean environment. The TazTag TazPad also has its own development environment with libraries and runtimes.

The following lines provide an easy to follow setup of a testbed, with a simple structure of necessity with the following elements: A network as bait is needed. Additionally there has to be a network for detection, running on appropriate hardware. The recognition network passes on the data for evaluation. As soon as this results in a detection, this is displayed. Hardware is also required for evaluation and display. All the points listed here can be represented in a diagram with a sequence of steps to take as follows:

1. Production network
2. Attack detection hardware
3. Detection forwarding
4. Attack target
5. Detection hardware
6. Evaluation and alarming

a clear architecture and the discussion on each of the testbed components:

1. The testbed consists of a productive network to describe this. The testbed consists of a production network. This production network is made up of the Meshnetics ZigBit. This is also described, images are also shown, and the test setup becomes visible. A measurement software runs on the production network, which is responsible for capturing the environmental parameters of all subscriber nodes of the network and forwarding them to a computer. So, there is a transition from the wireless network via a gateway to a host computer. The Meshnetics ZigBit have a chipset, as described in the previous section, by means of an Atmel AT86RF230, which allows transmission on 2.4 GHz. This also corresponds to the configuration of the detection hardware. Here, too, a detection is carried out on 2.4 GHz using Raven USB stick. Exactly the same with

the attacking components these also consist of the Raven USB sticks, which send the attack packages on 2.4 GHz. So much at the beginning about the send and receive built-in chips in the testbed.

2. The testbed also consists of a network for detection. The testbed, as mentioned in this section above, consists of an area of the detection network. This detection takes place in the components of the receiving module. There, the signals are decoded, and then placed as data packets on the serial interface. These data packets are received by the USB controller on the gateway computer and fed to the detection software.

3. The testbed exists for detection with this hardware. The detection hardware is as follows. The data packets are taken at the beginning from the drone, where they originate. They come almost directly in the frequency band and channel via the receiver chip of the Raven USB stick via the serial interface to the drone. This can be completely passed through or even be filtered to achieve a reduced number of packets because only a single attack pattern is planned to be detected; for example, if a detection pattern for an overload is to be detected, the total number of data transmissions that have been received is communicated. Furthermore, if an excessive use of broadcasts, for example, this is also communicated as follows: The drone passes the detected data via a network protocol to the central component which then passes on this data. This filtering on the drone might be necessary in order to pass on a reasonable amount of data and not to feed everyday traffic to the analysis in the first place. The hardware can be an ordinary host computer but can also be a computer on a single chip. Corresponding attempts with a Raspberry Pi were performed. This is a single chip computer, or a very ordinary host PC, important is an operating system runs on it, in this case this is Debian with GNU/Linux kernel. In the tests it was possible to use a Raspberry Pi 3B.

4. The testbed also needs an attack target which can be attacked by the attacker with various tools. This attack target is present in test setup 1 as an example of the Meshnetics network. Flooding with data packets turns the individual network nodes into an unusable state. In another trial setup and test, a wirelessly controlled lamp of a smart

home system is decoupled, which was coupled with a remote control and even has protection mechanisms.

5. The testbed has a component where the detection takes place in the core component. There, all packets of the database can be examined for anomalies. These anomalies can be defined in program themselves. In this core component, multiple detection filters can run in parallel. It consists of a populated database and the analysis methods that examine this database. The hardware required for this is scalable. Importantly, a Linux operating system, as with the drone, should be used. But just that the detection via the data packets can take place is important. This is the setup of the test bed as far. As the last step, which is now following, the alarm in the test bed.

6. The testbed, where the alarm takes place: The alarm is posted on the host of the core component by issuing an alarm on the console. An example output on a mobile device was also developed, during the research the reduction of interference due to many different components at the testbed it is recommended. The console is much better suitable, also in terms of error reduction.

### 5.6.1 Lab Setup Considerations

The setup in the lab for hardware related programming is based on a Windows installation installed. It is used for programming the firmware of the hardware in accordance with manufacturer recommendations. For the development of the attack and defence part of this research a GNU Linux distribution of Ubuntu is used with XFCE in version 16.04 LTS. The z3sec attacks were working Ok in this setup, for the beekeeper a previous Xubuntu version was used.

### 5.6.2 Setup Considerations for the Hardware

The development kit is a AVR Studio 4.18 SP3 for Meshnetics Meshbean and AVR Raven. This is due to the fact, that a newer installation is dependent on a newer build scheme of the projects. The development kits used in this work, the AVR Raven

kits, were built and designed for AVR Studio 4 which runs best on a Windows XP or Windows Embedded machine. It was possible with a few workarounds to get it running on Windows 7, but this was not experienced to be a very stable setup. Windows Vista is told to be the setup for “AVR Studio 5” then, and the then called “Atmel Studio 6” is the Windows 7 IDE (integrated development environment. Alternatively, as XP has run out of mainstream support, the sibling OS Windows Embedded can be considered to be used which is still supported with updates until April 2019 if there is a concern about using unsupported OS. The AVR Studio version 4.18SP3 was chosen over other setups, even though version 4.19 is available, because all the other setups were not guaranteed to perform well. This is due to the fact, that the new version has another C compiler built in, that is not the one that is supposed to be used and might lead to trouble. In hardware development, experience in the project work has shown that it is better stay close to the configuration the manufacturers told to use. The AVR Ravens were first announced in 2007 and were available in 2008. The setup as described was state-of-the-art back then.

#### 5.6.2.1 Experience with AVR Raven Default Setup

The results proved to be interesting; the AVR Raven evaluation boards were working out of the box with the demo environment provided by Atmel (WSN Demo for Stick, Router and Coordinator for the modules.)

#### 5.6.2.2 Experience with Meshnetics Meshbean Default Setup

The combination of Windows 7 and Meshbean2 did not work well at all when coming out of the box. The AVR Studio 6 took a long time until it was running properly, but trying to port the demo applications failed. Also with AVR Studio 5 this failed. What was working in the end was installing AVR Studio 4 in Safe mode with WinAVR and then using the original eZeeNet from 2007 or ZigBeeNet from 2008.

#### 5.6.2.3 Possible Wireless Sensor Network Stack Versions

Explanation of eZeeNet, SerialNet and ZigBeeNet:

- eZeeNet features simple-to-use networking for sensing or control application, basically it is a ZigBee-based private profile and works with mesh or tree topology. It comes with an Application Programming Interface (API) and offers security features, too.
- SerialNet is an out-of-the-box solution with serial interface, no ZigBee-specific implementation or programming is necessary. It is controlled via serial commands and works with star or mesh topology, together with optional security features.
- ZigBeeNet offers the full control over the ZigBee stack. This includes no limitations but is due to the possibilities available the less trivial to work with.

Regarding the wording of ZigBee, ZigBee Pro and later ZigBee 3.0, this is simply put the wording for marketing purposes. For the development these descriptions are not commonly used. A ZigBee PRO stack is compatible to the revision of the first ZigBee versions, the improvements were called PRO. BitCloud is an implementation of such an ZigBee Pro stack and was made by Meshnetics, then bought by Atmel in February 2009 (Atmel Corporation 2009)

### 5.6.3 Meshnetics Meshbean

The step by step installation of the Meshnetics Meshbean will be here at the beginning.

#### 5.6.3.1 Meshbean Installation Procedure

The installation is meant to be a guide for verification purposes and to let other researchers understand the work steps.

1. Unpacking the Meshnetics Meshbean Kit, having a closer look at documents and the CD-ROM with additional Software.

2. The Meshnetics website is shut down already, but there is a copy of the needed files in an Internet archive[2].

3. Installing the complete development kit v2.0.0.1

4. Installation of Compiler the WinAVR from 2010.10.01

5. Installing AVR Studio 4 with update to 4.18SP3.

The IDE for Meshnetics Meshbeans is ready to use after performing these steps.

### 5.6.4 AVR Raven

The installation of the AVR Raven will be described in two parts, for network driver and serial driver.

#### 5.6.4.1 Raven Network Driver Installation Procedure

The installation is meant to be a guide for verification purposes and to let other researchers understand the work steps.

1. Unpacking the AVR RZ RAVEN Kit, studying documents and the CD-ROM with additional Software and to the RAVEN website[3].

2. Study the AVR2015 document (8120.pdf) regarding installation.

3. Unzip the AVR2017.zip. Install AVRWirelessSetup.exe from May 2008.

4. Regarding AVR2015 installation of the USB device driver.

5. This was done before establishing the Wireless Sensor Network like it was presented in the example video[4].

[2] http://www2.ee.ic.ac.uk/t.clarke/projects/Resources/ZigBit_dev_tools_2.0_complete/

[3] www.atmel.com

[4] https://www.youtube.com/watch?v=148_M6N6Opk

The AVR Wireless Service uses an architecture described in the following document: doc8120.pdf (avr2015). The driver is implemented as a network driver and opens a socket by default at port 27000 on the local host. The software Wireless Service opens a TCP/IP connection on this service for interaction. The AVR Raven modules are running software, which is also in the firmware package. As described in the documentation provided, it was verified, that by default the Raven USB Stick has the blue LED turned on.

#### 5.6.4.2 Raven Serial Driver Installation Procedure

An other setup possibility is providing access to the AVR Raven via a serial connection.

1. Unzip Bitcloud AVR RAVEN 11.0 zip file
2. Flash the containing firmware on USB Stick and Raven modules.
3. Install the WSN demo 2.2.1.62 from the 11.0 zip file, alternatively use the 2.2.1.92 from 14.0 zip file, this was a working example, too.
4. for the USB stick the 6119.inf device driver has to be installed.

This will lead to a serial connection of the USB stick, attached to e.g. COM port 5 on a Microsoft Windows system.

The AVR Raven stick has a blue, a red and a yellow LED turned on when connecting to USB and a blue and a red LED when used in the WSN demo. The WSN demo application is a graphical demo of the modules interacting with each other.

### 5.6.5 TazTag TazPad

The fact to use the TazTag TazPad arose in the course of the year 2012. In 2006 the slides of the ZigBee Alliance already spoken about a mobile phone with ZigBee. It is pioneering to be able to interact with ZigBee devices from a single everyday object. In the past, the fact that buying a mobile phone or tablet with these properties was

impossible, as a workaround there was the reason to use other techniques to get data from the ZigBee network. Now, from the year 2012 on, it has arisen that TazTag as manufacturer is offering a smartphone and a Tablet for developers that involves ZigBee. This development platform consists of a smartphone or tablet with Android operating system. Android is a relatively new OS and provides an open source platform. A Linux is the basis, all functions of the phone are established in the case of applications that run in a closed system, a so-called sandbox.

The TazTag TazPad is a Tablet PC with new features in the mobile market. It is equipped with a fingerprint reader for user authentication and is capable of using NFC, ZigBee, Bluetooth, Wi-Fi and GSM as wireless communication. It can be extended with LAN adapter for stationary use and has a HDMI video output. This feature set is a perfect combination for being the mobile graphical user interface for the WSN honeypot. The Tablet is running Android and could therefore be used to program own applications like they are wanted. The provided SDK for developers gives access to all necessary interfaces.

Some of the interfaces were provided by the Android OS, others are coming from TazTag. TazTag does not give full access to the source code, of course, it is provided as precompiled libraries.

TazTag provides access on an Atmel ATmega128RFA1 chipset[5], an architecture created especially for mobile applications, their functionality still allows for a low power consumption at the other site. The use of mobile devices has a simple reason, it is simple and very practical and convenient to use existing structures, therefore the use of a mobile phone or Tablet PC is very preferable, because it is the notification of a user about an incident which affected the security of a wireless installation. As a conclusion the user is in the current design to be the receiver of a message and the notification is sent directly to the user and not elsewhere. It is not just a flashing light, not an alarm

[5]The ATmega128RFA1 is in principle a combination of a ATmega1281 microcontroller and the AT86RF231 ZigBee chipset

tone is what the user would like to have or will install but simply a notification on their everyday device, the smartphone.

The first steps with the TazPad device were the following:

- unboxing, check
- installation of environment, work in the development environment:

  Eclipse IDE

  Android SDK 4.0.3 (API 15)

  TazTag libraries 1.0.14_RC1
- first test runs with the sample programs.

With regard to the user experience the device was tested with regard to:

- the use of the device,
- its visual properties,
- haptic features and
- possibilities using audio.

The tests were promising and showed very good handling and stability, during many month of usage no system crash occurred. In standby mode with the display turned off the tablet was running for several days. For permanent usage a power connection is recommended. The tests started from unboxing and were done up to the application usage, see figure 5.21. The user experience was found to be acceptable, as the device contains the following features: it has an android operating system and is using a touch screen functionality, its tablet form fits to everyday usage and the rechargeable battery is considered good, the haptic is excellent due to different soft click sounds up to strong vibration features of the device, also it is equipped with a loudspeaker and audio connectors, see table 5.3.


*Figure 5.21:* TazPad with accessory

| User experience item | Present | Details |
|---|---|---|
| Usage | Good readable | Tablet with touch screen |
| Visual Properties | tablet with touch screen | Android OS |
| Haptic | vibration feedback | different strengths |
| Audio Properties | Loudspeaker and connectors | HDMI and 3.5mm audio jack |

*Table 5.3:* Hardware summary of TazPad

This was confirmed also by using the device and by reading its patent and specification.

The software visuals that need to run on the TazPad were designed as follows.

The alert should give the user informations:

1. In an overview mode
2. In a detail view

The first mode is the normal view, presenting a list of events that include the information of timestamp and the message category. The categories are either information relevant for the log, or they additionally present an alert.

In the detailed view the informations on each entry need to show timestamp, origin, system component, details on attack pattern.

### 5.6.6 Design Discussion for Specification of Testbed Setup

The availability of the IDE for Ravens and the IDE for Meshbeans posed a fundamental design question. Different parts of the network are best suitable for some kind of hardware. This leads to the question of a discussion about redesign of the used setup. When the development with Meshbean and Raven leads to the valuable insight, that it is easier to develop on the Raven platform, then the Ravens should be used as the honeypot network and the Meshbean Modules should be used as the productive network that has to be protected. In this setup it is more likely to have a stable environment that can be investigated. So the new setup is to have the testbed setup like this in the lab, the following setup was used to make the captures:

- Meshbean Modules Running WSNDemo
- Windows PC with Meshbean as Coordinator and WSNDemo Java application
- Linux PC with AVR Raven USB Stick as packet sniffer
- Linux PC with AVR Raven USB Stick as attack tool running KillerBee.

The tools that are available for experimental purpose are specialised to a distinct hardware setup. For example they have restrictions to the chipset that can be used for packet sniffing or they make use of an experimental operating system. This was the reason to first have a good literature research and afterwards buying the hardware that was used throughout the research. The AVR Raven and the Meshnetics hardware as well as others are chipset compatible.

The design changing decisions were made on the basis of different requirements. Some of the requirements were quite simplistic. For example, the AVR Raven boards have as the default power supply for the button cell. Button cells are known to

have a low capacity, so some of them have been used up and a prospect of possible alternatives has been sought from resource-saving measures.

The MeshBean boards have the identical radio chipset as the AVR Raven, but come with a connector for power supply or AA batteries therefore. Therefore it was useful for efficiency reasons to raise the production network on the basis of the MeshBean boards.

A network of MeshBean board represents the productive network in this structure. There should be no connection between the production network and the detecting network node. There are also other nodes on the production network that monitor network traffic.

Mentioned at the beginning of this chapter, a single objective was pursued. The goal is to check network traffic, this can also be called monitoring. No matter how a tool is called, the functionality of monitoring is now a key criterion for a wireless honeypot.

The aforementioned nodes use monitoring to detect attacks. In this setup the data transfer from the receiver to the evaluation is done serially via USB interface. At the end of the research an outlook was started to the evaluation on a Raspberry Pi 3 with Raspian set up, the computational power appeared surprisingly well. Therefore, here is an emerging topic for replacing a honeypot node with a Raspberry Pi obviously the next conclusive step.

### 5.6.7 Honeypot Process

Process flow of honeypot functionality:

- In normal operation the honeypot regularly reports its status to a central point the so-called signaller. In the case, it is necessary to use the Coordinator of ZigBee, once holding the data, until they can be accepted by a Tablet PC network as a relay.
- An attack is recognised by a honeypot node can be started directly with the noti-

fication. The channel should be permanently blocked in this case can be decided over three ways: 1. had been waiting, 2. switch to alternate channel, what be previously defined or 3. select an alternative way of notification, which must have been defined previously. (example: Wi-Fi or GSM)

- The Coordinator of the honeypot reached the message is evaluated and made available to the user. For this are several options to consider. This can be done through ZigBee or other channels.

## 5.7 Experiment Setup Revision for Attacks

The setup in use is based on a productive network of nine ZigBit Devices with the RF230 Chipset. They build a network and its Coordinator is connected to a host application running a Software displaying the sensor readings. Another testbed is the "Philips LivingColors" hardware as stated in a previous research proposal paper (Markert & Massoth 2013).

For the attacks as stated before the AVR Raven are used. Together with the firmware of the killerbee framework they were confirmed to be an ideal attack tool-set for the research purposes. Other researchers have written examples in additions to the killerbee framework, one of its additions from the Api-Do (2.3.2.3) extension found its way to be officially included, other researchers have written additional modules to attack such as the secbee or the z3sec (Morgner et al. 2017) attack toolset.

For the above stated target setup the following attacks were conducted.

### 5.7.1 Attack 1 - Passive Eavesdropping

Wireless is a shared medium, transmissions within short distance and good signal strength always can be read if they are received undisturbed. The killerbee framework allows passive eavesdropping, simply listening to the transmitted information. The following is an example of the output with the victims network, showing the information transmitted high-lightened (see figure 5.22).

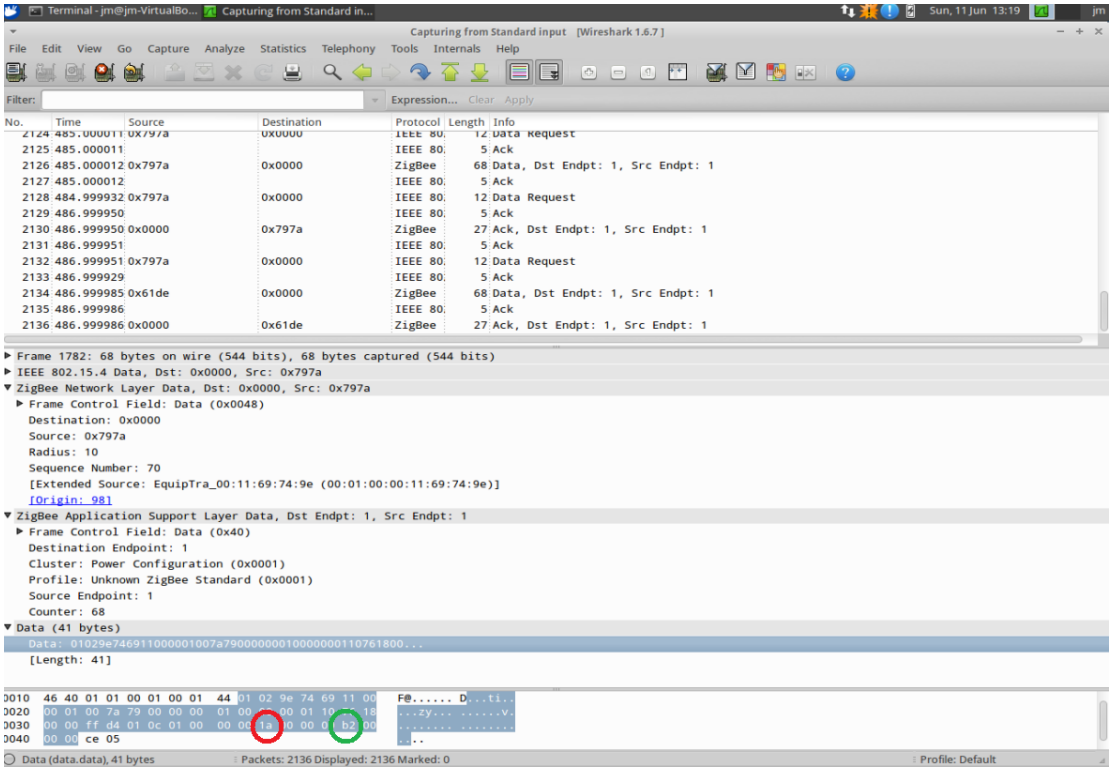



*Figure 5.22:* Temperature (red) and brightness (green) sniffed plaintext

The first value (red) the sensor node transmitted shows that 26 degree Celsius were read by the temperature sensor. The second value of the surrounding brightness in the lab as a value between 0 and 0xff (255).

### 5.7.2 Attack 2 - Sending a Crafted Packet

Right after the killerbee framework was initially released the wish to transmit informations manually and with a high degree of freedom in the packet structure led to the development of a ZigBee extension to the network tool scapy, a python driven toolset where network packets can be generated “by hand”. The following is an example code snippet publicly available that generates a Beacon message in the command line of scapy.

```
b = Dot15d4()/Dot15d4Cmd()
b.cmd_id = "BeaconReq"
```

```
juma@juma:~$ sudo zbstumbler -i 1:3 -c 25 -s 1
Warning: You are using pyUSB 1.x, support is in beta.
zbstumbler: Transmitting and receiving on interface '1:3'
New Network: PANID 0x4D65  Source 0x0002
        Ext PANID: c8:69:fe:8a:33:71:8d:0D      Stack Profile: ZigBee Enterprise
        Stack Version: ZigBee 2006/2007
        Channel: 25
```

*Figure 5.23:* Response from a ZigBee light link device

```
juma@juma:~$ sudo zbstumbler -i 1:3 -c 24 -s 1
Warning: You are using pyUSB 1.x, support is in beta.
zbstumbler: Transmitting and receiving on interface '1:3'
New Network: PANID 0x7610  Source 0x0000
        Ext PANID: aa:aa:aa:aa:aa:aa:aa:AA      Stack Profile: ZigBee Enterprise
        Stack Version: ZigBee 2006/2007
        Channel: 24
New Network: PANID 0x7610  Source 0x7532
        Ext PANID: aa:aa:aa:aa:aa:aa:aa:AA      Stack Profile: ZigBee Enterprise
        Stack Version: ZigBee 2006/2007
        Channel: 24
New Network: PANID 0x7610  Source 0x61DE
        Ext PANID: aa:aa:aa:aa:aa:aa:aa:AA      Stack Profile: ZigBee Enterprise
        Stack Version: ZigBee 2006/2007
        Channel: 24
```

*Figure 5.24:* Coordinator and routers of testbed answering the scan request

```
b.seqnum = 1
kb = KillerBee()
kb.inject(str(b))
```

### 5.7.3 Attack 3 - Scanning for Networks

Whenever ZigBee nodes receive a broadcast request, they send out the network information, meaning the ZigBee version, the Network identifier (PAN-ID), details about the structure and security aspects of the network. This feature can be used to identify networks actively, by setting the packet capturing device to a random ZigBee channel and sending a broadcast as described in the scenario two above.

The screenshots above were taken with use of the zbstumbler.

### 5.7.4 Attack 4 - Re-Transmit

If the steps one to three were successful beforehand and a working channel was determined the targets can be chosen. One of the attack patterns is re-transmitting packets

```
juma@juma:~$ sudo zbdsniff /home/juma/Z3sec/dependencies/killerbee/sample/zigbee
-network-key-ota.dcf
Processing /home/juma/Z3sec/dependencies/killerbee/sample/zigbee-network-key-ota
.dcf
NETWORK KEY FOUND: 00:02:00:01:0b:64:01:04:00:02:00:01:0b:64:01:04
      (Wireshark): 04:01:64:0b:01:00:02:00:04:01:64:0b:01:00:02:00
  Destination MAC Address: 00:d1:e4:a7:bb:f2:34:e7
  Source MAC Address:      00:9c:a9:23:5c:ef:23:b2
Processed 1 capture files.
```

*Figure 5.25:* Zbdsniff showing OTA key example

that were read and inject them again. This attack was described as replay-attack in the literature review of Wright & Cache (2015) and is done in killerbee with the following setup:

```
juma@juma:~\$ sudo zbreplay -r lightswitch-onoff.pcap -f 20 -s .1
zbreplay: retransmitting frames from 'lightswitch-onoff.pcap' on
interface '001:003' with a delay of 0.1 seconds.
4 packets transmitted
```

As seen, the packet from the file containing control sequences for home automation get re-transmitted and result in a repetition of the actions.

