Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Karl Edvard Balto

Hybrid IoT Cyber Range

Master's thesis in Master in Information Security
Supervisor: Basel Katt
Co-supervisor: Muhammad Mudassar Yamin, Andrii Shalaginov
December 2022

NTNU
Norwegian University of
Science and Technology

Karl Edvard Balto

Hybrid IoT Cyber Range

Master's thesis in Master in Information Security
Supervisor: Basel Katt
Co-supervisor: Muhammad Mudassar Yamin, Andrii Shalaginov
December 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Hybrid IoT Cyber Range

Karl Edvard Balto

December 15, 2022

Abstract

The use of IoT devices has increased fast lately, development of new devices moves fast, prices are forced down and thus the costs has to be reduced. IoT devices are trusted with more tasks that are critical, therefore it is important that the devices behave as intended, information is protected and the importance of security increases. It is not always the IoT device it self that is the target of an cyber attack, but it can be a tool for another attack. Users, especially home consumers, expect the devices to have high usability and is easy to use and set up. To reduce costs, complexity and time, the cut-downs are often within security.

To increase awareness and knowledge within IoT security, education, awareness, demonstrations and training is necessary. Small changes might result in large security benefits. With increased awareness and knowledge to the developers, manufacturers and users, they are able to do choices that can increase security. To increase knowledge and awareness in IoT security a training ground for IoT security is proposed, an IoT Cyber Range. Cyber ranges have lately got more attention, but not as much in the IoT field, at least not what is publicly available.

As the diversity in IoT devices is large, different vendors, different architectures and different components and peripherals, it is difficult to find one solution that fits all IoT devices. To some level, IoT devices can be emulated, but it is not feasible to create emulators for all types of devices, so to cover all needs why it is necessary to combine emulation with real hardware. Cyber ranges with this combination is called a hybrid cyber range.

This project surveys the requirements for a hybrid IoT cyber range and how to create such a range to fulfill those requirements

Sammendrag

Bruken av IoT enheter har økt kraftig de siste årene, utviklingen av nye enheter går fort, prisene presses og dermed må også kostnadene reduseres. IoT enheter får flere og flere oppgaver der det blir mer kritisk at enhetene virker etter intensjonen, der informasjon blir beskyttet og dermed at sikkerheten blir viktigere. Dersom man blir utsatt for et angrep er det ikke alltid at IoT enheten er målet, men enheten kan brukes til angrep videre. Brukere, spesielt private brukere, har en forventing om at installasjonen skal være enkel og at enheten skal være brukervennlig. For å redusere kostnader, kompleksitet og tid er det ofte sikkerheten som blir skadelidende.

For å øke bevissthet og kunnskap innen IoT sikkerhet er opplæring, holdningsarbeid, demonstrasjoner og øvelser nødvendige. Ofte kan det være enkle og små endringer som skal til for å øke sikkerhetsnivået betraktelig. Med økt bevissthet og kunnskap vil de som skal utvikle, produsere og bruke utstyr kunne ta valg som øker sikkerheten. For å øke nivået av sikkerhetsbevissthet og -kunnskap foreslås en øvingsfelt for IoT sikkerhet, en IoT Cyber range. Cyber range har i det siste blitt populært i sikkerhetsmiljøene, men det har vært mindre fokus på IoT, i alle fall blant det som er offentlig tilgjengelig.

Siden IoT enheter er veldig forskjellige, fra forskjellige leverandører, har forskjellige arkitekturer og komponenter er det vanskelig med en løsning som dekker alle typer IoT enheter. Til en viss grad kan IoT enheter også emuleres, men det er ikke gjennomførbart å lage emulatorer for alle variasjoner av IoT enheter, derfor er det også nødvendig å kombinere emulering med reelle IoT enheter for å kunne dekke alle behov. En cyber range med denne kombinasjonen kalles for hybrid cyber range.

Dette prosjektet undersøker hva slags krav som stilles til en hybrid IoT cyber range og hvordan en slik kan lages for å dekke kravene.