This attack is only working when the old ZigBee version is used, since ZigBee PRO it has a 4 byte frame counter in the auxiliary header of the network layer.

### 5.7.5 Attack 5 - Get Key

Obtaining a network key might be simple if the implementation is implemented insecure. There exist examples where it is possible to request the secret with the goal of then further attacks, such as un-encrypting traffic or interfering the session.

### 5.7.6 Attack 6 - Disturb Attack

In this attack scenario a lot of beacon requests are sent in a short period. The requests are small in frame size, the responses are extensive. This makes it possible to send a lot of small attacks, and the responses are more resource costly. This has the desig-

ned result, that the WSN nodes get far more requests than they can send responses, with the wanted outcome to crash the system by overloading its buffers with response packets that are crafted but cannot be sent fast enough.

```
zbfakebeacon -c24 -p 0x7610 -e 0x1234 -g
```

The first testing resulted in a crash of the attack tool and a crash of the sniffer that wanted to capture the attack (it was a virtual machine). The productive network crashed and the routing WSN devices needed reboot. The Enddevices did not crash. The Coordinator did not crash.

```
zbfakebeacon -c24 -p 0x7610 -e 0x1234 -g
```

The second testing testing resulted in a crash of the attack tool and a crash of the sniffer that wanted to capture the attack (it was a virtual machine). The productive network crashed and the routing WSN devices needed reboot. Some of the Enddevices were unresponsive. The Coordinator did not crash.

```
zbfakebeacon -c24 -p 0x7610 -e 0x1234 -g
```

The third testing resulted in a crash of the attack tool and a crash of the sniffer that wanted to capture the attack (it was a virtual machine). The productive network crashed and the routing WSN devices needed reboot. Some of the Enddevices did crash. The Coordinator needed reboot, too.

### 5.7.7 Attack 7 - Disconnect Attack

Using scapy with the killerbee framework attacks are possible. An example that was stated by researchers is a DoS attack by setting the CTR AES command in ZigBee. A script was released that makes use of this attack called “dos_aesctr_replay.py”, it was written by the makers of Api-do, see section 2.3.2.3. Another disconnect attack example is performed with the use of zbassocflood script.

```
jm@jm-VirtualBox:~/Downloads/killerbee-master/tools$ sudo python ./zbassocflood
-p 0x7610 -c 24
zbassocflood: Transmitting and receiving on interface '001:003'
++++....+..+.+++++...+++++++++.....................................................
...................................................................................
...................................................................................
...................................................................................
...................................................................................
```

*Figure 5.26:* Zbassocflood lab example

```
zbassocflood -p0x7610 -c 24
```

For the output see figure 5.26.

Attacks are running the best if the devices are not virtualised. The attacked network remains productive after the attack is closed, it gets back to a normal working and operational state within short time that is commonly used to build a network when powered up.

### 5.7.8 Attack 8 - Key Extraction

SecBee is an attack framework made by researcher Zillner (2015). It includes a number of attack scenarios to the network key. Its purpose it to extract the key material during the key exchange. This key can then be used to interfere with the encrypted communication.

As the tool was initially not designed to run with Killerbee devices a number of changes were needed to be done to get it working with Killerbee. Further changes then were needed to get the Killerbee framework together with the AVR Raven hardware running with the setup in the testbed.

### 5.7.9 Attack 9 - Attacks with z3sec Tool

Z3sec is an attack tool, that uses the knowledge from secbee. It is designed to run with Software Defined Radio (SDR) or the Killerbee framework with TmoteSky hardware. Numerous attacks are possible on the ZigBee Light Link specification. As the tool was initially not designed to run with AVR Raven devices, a number of changes were

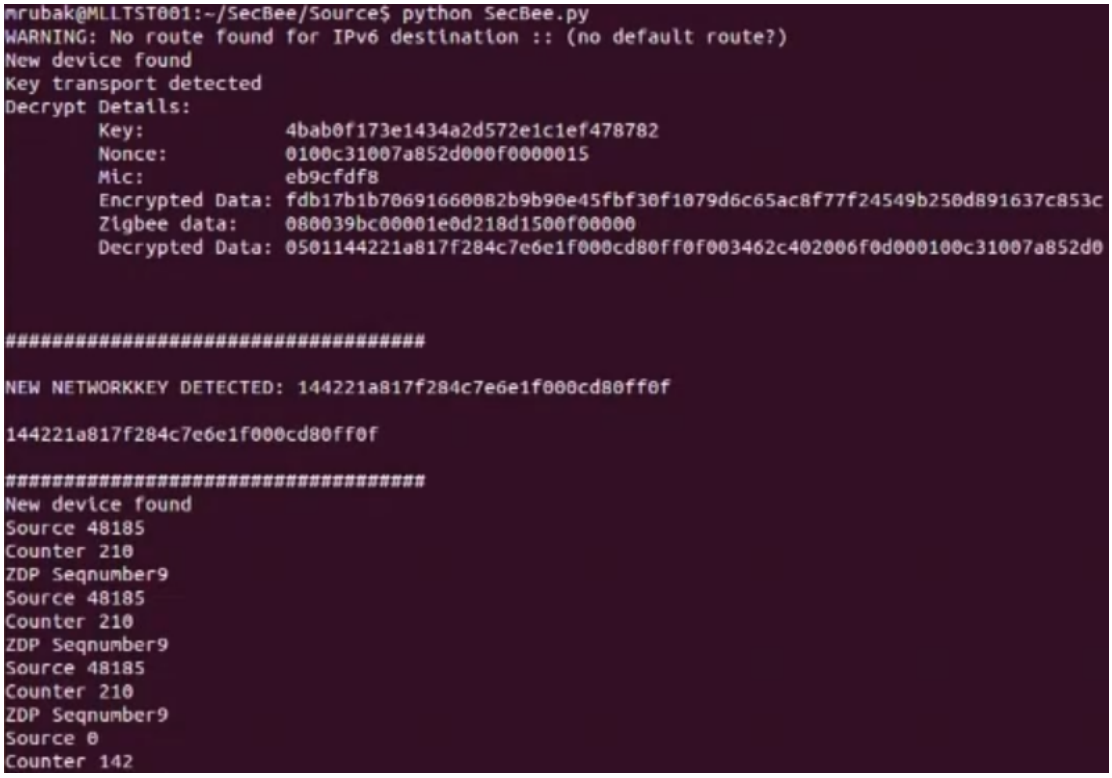



*Figure 5.27:* Secbee author example

needed to be done to get it working. Improvements on the z3sec tool introduced as part of the work here were needed to get z3sec running with the setup of the Killerbee framework together with the AVR Raven hardware.

Features are:

- Scanning for ZLL devices
- Triggering the identify action (blinking for a given length in seconds)
- Initiate a factory reset of the target
- Request a change of the channel in use.
- Unbind a target device and let it join a given network
- Create a new virtual ZLL device
- Turn light on or off

```
juma@juma:~/Z3sec$ sudo z3sec_touchlink --kb "1:4" -c "25" scan
Received 1 scan responses:

Scan responses overview:
# |RSSI |channel |src_pan_id |src_addr_ext            |
=================================================
0 |10   |25      |0x4d65     |00:17:88:01:00:1e:30:e4 |
```

*Figure 5.28:* Scan response lab example

```
juma@juma:~/Z3sec$ sudo z3sec_touchlink --kb '1:6' -c '25' scan
Setting up Radio:
KillerBee Radio (send+recv): 1:6
Warning: You are using pyUSB 1.x, support is in beta.
Scanning channel 25
Received 1 scan responses:

Scan responses overview:
# |RSSI |channel |src_pan_id |src_addr_ext            |
=================================================
0 |12   |25      |0x4d65     |00:17:88:01:00:1e:30:e4 |

juma@juma:~/Z3sec$ sudo z3sec_touchlink --kb '1:5' --kb '1:6' -c '25' identify -
-target 00:17:88:01:00:1e:30:e4 -d 5
Setting up DualRadio:
KillerBee Radio (send): 1:5
Warning: You are using pyUSB 1.x, support is in beta.
KillerBee Radio (recv): 1:6
Scanning channel 25
Sending identify request to: 00:17:88:01:00:1e:30:e4
juma@juma:~/Z3sec$
```

*Figure 5.29:* Z3sec identify mode

- Change the brightness (and colour) of the ZLL device

Sending packets that are made up and designed as wanted enables to send not only beacon messages but also commands to ZigBee devices. The following image 5.29 shows the detection of a ZigBee LightLink device and then the command to let the device run the identify mode. This mode consists of a blinking pattern, and its length in seconds can be set with the command. This attack was put into the attack tool z3sec, the identify mode of the device can only be interrupted by powering it down.

An achievement worth mentioning at the end is a stable running version of the attack tool z3sec on the Atmel AVR Raven hardware, this also included patching the behaviour of the Killerbee systemsoftware as a tool used within z3sec. The background to

this is, that the AVR Raven USB stick is capable of sending and receiving, however, not simultaneously; in effect using simplex communication mode. Commands to the stick that are related to the stick configuration, when sent during the transceiver processing the data, most often previously before the fixing result in a crash of the firmware.

## 5.8 Experiment Setup Revision for Detection

As stated in the design paper the attacks need to get detected. This was discussed to be a distributed system, that is capable to detect on a high number of possible locations in parallel. Also the discussion was about creating the signalling so that attacks get not only detected but even more shown to a kind of operational centre. Within the year 2014 a toolset was released, that offers the basis for a distributed attack detection called beekeeperwids[6]. It consists basically of two components:

1. zbdrone
2. zbwids

The zbdrone is a component that is designed to provide high redundancy as they are the detectors. They are used to capture attacks. Second component zbwids is the data aggregating component, where suspicious attack patterns can get analysed according to definable matching informations.

An example is shown below.

As this tool is designed to enable further improvement it was taken as a basis to build the detection component of the honeypot upon to. The details, the programming, the outcome, the testing and the results are the topic of the next chapters and their sections.

This chapter specifically addressed the concepts of wireless networks and discussed basic elements of the experiments. The chapter also dove into the applied construction

[6] https://github.com/riverloopsec/beekeeperwids/

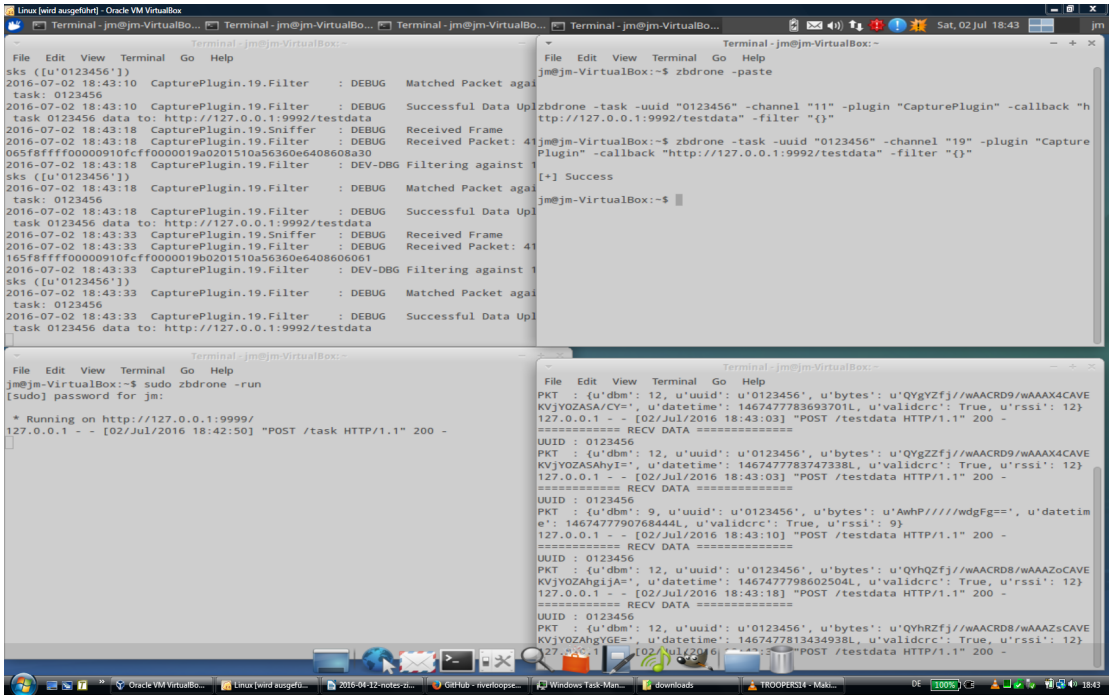


*Figure 5.30:* Beekeeperwids lab example

of the experiment in detail. The following basic design, through the analysis, up to the entire concept and specification, was discussed in this chapter. Then followed the planning of the implementation via the validation up to a workable prototype. Accordingly, a discussion on the treatment of the attacks aimed directly at the prototypes, followed by a controversial discussion about the attacker possibly bypassing the handling of his attacks.

Subsequently followed a comprehensive list of technical details of the testbed. These consist of various wireless sensor network components which together were necessary for the test setup. The discussion of the attacks that can be executed on the testbed rounded off the chapter and discussing the part of the testbed used for the detection of the attacks completed this chapter.

# Chapter 6

# Experimental results

THE following chapter discusses the results of the experiments in the testbed and presents the detection capability on the levels identified in the categorisation of the attacks. In addition, the results will also be discussed, and the conclusions that are now being drawn up will be summarised. The collected results so far will be the basis for analysis of honeypot strengths regarding the outcome of the project work.

## 6.1 Analysis of the Honeypot System Results

Common attack scenarios are tested on the honeypot for analysis. This is part of integration and verification. This section is about the research questions posed. First, there is a test of the honeypot and its reactions to common attack scenarios. Furthermore, it is tested how the honeypot reacts to more sophisticated attack scenarios. Those primary questions were already written down in previous chapters, as they were found to be appropriate questions. It is important to remember that those research questions were considered meaningful. Other research questions were not considered as appropriate. The appropriateness of for instance a signal jamming for a long time is not given. Therefore, only such attacks are considered that leave a possibility to react accordingly. Thus, the requirements from section 5.1.5 are fulfilled. The first requirement states, that detection has to be on a node-class device. This means, that of the hardware used everything for the detection should be the usual kind of hardware that is used for common wireless sensor network nodes. For the tests during this reseach it was considered and all devices used have the same chipset for the honeypot

production system that measures environmental parameters and also the detecting honeypot components. The second requirement states, that the detection and the production network have to be of the same type of devices. This considered, the hardware used for production and detection needs to be of the same kind. For the tests this was considered and all devices used have the same chipset for transmission for the honeypot production system that measures environmental parameters and the detecting USB stick. The third requirement states, that the detection is not allowed to be found out by the attacker. This requirement is followed in the experiments as the detection is passive in all of the cases. Therefore, it should not happen by design, that packets from the attack detection are sent in a manner, which would be obvious for the attacker to be entrapped. This third requirement is followed in the experiments as to shown in the following section. These requirements were considered the most appropriate to use, and these are also the ones that are followed in the experiments.

## 6.2 Evaluation as Planned

A last but very essential and closing task is remaining then, the summary of all of the project work, the results and findings of the Evaluation. The structure of this evaluation is created collating all papers and reports written earlier to give an overview of the work done. This assessment of the contribution is made to the field of WSN and ZigBee security. The evaluation is undertaken with regard to the detection capabilities. A distinction has to be made between a successful attack and detection, as well as a successful attack and no detection (this would then be a false negative detection). The category of unsuccessful attack and no detection as well as unsuccessful attack as and detection is considered to be irrelevant. The tests are performed by checking for applicability at the network protocol layer, when the question of the attack at every level of the network protocol is posed. In other words, all the problems of the layers are examined one after the other. This also makes it possible to check whether the detection with honeypot works at every layer. These layers were discussed earlier at section 2.1.7.

## 6.3 Evaluation of Prototype Misuse Detection Capability

Subsequent to the development of the detection capability and the integration of the collector and reporter to the honeypot framework, the whole system is put into testing and evaluation mode for a series of use cases and attack patterns. With the goal of detecting a high number of various cases of misuse, the network is guarded by the honeypot and predefined attacks will take place. All of the attacks will be documented to a log location and interpreted with the detected ones to establish a rate of success to failure. This rate can then be compared with others from simulated intrusion detection systems from literature research. This evaluation comparison will measure the success of the honeypot system in this work for wireless sensor networks. Together with its novelty value this will show, that a combination of existing work from different research topics is applied here to research in the field of honeypots on WSN. What are the steps of the evaluation, how it was done is presented here. Evaluated is a comparison between attacks and corresponding detection. These were performed according to the description in chapter 3. It was desired to know, whether the research questions are appropriate, and where the WSN honeypot is localised in a traditional network scheme. This research question has been considered in figure 3.5. The wireless sensor network honeypot should be located in a production network. For the categorisation into low or high interaction honeypot, the following can be stated: This is a high interaction honey pot with real decoys. Using virtual components is not seen as a possibility, as their existence could be found out. The research techniques were considered to be best when using experimental research however on a very open idea. This means, that other other researchers feedbacks are requested. Considered was a user acceptance testing, but the limited number of researchers on this topic and other so no large quantities of user experience can be expected for a quantitative approach. The question was whether it will be a fully new technique, but in fact it is more of a well decided combination of existing system components and the extension of their capabilities to push things further. The hybrid approach is not found to be true in its traditional meaning. As the

wireless sensor network as stated above is special, this detection is mostly based on a high interaction honeypot approach, but as said it is difficult to categorise a WSN honeypot in those two options, however, in tendency it is due to the production network as decoy more a high interaction WSN honeypot.

## 6.4 Evaluation of Prototype Quality of Security Enhancement

The conclusive part of the research takes place when qualitative testing is undertaken in the wild, as it were. Existing contacts to groups of other researchers will be used to invite them to test their tools against the honeypot system. This will result in insights into the stability and operability of the honeypot as an intrusion detection system. The outcome will be plenty of further logfiles and success rate values for attack and detection.

As stated in the chapter before, the basis for the detection mechanism is made on the beekeeperwids. Its design is thus that additional modules can be created that make a detection possible on basis that can be very well defined. As written, the proposal was to detect different attack types, and make the detection possible on all of the layers that can be thought of.

Of course, it was considered to use existing attack tools as a reference to obtain a set of attack patterns that can be used to generate detection filters. Therefore, a decision to analyse all of the known attack tools and generate according detection mechanisms seemed necessary was done. Examples are given below, showing the detection of all of the known and implemented attack patterns and their layers.

### 6.4.1 Detection of Attacks to the Physical Layer

As discussed before different attacks can happen to the physical layer (PHY). If there is a transmitter, which is more powerful than others nearby, then successful other transmissions do not happen any more on the frequency. A detection mechanism is possible in two ways, in a statistical analysis and in mathematical analysis. First attempt is with

```
datetime_now  = datetime.utcnow()
datetime_t60  = datetime_now - timedelta(seconds=60)
datetime_t120 = datetime_now - timedelta(seconds=120)

n60  = self.getPackets(valueFilterList=[('datetime','>',dateToMicro(datetime_t60))],
                       uuidFilterList=[uuid_task1], count=True)
n120 = self.getPackets(valueFilterList=[('datetime','<',dateToMicro(datetime_t60 )),
                                        ('datetime','>',dateToMicro(datetime_t120))],
                       uuidFilterList=[uuid_task1], count=True)

mean60 = n120/2.0 #60-120 seconds is a 60 second range so 2 * 60sec intervals
self.logutil.log("debug: Found (0) data packets in last 60 seconds, and (1) per 60 secs average over the prior 60 seconds (absolute (2)).".format(n60, mean60, n120))
# Calculate average of number of packets typically seen in a given timespan,
# and if new number is significantly lower (than 20% before) give alert message.
if n60 < (mean60*0.1):
    self.logutil.log(">>>>>ALERT: Noticed decreased data packets. (n60=(0), mean60=(1))".format(n60, mean60))
    self.registerEvent(name='InterferenceDetection', details={'channel':channel, 'n60':n60, 'n120':n120, 'mean60':mean60})
    self.generateAlert('InterferenceDetection')
```

*Figure 6.1:* Interference code example screenshot

the assumption that communication on the channel is taking place. Detecting a long period of non-transmissions can therefore be seen as a trigger. Also the circumstance of the second attempt can be seen as a result of an attack to the PHY: the checksum of ZigBee frames. If a lot of errors can be read over a specific period of time, then a disturbance by an attacker is to be expected.

These two possibilities led to the generation of a detection pattern for the detection tool and the mechanism was tested as shown below.

A lot of configuration testing took place to get a stable result, with the aim of reducing false positives and rendering a stable detection pattern.

### 6.4.2 Detection of Attacks to the Medium Access Control Layer

Attacks on the Medium Access Control (MAC) layer were seen in the previous chapter already. It is possible to generate frames that result in a removal of the associated node to the network. This attack has a specific pattern that exists as a detection pattern for the detection tool. The attack and the detection is shown below.

### 6.4.3 Detection of Attacks to the Network Layer

Attacks on the Network (NWK) layer similar to the MAC layer were seen in the previous chapter, too. It is possible to generate frames that result in a removal of the associated node to the network, this is an attack happening to the NWK layer. This attack has a specific pattern that exists as a detection pattern for the detection tool. The attack and the detection is shown below.

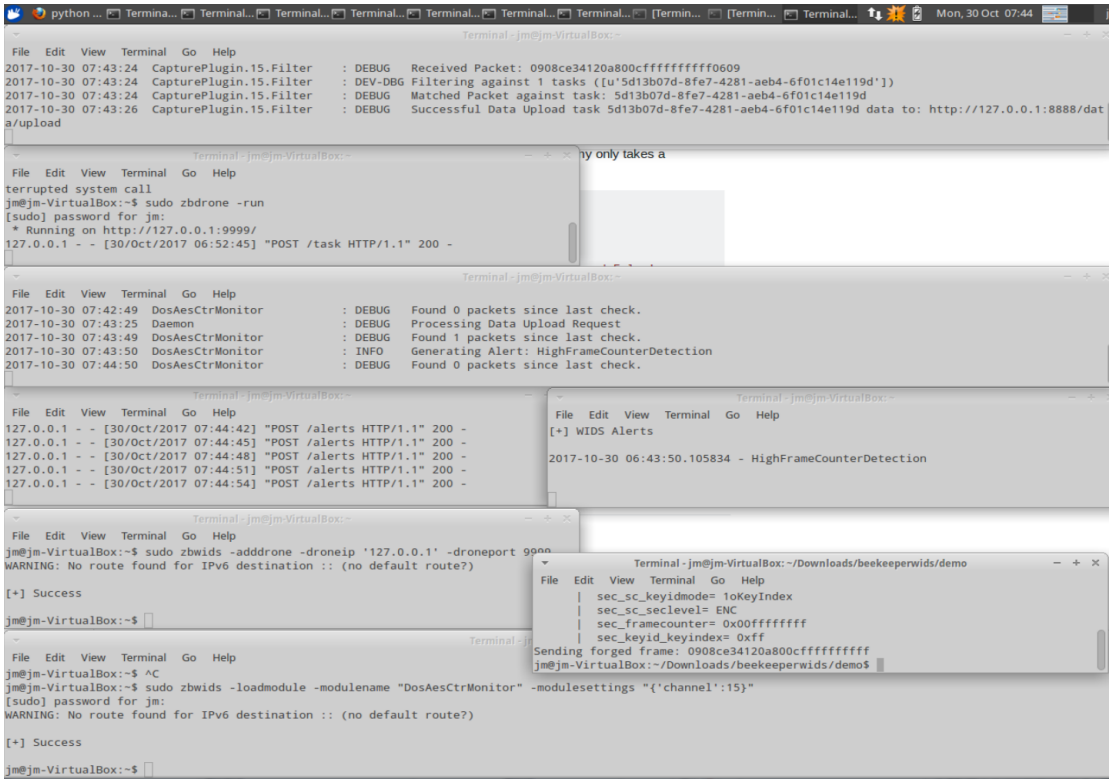



*Figure 6.2:* MAC layer attack detection example

But also the attacks that are used by the tool secbee and z3sec are targeting this network layer. Therefore additional detection patterns were written and put into the detection tool. The successful detection can be seen in the screenshots below:

### 6.4.4 Detection of Attacks to the Application Support Layer

The Application Support Layer (APS) attacks are based on the APS unbind attack, as shown in z3sec and secbee with which it is possible to let target devices join another network. This attack has a very special pattern that can get detected, it was therefore added to the detection tool.

A lot of configuration testing took place to get a stable result, with the aim of reducing false positives and rendering a stable detection pattern.

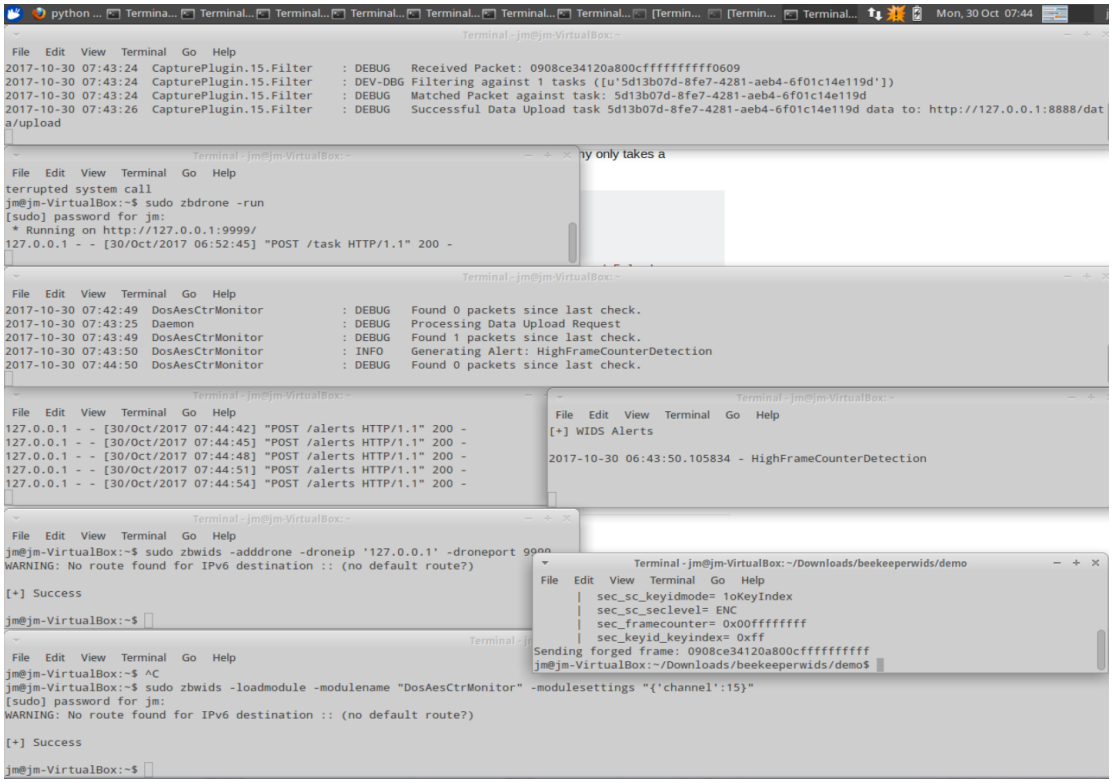



*Figure 6.3:* NWK layer attack detection example

## 6.5 Discussion of Experiments Output

It was shown, that it is possible to build a detection system with the purpose to act as a honeypot for wireless sensor networks. This honeypot was built as a prototype during this research and has shown, that it is possible to use the honeypot to detect attacks considered to happen on any layer that can be thought of. Results were shared with other researchers throughout the process and a concluding paper for a peer-reviewed conference will be prepared to get even more opinions from the research community.

The prototype was shown to be implemented on all the detection layers that were foreseen when the research started. Its capabilities are the better the more computational resources exist and are programmed to react on attacks. It was not shown on a battery driven network, this is subject for further research and porting the prototype from the lab to a battery driven testbed. Research after this presentation is not intended to stop, it is more of a direction of further research and development in further steps that is

```
juma@juma:~/Z3sec$ sudo z3sec_touchlink --kb '1:16' -c '25' join --target 00:17:
88:01:00:1e:30:e4 --channel_new 25 --addr_new 0x2342 --network_key a0a1a2a3a4a5a
6a7a8a9aaabacadaeaf
Setting up Radio:
KillerBee Radio (send+recv): 1:16
Warning: You are using pyUSB 1.x, support is in beta.
Scanning channel 25
Sending network join request to: 00:17:88:01:00:1e:30:e4
###[ 802.15.4 ]###
  fcf_reserved_1= 0L
  fcf_panidcompress= False
  fcf_ackreq= True
  fcf_pending= False
  fcf_security= False
  fcf_frametype= Data
  fcf_srcaddrmode= Long
  fcf_framever= 0L
  fcf_destaddrmode= Long
  fcf_reserved_2= 0L
  seqnum    = 206
###[ 802.15.4 Data ]###
     dest_panid= 0xffff
     dest_addr = 00:17:88:01:00:1e:30:e4
     src_panid = 0x366
     src_addr  = 0b:9b:26:0a:07:f4:8b:be
###[ Zigbee Network Layer for Inter-PAN Transmission ]###
        reserved  = 0L
        proto_version= 2L
        frametype = 3L
        reserved  = 0L
###[ Zigbee Application Layer Data Payload for Inter-PAN Transmission ]###
           frame_control=
           delivery_mode= unicast
           frametype = 3L
           cluster   = ZLL_commissioning
           profile   = ZLL_Light_Link
###[ Zigbee LightLink Commissioning Cluster Frame ]###
              reserved  = 0L
              disable_default_response= 1L
              direction = client2server
              manufacturer_specific= 0L
              zcl_frametype= 1L
              transaction_sequence= 66
              command_identifier= network_join_router_request
###[ ZLL: Network Join Router Request ]###
                 inter_pan_transaction_id= 0x4623b74
                 pan_id_ext= 10:55:08:a9:4d:8b:63:e5
                 key_index = 4
                 encrypted_network_key= 0x82ecd373f297771324f8207682aee955L
                 network_update_id= 1
                 channel   = 25
                 pan_id    = 0x366
                 network_address= 0x2342
                 group_id_begin= 0x0
                 group_id_end= 0x0
                 free_network_address_range_begin= 0x0
                 free_network_address_range_end= 0x0
                 free_group_address_range_begin= 0x0
                 free_group_address_range_end= 0x0
Scanning channel 11
Scanning channel 15
Scanning channel 20
Scanning channel 25
Target was successfully joined to network 0x366.
```

*Figure 6.4:* Z3sec usage to let a device join a new network

necessary and can be foreseen for future research groups and projects.

The research was straightforward in the idea, at the same time it was challenging due to the restrictions present. On one hand, there is the novelty aspect, that the honeypot approach was not investigated as detailed before in the structure and precision done in this work, which aspect also means scarce primary references and literature on similar topics to cite.

As it was presented here, the honeypot in ideal form consists of a normal wireless sensor network, that seems to perform normal sensing processing and transmitting of data, with an additional module capable to read all the networking packets.

The networking aspect was indeed crucial, which led to a review of the research goals and objectives; for instance whenever the wireless sensor network is attacked by a denial of service attack, then reporting cannot take place on the same communication link as it is affected by that denial of service as well.

This insight led to these two possibilities:

1. perform the canary approach[1], as long as the canary is present, then no denial of service attack is occurring, or
2. refrain from using the same communication link, i.e. use an extra channel 5.1.2 for feedback.

It is obvious that a research relying on an assumption that is valid only sometimes, is not a scientific one; therefore the approach to use the same communication link as a feedback channel on which the attack will occur cannot be seen as a reliable one.

Alternatives for the feedback channel were discussed in section 5.1.2, in the same medium (frequency and signal encoding) they can be summarised as unreliable; on another transport medium such as light (laser, infrared and comparable) there could

[1] The canary in the coal mine is the idea of placing a canary in a mine to detect poisonous gas, in this case carbon monoxide

be physical obstacles, for instance a distance too close (in the case of IR) or else prohibitive cost (with laser).

Further investigation shows, that a honeypot for wireless sensor network most likely may need to have a wire-bound feedback. This can be seen as a topic for further work and a future research topic.

In this section the approach will be verified to have conformed to the research goal, to follow the blueprint for a science of cybersecurity, as proposed by Schneider (2011).

It was shown, that this approach was followed and that the results are promising to lead to a new direction of a science of cybersecurity; and as this is a new outcome, presents a novel contribution and is reproducible by performing exactly the steps documented, additionally the source code written for the detection routines, for the analysis, and for the conclusion are available on request, and for possible future collaboration, there are definite opportunities at the research institutes in Darmstadt, where this work began.

### 6.5.1 Validation of Assumptions

An assumption to start the research was, that there is a need of security in wireless sensor networks, and honeypots are possible to fill this gap. Thus, more research needed to be done to present a honeypot for wireless sensor networks. The concepts of other wire-bound honeypots could be taken for a first start and be refined with additional recommendations on wireless networks.

### 6.5.2 Observation of Emergent Behaviours

The upcoming points will now serve to prove, that the vision about generating a prototype that fulfils all the desired specification requirements was realistic. In this case, if the generation of a prototype is desired, then as stated before, the use of formal specification and proof of correctness would be the wrong approach. The science of cybersecurity shows a way to generate knowledge by experiments, and the construction of a prototype will be be failure-free. Therefore the goal is to contribute knowledge, which

can then be used in a future phase to deliver a product through real-world development using plenty of resources and possibly years of development time. But before that, a requirements specification and a couple of metrics is created here, to make sure that there is a possibility to measure the outcome of such future development. In chapter 4, there were written a lot of requirements and points that needs to be regarded during the prototype generation. This is done in the ambitious approach to show the results hold and provide additional knowledge, that the results that were reached are also conforming to the specification, and for the requirements that were not fulfilled as a whole, to show the restrictions that are present in this environment of wireless sensor networks, that make it necessary to find another approach to a solution. This knowledge fills the section “limitations of the work” and also the section “further research proposed steps”.

This chapter includes the results of the experiments in the testbed. The detection capability in the layers is shown, which is referring to the categorisation of the attacks from chapter two. It showes the detection capability on the levels identified in the categorisation of the attacks. In addition, the nature of the results is incorporated into the discussion, and many of the conclusions are drawn from it, which are incorporated into the discussion.

# Chapter 7

# Conclusion

As written in the section 4.7 the path of experimental setup is followed. This is done with the goal of knowledge generation, which can then be seen as furthering the science of cybersecurity. This last chapter starts with a summary of the results, then continues with the achievements of this research, in direct comparison with the goals set in the beginning, up to the topic of the knowledge generated. The limitations must of course be mentioned and the chapter then ends with conclusions and recommendations for subsequent further research. The experimental setup in the testbed shows, that attacks are recognised and captured, and that this is done on all of the layers that were regarded in theory in the literature research phase in chapter 2. The attacks that are possible on these layers in chapter 5 result in a basic design, a specification of a detailed design of the prototype, and then to an experimental setup as testbed. The conclusion from the results is intended to be put into direct correlation to the previous report titled “Honeypot framework for ZigBee network architectures”. This paper will be the conclusion of the project work, especially regarding performance of the honeypot for wireless sensor networks. One of the topics should be a statement on the general capability of honeypots in wireless sensor networks. As further planned paper will target the interpretation of the findings, titled “Results of honeypot framework for ZigBee network architectures”.

The conclusion of the honeypot for WSN research project is, that it is a takeoff into a direction of development which proved to be correct. Most of the research assumptions were shown to be correct, and possible directions of improvement were elaborated.

## 7.1 Research Achievements

This thesis deals with such networked systems in the case of "wireless sensor networks" and highlights topics on the malicious interaction with those systems. It also primarily shows an extension of existing security measures by the introduction of a honeypot for wireless sensor networks.

This is a question of research, whether it is possible to consider the security of a WSN as endangered and to increase the security in the WSN with the help of additional protection. It turned out that the literature describes a variety of attacks and that they are not only academic in nature and can exist in theory, but have actually found a place in attack tools. The researchers on this topic are working on very different aspects, while there is the development of hardware on the one hand, on the other hand individual tools are also produced. But also, by merging several such tools, major projects are done. An example is the killerbee framework created by a single developer as a basis. It was promoted by many individuals, supplemented by further tool scripts, and is maintained today. The question now arises as to whether the research question has been answered. It is broadly based on sub-items and also includes considerations on how, for example, improvement can be promoted.

The research aims from section 1.4 were met by the objectives as follows:

**Objective 1:** Review and summary of the usual strategies for detecting attacks on wireless sensor networks and presenting existing solutions, as well as highlighting promising strategy.

The review of related work within chapter 3.2 gave a broad overview of the attack detection possibilities and revealed the potential use of honeypot technologies in wireless sensor networks as a new strategy, while at the same time stating that a new approach also needs a new overview on the challenges implied by this introduction to a new area.

**Objective 2:** Perform literature review to understand trade-offs in wireless sensor networks security.

Chapter 2 includes a thorough literature review on current wireless sensor network basics with the goal to include the limitations due to hardware constraints and furthermore includes lists of vulnerabilities and attacks. Also, it describes the challenges in wireless sensor networks, and in the section 2.2.7 it shows the categorisation of attacks, which was done with the goal to get measurable achievements. This is due to a simplification, because if a category of attacks can be detected, a part of this objective is reached.

**Objective 3:** For honeypots' common definitions and setups, perform an analysis and the concept translation to the wireless sensor network domain evaluation, then design and implement a prototype.

In Chapter 4, the analysis of the current definition in wire-bound IP based networks was conducted; additionally the interpretation for wireless IP based networks was completed to get a solid basis for the following concept translation that was undertaken. Results of the translation have shown that most definitions from the IP-networks area could be translated to the wireless sensor networks domain.

**Objective 4:** Prototype development to detect the proposed attacks as a model of completeness by using determined scenarios.

Chapter 5 shows the experimental setup in the testbed and the achievements in detection are presented. The results in the model of completeness in the determined scenarios are shown. As was shown in the chapter on attack categorisation, there are four attack layers which were taken into account for the honeypot capabilities. The novelty of the honeypot prevents a comparison in a direct manner; however, it was shown that this approach was correct to be a promising one.

## 7.2 Key Findings - Contribution to the Knowledge

It is a question of what the contribution to knowledge is here. It can be said that there was no earlier similar work. The wide-ranging view of all topics of this thesis is a comprehensive work. The importance of existing research-outcomes remains, but important aspects are added. For example, there is an understanding that the Raven USB Stick actually works with the z3sec attack tool. Corresponding changes were made in the context of the thesis. Furthermore, this attack via z3sec was integrated into the beekeeper's detection scheme. Signalling to the operator of the network is also an important additional point. The basis for networking is created. Further effort is needed to carry out such work. Knowledge again, as stated, is a central part of this section. What has been contributed to the knowledge that was provided? It has been checked whether rules of the wired network apply, but it is not possible to access, control or protect data in a central location, because it is a radio network and thus a shared medium. It is more difficult to introduce security in a radio network. The realisation remains that similar to a network built with a network hub, corresponding problems exist. This no longer corresponds to the technique of a wired network that nowadays is built up with the help of a switch. So, it is to represent a WSN rather with a network that has a hub, i.e. all devices can read data from all other devices, which basic assumption takes a researcher much further, and this knowledge carries further remarks as well. If now there is the knowledge that all networks are essentially of this kind, i.e. they are as it was 30 years ago in the LANs quite usual, so it is clear that it was in the essential characteristics of that time which was supplanted by improvements in technology here, should be considered necessary again. It is also clear that safeguards that were in use at the time could then be reinstated. However, it is not unfortunately possible to improve the network here because the fact of the wireless shared medium will not change, as a result the network properties cannot be changed, the WSN will not be changed to a wirebound network. This is also with a realisation precisely from this thesis, that there are other intelligent additional security measures needed to increase security again.

As a result it is time to talk about the results. To this end, it is necessary to consider all aspects of the present research. On the one hand it was possible to use the work of previous research, on the other hand it was possible to combine and supplement it skilfully. A framework was created with the purpose of a contribution to knowledge. It consists of a knowledge of the present technique of the knowledge of attacks, about knowing defences. It is about the knowledge of adequacy and legality. It is about knowing attack and detection, it is also about the knowledge of the functionality and the functionality of the adequacy, whether it is adequate.

With regard to theoretical advancement, the question now arises as to what the theoretical improvement is. Many results suggest that this work is different from what would have been set up in a LAN with an IP-based network, for example. Concrete improvements are available: in adapting existing attack tools, increasing stability and making them workable. Furthermore, in the associated possibility to carry out attacks, it was then also proven to recognise these attacks. In principle, every attack can be detected when enough data is collected. Furthermore, it is necessary to map detection patterns to this data. In a pure honeypot, where slightly higher interaction is not expected to take place, than what the system generates in background noise. So, each additional interaction likely represents an attack. In this way, intrusion detection is always possible. Any interaction with the honeypot then is an attack. However, if the WSN is a production network that is actually in operation, making use of a honeypot for this WSN is quite difficult. In theory, any attack can be identified with a honey pot. Once again, it has become clear that the approach, as with a wired network, is not effective. In a wired network, as said not any interaction is a potential attack. However, the goal of a WSN is in the cooperative action of different components with each other. The protocols require binding and dissolving the communication partners. It is always possible to assume an attack in theory. However, it raises the fundamental question of whether the honeypot for WSN should remain a purely theoretical tool, or whether it is also used for securing production networks. The aim of this work is to shed light on this theoretical aspect. According to the present state of knowledge, the difficult securing of the components

speaks for the use outside of productive networks. Especially with the research paper of Ronen et al. (2018), it seems problematic to leave the networks vulnerable. However, it cannot be effective to make all networks unproductive. So, it takes a thorough consideration about what needs to be found out: which compromises in security are to be tolerated, which losses are acceptable, which overhead is acceptable. Of course, the easiest way for this work would have been to leave the network completely generic. However, this is an obstacle, especially in the case of more complex attacks. as an example, the network rejoin attack from the z3sec: This runs over several stages. On the one hand, when the device is hijacked, it is also integrated into a new network, and then it is under the control of the attacker. This attack requires a lot of data packets to be sent through the area. If the device is not to be attacked, it does not send a response if the device does not exist, no replies are sent so the attacker does not receive any feedback, so the attack remains virtually in its infancy. It may be that the first or second step of the attack executed, but does not follow the usual steps. This example illustrates quite well that a purely unproductive network without actual proper interaction cannot be suitable to detect attacks that show a high level of complexity. Such simulated unproductive network can be useful however to detect simple attacks and to report. Both scenarios have their pro and cons, the theory can be put forward with it, as I said, the simple attacks in unproductive networks can be detected, but it is necessary to have productive networks, because only they can interact accordingly and let the attack be happen completely in the actual production network. If an attacker tries this, it can then be recorded here. This finding was not available before this work. The way chosen here was an iterative approach. First of all, there was the idea, and a report about it (Research Proposal). Then, the refinement of this idea and a resulting paper for a workshop about it (SEIN). Next up, more background research on the topic and a further paper about the outcome of that (CERC). It was deemed important to work together with others and thus to start exchanging ideas with other research groups about this topic, which received positive feedback and suggestions on how to continue. Consequentially, the approach chosen here and the steps planned to take were presented and a conference

paper written on it (MoMM). The detection concept on all of the layers was planned, followed by a paper about it (DEXA). Regarding the fundamentals of a honeypot in a wireless sensor network, it was deemed necessary to publish a paper to base work upon (ECSA).

Every system is expected to be failure prone to a certain degree. There are existing methods for improvement and detection of technical vulnerabilities, where concepts such as firewalls and malware-detection are well known examples. With regard to malicious interaction, this is already existent in the “Internet of Things” (IoT). Now, the environment in the IoT is completely different from WSNs, as IoT devices are mostly miniaturised and low energy profile devices. These may have an Internet gateway, but other parts do not need to be wire-bound. Wireless sensor networks, on the other hand, form networks with mesh routing or star topology.

With the digitalisation of the future so-called smart home, and the Internet of things forming a network for a comfortable living new attack vectors become noticeable that need to be addressed during development and runtime. With the honeypot for wireless sensor networks an important step for improvement has been undertaken.

It was shown to be possible to do the transformation from a wire-bound technology over wireless implementation considerations to a wireless sensor network setup and to achieve results.

## 7.3 Limitations

The limitations exist in different aspects. First of all, the research work was undertaken with the goal to find out what is possible to do. When asking for the limits this means that a thorough combination of the limitations that exist in the following places has to be shown. If it is considered, that the whole knowledge of the work consists of the fact that on the one hand the background must be shown as to how a monitoring solution can work at all. It is obvious that these insights can only arise if a monitoring solution is also operated. The operation of this solution was originally intended as a completely

new development, but this is of course very meaningless, if there is already a corresponding solution, the beekeeper, which interestingly then also exactly corresponds to the criteria that also in advance also had been created. And so it just turned out that the wheel no longer had to be completely reinvented, but many aspects already existed. Also, that the solution with which is being worked on has already existed in various areas once the securing of the physical layer came to be. So it is no wonder that a whole range of sniffers already existed in packet capture tools. These possibilities are beautiful and all work, and have all advantages and disadvantages. The main aspect to be named here is the radio band, i.e. the frequency spectrum which is being worked on, as this can of course change and it is a fact that for common chips, it is just difficult to be able to do everything at the same time. That's why researchers then moved on to monitoring more than one channel by using more than one receiver. These points were explained in chapters 2 and 3 and it essentially means that there is a limitation precisely because only one receiver was used in the present test setup. Of course the whole can be extended here, more can be done, more hardware can also be installed, more drones can run on different channels.

With a honeypot for wireless sensor networks a new method is introduced with the goal to contribute to the science of cybersecurity. The following limitations of this research are present.

The use of a formal specification of a honeypot was discussed in the research concept section and still seen to be a way into the right direction to get a deeper understanding of the topic.

In the section 5.1.3 it was discussed that the WI feature is only to be implemented if there is an Intranet and the Data Control requirement of a wireless honeypot needs to be satisfied only if there is data that can be controlled. In the case of a research honeypot, that is primarily set up to understand attackers this internal network does not exist. It can be therefore seen as a positive aspect, that productive data can not get lost. This is done through security by isolation, other authors also name it segmentation

or separation.

For a final and stable product a rewrite is suggested and as this thesis is not meant to contain pure product development, this step is a point to considered as limitation and for further research.

## 7.4 Further Research Recommendations

Further research is seen in the model changing as an automatic feature of a honeynet as it was discussed in chapter 2 as well as also the use of formal specification. As a next step in future research the focus should surely be a collaboration platform as well as a resource downsizing experiment.

With respect to the current work undertaken, a set of new ideas arose in the course of the thesis. There is the possibility to do more research on the use of single board platforms such as the Raspberry PI 3 or newer for data analysis, something that has to be done on a device with the database of captured traffic. This research was begun already before handing in the thesis, and running a Linux operating system (Debian) with the killerbee sticks was shown to be working on a Pi model 3B in 2018. The limited resources were not seen as an issue, although no stress test was done to the platform, the reliability was experienced as high, but not all tested components were working. With every tested framework, attack and detection tools, some software dependencies were not met. No fully working setup could be achieved, always a failure was experienced. To be precise, a suggested research would include the new Raspbian based on version 10 (Buster) with 64-bit support for ARM, the new Python 3.8, the new Killerbee 2.7.1, and then get the attacks and detection working with all the new dependencies. Furthermore, there were other new attacks presented at the zigdiggity-tool before this thesis was finished, which should also in future be evaluated and put into the detection scheme.

Also, the use of GPS coordinates at the honeypot nodes was considered, especially when the devices could be mobile, then the location could be of interest. Connecting

a GPS device to an embedded platform is an easy task and was done before already. The connection is a simple serial protocol, is specified as NMEA 0183 and in summary states, that the data is transferred at 4800 baud on a serial connection. This means, that every GPS device that is compatible to the standard sends the coordinates on the serial line at 4800 baud and when this is read it sends the new coordinates as well. GPS chips were bought for the project, namely called SIM18. However, in the end they were not put into the project, because it was not the main research aim.

Also, the use of other frequencies for the reporting channel was thought of and a component for the 900MHz band was chosen, the ANY900 from Dresden electronics. Their form factor is about the size of the Atmel WSN components. For more computational resources, newer development boards from Atmel were tested, such as the ATmega256RFR2 XSTK or the EK kit for evaluation purpose. However, switching the platform was not seen as necessary there.

To simulate a production environment in the wireless sensor network, a device was thought of that could be safeguarded. With respect to the use of such systems this could either be a smart home or a smart factory. A device running there was thought of to be simulated by using a 3D printer, because it is a remotely controllable device including a computational unit, that is remotely configured and surveilled, to produce something such as is done by every automatisation technique.

Another idea is the use of high-end devices to receive and to send data at frequencies that are different from the band where the attack is happening. Such devices can be found in software defined radios (SDR). This means, that the whole receiving and transmitting is configured by software. Where before the transceiver was basically a set of electrical circuits with well-chosen attributes, the SDR is capable of sending and receiving at almost all frequencies and it can be configured to use any modulation and any encoding on top. This means, that with an SDR it is possible to listen to music at the FM radio, it is possible to watch TV with DVB, but it is also possible to listen to the traffic in the WSN. The interest is, whether it could be possible to enrich a WSN

honeypot with an SDR, to make sure really nothing is overseen at the receiving side of it.

It would be an interesting test to find out if an appropriately high number of detectors and sensors can be located at the site for an attack on all channels and whether the corresponding software can also process this information flood. It is also interesting to see what possibilities there are, especially when using software defined radio (SDR). This is a way of interpreting, as already written in another paragraph, to receive signals, but of course also to be able to send them. The number of channels used by the receiver as it is currently set up is limited, it is not possible to monitor all channels at the same time, so it would be an interesting challenge to monitor all channels at the same time. It would also be an interesting challenge to determine with what number of signal transmitters, i.e. sensors in the Wireless Sensor Network, this can be accomplished.

This concludes the chapter, which was started as announced with the summary of results, then listed the achievements of research in direct comparison with the goals created at the beginning of the thesis in chapter one, followed by the topic of the knowledge generated. Finally, the limitations have been mentioned and after conclusions, has subsequently recommended further research.

# List of references


Alcaraz, C. & Lopez, J. (2010), 'A Security Analysis for Wireless Sensor Mesh Networks in Highly Critical Systems', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **40**(4), 419–428.

Andriesse, D., Rossow, C., Stone-Gross, B., Plohmann, D. & Bos, H. (2013), Highly resilient peer-to-peer botnets are here: An analysis of gameover zeus, *in* 'MALWARE 2013 8th International Conference on Malicious and Unwanted Software: "The Americas", IEEE, 2013', pp. 116–123.

Aschenbruck, N., Bauer, J., Bieling, J., Bothe, A., Schwamborn, M., Prof. Dr. Martini, P., Dr. Pfisterer, D., Hakim, K., Prof. Dr. Fischer, S., Buschmann, C., Gehring, F. & Wieschebrink, C. (2011), 'Wireless sensor networks-labor'.
**URL:** *http://net.cs.uni-bonn.de/fileadmin/ag/martini/projekte/wsnlab/ wsnlab_abschlussbericht.pdf*

Atmel Corporation (2009), 'Atmel acquires meshnetics zigbee intellectual properties - atmel's microcontrollers and rf transceivers combined with zigbee pro technology deliver best in class, low energy consumption zigbee solutions'.

Atmel Corporation (2012), 'Atmel AVR 2050: Atmel BitCloud Developer Guide'.

AVR (2008), 'AVR Raven', Website.
**URL:** *http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4291*

Barnett, R. (2006), *Preventing Web Attacks with Apache*, Pearson Education.

Baumann, R. (2002), 'Honeypots'.

Bechtold, T. & Heinlein, P. (2004), *Snort, Acid & Co*, Root's reading, Open Source Press.

Benson, T. & Chandrasekaran, B. (2017), Sounding the bell for improving internet (of things) security, *in* 'Proceedings of the 2017 Workshop on Internet of Things Security and Privacy', IoTS&P '17, ACM, New York, NY, USA, pp. 77–82.
**URL:** *http://doi.acm.org/10.1145/3139937.3139946*

Bhuse, V. & Gupta, A. (2006), 'Anomaly intrusion detection in wireless sensor networks', *J. High Speed Netw.* **15**(1), 33–51.
**URL:** *http://dl.acm.org/citation.cfm?id=1140563.1140567*

Böck, H., Zauner, A., Devlin, S., Somorovsky, J. & Jovanovic, P. (2016), Nonce-disrespecting adversaries: Practical forgery attacks on GCM in TLS, *in* '10th USENIX Workshop on Offensive Technologies (WOOT 16)'.

Bringer, M. L., Chelmecki, C. A. & Fujinoki, H. (2012), 'A survey: Recent advances and future trends in honeypot research', *International Journal of Computer Network and Information Security* **4**, 63–75.

Bundesamt für Sicherheit in der Informationstechnik (2009), 'Drahtlose Kommunikationssysteme und ihre Sicherheitsaspekte', Website.
**URL:** *https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Broschueren/DrahtlosKom/drahtkom_pdf.pdf*

Cache, J., Wright, J. & Liu, V. (2010), *Hacking Exposed Wireless, Second Edition*, Hacking Exposed, McGraw-Hill.
**URL:** *http://books.google.de/books?id=_22wK66-Dh8C*

Cai, H., Jia, X. & Sha, M. (2011), 'Critical sensor density for partial connectivity in large area wireless sensor networks', *ACM Trans. Sen. Netw.* **7**(4), 35:1–35:23.
**URL:** *http://doi.acm.org/10.1145/1921621.1921629*

Cesare, S. (2014), 'Breaking the security of physical devices', *Presentation at Blackhat* **14**.

Chen, X., Makki, K., Yen, K. & Pissinou, N. (2009), 'Sensor network security: a survey.', *IEEE Communications Surveys and Tutorials* **11**(2), 52–73.
**URL:** *http://dblp.uni-trier.de/db/journals/comsur/comsur11.html#ChenMYP09*

Clark, D. D. & Wilson, D. R. (1987), A Comparison of Commercial and Military Computer Security Policies, *in* '1987 IEEE Symposium on Security and Privacy', pp. 184–195.

Clarkson, M. R. & Schneider, F. B. (2010), 'Hyperproperties', *Journal of Computer Security* **18**(6), 1157–1210.
**URL:** *https://doi.org/10.3233/JCS-2009-0393*

Cole, E., Krutz, R. L. & Conley, J. (2005), *Network Security Bible*, Wiley.
**URL:** *http://amazon.com/o/ASIN/B00817SBEG/*

Combs, G. (2012), 'Wireshark project', Website.
**URL:** *www.wireshark.org*

Control4 (2009), 'Control4 surpasses milestone of one million zigbee products', Website.
**URL:** *http://www.control4.com/about-us/press/2009/08/24/one-million/*

Cunha, A. (2007), 'On the use of ieee 802.15.4/zigbee as federating communication protocols for wireless sensor networks', Website.
**URL:** *http://repositorio-aberto.up.pt/bitstream/10216/10939/2/Texto%20integral.pdf*

Das, V. V. (2009), Honeypot scheme for distributed denial-of-service, *in* 'Proceedings of the 2009 International Conference on Advanced Computer Control', ICACC '09, IEEE Computer Society, Washington, DC, USA, pp. 497–501.
**URL:** *http://dx.doi.org/10.1109/ICACC.2009.146*

Davis, M. (2009), 'Smartgrid device security - adventures in a new medium', Website.
**URL:** *https://www.blackhat.com/presentations/bh-usa-09/MDAVIS/BHUSA09-Davis-AMI-SLIDES.pdf*

Denning, T., Kohno, T. & Levy, H. M. (2013), 'Computer security and the modern home', *Communications of the ACM* **56**(1), 94.

DePetrillo, N. (2009), 'Power hungry people - making sense of new critical infrastructure threats', Website.
**URL:** *http://data.proidea.org.pl/confidence/6edycja/materialy/prezentacje/ CONFidence2009_nick_de_petrillo.pdf*

Dexheimer, T., Dukanovic, S., Henkel, T., Laabs, M. & Markert, J. (2014), Die nächste Industrielle Revolution kommt mit Sicherheit, *in* 'KommA 2014 - Jahreskolloquium Kommunikation in der Automation', Lemgo, Germany.

Dornseif, M., Gärtner, F. C. & Holz, T. (2004), 'Vulnerability assessment using honeypots', *Praxis der Informationsverarbeitung und Kommunikation* **27**(4), 195–201.

Dowling, S., Schukat, M. & Melvin, H. (2017), A zigbee honeypot to assess iot cyberattack behaviour, *in* '2017 28th Irish Signals and Systems Conference (ISSC)', pp. 1–6.

Dwivedi, A. K. & Vyas, O. P. (2011), 'An exploratory study of experimental tools for wireless sensor networks', *Wireless Sensor Network* **3**(7), 215–240.
**URL:** *http://dx.doi.org/10.4236/wsn.2011.37025*

Eady, F. (2007), *Hands-On ZigBee: Implementing 802.15.4 with Microcontrollers (Embedded Technology)*, Newnes, Newton, MA, USA.

Eaton Corporation (2007), 'Eaton home heartbeat', Website.
**URL:** *http://www.homeheartbeat.com/HomeHeartBeat/*

Eaton Corporation (2012), 'Eaton vault sense', Website.
**URL:** *http://www.eaton.com/Eaton/ProductsServices/Electrical/ ProductsandServices/ElectricalDistribution/NetworkProtection/Communications/ VaultSenseWirelessCurrentSensor/PCT_367526*

Eckert, C. (2009), *IT-Sicherheit - Konzepte, Verfahren, Protokolle (6. Aufl.)*, Oldenbourg.
**URL:** *http://www.oldenbourg-wissenschaftsverlag.de/olb/de/1.c.1679883.de*

Eilers, C. (2014), 'Internet of (in)security?', *Entwickler Magazin* **1**(1), 67–72.
**URL:** *https://entwickler.de/produkt/entwickler-magazin-spezial-vol-1-internet-things/*

Elahi, A. & Gschwender, A. (2009), *ZigBee Wireless Sensor and Control Network*, Prentice Hall Communications Engineering and Emerging Technologies Series, Pearson Education.
**URL:** *http://books.google.de/books?id=481PeRPyiEoC*

Emami-Naeini, P., Dixon, H., Agarwal, Y. & Cranor, L. F. (2019), Exploring how privacy and security factor into iot device purchase behavior, *in* 'Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems', CHI '19, ACM, New York, NY, USA, pp. 534:1–534:12.
**URL:** *http://doi.acm.org/10.1145/3290605.3300764*

Fabbricatore, C., Zucker, M., Ziganki, S. & Karduck, A. (2011), Towards an unified architecture for smart home and ambient assisted living solutions: A focus on elderly people, *in* 'Digital Ecosystems and Technologies Conference (DEST), 2011 Proceedings of the 5th IEEE International Conference on', pp. 305 –311.

Faludi, R. (2010), *Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing*, 1 edn, O'Reilly Media.
**URL:** *http://amazon.com/o/ASIN/0596807732/*

Farahani, S. (2008), *ZigBee Wireless Networks and Transceivers*, Newnes, Newton, MA, USA.

Fernandes, E., Jung, J. & Prakash, A. (2016), Security analysis of emerging smart home applications, *in* '2016 IEEE Symposium on Security and Privacy (SP)', pp. 636–654.

Fraunholz, D., Lipps, C., Zimmermann, M., Duque Antón, S., Mueller, J. K. M. & Schotten, H. D. (2018), Deception in information security: Legal considerations in the context of german and european law, *in* A. Imine, J. M. Fernandez, J.-Y. Marion, L. Logrippo & J. Garcia-Alfaro, eds, 'Foundations and Practice of Security', Springer International Publishing, Cham, pp. 259–274.

Freakduino (2011), 'Freakduino project', Website.
**URL:** *http://freaklabs.org/*

Ghosal, A. & Halder, S. (2013), Intrusion detection in wireless sensor networks: Issues, challenges and approaches, *in* S. Khan & A.-S. Khan Pathan, eds, 'Wireless Networks and Security', Signals and Communication Technology, Springer Berlin Heidelberg, pp. 329–367.

Ghosh, P., Mayo, M., Chaitankar, V., Habib, T., Perkins, E. & Das, S. K. (2011), Principles of genomic robustness inspire fault-tolerant wsn topologies: a network science based case study, *in* '2011 IEEE international conference on Pervasive computing and communications workshops (PERCOM workshops)', IEEE, pp. 160–165.

Giannetsos, T. & Dimitriou, T. (2013), Spy-sense: Spyware tool for executing stealthy exploits against sensor networks, *in* 'Proceedings of the 2Nd ACM Workshop on Hot Topics on Wireless Network Security and Privacy', HotWiSec '13, ACM, New York, NY, USA, pp. 7–12.
**URL:** *http://doi.acm.org/10.1145/2463183.2463186*

Giannetsos, T., Dimitriou, T., Krontiris, I. & Prasad, N. R. (2010), 'Weaponizing wireless networks: An attack tool for launching attacks against sensor networks'.

Gislason, D. (2008), *Zigbee Wireless Networking*, pap/onl edn, Newnes, Newton, MA, USA.

Glanz, A. & Jung, O. (2010), *Machine-to-Machine-Kommunikation*, Campus Verlag GmbH.
**URL:** *http://amazon.com/o/ASIN/3593392240/*

Goodspeed, T. (2007*a*), 'Memory-constrained code injection', Website.
**URL:** *http://travisgoodspeed.blogspot.com*

Goodspeed, T. (2007*b*), 'Msp430 buffer overflow exploit for wireless sensor nodes', Website.
**URL:** *http://travisgoodspeed.blogspot.com*

Goodspeed, T. (2009), 'Improving the msp430 fet', Website.
**URL:** *http://goodfet.sourceforge.net/*

Goodspeed, T., Bratus, S., Melgares, R., Speers, R. & Smith, S. (2012), Api-do: Tools for Exploring the Wireless Attack Surface in Smart Meters, *in* 'System Science (HICSS), 2012 45th Hawaii International Conference on', pp. 2133 –2140.

Grimes, R. (2005), *Honeypots for Windows*, APress.

Grudziecki, T., Jacewicz, P., Juszczyk, L., Kijewski, P. & Pawlinski, P. (2012), Proactive detection of security incidents ii - honeypots, Technical report, European Network and Information Security Agency.
**URL:** *http://www.enisa.europa.eu/activities/cert/support/proactive-detection/proactive-detection-of-security-incidents-II-honeypots/at_download/fullReport*

Gu, Q. & Noorani, R. (2008), Towards self-propagate mal-packets in sensor networks, *in* 'Proceedings of the first ACM conference on Wireless network security', WiSec '08, ACM, New York, NY, USA, pp. 172–182.
**URL:** *http://doi.acm.org/10.1145/1352533.1352563*

Gupta, P., Rawat, P., Malik, S. & Gupta, S. (2012), 'Securing WMN Using Honeypot Technique', *International Journal on Computer Science and Engineering (IJCSE)* **4**(2), 235–238.

Hai, T. H., Huh, E.-N. & Jo, M. (2010), 'A lightweight intrusion detection framework for wireless sensor networks', *Wirel. Commun. Mob. Comput.* **10**(4), 559–572.
**URL:** *http://dx.doi.org/10.1002/wcm.v10:4*

Hamdi, M., Meddeb-Makhlouf, A. & Boudriga, N. (2008), 'Multilayer statistical intrusion detection in wireless networks', *EURASIP Journal on Advances in Signal Processing* **2009**(1).

Hassan, A. & Al Ali, M. (2011), Collecting malware from distributed honeypots âĂŤ honeypharm, *in* '2011 IEEE GCC Conference and Exhibition (GCC)', pp. 351–352.

Holz, T. & Raynal, F. (2005), Detecting honeypots and other suspicious environments, *in* 'Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop', pp. 29–36.

Honeynet Project (2002), *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*, Addison-Wesley.

Honeynet Project (2004), 'Honeynet definitions, requirements, and standards (ver 1.6.0)'.
**URL:** *http://old.honeynet.org/alliance/requirements.html*

Huang, Z. & Lu, T. (2012), A particle swarm optimization algorithm for hybrid wireless sensor networks coverage, *in* 'Electrical Electronics Engineering (EEESYM), 2012 IEEE Symposium on', pp. 630 –632.

IEEE - Institute of Electrical and Electronics Engineers (2005), *802.15.1-2005 - IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Specific Requirements. - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)*, IEEE (std.), IEEE.

IEEE - Institute of Electrical and Electronics Engineers (2006), 'Ieee 802.15.4-2006 ieee standard', Website.
**URL:** *http://standards.ieee.org/getieee802/802.15.html*

ISA (2011), 'INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS–: Wireless Systems for Industrial Automation: Process Control and Related Applications (Based on ISA 100.11a)'.

Islam, K., Shen, W. & Wang, X. (2012), Security and privacy considerations for wireless sensor networks in smart home environments, *in* 'Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on', pp. 626–633.

Islam, M. S., Khan, R. H. & Bappy, D. M. (2010), 'A hierarchical intrusion detection system in wireless sensor networks', *IJCSNS International Journal of Computer Science and Network Security* **10**(8), 21–26.

Iwendi, C. O. & Allen, A. R. (2011), Cia security management for wireless sensor network nodes, *in* M. Merabti & O. Abuelmaatti, eds, 'Proceedings of the 12th Annual PostGraduate Symposium on the convergence of Telecommunications, Networking and Broadcasting', Liverpool John Moores University, pp. 123–128.

Jenkins, I. R., Shapiro, R., Bratus, S., Goodspeed, T., Speers, R. & Dowd, D. (2014), Short paper: Speaking the local dialect: Exploiting differences between ieee 802.15.4 receivers with commodity radios for fingerprinting, targeted attacks, and wids evasion, *in* 'Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks', WiSec '14, ACM, New York, NY, USA, pp. 63–68.
**URL:** *http://doi.acm.org/10.1145/2627393.2627408*

Jenkins, I. R., Shapiro, R., Bratus, S., Speers, R. & Goodspeed, T. (2014), Fingerprinting IEEE 802.15.4 Devices with Commodity Radios, Technical Report TR2014-746, Dartmouth College, Computer Science, Hanover, NH.
**URL:** *http://www.cs.dartmouth.edu/reports/TR2014-746-rev2.pdf*

Jin-Shyan Lee, Y.-W. S. & Shen, C.-C. (2007), A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi, *in* 'IECON 2007 - 33rd Annual Conference of the IEEE Industrial Electronics Society', pp. 46–51.

Kaplantzis, S. (2006), 'Security Models for Wireless Sensor Networks', *PhD Conversion Report, Centre of Telecommunications and Information Engineering, Monash University, Australia* .
**URL:** *http://users.monash.edu.au/˜skap3/transfer-final-rev.pdf*

Kaplantzis, S., Shilton, A., Mani, N. & Sekercioglu, Y. A. (2007), Detecting selective forwarding attacks in wireless sensor networks using support vector machines, *in* '2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information', pp. 335–340.

Karlof, C. & Wagner, D. (2003), Secure routing in wireless sensor networks: attacks and countermeasures, *in* 'Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.', pp. 113–127.

Kim, I., Oh, D., Yoon, M. K., Yi, K. & Ro, W. W. (2013), 'A distributed signature detection method for detecting intrusions in sensor systems'.
**URL:** *http://doi.acm.org/10.3390/s130403998*

Klues, K., Hoffert, J. & Orjih, O. (2005), 'Configuring the IEEE 802.15.4 MAC Layer for Single-sink Wireless Sensor Network Applications', Website.
**URL:** *http://sing.stanford.edu/klueska/Black_Site/Publications_files/ kluesConfig802154.pdf*

Koch, A. (2008), 'Die rechtlichen rahmenbedingungen von hackback'.
**URL:** *http://www.irnik.de/publikationen/vortraege/2008-06-04_Vortrag_CCG-Seminar.pdf*

Kraußе, M. & Konrad, R. (2014), *Drahtlose ZigBee-Netzwerke: Ein Kompendium*, Springer Fachmedien Wiesbaden, Wiesbaden.
**URL:** *http://dx.doi.org/10.1007/978-3-658-05821-0*

Kreibich, C. & Crowcroft, J. (2004), 'Honeycomb: Creating intrusion detection signatures using honeypots', *SIGCOMM Comput. Commun. Rev.* **34**(1), 51–56.
**URL:** *http://doi.acm.org/10.1145/972374.972384*

Krontiris, I., Benenson, Z., Giannetsos, T., Freiling, F. C. & Dimitriou, T. (2009), Cooperative intrusion detection in wireless sensor networks, *in* 'Proceedings of the 6th European Conference on Wireless Sensor Networks', EWSN '09, Springer-Verlag, Berlin, Heidelberg, pp. 263–278.
**URL:** *http://dx.doi.org/10.1007/978-3-642-00224-3_17*

Lee, G., Lim, J., Kim, D.-k., Yang, S. & Yoon, M. (2008), An approach to mitigating sybil attack in wireless networks using zigbee, *in* '2008 10th International Conference on Advanced Communication Technology', Vol. 2, pp. 1005–1009.

Markert, J. & Massoth, M. (2013), Honeypot Framework for Wireless Sensor Networks, *in* 'Proceedings of International Conference on Advances in Mobile Computing & Multimedia', MoMM '13, ACM, New York, NY, USA, pp. 217:217–217:223.
**URL:** *http://doi.acm.org/10.1145/2536853.2536941*

Markert, J. & Massoth, M. (2014), Honeypot Effectiveness in Different Categories of Attacks on Wireless Sensor Networks, *in* 'Proceedings of Database and Expert Systems Applications (DEXA), 2014 25th International Workshop on', DEXA '14, IEEE.

Markert, J. & Massoth, M. (2015), Honeypot Wording and Definitions in Wireless Sensor Networks, *in* 'Conference Proceedings Paper-Sensors and Applications', ECSA 2015, Basel, Switzerland, p. 5.
**URL:** *http://sciforum.net/conference/ecsa-2/paper/3200/download/pdf*

Markert, J., Massoth, M., Fischer, K.-P., Furnell, S. & Bolan, C. (2011), Attack vectors to wireless zigbee network communications - analysis and countermeasures, *in* 'Proceedings of the Seventh Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2011), Furtwangen, Germany', pp. 57–66.

Masica, K. (2007), 'Recommended practices guide for securing zigbee wireless networks in process control system environments', Website.
**URL:** *http://csrp.inl.gov/Documents/*

Meier, M. (2007), *Intrusion Detection effektiv!: Modellierung und Analyse von Angriffsmustern*, 1 edn, Springer.
**URL:** *http://amazon.com/o/ASIN/3540482512/*

Min, W., Zhang, X., Ping, W., Kim, K. & Kim, Y. (2010), Research and Implementation of the Security Method Based on WIA-PA Standard, *in* '2010 International Conference on Electrical and Control Engineering (ICECE)', pp. 1580–1585.

Misic, V., Fang, J. & Misic, J. (2005), Mac layer security of 802.15.4-compliant networks, *in* 'Mobile Adhoc and Sensor Systems Conference, 2005. IEEE International Conference on', pp. 8 pp. –854.
**URL:** *http://asc.di.fct.unl.pt/tasd/archivemod3/Misic05.pdf*

Mitchell, R. & Chen, I.-R. (2014), 'A survey of intrusion detection in wireless network applications', *Computer Communications* **42**, 1–23.

Mokube, I. & Adams, M. (2007), Honeypots: Concepts, approaches, and challenges, *in* 'Proceedings of the 45th Annual Southeast Regional Conference', ACM-SE 45, ACM, New York, NY, USA, pp. 321–326.
**URL:** *http://doi.acm.org/10.1145/1233341.1233399*

Morgner, P., Mattejat, S., Benenson, Z., Müller, C. & Armknecht, F. (2017), Insecure to the touch: attacking zigbee 3.0 via touchlink commissioning., *in* G. Noubir, M. Conti & S. K. Kasera, eds, 'WISEC', ACM, pp. 230–240.

Mostarda, L. & Navarra, A. (2008), 'Distributed intrusion detection systems for enhancing security in mobile wireless sensor networks', *Int. J. Distrib. Sen. Netw.* **4**(2), 83–109.
**URL:** *http://dx.doi.org/10.1080/15501320802001119*

Muraleedharan, R. & Osadciw, L. A. (2009), An intrusion detection framework for Sensor Networks using Honeypot and Swarm Intelligence, *in* 'Proceedings of the 6th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services', IEEE.

Neil Peterson and Jeff Potter (2011), 'Strength-in-depth security for WirelessHART and Wi-Fi networks', *Industrial Ethernet Book Issue 66 / 40* .

NIST (2001), 'Federal information processing standards - advanced encryption standard (aes)', Website.
**URL:** *http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf*

O'Flynn, C. (2012), '15dot4-tools', Website.
**URL:** *http://sourceforge.net/projects/dot4-tools/*

Oosterhof, M. (2018), 'Cowrie ssh/telnet honeypot'.
**URL:** *https://github.com/micheloosterhof/cowrie*

Parbat, B., Dwivedi, A. K. & Vyas, O. P. (2010), 'Data Visualization Tools for WSNs: A Glimpse', *International Journal of Computer Applications* **2**(1), 14–20. Published By Foundation of Computer Science.

Philips Applied Technologies (2010), 'Zigbee health care reference design - rapid route to zigbee products', Website.
**URL:** *http://www.apptech.philips.com/zigbee-healthcare/*

Prathapani, A., Santhanam, L. & Agrawal, D. P. (2009), Intelligent honeypot agent for blackhole attack detection in wireless mesh networks, *in* 'IEEE 6th International Conference on Mobile Adhoc and Sensor Systems, MASS 2009, 12-15 October 2009, Macau (S.A.R.), China', pp. 753–758.
**URL:** *https://doi.org/10.1109/MOBHOC.2009.5336925*

Provos, N. & Holz, T. (2007), *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*, Pearson Education.

Qassrawi, M. T. & Hongli, Z. (2010), Deception Methodology in Virtual Honeypots, *in* 'Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on', Vol. 2, IEEE, pp. 462 –467.

Qiang, Z. & Yuqiang, S. (2013), The research of a cooperative model intrusion detection system, *in* W. Zhang, ed., 'Advanced Technology in Teaching', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 507–515.

Radcliffe, J. (2006), 'Cyberlaw: Honeypot Edition: Your guide to the legal issues and honeypots'.

Radcliffe, J. (2007), 'CyberLaw 101: A primer on US laws related to honeypot deployments'.
**URL:** *http://www.giac.org/paper/gsec/5423/cyberlaw-101-primer-laws-related-honeypot-deployments/110211*

Radmand, P., Domingo, M., Singh, J., Arnedo, J., Talevski, A., Petersen, S. & Carlsen, S. (2010), ZigBee/ZigBee PRO Security Assessment Based on Compromised Cryptographic Keys, *in* '2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing', IEEE, pp. 465–470.

Raj, J. S. S. & Thilagavathy, D. (2012), 'Security Threats And Jamming Attacks Of Multi Channel Wireless Sensor Networks', *International Journal of P2P Network Trends and Technology- Volume2Issue1- 2012* **2**(1), 27–31.

Ronen, E., Shamir, A., Weingarten, A. O. & O'Flynn, C. (2018), 'Iot goes nuclear: Creating a zigbee chain reaction', *IEEE Security Privacy* **16**(1), 54–62.

Roundy, S., Steingart, D., Frechette, L., Wright, P. & Rabaey, J. (2004), Power sources for wireless sensor networks, *in* H. Karl, A. Wolisz & A. Willig, eds, 'Wireless Sensor Networks: First EuropeanWorkshop, EWSN 2004, Berlin, Germany, January 19-21, 2004. Proceedings', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–17.
**URL:** *https://doi.org/10.1007/978-3-540-24606-0_1*

Rubin, B. & Cheung, D. (2006), 'Computer security education and research: Handle with care', *Security Privacy, IEEE* **4**(6), 56–59.

Salus Controls (2009), 'Salus programmable room thermostat ert 50 wireless zigbee technology', Website.
**URL:** *http://www.salus-controls.de/index_e.htm*

Sandee, M. (2015), 'Gameover zeus: Backgrounds on the badguys and the backends', *Rapport technique, Fox-IT, Delft, Pays-Bas* .

Schneider, F. (2011), Blueprint for a science of cybersecurity, Technical report, Cornell University. Also to appear in The Next Wave.
**URL:** *http://www.truststc.org/pubs/873.html*

Scottberg, B., Yurcik, W. & Doss, D. (2002), Internet honeypots: protection or entrapment?, *in* 'Technology and Society, 2002. (ISTAS'02). 2002 International Symposium on', pp. 387–391.

Sedjelmaci, H. & Feham, M. (2011), 'Novel Hybrid Intrusion Detection System For Clustered Wireless Sensor Network', *International Journal of Network Security & Its Applications* **3**(4), 1–14.

Shin, S., Kwon, T., Jo, G.-Y., Park, Y. & Rhy, H. (2010), 'An experimental study of hierarchical intrusion detection for wireless industrial sensor networks', *Industrial Informatics, IEEE Transactions on* **6**(4), 744 –757.

Siemens Industry Inc. (2007), 'Siemens building technologies introduces new wireless room temperature sensor with mesh network technology for real-world control applications', Website.
**URL:** *http://www.buildingtechnologies.siemens.com/bt/us/Press/press_release/2007/Pages/WirelessRoom.aspx*

Siles, R. (2007), 'Honeyspot: The wireless honeypot - monitoring the attacker's activities in wireless networks', Website.
**URL:** *http://honeynet.org.es/papers/honeyspot/HoneySpot_20071217.pdf*

Singh, S. & Verma, H. K. (2011), 'Security For Wireless Sensor Network', *International Journal on Computer Science and Engineering (IJCSE)* **6**(3), 2393–2399.

Skrzewski, M. (2012), Network malware activity – a view from honeypot systems, *in* A. Kwiecień, P. Gaj & P. Stera, eds, 'Computer Networks', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 198–206.

Smart Grid Australia (2011), 'Newcastle to be australia's first smart grid city', Website.
**URL:** *http://www.smartgridaustralia.com.au/SGA/News/August_2011/SGA/3_News/News_-_August_2011.aspx*

Sokol, P. (2015), 'Legal issues of honeynet's generations', *Proceedings of the 2014 6th International Conference on Electronics, Computers and Artificial Intelligence, ECAI 2014* pp. 63–69.

Sokol, P. & Andrejko, M. (2015), Deploying honeypots and honeynets: Issues of liability, *in* P. Gaj, A. Kwiecień & P. Stera, eds, 'Computer Networks', Springer International Publishing, Cham, pp. 92–101.

Sokol, P. & Host, J. (2016), Evolution of legal issues of honeynets, *in* E. Pricop & G. Stamatescu, eds, 'Recent Advances in Systems Safety and Security', Springer International Publishing, Cham, pp. 179–200.
**URL:** *https://doi.org/10.1007/978-3-319-32525-5_10*

Sokol, P., Husák, M. & Lipták, F. (2015), Deploying honeypots and honeynets: Issue of privacy, *in* '2015 10th International Conference on Availability, Reliability and Security', pp. 397–403.

Sokol, P., Míšek, J. & Husák, M. (2017), 'Honeypots and honeynets: issues of privacy', *EURASIP Journal on Information Security* **2017**(1), 4.
**URL:** *https://doi.org/10.1186/s13635-017-0057-4*

Song, J., Han, S., Mok, A., Chen, D., Lucas, M., Nixon, M. & Pratt, W. (2008), WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control,

*in* '2008 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)', pp. 377–386.

Sorensen, S. (2009), *The Sustainable Network: The Accidental Answer for a Troubled Planet*, Sustainable Living Series, O'Reilly Media.
**URL:** *http://books.google.ca/books?id=WGPZ6BLBpkMC*

Spitzner, L. (2003*a*), 'The honeynet project: Trapping the hackers', *IEEE Security & Privacy* **1**(2), 15–23.

Spitzner, L. (2003*b*), *Honeypots: Tracking Hackers*, Addison-Wesley.

Stetsko, A., Folkman, L. & Matyas, V. (2010), Neighbor-based intrusion detection for wireless sensor networks, *in* '2010 6th International Conference on Wireless and Mobile Communications', pp. 420–425.

Stoll, C. (1989), *The cuckoo's egg*, Doubleday.

Tamminen, U. (2009), 'Kippo ssh honeypot', Website.
**URL:** *https://code.google.com/p/kippo/*

Thomsen, J. B. (2010), 'Marama.wiresharkutility.qs.001.v0.10 quick start guide for wireshark zigbee sniffer utility', Website.

Unterschütz, S. & Turau, V. (2012), Fail-Safe Over-The-Air Programming and Error Recovery in Wireless Networks, *in* 'Workshop on Intelligent Solutions in Embedded Systems (WISES 2012)', pp. 27 – 32.

Wagener, G., State, R., Dulaunoy, A. & Engel, T. (2011), 'Heliza: talking dirty to the attackers', *Journal in Computer Virology* **7**(3), 221–232.

Wang, Y., Wang, X., Xie, B., Wang, D. & Agrawal, D. (2008), 'Intrusion detection in homogeneous and heterogeneous wireless sensor networks', *Mobile Computing, IEEE Transactions on* **7**(6), 698 –711.

Wei, M. & Kim, K. (2012), 'Intrusion detection scheme using traffic prediction for wireless industrial networks', *Journal of Communications and Networks* **14**(3), 310–318.

Wenda, D. & Ning, D. (2012), A honeypot detection method based on characteristic analysis and environment detection, *in* R. Chen, ed., '2011 International Conference in Electrics, Communication and Automatic Control Proceedings', Springer New York, New York, NY, pp. 201–206.
**URL:** *http://dx.doi.org/10.1007/978-1-4419-8849-2_26*

Wood, A. & Stankovic, J. (2002), 'Denial of service in sensor networks', *Computer* **35**(10), 54 – 62.

Wright, J. (2009), 'Killerbee: Practical zigbee exploitation framework', Website.
**URL:** *http://www.willhackforsushi.com/presentations/toorcon11-wright.pdf*

Wright, J. & Cache, J. (2015), *Hacking Exposed Wireless, Third Edition: Wireless Security Secrets & Solutions*, McGraw-Hill Education.

Yang, Y., Zhu, S. & Cao, G. (2008), Improving sensor network immunity under worm attacks: a software diversity approach, *in* 'Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing', MobiHoc '08, ACM, New York, NY, USA, pp. 149–158.
**URL:** *http://doi.acm.org/10.1145/1374618.1374640*

Yek, S. (2003), Measuring the effectiveness of deception in a wireless honeypot, *in* '1st Australian Computer, Network & Information Forensics Conference, 25 November 2003, Perth, Western Australia', Edith Cowan University.

Yek, S. (2004), Implementing network defence using deception in a wireless honeypot., *in* '2nd Australian Computer, Network & Information Forensics Conference, Fremantle, Western Australia', Edith Cowan University, pp. 4–15.

Yek, S. (2005), Blackhat fingerprinting of the wired and wireless honeynet., *in* C. Valli & A. Woodward, eds, '3rd Australian Computer, Network & Information Forensics

Conference, Perth, Western Australia', School of Computer and Information Science, Edith Cowan University, Western Australia, pp. 115–125.
**URL:** *http://dblp.uni-trier.de/db/conf/ausforensics/ausforensics2005.html#Yek05*

Yüksel, E., Nielson, H. R. & Nielson, F. (2008), Zigbee-2007 security essentials, *in* 'Proceedings of the 13th Nordic Workshop on Secure IT Systems : NordSec 2008', DTU Informatics, Building 321, Denmark, pp. 65–82. Presented at: The 13th Nordic Workshop on Secure IT Systems : NordSec 2008 : Kongens Lyngby, Denmark, 2008.

Yüksel, E., Nielson, H. R. & Nielson, F. (2011), 'A secure key establishment protocol for zigbee wireless sensor networks', *The Computer Journal* **54**(4), 589–601.
**URL:** *http://comjnl.oxfordjournals.org/content/54/4/589.abstract*

Zand, P., Chatterjea, S., Das, K. & Havinga, P. (2012), 'Wireless Industrial Monitoring and Control Networks: The Journey So Far and the Road Ahead', *JSAN (Journal of Sensor and Actuator Networks)* **1**(3), 123–152.

Zhang, Y., Lee, W. & Huang, Y.-A. (2003), 'Intrusion detection techniques for mobile wireless networks', *Wirel. Netw.* **9**(5), 545–556.
**URL:** *http://dx.doi.org/10.1023/A:1024600519144*

ZigBee Alliance (2006), 'First zigbee certified products now available', Website.
**URL:** *https://docs.zigbee.org/zigbee-docs/dcn/06/docs-06-4572-00-0mwg-first-zigbee-certified-products-now-available.pdf*

ZigBee Alliance (2008), 'Latest zigbee specification including the pro feature set', Website.
**URL:** *http://zigbee.org/Products/DownloadZigBeeTechnicalDocuments.aspx*

ZigBee Alliance (2012), 'Zigbee specification zigbee document 053474r20', Website.
**URL:** *http://www.zigbee.org/?wpdmdl=2168*

ZigBee Resource Guide (2017), 2017-2018 resource guide, Technical report, Webcom Communications Corp., Greenwood Village, CO.

**URL:** *http://www.nxtbook.com/nxtbooks/webcom/zigbee_rg2017/*

Zillner, T. (2015), 'Zigbee exploited - the good , the bad and the ugly'.

# Index

# Bound copies of published papers

FOR submission the papers during research were attached to this document.

# Attacks to ZigBee and Wireless Sensor Networks Honeypots for Detection and Response

J. Markert[1], M. Massoth[2], K.-P. Fischer-Hellmann[2], S. M. Furnell[1]

[1]Plymouth University, Plymouth, United Kingdom

[2]Hochschule Darmstadt – University of Applied Sciences, Darmstadt, Germany

e-mail : jurgen.markert@plymouth.ac.uk

## Abstract

ZigBee is a new communication technology and offers opportunities not only for developers, but also emerges as a possible risk containing flaws for an attacker to focus on. There is a trend in this field to develop methods for tracking intrusions and unauthorized access by an attacker, and for detecting attempted attacks against the stability of wireless sensor networks with intrusion detection systems. These systems represent the current state of the art in the field of research on detecting attacks on wireless sensor networks. We therefore propose implementing honeypots for ZigBee networks, which would be an alternative approach to intrusion detection systems, and which would prove to be ideal for the development of appropriate countermeasures. We feel that not enough research has been done so far in the development of ZigBee honeypots to offer opportunities for specific analysis and intercepting attacks.

## Keywords

Wireless Sensor Network, ZigBee, Intrusion Detection System, Honeypot

## 1. Introduction

The next industrial revolution will be the internet of things, machine to machine communication, “smart homes”, “smart buildings” or even “smart cities” (Sorensen 2010). The concept is deceptively simple: Everything can be connected to anything else. Anything provides services and can be monitored and controlled. This evolution is already laid out and well on its way. IPv6 and other technical prerequisites are already available to cope with the challenge of having more and more systems to be addressed in a single network.

Yet, these networks don't necessarily have to be wire-bound. A lot of networking technologies have their focus on wireless communication channels. Many companies have worked together in the development of Bluetooth as a wireless standard. ZigBee was likewise introduced and developed to be an ideal standard for wireless sensor and control networks (ZigBee Alliance 2008). ZigBee is an additional feature set to the IEEE 802.15.4 standard and defines additional layers on top of it (IEEE 2006).

IEEE 802.15.4 defines physical radio and a MAC layer, providing a simple packet data protocol for lightweight wireless networks. ZigBee adds the layers for logical network, security and application software. ZigBee determines the API, and the ZigBee Alliance certifies ZigBee-compatible devices guaranteeing their interoperability (Elahi 2010).

New ways of interaction become available through ZigBee, offering great advantages and a lot of potential to save energy and make life much more comfortable, but also requires that underlying structures be robust, reliable, safe, and secure. The key aspects of network technologies are commonly the same: application developers are expected to protect communications in order to attain the classic information security requirements: confidentiality, integrity, and availability.

## 2. Motivation

The analysis of the security and reliability of ZigBee networks has shown that these aspects have not yet been resolved satisfactorily, depending on the chosen implementation. But not all of the above requirements are present in the various configurations. The background to this is as easy as might be: the intended purposes of wireless sensor networks vary widely. One of them is low power consumption, running independently for the longest possible time, another one is for these devices to be as interoperable, compatible and easy to use as possible. This is in direct conflict of nearly every security feature, because security needs computational power and thus results in a higher energy consumption. These

features also limit their ease of use and render interoperability problematic: Encryption details have to be distributed and managed, requiring a bigger feature set, more network traffic.

The paper by Fabbricatore et al. (2011) describes these problems in the area of smart homes. The authors see several problems for existing home automation solutions involving ZigBee. They propose using additional cryptographic extensions on top of ZigBee, or to change the protocol, should ZigBee remain insecure for their purposes.

## 3. Attack categories

Common attacks on wireless sensor networks (WSN) as described in current publications and literature are part of a range of working and well-known attacks on ZigBee networks: Eavesdropping, replay attack, sinkhole attack, selective forwarding, network flooding, firmware modification and code injection (DePetrillo 2009), (Goodspeed 2007), (Gu & Noorani 2008), (Masica 2007), (Wright 2009), (Yang et al. 2008). These attacks can be subdivided into three categories:

### 3.1. Resource drain attacks

The first category shall be termed “resource drain attacks”. These attacks target the power of network devices as shown in figure 1 (replay-attack). In a battery-driven system, the main goal of an attacker is to exploit the system's power consumption against the system itself. An attacker might render wireless sensor network unusable by making parts of it run out of battery power through the modification of networking schematics, with the propagation of new routes, and by incessantly sending and requesting data. Batteries are a limited resource, network capacity is likewise limited. An increasing network load limits cripples the already limited throughput of the network.

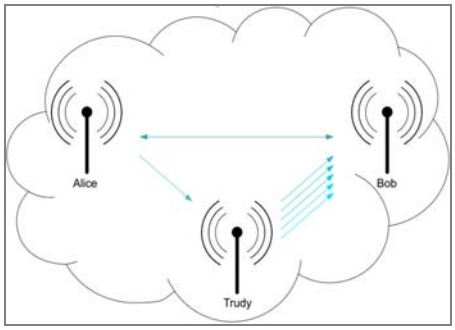



**Figure 1: Power drain through replay attack**

The authors Raj and Thilagavithy (2012) propose a solution to a specific type of network load caused by so called jamming attacks in wireless sensor networks. Due to the amount of commonly available radio channels in these networks, shifting to another network channel is suggested each time a jamming attack is detected. The authors claim that a residual network activity should keep the channel busy for a limited time during the short interval in which the jamming is not taking place as to make the jamming attacker believe that the jamming attack is successfully disturbing the WSN communication. This behaviour could also be described as deception, or, simply put, a honeypot or honeynet. This could even also attract an attacker and act as a decoy for the attacker.

### 3.2. Network attacks

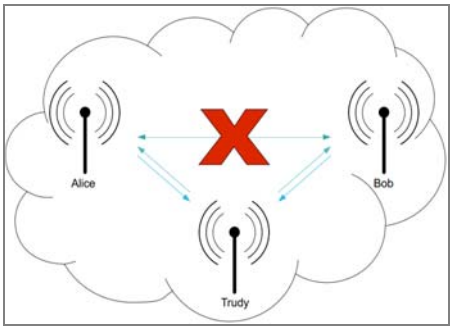



**Figure 2: Network redirection for man in the middle scenario**

The second category is called “network attacks”, describing the attacker's goal of modifying the network's routing structure. By this, the attacker aims to gain valuable information through redirection. He becomes therefore a “man in the middle” listening to any traffic.

### 3.3. Hardware-specific attacks

The third category is called “hardware-specific attacks”, standing for the goal to bring the networking nodes under complete control of the attacker (e.g. firmware modification as shown in fig. 3). This could be achieved with physical attacks like attaching wires in order to extract encryption keys from a network node. Attacks on hardware can also be an attack on firmware, resulting to firmware-modification. This is possible through security flaws in software-components of the nodes.

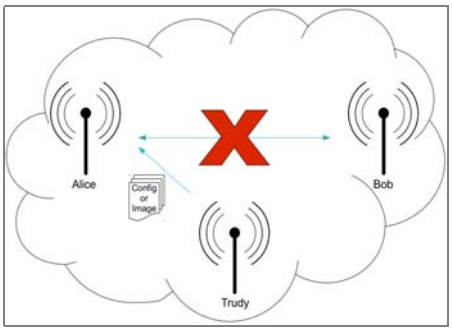



**Figure 3: Firmware-modification attack**

All of the three attack types mentioned above already exist. There is even a “proof of concept” worm that exploits network node communications and propagates itself over the network.

Goodspeed et al. (2012) have presented new results in the area of WSN hacking. They improved the killerbee API and wrote extensions to execute “wardriving” scenarios on ZigBee networks. They extended existing tools until all 16 ZigBEE channels could be monitored in parallel. They then combined this with a GPS signal logger for “wardriving” around the campus of Dartmouth. They also wrote semi-automatic tools to attack the networks with various different attack types. This development should make abundantly clear, that the security threats for ZigBee are real and serious and it is the conclusive proof that solutions to detect the attacks is are of crucial necessity.

## 4. Countermeasures by design

Several security features are implemented in ZigBee by design. “ZigBee PRO 2007” defines security mechanisms for authentication, encryption and trust centre functions (Elahi 2010).

Research on these features has however revealed some shortcomings. Security features might be circumvented or their imperfection could be used against the system. Successful attacks might last as long as the attacker wants. These can even not be analysed, because usually, there is no log of network activities and hence, attackers will leave no traces. There is no “packet capture” of attacks, and the probed exploit code is not available in the wake of an intrusion attempt. An attacker will not get noticed and therefore, will remain unidentified. It is hardly possible to gather the leftover information for forensic analysis.

There are clearly still problems in the process of improving the security features of ZigBee networks. Having blind faith and trust in the existing countermeasures is greatly unjustified. Radmand et al. (2010) showed in their comprehensive paper, that the cryptography used in ZigBee is not encompassing enough to cover all possible attacks. Some of the problems like confidentiality are solved encrypting the data stream. Yet, problems on the network layer such as replay attacks still remain. They also state, that the manufacturers should provide a minimum of security, due to problems with encryption keys stored in plain-text in the nodes. These could be extracted by an attacker tampering with physically captured ZigBee network nodes. There is hence a need for an intrusion detection system, and also for system reporting break in attempts.

## 5. Intrusion Detection System

Intrusion attempts are reported in classical network security solutions by so-called intrusion detection systems (IDS). They use only passive inspection to monitor all network activities for violation of regular use, and have no other function in the net. Intrusion detection systems rely on predefined policies describing normal network activity and normal behaviour. An intrusion detection system is generally a single point in the network, but it might additionally also be distributed on several hosts of the network.

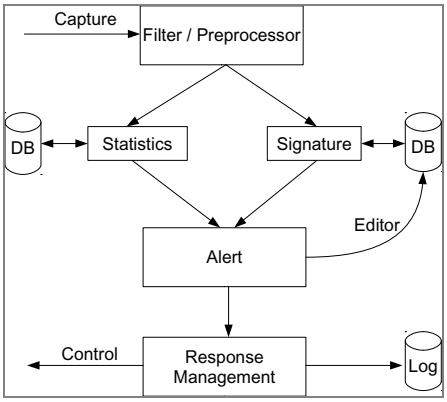



**Figure 4: Intrusion detection system (IDS) working scheme**

It might also stop intrusion or filter or modify malicious traffic as an active part of the network and is in this case most often called an intrusion prevention system (IPS). In an IPS, the controlling aspect of figure 4 is much more elaborate than in an IDS. All of these setups provide logging and reporting. In an IDS, every data packet has to be regarded and analysed. Its purpose is to spot the packets of an attack within a large amount of regular and normal network traffic. Processing every packet, its inspection, and the detection of attacks by matching patterns all require a lot of resources. In the context of ZigBee networks, and therefore battery driven network nodes with very limited resources, this is a clear drawback of an intrusion detection system approach in wireless sensor networks.

Kaplantzis et al. (2007) did a network simulation and had success in detecting selective forwarding attacks in wireless sensor networks by support vector machines (SVM), too. They also concluded that these attacks are the most difficult to accurately detect thus verifying assumptions made in previous research.

Sedjelmaci and Feham (2011) presented a novel IDS for the use in clustered WSNs. Using SVM, just like Kaplantzis et al. (2007) did, the essential point is to teach the detector nodes about normal network behaviour, such that they achieve a detection ratio of over 98% regarding abnormal network traffic.

Iwendi and Allen (2011) wrote a comprehensive paper on attacks on WSNs and demonstrated a way to simulate attacks with the intention of developing appropriate countermeasures. They propose in their conclusion the development of a security protocol for defensible WSN.

There is a definite need for security in wireless sensor network communications, and possible improvements should involve intrusion detection systems. The emerging outline can be briefly described as a sensor working on different network layers. It purpose is to detect anomalies in the net and to report these attacks. This is supposed to provide a new technical instrument for reacting to this new attack scheme. This feature set forms the common core of an intrusion detection and prevention system, also called a honeypot.

## 6. Honeypots

A honeypot can be described with a single phrase: It is a trap. It is in general an active system in a network monitoring every connection to this part of the network. The results are written to log files and reported accordingly. The honeypot looks like a normal node of the network to every system passing by. To an attacker, however, it will appear as a valuable target.

The advantages of a honeypot might be summarized as follows: instead of inspecting and analysing every packet of the network like an IDS, and then deciding whether this has been an attack or not a honeypot simply regards every connection as a likely attack. This behaviour saves a lot of resources because not every packet needs to be processed,

inspected and filtered for attacks. Even in the context of ZigBee networks and therefore battery driven networks, the honeypot system could make do with very limited resources. One of the drawbacks of a honeypot system approach in wireless sensor networks could be that it exposes additional targets which might get corrupted and then be taken over by an intruder. But since any other part of the net could equally get compromised, this is a small drawback. The advantages of a honeypot far outweigh its disadvantages.

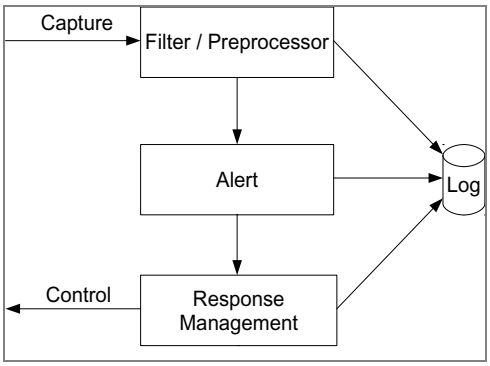



**Figure 5: Honeypot working scheme**

Prathapani et al. (2009) propose a honeypot system for the wireless mesh network standard 802.11s, then still in draft status. They set up the network simulator ns-2 and showed in their environment a very significant and successful result in network throughput with a high detection and low false positive ratio during a blackhole attack. Although these results were not proven to be correct in a real network, we assume that a similar result might be achieved in wireless sensor networks based on ZigBee standard communication.

Gupta et al. (2012) propose a honeypot technique for wireless mesh networks. A group of several honeypots are called a honeynet. The authors differentiate between different types of honeypots such as honeypots for research and honeypots for the productive environment. In detail they propose using a concept called honeyPHARM described by Hassan and Al Ali (2011). This solution has been specifically invented for the 802.11s draft. The authors Hassan and Al Ali (2011) propose a honeyPHARM, a distributed honeynet for the collection of malware in networks. Collected data will be used to discover new attack methods. This will be an addition to the nephentesPHARM system.

The authors Muraleedharan and Osadciw (2009) propose a framework using a honeypot technique combined with swarm intelligence for a battlefield monitoring application in a wireless sensor network. They state that traditional security schemes can not be applied in the field of WSN due to resource constraints. They had planned to build a simulation to prove this proposal, and validate the results.

The summary of all of these ideas leads to a proposed solution development of a honeypot for ZigBee networks without the drawbacks of a classical IDS. The concept of the honeypot will be presented below.

Several technical details are worthy of discussion. One of the proposed features of a honeypot should be that it might be temporarily part of the net at various different times. It should not always neighbor the same node in the network, because these nodes in the net might not be the target of the attacker.

One of the proposed features could also involve cloning. Those resources of special appeal to the intruder might duplicate themselves virtually during the course of an attack, expanding the target range and thus reducing the likelihood of an attack on the real network.

The system additionally features a feedback channel for reporting attacks. This can be probably done via a covert channel or other medium via a gateway. This might be done on other frequencies or channels like e.g. GSM, 5GHz WiFi, 868 MHz ZigBee, LAN and others.

A honeypot system should be a hardened system. Singh and Verma (2011) list a number of possibilities for hardening the WSN in the paper “Security For Wireless Sensor Network”. They also show the list of unresolved problems, and state that these problems are difficult to solve.

A honeypot must not be recognizable as such to an attacker. An attacked honeypot should therefore react in the same way as any other attacked part of the wireless sensor network. This leads to the assumption, that its hardware capabilities should be the same. For safety and reliability reasons, it could be equipped with a larger battery or a wired power supply.

In their paper, Mostara and Navarra (2008) suggest assigning roles in wireless sensor networks with specific features for honeypots. They propose progressively changing the roles of the honeypot nodes over the lifetime of the network. This feature should be part of the WSN honeypot as well.

Regular clients are forbidden from using the honeypots for networking purposes, since connections to these honeypots would be regarded as an attack. Since the roles of nodes in this network should change over time between honeypot nodes and regular nodes, a predefined schedule will be used to guarantee a low false positive intrusion detection rate. This method renders the network harder to attack.

Detecting new attack methods is definitely possible with honeypots. The rest of the network will be left in a functional state while the honeypots are under attack. As an additional benefit, new countermeasures can be prepared while the attack is still ongoing. All these presented uses of honeypots are suitable to increase the reliability and availability of ZigBee networks.

Moreover, the recording unknown attacks will undoubtedly lead to the development of new countermeasures and hardening. The proposed honeypot concept for ZigBee networks combines new detection mechanisms offering new countermeasures for the stability and reliability of the network.

The installation and maintenance of a single honeypot or of a distributed honeypot network clearly requires additional effort. Yet, this effort should be made on top of common improvements to the wireless sensor network like configuration hardening and regular firmware updates. Some implementations offering over-the-air programming features for updating nodes already exist.

The detection of an attack by a honeypot and the insight in the behaviour of an intruder can be used to devise appropriate defence mechanisms or to generate updates for the complete network. Considering the publications over the last year discussing new attack schemes, there is an expected growing market for researchers to do research on security flaws simply due to the also growing opportunities for intruders to mess with the ZigBee networks (Cache et al. 2010). Honeypots will help in detecting and resisting these unavoidably upcoming attacks. It should be understood, though, that the security of a ZigBee network is not static, but rather an ongoing process to be kept up during its whole lifetime.

## 7. Conclusions

ZigBee offers some security features out of the box. Its weaknesses, however, have been explored only in part to the present day. It appears that only few people have been doing research in this area at all, and no one has ever applied honeypot methods in a ZigBee network. Our intended research will explore previously unconsidered and unregarded aspects of security threats against ZigBee networks and develop new defence mechanisms with the use of honeypots.

## 8. References

Cache, J., Wright, J., Liu, V. (2010), “Hacking Exposed – Wireless”, MC Graw Hill.

DePetrillo, N. (2009), “Power Hungry People - Making Sense of New Critical Infrastructure Threats”, http://proidea.maszyna.pl/CONFidence09/2/CONFidence2009_nick_de_petrillo.pdf.

Elahi, A., Gschwender, A. (2010), “ZigBee Wireless Sensor and Control Network”, Prentice Hall.

Fabbricatore, C., Zucker, M., Ziganki, S. & Karduck, A. P. (2011), “Towards an Unified Architecture for Smart Home and Ambient Assisted Living Solutions”.

Goodspeed, T. (2007), “MSP430 buffer overflow exploit for wireless sensor nodes”, http://travisgoodspeed.blogspot.com

Goodspeed, T., Bratus, S., Melgares, R., Speers, R. & Smith, S. W. (2012), “Api-do: Tools for Exploring the Wireless Attack Surface in Smart Meters.”, 2012 45th Hawaii International Conference on System Sciences., IEEE, pp. 2133–2140.

Gu, Q., Noorani, R. (2008), “Towards self-propagate mal-packets in sensor networks”, WiSec '08: Proceedings of the first ACM conference on Wireless network security, ACM

Gupta, P., Rawat, P., Malik, S. & Gupta, S. (2012), “Securing WMN Using Honeypot Technique”, (4), 235–238.

Hassan, A. & Al Ali, M. (2011), “Collecting Malware From Distributed Honeypots - HoneyPHARM”.

IEEE, Institute of Electrical and Electronics Engineers - IEEE 802.15.4-2006 IEEE Standard (2006), http://standards.ieee.org/getieee802/802.15.html

Iwendi, C. O. & Allen, A. R. (2011), “CIA Security Management for Wireless Sensor Network Nodes”.

Kaplantzis, S., Shilton, A., Mani, N. & Sekercioglu, Y. A. (2007), “Detecting Selective Forwarding Attacks in Wireless Sensor Networks Using Support Vector Machines.”

Masica, K. (2007), “Recommended Practices Guide For Securing ZigBee Wireless Networks in Process Control System Environments”, http://csrp.inl.gov/Documents

Mostarda, L., Navarra, A. (2008), “Distributed Intrusion Detection Systems for Enhancing Security in Mobile Wireless Sensor Networks”, International Journal of Distributed Sensor Networks, Vol. 4, 2008.

Muraleedharan, R. & Osadciw, L. A. (2009), “An intrusion detection framework for Sensor Networks using Honeypot and Swarm Intelligence.”, Proceedings of the 6th Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. IEEE.

Prathapani, A., Santhanam, L. & Agrawal, D. P. (2009) “Intelligent Honeypot Agent for Blackhole Attack Detection in Wireless Mesh Networks”.

Radmand, P., Domingo, M. & Singh, J. et al. (2010), “ZigBee/ZigBee PRO Security Assessment Based on Compromised Cryptographic Keys.”, 2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing., IEEE, pp. 465–470.

Raj, J. S. S. & Thilagavathy D. (2012), “Security Threats And Jamming Attacks Of Multi Channel Wireless Sensor Networks.”, International Journal of P2P Network Trends and Technology- Volume 2 Issue 1- 2012 (2), 27–31.

Sedjelmaci, H. & Feham, M. (2011), “Novel Hybrid Intrusion Detection System For Clustered Wireless Sensor Network.”, International Journal of Network Security & Its Applications 3 (4), 1–14.

Singh, S. & Verma, H. K. (2011), “Security For Wireless Sensor Network.”, International Journal of Network Security & Its Applications (3), 2393–2399.

Sorensen, S. (2010), “The Sustainable Network”, O'Reilly

Yang, Y., Zhu, S., Cao, G. (2008), “Improving sensor network immunity under worm attacks: A software diversity approach”, MobiHoc '08: Proceedings of the 9th ACM International Symposium on Mobile Ad Hoc Networking and Computing.

Yüksel, E., Nielson, H. R., Nielson, F. (2010), “A Secure Key Establishment Protocol for ZigBeeWireless Sensor Networks”, Oxford University Press.

ZigBee Alliance (2008), “Latest ZigBee Specification Including the Pro Feature Set”, http://zigbee.org/Products/DownloadZigBeeTechnicalDocuments.aspx, Std. 053 474r17

# Honeypot Framework for Wireless Sensor Networks

Jürgen Markert
School of Computing and Mathematics
Plymouth University
UK, Plymouth
jurgen.markert@plymouth.ac.uk

Michael Massoth
Fachbereich Informatik
Hochschule Darmstadt
Germany, Darmstadt
michael.massoth@h-da.de

## ABSTRACT

In the field of wireless sensor networks there is a course to develop methods for tracking intrusions and unauthorised access by an attacker as well as detecting attempted attacks against the stability of wireless sensor networks, by using intrusion detection systems. So far, very little research has been done in the development of honeypots.Such systems represent the current state of the art in the field of research on detecting attacks on wireless sensor networks. It is considered that insufficient research has been conducted in the development of wireless sensor network honeypots, thus offering not enough opportunities for specific analysis and intercepting attacks.

## 1. INTRODUCTION

New communication technologies offer new opportunities for developers. Yet, they also pose new risks via potential attacks by an intruder. The purpose of this paper is to present a solution to this current state in the security area of wireless sensor networks (WSN). The planned improvements resulting from this research will be illustrated. Wireless sensor networks are a new communication technology and offer opportunities not only for developers, but also emerge as possible risk containing flaws for an attacker to focus on. The major findings from research in recent literature on the subject regard the improvement of the security features of these networks, based upon formal specification, simulation and tests. Another trend pursues the development of methods to detect intrusions or intrusion attempts conducted by an attacker against the wireless sensor network. The honeypot for WSN is useful for security purposes in any smart building. This could be a smart home or a smart technology complex of a company.

## 2. MOTIVATION

In the end, an intrusion detection system (IDS) always needs to be trained in any way with examples. In the past it was like this: producers of commercial IDS systems have been asking for, and purchasing insights about, new security weaknesses so that their customers were protected as soon as possible from the threats. Miller [18] is a security researcher and describes his expertise with the security market. He found vulnerabilities and wrote corresponding exploits and summarises his experiences while selling a 0-day exploit. While manufacturers with a "responsibly disclosed" security hole were still in the process to fix vulnerabilities, IDS / IPS have already been able to defend against a corresponding attack.
So a honeypot enables a learning effect for network security, and can help to identify new attack patterns and strategies. It can help to explore and discover attacks which lead to privilege escalations like most exploits do. This would be the obvious way to train an IDS. If a new bug is found, if a vulnerability is found, that an IDS does not know, then the IDS security is overridden. Whatever the case, the IDS / IPS must always be supplied with actual and accurate information. The attack recognition must come from somewhere. And this is where honeypots offer a playground in which attackers can, among other things, try out their new attacks, providing training information for an IDS/IPS.
Another author has also spoken of a different experience: what's much present gets much attacked [28]. This means specifically that the platforms that are found most often are going to be studied very hard.

It should be noted that there are several new attacks and ideas by developers regarding security flaws in the WSN field. The KillerBee API [27] is a simple but powerful tool, and it is easy to build add-ons on top of it. This API and the tools for it provide a basis for further development. Results may be given back to the community.
15dot4 is a tool to listen to ZigBee and 802.15.4 communication [19], it provides an input device for the well known network monitoring tool Wireshark [4] to monitor and capture the network traffic in realtime. Both tools feature usage with state of the art development kits, for example the AVR Raven [2] and they might be ported to other platforms as well.
Free projects like the Freakduino [6] make it easy to build WSN components based on the cheap Arduino development kits. These kits have powerful tools available, bringing WSN technology to everybody interested in working with it and doing research on the technology.
With hardware-specific attacks the goal is to bring the networking nodes under complete control of the attacker (e.g. firmware modification or code injection). This could be achieved with physical attacks like attaching wires in order


*MoMM2013, 2-4 December, 2013, Vienna, Austria.*

to extract encryption keys from a network node. Attacks on hardware can also be an attack on firmware, resulting in firmware modification. This is possible through security flaws in software-components of the nodes.
There is even a "proof-of-concept" worm that exploits network node communications and propagates itself over the network [9], also in smart meters [5].
New results in the area of WSN hacking were recently presented. Improvements to the KillerBee API with extensions to enable "war driving" scenarios on wireless sensor networks were developed [10]. They extended existing tools until all 16 ZigBee channels could be monitored in parallel. The authors then combined this with a GPS signal logger for "war driving" around the campus of Dartmouth. They also wrote semi-automatic tools to attack the networks with various different attack types.
All of the attack types mentioned above already exist. This development should make abundantly clear, that the security threats for WSN are real and serious and this is conclusive proof that solutions to detect attacks are of crucial necessity.

# 3. RELATED WORK

Our wireless sensor network honeypot project started with research into the state of the art in wireless sensor networks and current security threats to them. The focus was on information security requirements in wireless networks. Essential recent improvements in WSN security were compiled and tasks remaining identified.
The potential for IDS with special regard to honeypot architectures was presented in a conference paper in 2011, titled "Attack Vectors to Wireless ZigBee Network Communications - Analysis and Countermeasures" [16], describing the project results by then. With an initially held assumption of a lack in security features regarding intrusion detection for WSN and ZigBee, research on featured or proposed intrusion detection mechanisms was done. This resulted in another paper, written in 2012, titled "Attacks to ZigBee and Wireless Sensor Networks - Honeypots for Detection and Response" [15]. The latest steps were taken by generating an overall design for a honeypot in wireless sensor networks, especially with regard to the development of the honeypot system design. Definition of detection mechanisms, algorithms and a corresponding reporting system is also put into this paper, containing early stubs of system components. As a result this paper was completed to combine the recent project work.

## 3.1 Improvements

The authors Pazynyuk et al. [20] propose a QoS scheme to secure wireless sensor networks. Their evaluation experiments revealed significant improvements. Masica [17] gives recommendations for WSN using ZigBee. These can be seen as a transfer of best practices from wireless networks to the area of WSN. In the article by Khan et al. [13], security in wireless sensor networks is discussed on pages 43-49, regarding attacks and defenses. This paper is very comprehensive and gives a good summary of the topic. The authors propose more research on QoS and Cryptography in WSN.

## 3.2 Jamming

Wang et al. [25] propose a mathematical model to improve the survivability of wireless mesh networks under intelligent attacks. They focus on countermeasures in a jammed wireless network and did a simulation on their proposal for evaluation.

## 3.3 Hardening

The authors Lin et al. [14] describe in this paper the possibilities to increase a wireless sensor network's robustness to sustained malicious and jamming attacks.
Gill [8] describes the options to increase a WSN's security, especially when used in the field of home automation.

## 3.4 Mobile phone honeypot

The authors of [7] describe the Smartpot, their mobile smart-phone honeypot. Their research is based on emulating a mobile phone for an attacker scanning a network, but offering no real service. The scanning target itself and pretending to be a mobile phone in an IP network is the honeypot functionality. In another paper the authors describe a honeypot in mobile networks which is using OpenBTS[24].

## 3.5 Deception

The author Gopinath writes in her thesis about deception success in wireless sensor networks [11]. She clarifies that traditional honeypots cannot be adopted to wireless sensor networks. "But these theories developed do not translate to sensor networks, due to differences between sensor networks and conventional systems."

## 3.6 IDS and Honeypot

In the book by Chaouchi and Laurent-Maknavicius [3] the chapter 15 deals with WSN. After the introduction, a case study is conducted, and the authors build an exemplary distributed IDS. The authors Prathapani et al. [21] propose a novel detection strategy to detect black hole attacks in wireless mesh networks. Their work regards this single problem and a honeypot solution for it in IP based networks. Therefore, their honeypot for wireless mesh networks is a valuable source of information, but can not be adopted as is into the domain of wireless sensor networks. The paper by Shamsh [22] describes honeypots in Mobile Ad-hoc NETworks (MANET) and an election algorithm for the honeypot nodes in the network. Siles [23] defines a HoneySpot as a Honeypot Hotspot for IEEE 802.11 networks. By the term Honeytoken the authors White et al. [26] generally refer to specifically prepared data copied in violation of rules, laws or regulations. This can for example be an additional entry in a database or superfluous extra communications. In this paper he describes a system with a set of Honeytokens transmitted as an additional communication "noise" that can in-house be automatically filtered out, but pose a problem for an external attacker.

# 4. SUMMARY OF WSN HONEYPOT TECHNICAL DETAILS

The areas of the research work in its scope is the research in the area of network security of wireless sensor networks and their fields of application. It has the border to conduct research into defensive security instead of development of new attacks. This shall be done without simulation, but in a prototype. The focus is precise, the research here should be centred on novel intrusion detection mechanisms, different from known solutions in the field. Conceptual, there must

also be a solution for lightweight intrusion detection systems in the area of devices with only little resources. The assumption of this paper is, that this solution can be found in the area of honeypots. The goal is to show or falsify a novel intrusion detection system for tracking hackers in wireless sensor networks. At the end its about data, to show results, numbers are needed. The details are discussed further on in this paper.

### 4.1 IDS and honeypot

The authors Kachirski and Guha [12] present an effective intrusion detection with multiple sensors in Wireless Ad-Hoc networks. This work is an early research in this domain and can in parts be taken into account for the area of WSN. In contrast to the intrusion detection system attempts shown, the use of the honeypot approach is much more comprehensive.

## 5. HONEYPOT PROTOTYPE

In the previous research stage, emphasis was placed on the need for enhanced security in wireless sensor networks, combined with the demand for some kind of ongoing monitoring, looking for unusual events in the traffic seen. To determine the urgency of the approach suggested, a literature research was conducted. From the results, required measures were determined [16] and a perspective shown as to the integration of the proposed honeypot as a security base for wireless sensor networks [15].
Improving security methods to detect misuse of a network can be done in two ways, using an intrusion detection system, or alternatively a honeypot. This despite the fact that there are theoretical intrusion detection mechanisms able to detect a distinct intrusion scheme with the use of pattern recognition in an invariable network structure.
The requirement for the network nodes in an IDS approach is high on resources in memory, computational power and energy consumption, lowering the overall performance of the WSN. With this in mind, it is important in the design of a security enhancement to regard the performance and resource needs. The detection of misuse is possible solely with ongoing monitoring of the network. To reach this goal, the system therefore has to make available a detection mechanism, combined with methods for reporting and notification. For the full function honeypot a complex implementation scenario can be foreseen, with a widely distributed honeynet, collecting information from lots of member honeypots. An early prototype for this honeynet will be rather reduced in the number of available interacting networking nodes in hardware, and tested in a laboratory environment first.
To be more specific, if misuse of a network is detected by a honeypot node, then an alarm event will be triggered and the current notification routines are activated, which may be local or signaling to a remote user, as well as forwarded to a central alert system. Indeed, showing an alert is one of the aspects, which will be a focus in the ongoing research. In the end, the efficacy of this honeynet will be quantified in testing and evaluating the prototype implementation.
With respect to the research draft from the beginning of these studies, the proposal is to implement and test the honeypot principle in a WSN with a reporting system, as recommended in that draft paper.

## 6. SPECIFICATION OF THE DISTRIBUTED HONEYPOT ARCHITECTURE

The specification of this architecture is discussed in this paper and will rely on the results named there. The goal is to specify complete details for implementing the system in all of its parts, in the network as well as in the users' node. The next section focuses on concepts and looks at the details of the technical prerequisites; the full functional specification will give an explicit overview of the system design as a whole.
To provide details for the next step, drafts of the graphical user interface and other connections will be determined and drawn. This is going to be done for all of the central elements of the system: the detector, the reporter, the combiner and the signaller. It is expected, that formal development of these central elements will depend (or rely) on these details.

## 7. HONEYPOT PARTS DEVELOPING

Prototype parts are created to prove the honeypot efficacy, according to the technical and connector specifications. The whole system is developed for the current state of the art Atmel microcontroller hardware and a graphical user interface using either LabVIEW 8 or 2012 for Windows XP/7 or an Android app for a TazTag TazPad tablet or a TazTag TphONE smartphone. TazTag products for the mobile market are the first to have a builtin ZigBee network adapter component. AVR Studio 6 is used for the microcontroller program code development and Eclipse with SDK is used to provide Java apps/programs for the mobile world.
All of this development will be at the forefront of current hardware hacks; the development of a honeypot for wireless sensor networks is new, as well as the use of TazTag technology for the user side elements.

### 7.1 Detector / Sensor

The detection part is developed within an Atmel microcontroller environment. When the honeypot nodes are up and running and the individual nodes are reporting to the central component of the network, then a basic honeypot functionality is achieved.
This detection will determine any misuse of the detector as a part of the honeypot. This will be done according to different levels of interaction.
The misuse detected will be differentiated and reacted upon correspondingly. The simplest case would be for instance a probing of a honeypot device, which can also be caused by an automatic service on another device. High interaction with the honeypot device, though, would be medium on the scale. The highest level of misuse is the deploying of new firmware or other code, which behaviour should always be regarded as a direct violation of all security rules. The honeypot detector as described in the previous sections is responsible for a permanent and ongoing security detection, more precisely a misuse detection and categorisation, so that the security can be kept on a high level.
With an ongoing security enforcement, a foundation is laid for the benefit of wireless sensor networks. The honeypot detector has a reporting scheme that could also easily be adopted for other research groups' projects, so additions to this work are kept open for collaboration and an exchange of research outcome is possible with these research groups.

## 7.2 Reporter

The Reporter implementation is a set of functions to report the event that was triggered by the misuse of a honeypot system, and will be reported to the combiner to forward it for signalling. Forwarding the information shall take place in a secure manner, and end to end encryption of the report is planned to work already in the prototype state of the reporter. The reporter can be part of the honeypot host, but can be also seen as separated due to the fact that the reporting network traffic could take place on another medium like GSM networking.

## 7.3 Combiner / Collector

The Combiner or Collector is the sink of the honeypot network, as it were; the place where all the detected anomalies from the honeypot sensors are collected and will be forwarded for signaling. This being a very important part of the network, it is clear that it has to be secured in a hardened and very fail-proof manner. The Collector shall also have more than enough storage to buffer accumulated data until it can be forwarded to the signaller; when there is no direct connection currently possible, the collector's buffer for security incidents should not overflow.

## 7.4 Signalling and managing

The signaller is planned to be implemented as a prototype, and its purpose is to show the honeypot status to an operator. This implementation is going to be developed with the Taztag SDK for a tablet PC running Android. An evolution to completion can be handed over to project groups for further development. The fundamentals of the signaller are signalling alerts complete with misuse categorisation and the overview of the total honeypot network status. Later development could implement authenticated update mechanisms for the devices and further analysis options for the reported alerts.

# 8. DESIGN OF THE HONEYPOT PROTOTYPE

With the basic design of the sketch of the honeypot the detailed design was created.

## 8.1 Client for Reporting

As the previous section described the layers of components to be worked on, this part describes the honeypot client interface. There has to be a destination for the alerts a honeypot generates, and these alerts should also be reacted on and used to develop appropriate countermeasures. Therefore there will be a host based client component with a graphical user interface to display those alerts and reports. This client's hardware will be a standard laptop, connected to the honeypots via a gateway.

## 8.2 Gateway

The client computer has to be connected with the monitored network. In the development phase this should not be cost-intensive, so it is clear that this will be done via a direct connection, for instance by serial communication. All the transfered data could of course also be provided for example via GSM network, SMS based texting or email communication via the Internet. Due to the increased cost this will not be done in the research phase. The capabilities of the MeshBean2 boards make these an ideal network component for this task, as well as an ideal gateway due to their possible USB connection for serial communications.

## 8.3 Client for Reporting including Gateway - Tablet PC with WSN

The opportunity to use the TazTag TazPad arose in the course of the year 2012. The ZigBee Alliance have already spoken in 2006 in the presentation slides about a mobile phone with ZigBee. It would be a wonderful thing to be able to control all ZigBee devices from an everyday object. In the past, buying a mobile phone or tablet with these properties was impossible; as a workaround there was the reason to use other techniques to get data from the ZigBee network. Now, in 2012, TazTag as manufacturer is offering a smartphone and a Tablet for developers that includes ZigBee. Their development platform consists of a smartphone or tablet with Android operating system. Android is a relatively new OS and provides an open source platform. A Linux kernel is the basis, all functions of the phone are implemented as applications that run in a closed system, a so-called sandbox.

## 8.4 Tablet overview

The TazTag TazPad is a Tablet PC with new features in the mobile market. It is equipped with a fingerprint reader for user authentication and is capable of using NFC, ZigBee, Bluetooth, WiFi and GSM wireless communication. It can be extended with a LAN adapter for stationary use and has a HDMI video output. This feature set is a perfect combination for being the mobile graphical user interface client for the WSN honeypot. The Tablet is running Android and could therefore be used to program additional applications as required. The provided SDK for developers gives access to all necessary interfaces. Some of the interfaces were provided by the Android OS, others are coming from TazTag. TazTag does not give full access to the source code, of course; this is provided as precompiled libraries.

## 8.5 Usability

The use of mobile devices has a simple background, it is simple and very practical and convenient to use existing infrastructures therefore the use of a mobile phone or Tablet PC available is very preferable. Whatever is used in the end, the goal has once again to be kept in mind: the notification of a user about an incident which affected the security of a wireless installation. It is therefore only logical and understandable to transmit the notification directly to the user and not elsewhere. Not a flashing light, not an alarm tone is what the user would like to have or will install but simply a notification on their everyday device, the smartphone.

## 8.6 TazPad user experience

The following were the appropriate points of interest when chosing the users end device. First of all we have had a look at the general use of the device, if it is usable out of the box. The properties for visualisation were of interest, because it has to display information in a correct manner. Also the haptic properties were important and as a last point, the audio experience, the possibilities in the area of audio are of interest for producing an alarm sound. All of these points were successfully passed, the device is really usable and useful for further development.

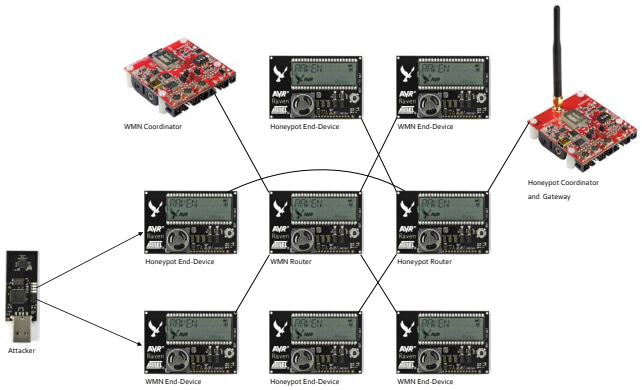



**Figure 1: Detailed design of honeypot network architecture**

## 8.7 Honeypot working scheme

This is a chronological description of honeypot functionality:

1. In normal operation the honeypot regularly reports its status to a central point, the so-called Combiner/Collector. In this case, it is necessary to use the Coordinator of ZigBee, once holding the data, until they can be accepted by a Tablet PC networks as a repeater.

2. An attack recognised by a honeypot node will result directly in a notification.

3. If the channel is not permanently blocked the notification starts directly.

4. In any other case the honeypot node can decide between currently three ways:

   - continue waiting,
   - switch to alternate channel, that was previously defined, or
   - select an alternative medium for notification, which must have been defined previously (example: WiFi or GSM).

5. When the Coordinator of the honeypot is reached, the message is evaluated and made available to the user. For this, several options are to be considered. This can be done through ZigBee or other media.

## 8.8 Honeypot nodes

As described previously in this paper, the nodes have to be configured in a way so that an attacker can not distinguish between "normal" nodes and the ones with honeypot functionality. The AVR Raven boards can be used as ZigBee network nodes and in addition further Raven modules can be used as honeypot nodes. The diagram of the design concept is shown in Figure 1.

## 8.9 Development and testing

The description in this section provides an outlook into a preferred development and testing scenario. Running tests with attacks to the network will provide necessary results to be interpreted and used to improve and refine the prototype. One of the key aspects next to functionality is the system's effectiveness. The testing of functionality under previously considered attacks and duration is essential. When the assumption is proven that use of the honeypot yields efficiency of detection of attacks this will be a success of the proposed solution. The result will be a well-tested honeypot for a wireless sensor networks prototype. A very challenging aspect is the fact that no one before succeeded in presenting a working honeypot system for ZigBee. Papers and reports on a vague idea of this system were presented and also cited in this section, but the challenge here is to provide a working prototype and to present the tests and their results that resulted with this new honeypot system.

# 9. DEVELOPMENT OF HONEYPOT PROTOTYPE

With regard to the limited resources currently available for the research stage, as well as funds for hardware, for instance development kits; at first only a small part of the ideas mentioned is going to be developed. From all of the proposed methods for the honeypot, focus will be first on the detection, later on mechanisms of firmware upgradeability. For signalling and managing, the focus will be on signalling the status, later on managing the system in depth.
The detection is going to be implemented with reproducible decisions, meaning that detection of a high-rated alert event will be categorised on its next occurrence in the same category. Network security is more than a setting up once and then never again touching a running system. The ongoing misuse detection of a honeypot is giving the operator of the wireless sensor network a tangible status of past security events and an outlook on upcoming incidents and events.
The specified design will be used to develop a honeypot prototype in the research. This work will introduce a prototype of a real system to detect and react on active intrusion attempts to ZigBee networks. The literature research revealed that previous attempts were at most just regarding single attack scenarios in simulated environments.

The development of a honeypot prototype will start with implementation of the mechanisms to the design of the proposed product. As development platform the Atmel AVR (and former Meshnetics) products called Raven and ZigBit seemed to be well-equipped with state of the art chipsets and a well established developer community, also regarding ZigBee related topics.

In total a set of 3 MeshBean2 boards from Meshnetics as well as 4 AVR Raven USB sticks and 8 AVR Raven mobile nodes will be used for this task. The development boards contain the same chipsets and are already code compatible by the microcontroller used: an Atmel ATmega 1281 and Atmel ATmega 1284p, as well as the by chipset used for wireless communication, an Atmel AT86RF230. The Atmel AT90USB1287 features additional on-chip USB capable communication for the Raven USB stick.
The detailed design is the detailed elaboration of the sketch of the honeypot concept, which is something like the construction plan of the honeypot.

As such, it is a classic situation from transmitter to receiver, in which the sink is always the coordinator, the gateway towards the host. The question which arises at this point is the amount of detail necessary. The participating modules are shown here schematically.
The Raven and Meshbean modules come with details as to their design, this makes the credible statement that these are in principle based on the same hardware and software.

## 9.1 IDE

Also the programs, tools and techniques used should be listed at this point. For programming the Atmel AVR Studio 6 with debugging functionality is used. Its use is free of cost and therefore ideal for development in a low cost research setup. For debugging purposes an Atmel JTAG device is essential. The AVR JTAGICE mkII is manufactured by Atmel. It is a very powerful development tool for programming and can also be used to debug the programs directly on the microcontroller via JTAG interface. It is very important to mention this, because development without a serious and proper debugging hardware is nearly impossible, using OTAP and just serial programming is really very slow.

The tests on ZigBee networks does not need to be only on my project's own development kit hardware. ZigBee network communication nodes for tests could be hardware from Philips, like 4 different remote controls and 10 end devices from the product range of Philips LivingWhites (light bulbs and power sockets) as well as Living Colors (ambient light), which were available to be investigated in detail. But other ZigBee products in consumer electronics can be targeted, too.

## 9.2 WSN Tablet - Technical description:

TazTag provides access to an Atmel AT86RF231 chipset [1], a single chip architecture created especially for mobile applications. Although the chipset functionality is comprehensive, the single chip architecture allows for running with low power consumption. The processor is a an Atmel SAM 9263.

# 10. SUMMARY

## 10.1 Evaluation of prototype misuse detection capability

Subsequent to the development of the detection capability and the integration of the collector and reporter to the honeypot framework, the whole system will be put into testing and evaluation mode for a series of use cases and attack patterns. With the ambition to detect a high number of various cases of misuse, the network will be guarded by the honeypot and predefined attacks will take place. All of the attacks will be documented to a log location and interpreted with the detected ones to establish a rate of success to failure. This rate can then be compared with others from simulated intrusion detection systems from the literature research. This evaluation comparison will measure the success of our honeypot system for wireless sensor networks. Together with its novelty value this will show that a combination of existing work from different research topics can be applied to research in the field of honeypots on WSN.

## 10.2 Evaluation of prototype quality of security enhancement

The conclusive part of the research takes place when qualitative testing is undertaken in the wild, as it were. Existing contacts to groups of other researchers will be used to invite them to test their tools against the honeypot system. This will result in insights into the stability and operability of the honeypot as an intrusion detection system. The outcome will be plenty of further logfiles and success rate values for attack and detection. The feedback from other researchers testing the system's performance will give additional information about the success of the research project. Although the focus of this research work is the combination of the two worlds of honeypots and wireless sensor networks, further research can be conducted in parallel to this. There is a lot of open questions to solve and others will be asked to join in for possible contributions to this as well.

# 11. CONCLUSION

The conclusion is an analysis of the honeypot system results. The collected results so far will be the basis for analysis of honeypot strengths regarding the outcome of the project work. Common attack scenarios like the replay or garbage attack are tested on the honeypot for analysis. This is part of integration and verification. For evaluation, this routine will be increasingly harsher to analyse the honeypot's robustness and interpret the outcome. The honeypot mechanisms could further be analysed especially with regard to performance.
This paper gives a descriptive model of a honeypot as a security improvement. It shows the procedures and techniques used to specify all its details. A honeypot can be a specific appliance in the field of wireless sensor network security. It works on large scale wireless sensor networks that still have free transmitting slots left. Its operation is automatic and maintenance-free, because it only reports incidents. Of course, battery powered systems have to have batteries replaced as before. We demonstrated the feasibility of composing a honeypot with a wireless sensor network and developed an architecture to improve detection in wireless sensor networks, without having to do frequent detection pattern updates. The Honeytoken approach by [27] uses a comparable approach to misleading attackers and thus securing network traffic, but has no detection and no learning of attack patterns can be achieved.

# 12. OUTLOOK

With the honeypot for WSN we introduced a new style of security architecture to this domain, whose implementation demonstrated the architecture and the honeypot itself as part of the results. We are continuing to conduct comprehensive experiments to further evaluate the performance of the honeypot. The evaluation shall be done with regard to overall ZigBee network performance with and without an attack, and again with the honeypot system added with regard to the overall system performance with and without an attack to the network. The results obtained will yield the basis for the evaluation phase. Further conclusion of the project work will be reported in an upcoming paper. The conclusion of the results will be put into direct correlation to this paper titled "Honeypot framework for wireless network architectures". It is going to be the conclusion of the project

work, especially regarding the performance of the honeypot for wireless sensor networks. One of the topics should be a statement on the general capability of honeypots in wireless sensor networks. As one planned output, another paper will target the interpretation of the findings, titled "Results of honeypot framework for ZigBee network architectures".

## Acknowledgement

The authors would like to thank:
Prof Dr S.M. Furnell - Plymouth University
Dr K.-P. Fischer-Hellmann - University of Applied Sciences Darmstadt

## 13. REFERENCES

[1] Atmel Corporation. Taztag Adds ZigBee Functionality With Atmel Product to New TazCard Contactless Portable Device. 2010.

[2] AVR. AVR Raven. Website, 2008. Available online at http://www.atmel.com/dyn/products/tools_card.asp?tool$_i d = 4291$.

[3] H. Chaouchi and M. Laurent-Maknavicius. Wireless and Mobile Networks Security. 2007.

[4] G. Combs. Wireshark project. Website, 2013. Available online at www.wireshark.org.

[5] M. Davis. SmartGrid Device Security - Adventures in a new medium. Website, 2009. Available online at https://www.blackhat.com/presentations/bh-usa-09/MDAVIS/BHUSA09-Davis-AMI-SLIDES.pdf.

[6] Freakduino. Freakduino project. Website, 2011. Available online at http://freaklabs.org/.

[7] M. Freeman and A. Woodward. SmartPot - Creating a 1st Generation Smartphone Honeypot. *Proceedings of the 7th Australian Digital Forensics Conference*, 2009.

[8] K. Gill. Enhancing the Security of Wireless Sensor Network based Home Automation Systems. 2009.

[9] T. Goodspeed. MSP430 buffer overflow exploit for wireless sensor nodes. Website, 2007. Available online at http://travisgoodspeed.blogspot.com.

[10] T. Goodspeed, S. Bratus, R. Melgares, R. Speers, and S. W. Smith. Api-do: Tools for Exploring the Wireless Attack Surface in Smart Meters. In *2012 45th Hawaii International Conference on System Sciences*, pages 2133–2140. IEEE, 2012.

[11] V. Gopinath. Success Analysis Of Deception In Wireless Sensor Networks. 2010.

[12] O. Kachirski and R. Guha. Effective intrusion detection using multiple sensors in wireless ad hoc networks - System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference. 2003.

[13] K. Khan, W. Goodridge, and D. Ragbir. Security in Wireless Sensor Networks. 2012.

[14] F. Lin, Y.-S. Wang, J.-W. Wang, and C.-H. Chan. Effective network defense strategies to maximize system survivability of wireless mesh networks under malicious and jamming attacks. In *Computational Aspects of Social Networks (CASoN), 2012 Fourth International Conference on*, pages 297–302, Nov.

[15] J. Markert, M. Massoth, K.-P. Fischer-Hellmann, and S. Furnell. Attacks to ZigBee and Wireless Sensor Networks - Honeypots for Detection and Response. In *Proceedings of the Collaborative European Research Conference (CERC 2012), Darmstadt, Germany*, pages 29–36, 2012.

[16] J. Markert, M. Massoth, K.-P. Fischer-Hellmann, S. Furnell, and C. Bolan. Attack Vectors to Wireless ZigBee Network Communications - Analysis and Countermeasures. In *Proceedings of the Seventh Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2011), Furtwangen, Germany*, pages 57–66, 2011.

[17] K. Masica. Recommended Practices Guide For Securing ZigBee Wireless Networks in Process Control System Environments. Website, 2007. Available online at http://csrp.inl.gov/Documents/.

[18] C. Miller. The Legitimate Vulnerability Market: Inside the Secretive World of 0-day Exploit Sales. 2007.

[19] C. O'Flynn. 15dot4-tools. Website, 2012. Available online at http://sourceforge.net/projects/dot4-tools/.

[20] T. Pazynyuk, J. Li, G. S. Oreku, and L. Pan. QoS as Means of Providing WSNs Security. In *2008 Seventh International Conference on Networking ICN*, pages 66–71, 2008.

[21] A. Prathapani, L. Santhanam, and D. P. Agrawal. Detection of blackhole attack in a Wireless Mesh Network using intelligent honeypot agents. *The Journal of Supercomputing*, 64(3):777–804, 2013.

[22] S. Shamsh. Roaming Honeypots along with IDS in Mobile Ad-Hoc Networks. 2013.

[23] R. Siles. HoneySpot: The Wireless Honeypot - Monitoring the Attacker's Activities in Wireless Networks. Website, 2007. Available online at http://honeynet.org.es/papers/honeyspot/HoneySpot_20071217.pdf.

[24] Y. Song, X. Zhu, Y. Hong, H. Zhang, and H. Tan. A Mobile Communication Honeypot Observing System. In *2012 4th International Conference on Multimedia Information Networking and Security (MINES)*, pages 861–865, 2012.

[25] Y.-S. Wang, F. Y.-S. Lin, C.-H. Chan, and J.-W. Wang. Maximization of Wireless Mesh Networks Survivability to Assure Service Continuity under Intelligent Attacks. In *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 583–590.

[26] J. White, J. Brown, S. Ramaswamy, S. Geoghan, and M. Itmi. Securing P2P Wireless Communications by Deploying Honeytokens in a Cooperative Maritime Network. 2009.

[27] J. Wright. Killerbee: Practical ZigBee Exploitation Framework. Website, 2009. Available online at http://www.willhackforsushi.com/presentations/toorcon11-wright.pdf.

[28] X. Zheng, T. Li, and Y. Fang. Strategy of fast and light-load cloud-based proactive benign worm countermeasure technology to contain worm propagation. *The Journal of Supercomputing*, 62(3):1451–1479, 2012.

# Honeypot Effectiveness in Different Categories of Attacks on Wireless Sensor Networks

Jürgen Markert
School of Computing and Mathematics
Plymouth University
UK, Plymouth
Email: jurgen.markert@plymouth.ac.uk

Michael Massoth
Faculty of Computer Science
University of Applied Sciences Darmstadt
Germany, Darmstadt
Email: michael.massoth@h-da.de

***Abstract*—Wireless sensor networks (WSN) are entering the daily life of many. Like wireless LAN before, these are a new playground for developers, but also offer new ways for misuse of this technology. For the improvement of security and to get more valuable data about new attacks, a honeypot for WSNs is in development. This paper presents an effectiveness analysis of a honeypot for WSN and shows detection capabilities in the categories of known and unknown attacks.**

**Keywords: Wireless Sensor Network, Honeypot, Incident Analysis, Intrusion Detection System, WSN, IDS, Preventive Measure, Defensive Approach**

## I. Introduction

By now, everyone has certainly already heard at least something about IPv6, Machine to Machine Communication (M2M), Cyber-Physical Systems (CPS), Internet of Things (IoT) or Industry 4.0, Smart Home, Smart Grid or Smart City. These are all innovations that have started to enter into our lives, new opportunities and standards that aim to simplify, but also constitute a new platform and playground for those interested in technology. The topics mentioned above are all about the communication of devices with each other. Man on the other hand is only in part initiator or beneficiary of this communication.

For example, you can remotely control new light models the so-called Ambient Light by Philips and others. There are also transmitters (e.g. TV remote control from the company named "remotecontrol") for many TV sets of other manufacturers. Samsung uses ZigBee to offer remote controls to control TV sets by Samsung. For smart home use, heating devices can also be controlled from the personal computer; heating intervals are chosen with a configuration program and then transmitted via WSN onto the radiator controller, which then heats the rooms according to these choices. This has the potential for significant savings. In exemplary tests in a three person household a decrease of 14,000 kWh to 10,500 kWh of energy (gas) per year can be observed.

The German government has a project in its High-Tech strategy called "Industry 4.0". Traditional manufacturing is proposed to be computerised to establish the "smart factory". In this concept the "Internet of Things" (IoT) and the "Cyber Physical Systems" (CPS) are used as a conceptual background. Sensor networks are traditionally put into both these categories, so it can be stated that securing these systems is also securing the manufacturing standards of the future. Something is needed for WSNs to help keep companies secure using these and also detect intrusion attempts. This could otherwise result in manufacturing information loss and thus certainly a loss of profit.

We are dealing with the security concepts for WSN, more precisely with the question of whether the use of honeypot architectures in WSN is correct in principle and can also measure and improve security in a WSN.

## II. Related Work

The honeypot for wireless sensor networks (WSN) is an ongoing project. We would like to give a very short summary of recent work steps, followed by an overview on the current state of project work.

At the start of our research we found evidence that there is a lack of security for WSN. We showed the shortcomings and the demand for new security techniques, formulated the idea and conceived a plan to fill this gap with honeypots for WSNs [1]. Subsequently, requirements were written and research to establish a first model of the honeypot system was done. This model showed a course of development with regard to different attack scenarios, which were also found to be critical by other researchers in the field of WSN security research [2]. After going more into detail about how this should look like and showing the components and their interaction, we put the latest results together into a paper sketching a framework [3].

Regarding WSN honeypots, there are other groups starting to work on this topic. A research group from the University of Galway, Ireland, is focussing on security in WSNs [4] and started their project on honeypots and honeynets in WSNs in the year 2013. We are collaborating to share our latest research outcome.

The results of this second research group on honeypots for WSNs are available for their prototype build [5], and they share the experience that some of the classical honeypot features could be adopted to the domain of WSNs [6].

## III. Motivation

We would now like to explain our motivation, then present our solution and thereafter go into the discussion.

The setup you have to visualise as follows:

We have a WSN, which consists of a number of nodes, communicating among themselves, in normal operational state. But there is also the honeypot network, which is basically exactly like the same network mirrored or cloned.

The action of an attacker on the network, and thus at least on one network node, is captured in the model like any interaction.

This leads to a discussion on the main purposes of the honeypot; one purpose is new attack detection. The intention is to make it possible to capture a new attack. Without a device that captures attacks, like a honeypot does, a new attack method could be applied on a victim's WSN without any record of the event happening.

The detection of every attack, be its type known or unknown before, is made possible with the help of a WSN honeypot. Any attack detection would be good, but there is a constraint that has to be regarded as to the boundary of limits; a honeypot can only be useful when there is interaction with it! This will be important in the discussion section.

New attack signature gathering is another main purpose of a WSN honeypot. It is intended to capture new attacks and therefore to gain new insights into attacks. This might lead to new signatures for an intrusion detection system, or to possibly achieve a hardening of the WSN nodes via an update of the firmware.

So the result is that the best thing a honeypot could do is to detect every attack and understand the hackers methods as well as to detect new and unknown attack patterns which can be used to harden the WSN.

"Traditionally a honeypot has been a single system connected to an existing production network in order to lure attackers." [7]. In our opinion it is the detection of new attacks, and therefore gaining knowledge about new attack vectors which would otherwise never be captured and examined, which motivates us to create the honeypot for WSNs.

## IV. WSN Honeypot Review

Our design of the honeypot [3] is the following: Every element of the network is a part of the honeypot. Because every node in a WSN can participate in communication, this results in every single network node being potentially exposed to an attack. So the honeypot functionality must not be considered separately in the implementation of the regular software of a productive WSN, but should constitute part of the operating system of all nodes of the WSN honeypot. All network nodes and their honeypot parts work together.

The goal of our current measures is the refining of the overall design of the honeypot. The above-mentioned honeypot framework for WSN [3] shows a technical overview, but does not tell much about the conceptual background that this paper would like to present. We have a whole list of different attack types, some of which were adapted from other networks, from wire-bound traffic attacks or wireless networks, to the field of WSN; others were invented only in theory and are not yet proven to be working correctly in a real WSN. The different attack types can be categorised for different purposes an attacker might have.

For preparing the appropriate countermeasures to an attack scenario, several discussion points have to be evaluated. As a result it becomes clear that for defensive measures a categorisation of attack results in the different WSN layers attacked has to be worked out.

### A. Goal or Layer Oriented Approach?

1) One goal is the purpose to know which intention an attacker might have and therefore which set of different attacks he will use to get to his goal, to reach it. Different authors have written about this, among them Benenson et al. [8]. In a paper from 2010, an overview was already presented by [9], also doing a categorisation of attacks, although that was not as broad an overview as other surveys.
2) Another approach is - next to the goal of an attacker oriented approach - the layer oriented dividing of attack types according to the networking layer they target. So if an attack is done for instance by jamming the communication frequency, this is targeting the networking layer 1, the physical layer.

## V. WSN layers

The WSN layers can be described by dividing them into:

1) PHY, the physical layer,
2) MAC, the media access control layer,
3) RTG, the routing layer,
4) NODE, the hardware layer.

A work describing this approach is that from Aschenbruck et al. [10] and they also give a number of detection mechanisms they use for their intrusion detection system (IDS) prototype. We would like to present a short summary of their experiences by showing a list of attacks to the physical network layer and additionally our comments.

1) It is definitely hard to detect sniffing and jamming on layer 1, the PHY layer, although there are already possible solutions that were presented recently. Wang et al. [11] focus on countermeasures in a jammed wireless network and did a simulation on their proposed mathematical model to show the improvement of survivability during intelligent attacks. It is unclear whether this type of attack can be detected by a honeypot for WSNs. Due to the fact that jamming does not have to be a "meaningful" transmission of packets, but possibly just a simple disturbance on the frequency, this connectionless and packet-less attack is the perfect example for the need to have an anomaly based intrusion detection system.
2) On the MAC layer, attack types occurring are similarly hard to detect. One attack is by an attacker sending valid-formed packets on the frequency, but without pauses, which would be expected in a CSMA-CA based network. This could lead to a node wanting to send a packet and go to sleep subsequently into a situation where it cannot send packets at all and will have to stay "awake"

to wait for an opportunity to send, but unable to preserve energy.

3) The third layer, the RTG for routing, is a very interesting layer for honeypot routines, because it always makes use of actual network logic. Attack types on this layer can be seen as having guaranteed interactions with nodes, meaning requests and responses being well-interpreted, and not discarded or ignored like it could happen in the second layer, "MAC".
4) The fourth layer, NODE, is some kind of a mixed layer, because here different attack types can be combined together that somehow are connected to each other and fit well into the group of direct attacks on a node, but are of completely different attack types. An attack is for instance, opening a node's hardware case and reading the firmware via JTAG. This is not an attack which can be detected by a piece of software alone; other detection methods have to be used in this case [12]. Further work has to be done here to clear up which attack types have to be taken into account, and others that can be considered to be not regarded, although a direct interaction with hardware is performed.

The next step is a discussion about the attacks and their detection by the use of a WSN honeypot. How does the WSN honeypot fit in this layer view approach? This layer view seems to be a successful approach to continue the discussion about detection mechanisms on the WSN layers, but it needs to be refined for the WSN honeypot. This means that the architectural decisions for our honeypot design are put into question again.

We have to lay down the following rule for further discussion:
"A honeypot in wireless sensor networks waits for unprivileged interaction". This will be discussed later on. So interaction is needed to trigger a honeypot. When the honeypot functionality is already triggered, i.e. the node detected an anomaly occurring, this has to have some meaning. If this is just some disturbance, like a lot of noise on the frequency due to jamming, this could not trigger our honeypot but instead an alarm in an anomaly-based IDS.

One outcome of the discussion, which is very much worth mentioning here based on the rule laid down beforehand, is that there is no possibility to implement honeypot functionality on the PHY layer of a WSN. Not every interaction on the PHY layer from an attacker is unprivileged interaction, i.e. a directed attack. Simply jamming the frequency is an undirected attack without interaction. All the prerequisites and definitions about the facts of a WSN and a honeypot lead to one single result: There is no possible honeypot on the PHY layer with a WSN honeypot.

Thus it can be stated that anomaly based intrusion detection functionality should probably not be implemented on a honeypot.

But for the MAC layer there is an idea, which could lead to a detection possibility. Whenever the honeypot node detects numerous traffic from a single node due to it sending many packets in an aggressive manner without waiting the usual time intervals that are designed to be interaction-friendly, then a trigger could be activated stating that this is due to an attacker and its behaviour. This is similar to the denial of sleep scenario; whenever such a situation is triggered, an attack is occurring. These attacks need interaction, so the use for a honeypot is a given.

On the RTG layer, there is a group consisting of three major attack types; the sinkhole, the wormhole and the sybil attack.

The sinkhole attack is one of the routing attacks. Karlof and Wagner [13] described this attack type as: "Sinkhole attack typically work by making a compromised node look especially attractive for surrounding nodes with respect to the routing algorithm." Its use has a specific pattern that can be recognised and detected by a honeypot node.

The wormhole attack has the objective to propagate a new route over a dedicated medium to apparently speed up the networking as the "best route" in a network. For instance, this could be the proposal to use a direct connection via a WiFi bridge instead of ten hops in a WSN. After being accepted as the best route, a man in the middle scenario can be established by the attacker.

The sybil attack uses the technique of cloning a single node to pretend to be a number of nodes on the network [14]. This security threat could be detected with a honeypot as well, because the cloned nodes would then establish network interaction, which will trigger the honeypot.

This third layer, "RTG", is the best layer for testing the honeypot architecture.

Regarding the fourth layer, "NODE", the physical dislocation of a node is detectable in a static WSN, where the nodes stay at the locations where they were deployed initially. Destroying or removing a node completely can trigger an alarm if each node sends a heartbeat message frequently enough.

In case of reprogramming the node, it is evident that this would be detected if conducted via over the air programming (OTAP). The reprogramming as a result of an attack on a honeypot would trigger an alarm. Capturing the node's hardware and reprogramming it could only be prevented by using measures like physical unclonable function (PUF) on the hardware design.

This was the start of the design review, which we would now like to continue and refine.

Now the question is, how is the cloned mesh implemented to measure an attackers interaction.

1) First, there would be some hard facts, such as the ID of the node. If there is a node with a new and hitherto unknown ID, this is very likely an intruder. Of course, this requires that there is a list of all the honeypot network nodes, also called honeynet, with registered valid IDs. This corresponds to the known whitelisting method or approach.
2) Another approach is to use "security by obscurity". That could mean for example, that for every network node a fixed but not externally known behaviour is

expected, which differentiates it from an attacker's node. For example, a report of an ambient temperature sensor could every second time contain an even digit in a predefined place.

## VI. Discussion

We would like to rework several major points for discussion. It seems that the most important work will be to make a honeypot for WSN particularly desirable for an attacker. This is a very valuable topic for further research. Here we could discuss various approaches, as this aspect has so far not been considered in detail. But some points can now already be worked out that would have to be considered: increasing the transmission power of honeypots, or positioning of the honeypots within easy access range to public places, near the border to factory premises or similar; these possibilities have to be regarded in detail.

Another point arising for discussion is about throughput; is there really enough bandwidth for an additional network of similar size? For a first answer to this it could be stated that one could measure the co-existence of honeypots, i.e. to apply the throughput test in order to show that a honeynet can exist as an additional network next to a production network.

A couple of further questions concern alternatives to honeypots. Wouldn't it be possible to use sniffers to record all of the traffic for analysis? This could create a derivative of the network infrastructure. Is that not a kind of honeypot? But the answer is a simple no. A sniffer is not a honeypot and also not like a honeypot; it records data traffic of one node and processes and displays it. The honeypot for WSN is designed as a large-scale deployed system, to be able to detect an attack to a node of the honeypot anywhere on the (e.g.) factory premises. A sniffer would need a a large amount of memory to store the network data, and WSN nodes generally don't have this. Also, if the node would be equipped with a gateway to forward the captured data, it would only be a single point in the network that records the network traffic. And that would in a WSN only have sufficient signal strength within a radius of a few meters. So just one sniffer is not a proper alternative for a honeypot solution, but has other purposes.

And the discussion point coming up next as a result of the previous one is: if a single honeypot is insufficient, why must it be a whole network of it? The network coverage was discussed before, which is the main reason for this approach of a WSN honeypot, because attacks on a single node could remain undetected.

Regarding a question on how important a feasibility study would be at this point to show or disprove that a honeypot functionality can be best integrated into an existing network: So far we have always been answering in the negative, but never proven that this cannot be integrated into an existing network. We are currently of the opinion that we separated the honeypot from other complex designed WSN applications, with the main reason to make its results measurable.

But would it not be possible to attach a honeypot flanged to existing systems? In principle, yes, but up to now we decided for strong opposition to this for design reasons. We assume this increases the difficulty to discern the results; it might weaken the results how the honeypot components behave exactly when they compete with other applications of WSN for the limited resources of nodes.

Should the honeypot have been shown to recognise anything? We believe it is sufficient for now to create groups of attack scenarios that have a pattern in common. If we can argue that these are indeed groups and why, then we can show examples to use to attack these groups and to test the honeypot functionality for the whole group of attacks.

Do we need to implement a honeypot for all types of devices? Yes clearly, for example, an attacker could for his purposes indeed show interest in not only end devices, but for example sensors, or actuator nodes which implement control commands.

Would a throughput test give evidence that a honeypot is feasible and be capable to show the honeypot completeness? No, a throughput test is unsuitable for this question, but the throughput test should be used to show that honeypot and production network can exist in parallel.

Is the complexity of the systems really that big that you cannot be prepared for all possible attacks? This question can, if necessary, answered by transferring from past experience, only that is for sure.

Two fundamental insights from this discussion are formulated in the sentences:

A honeypot in a WSN is capable to detect direct interaction.

Direct attack attempts should be detected by every wired or wireless honeypot.

## VII. Outlook

One idea at this stage is the throughput test which could be done during an attack on the production net and also during an attack on the honeypot network. The expectation would be that throughput would decrease significantly during an attack on the production net, but not in the same amount as when the honeypot network is targeted. We would like to point out one of the open research topics, as mentioned in the discussion, the question on how to make interaction with a WSN very desirable.

## VIII. Conclusion

This work intentionally listed the attack types a WSN could likely be facing and presented the defensive approaches a WSN honeypot could present to an attacker. The sophisticated defensive measure is found in direct interaction with the WSN honeypot. Several potentials to detect other attacks on the hardware layer and on lower networking layers are given. Additionally, classical statistical methods can be used for detecting non-intelligent network disturbances. A WSN honeypot offers methods to capture attack attempts. With the possibility to give an opportunity for the examination of security incidents, the analysed data can be used to develop further preventive measures to defend WSN security.

## ACKNOWLEDGMENT

The authors would like to thank:
Prof Dr S.M. Furnell - Plymouth University
Dr K.-P. Fischer-Hellmann - University of Applied Sciences Darmstadt

## REFERENCES

[1] J. Markert, M. Massoth, K.-P. Fischer-Hellmann, S. Furnell, and C. Bolan, "Attack Vectors to Wireless ZigBee Network Communications - Analysis and Countermeasures," in *Proceedings of the Seventh Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN 2011), Furtwangen, Germany*, 2011, pp. 57–66.

[2] J. Markert, M. Massoth, K.-P. Fischer-Hellmann, and S. Furnell, "Attacks to ZigBee and Wireless Sensor Networks - Honeypots for Detection and Response," in *Proceedings of the Collaborative European Research Conference (CERC 2012), Darmstadt, Germany*, 2012, pp. 29–36.

[3] J. Markert and M. Massoth, "Honeypot Framework for Wireless Sensor Networks," in *Proceedings of International Conference on Advances in Mobile Computing & Multimedia*, ser. MoMM '13. New York, NY, USA: ACM, 2013, pp. 217:217–217:223. [Online]. Available: http://doi.acm.org/10.1145/2536853.2536941

[4] E. Holohan and M. Schukat, "Authentication Using Virtual Certificate Authorities: A New Security Paradigm for Wireless Sensor Networks," in *Network Computing and Applications (NCA), 2010 9th IEEE International Symposium on*, July 2010, pp. 92–99.

[5] K. Lasota, E. Niewiadomskia-Szynkiewicz, and A. Kozakiewicz, "Adaptation of honeypot solutions for WSN - Adaptacja rozwiazan honeypot dla sieci czujników," *Studia Informatica*, vol. Vol. 33, nr 3A, pp. 139–148, 2012.

[6] K. Lasota, E. Niewiadomska-Szynkiewicz, and A. A. Kozakiewicz, "Honeypot for mobile sensor networks - Mobilny honeypot dla sieci sensorycznych," *Przeglad Telekomunikacyjny- Wiadomosci Telekomunikacyjne*, no. 8-9/2012, pp. 699–704, 2012.

[7] H. Project, *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*. Addison-Wesley, 2001. [Online]. Available: http://books.google.de/books?id=p4fuAAAAMAAJ

[8] Z. Benenson, P. M. Cholewinski, and F. C. Freiling, "Vulnerabilities and attacks in wireless sensor networks," in *Wireless Sensor Network Security*, ser. Cryptology and Information Security Series, 2008.

[9] M. S. I. Mamun and A. F. M. S. Kabir, "Hierarchical design based intrusion detection system for wireless ad hoc network," *IJNSA*, vol. 2, 2010.

[10] N. Aschenbruck, J. Bauer, J. Bieling, A. Bothe, and M. Schwamborn, "A Security Architecture and Modular Intrusion Detection System for WSNs," in *Networked Sensing Systems (INSS), 2012 Ninth International Conference on*, June 2012, pp. 1 –8.

[11] Y.-S. Wang, F. Y.-S. Lin, C.-H. Chan, and J.-W. Wang, "Maximization of Wireless Mesh Networks Survivability to Assure Service Continuity under Intelligent Attacks," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 583–590.

[12] A. Becher, Z. Benenson, and M. Dornseif, "Tampering with Motes: Real-world Physical Attacks on Wireless Sensor Networks," in *in 3rd International Conference on Security in Pervasive Computing (SPC*, 2006.

[13] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks: Attacks and countermeasures," in *In First IEEE International Workshop on Sensor Network Protocols and Applications*, May 2003, pp. 113–127.

[14] W. Z. Khan, M. Y. Aalsalem, M. N. B. M. Saad, and Y. Xiang, "Detection and mitigation of node replication attacks in wireless sensor networks: A survey," *International Journal of Distributed Sensor Networks*, vol. 2013, 2013.

OPEN ACCESS

2nd International Electronic Conference on Sensors and Applications 15–30 November 2015

www.mdpi.com/journal/sensors

*Conference Proceedings Paper–Sensors and Applications*

# Honeypot Wording and Definitions in Wireless Sensor Networks

**Jürgen Markert [1,*] and Michael Massoth [2]**

[1] Centre for Security, Communications and Network Research, School of Computing, Electronics and Mathematics, Plymouth University, Drake Circus, Plymouth, PL4 8AA , UK; E-Mails: jurgen.markert@plymouth.ac.uk

[2] Computer Science Department, Hochschule Darmstadt - University of Applied Sciences, Haardtring 100, 64295 Darmstadt, Germany

* Author to whom correspondence should be addressed; E-Mail: jurgen.markert@plymouth.ac.uk

*Published: 9 November 2015*

---

**Abstract:** Wireless Sensor Networks (WSN) as emerging technology are becoming even more important through the technisation of industry, our mobility driven lives and our "smart homes". In a world where a plethora of our devices interact to serve our purposes, it's clear that competing groups may want to use these devices for their own objectives. Therefore we focus on honeypots for WSNs as an opportunity to become acquainted with the techniques used to misuse or even to take-over WSNs. This knowledge has the potential to improve the WSN designs and even those already deployed in the field. Honeypots in information security can be found in computer networks where a client-server model is applied and client-honeypots are very different from server-honeypots. In WSNs the client-server model is not applicable. This was the motivation behind determining whether it would be a honeypot at all if brought into this domain. What if we have to call it something else? The uncertainty of a missing naming convention leading to ambiguous conversations about this new topic needs discussion. In this paper we show the classic definition, discuss the categorisation and give recommendations for further use as well as the wording that will be used further on in the following research.

**Keywords:** Honeynet, Honeypot, WSN, Wireless Sensor Network, Network Security, IDS, Intrusion Detection System

---

## 1. Introduction

Honeypots for WSNs [1] are an emerging and mostly unexplored research field. We give an overview on the topic of honeypots and compares honeypots with the purpose of starting a discussion on the differences between WSN honeypots to wireless ones as well as those in wire-bound networks. Numerous definitions and terminology exist. This discussion is on their use in WSN and offers guidance or recommendations for further use in the field of honeypots for WSN. This clarifies our description of a WSN honeypot. The meaning of the term WSN honeypot is established, because, so far, a concise definition does not exist.

A honeypot is a well and widely known term in information security. Like the authors of [2] have practised, it stands for a computer system that is closely monitored, reachable as a decoy and has only one purpose: to entice the bad guy into this pit trap. The result is knowledge. It is important to understand there was (an entity with) the intention to utilise the system for (its own) other purposes. Additionally, a closely monitored system will also give valuable data for analysis, for instance the attack vector, the strategy, or the tools and techniques used. A set of more than one honeypot acting jointly is called honeynet [3].

Classically, the discussion is about high-interaction honeypots vs. low-interaction honeypots, when speaking about honeypots in wired network environments [4]. These are then further divided into client-side and server-side honeypots, depending on the deployment of the honeypot as being a client system or a server system [5]. This detail is very important since a client is exposed to other threats than a server system. As an example, on a personal computer client an attack vector is triggered when a malicious website is visited (e.g. by a drive-by-download). On the other hand, a server system offers services likely to contain vulnerabilities, that can be exploited to gain control of the system or deploy other malware for further attacks.

High-interaction server honeypots are set up to focus on attacks carried out manually and new methods of the attacker. Therefore a highly interactive honeypot is set up in a way that achieving control of it appears very desirable, i.e. it would offer significant bandwidth for a potential attacker or seem to contain documents or data of interest. High-interaction honeypots are often set up in the same network as the “normal” server. The normal server is hardened as good as possible and the honeypot is under continuous monitoring. There are honeypot software solutions for these monitoring purposes. Highly interactive honeypots are very resource-intensive in maintenance, repair and operations (MRO) [4].

On the client-side the high-interaction honeypots are partly automated web crawling systems with real client software. They are configured to visit websites and capture incoming attacks as well as their payloads in a sand-boxed system. These honeypots preferably run on a virtual machine for convenience. Further their web browsers are designed to run in sandboxes.

Low-interaction honeypots are defined as systems that emulate one or more systems [4],[5].

When regarding low-interaction server honeypots, these are services that are emulated. A single interaction with them is enough to capture an intrusion attempt. A low-interaction honeypot could also seem to be a whole network, emulated by a single honeypot system. The emulated services offer only a subset of the features that would be available on a real system. Those systems are most often used to capture packets of automated or new attacks, to get new variations of computer network worms and to get

the latest version of a malware, this could for instance be a software trojan application. An attacker has various possibilities to detect being connected to a honeypot, but this fact could be ignored completely, because the intrusion detection attempt is already valuable information enough at a low-interaction server honeypot.

A low-interaction client honeypot is for example a web crawling client emulation that visits websites to capture drive-by-download attempts of infected malicious websites.

## 2. Discussion

For the start of a naming discussion after a preliminary short summary of the various details of established honeypot concepts. With the established fact that a honeypot has no other purpose than to attract and monitor an attackers' behaviour (compare to honeypot definition by Spitzner [4]) We compare the WSN honeypots to this, exactly this is what is done with the "Honeypot Framework for Wireless Sensor Networks" [1]. Due to this fact that every connection to a honeypot is by definition an attack, because it has no other purpose than to attract and to monitor the attack, an intrusion detection attempt is already valuable information enough. So if the name "honeypot" is indeed the right term for a honeypot independent of being in a wired-network, a wireless network or a meshed network like a WSN is, it can be called a honeypot, but not something else. Despite the fact, that in a WSN the routing is different compared to IP networking and we don't use ports and services, so a service like a honeypot daemon for a single networking service is not used, it will be called honeypot.

### *2.1. Low- and high-interaction honeypots*

With the definition that a honeypot in a WSN is a generally accepted term for a system that attracts an attacker and otherwise has no real use for the original purpose of the WSN, can a statement be formulated to say something about the fact of it being a high-interaction honeypot or is it a low-interaction honeypot?

Like it was shown in the introduction, that low-interaction honeypots are simulating services on real systems, but they are not physically there, they are not real hardware. It is abundantly clear, that the devices in a WSN do not offer services like a mail- or web-server in an IP based network does. Due to the fact that a node in a WSN is real hardware and does not emulate anything, a honeypot in WSN must be a high-interaction honeypot.

### *2.2. Honeypots form a honeynet*

Likewise it was stated in the introduction, that more than one honeypot working together is called honeynet, this goes further on. Is it a honeypot if it is distributed and should it be named a honeynet or is a swarm of nodes that ba acting together are the honeypot? In a WSN all the sensors of the network together have the honeypot capabilities. A WSN covers a geographical region, its nodes can only transmit and receive signal within a limited radius. Therefore the taken approach due to the physical region a WSN can only be is that this WSN running honeypot routines and algorithms should together be called a WSN honeypot. A single node of the WSN is not the honeypot, it is all the nodes working together. Our recommendation here is to stick to the naming convention that is already established; a set

of honeypots working together is called a honeynet. More than one WSN honeypot acting together may then be called a WSN honeynet.

This is also how a honeynet for wire-bound networks is designed. Numerous systems work together to collect the attacks, honeynets can work together globally. This design can be adapted for a WSN honeynet, consisting of WSN honeypots.

The WSN honeynet is a solution that can be reached by numerous WSN honeypots, for instance one in every metropolitan region, sending their distributed alerts to a database for analysis. In this setup the analysis of attacks is possible and therefore new improvements can be developed for securing the WSNs.

### *2.3. Client-Server Model*

In a discussion whether it is a client or a server honeypot the classical definitions of client and server help us to come to conclusion. We assume that the WSN honeypot functionality is put (among others) in an end-device node of the WSN. As a result, this is by definition not a client honeypot. WSNs don't have clients comparable to those on a personal computer. Each of the nodes in a WSN has networking capabilities and offers services e.g. being an actor for controlling the heating control unit of a radiator. But is it a server? A server offers services and responds to requests.

With regard to the commonly known wording for the client-server model, the WSN do not fit there. One of the comparable models is the peer to peer networking architecture that has a decentralised organisation where every node can communicate to every other, directly or indirectly via hops. Looking at other networking models, a WSN is rather either a peer to peer or a meshed network.

So a WSN honeypot is neither a client nor a server honeypot.

### *2.4. Physical or virtual honeypot*

Is it a physical or a virtualised honeypot? With regard to the scarce resources in a WSN it will most likely in any set-up be a physical honeypot. This can be stated, because research on virtualised wireless sensor network nodes is further ongoing.

### *2.5. Effectiveness on different layers*

Likewise a honeypot needs directed interaction so that if an event is triggered, a WSN honeypot has the same premise. It waits for interaction, so that the event can be analysed.

An estimated effectiveness discussion was previously already discussed and we would like to give a reference to this [6].

## 3. Results and Discussion

So as a matter of fact it has to be stated, that it is neither a client nor a server honeypot, but a WSN honeypot, nonetheless, this results in:

1. A honeypot in a WSN should be named what is is: a WSN honeypot.

2. The WSN honeypot is most likely a high-interaction honeypot, nothing is emulated.

3. All the nodes are physical, nothing is virtualised.
4. A differentiation between client and server is impossible, a WSN is different in this aspect.
5. A WSN honeypot needs direct interaction to be triggered.

## 4. Conclusion

Therefore the fact of matter is, as the result of the discussion on the naming conventions for honeypots in WSNs it has to be stated, that it is neither a client nor a server honeypot, but a WSN honeypot, nonetheless. Its use is worthy and needed, paying attention to honeypots shall be on every organisations list.

## Acknowledgments

The authors would like to thank the CSCAN, Centre for Security, Communications and Network Research and the GSD, Graduate School Darmstadt.

## Conflicts of Interest

The authors declare no conflict of interest.

## References

1. Markert, J.; Massoth, M. Honeypot Framework for Wireless Sensor Networks. Proceedings of International Conference on Advances in Mobile Computing & Multimedia; ACM: New York, NY, USA, 2013; MoMM '13, pp. 217:217–217:223.
2. Cheswick, W.R.; Bellovin, S.M. Firewalls and Internet Security: Repelling the Wily Hacker, 1994.
3. Honeynet Project. *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*; Addison-Wesley, 2001.
4. Spitzner, L. *Honeypots: Tracking Hackers*; Addison-Wesley, 2003.
5. Provos, N.; Holz, T. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*; Pearson Education, 2007.
6. Markert, J.; Massoth, M. Honeypot Effectiveness in Different Categories of Attacks on Wireless Sensor Networks. Proceedings of Database and Expert Systems Applications (DEXA), 2014 25th International Workshop on. IEEE, 2014, DEXA '14.