Contents

Figures

Tables

Acronyms

ADB Android Debugging Bridge. 9

AI Artificial Intelligence. 18, 27

CPS Cyber-Physical-System. 1, 16

DDoS Distributed Denial of Service. 3

DoS Denial of Service. 19

DSR Design Science Research. 14, 38, 43

GPIO General Purpose Input/Output. 28, 29

IIoT Industrial Internet of Things. 1, 8, 16

IoT Internet of Things. 1, 2

IRT Incidence Response Team. 17

JTAG Joint Test Action Group. 9, 21, 37

MTU Maximum Transmission Unit. 26

NIST National Institute of Standards and Technology. 1, 6

OSI Open Systems Interconnection. 29

RTOS Real-Time Operating System. 9

SoC System On Chip. 20, 23, 36, 37

SPI Serial Peripheral Interface. 21, 36

SWD Serial Wire Debug. 21, 22, 37

TAP Test Access Port. 22

UART Universal Asynchronous Reciever-Transmitter. 21, 28–30, 34, 36, 38, 48

Chapter 1

Introduction

The Internet of Things (IoT) are a collection of "things" (devices, sensors, objects) that are connected through network or Internet for the purpose of automation and/or ease human tasks. These devices can be sensors or controllers for processing temperature, humidity, motion, image, sound etc. to give intelligence for a system or a human so it can do a task based on the input. These systems are often reffered to as smart homes, smart health, smart industry/industry 4.0/Industrial IoT(IIoT), smart cities, smart transportation or smart aviation, in short smart everything. The devices can be actuators, sensors, power controlling relays, bulbs, pacemaker, camera, weather station, motors, door locks and so on.

IoT devices can affect the physical world, which is referred to as Cyber-Physical-System (CPS). With CPS, an adversary, can through network operations cause damage in the physical world. Not only can it cause physical damage, but it can also collect information about the actual physical world, including information that affect persons privacy. As IoT devices take over more of human tasks in controlling measurements and actuation tasks, it is very important to assure high availability and high confidence in the installation. IoT devices are often placed in locations without constant power supply and without wiring at all. This results in IoT devices depending on battery as power source and wireless transmission as communication medium, both with their benefits and challenges.

The IoT development is happening at high speed and it is challenging for the information security field keep up the speed. To reduce costs and time during development and production of devices information security is not a priority, and since the consumer often, especially private customer, is more focused on price rather than security, the manufacturer has to compete in costs.

To increase the cyber security awareness among the users, developers and manufacturers, increasing their knowledge within cyber security is a possible measure. To increase these security skills and knowledge in the IoT security field there is a need for learning and training, and skills should be a desired asset when working within developing, learning, education, forensics and research. For this purposes Cyber ranges or testbeds are often developed and used.

NIST defines a Cyber Range as [1]:

> Cyber ranges are interactive, simulated representations of an organization's local network, system, tools, and applications that are connected to a simulated Internet level environment. They provide a safe, legal environment to gain hands-on cyber skills and a secure environment for product development and security posture testing.
>
> A cyber range may include actual hardware and software or may be a combination of actual and virtual components. Ranges may be interoperable with other cyber range environments. The Internet level piece of the range environment includes not only simulated traffic, but also replicates network services such as webpages, browsers, and email as needed by the customer.

Malware and information security attacks have a potential to damage production systems, a Cyber Range is an environment to ensure isolation from other systems, thereby not affecting production systems. The environment should also be a complete environment, where users can be trained, and devices can be tested and demonstrated, without being limited by the lack of functionality or realism.

1.1 Problem description

As pointed out earlier, there is a need for training within the IoT security field, and Cyber Ranges seem to fit as a solution. While learning through education, reading, lectures and awareness campaigns do increase knowledge and awareness, hands on exercises, demonstrations and training can give other effects in learning. Conducting training for all stakeholders in information security is often done by labs and exercises [2]. Cyber ranges can do both, as defined by NIST above. Cyber Ranges must also be designed to give the users an environment where the focus is on learning IoT security and not spending most of the time administering and setting up and/or preparing the Cyber Range. IoT is a challenging field as there are rarely standards available, and even if the standards exit, they might not be used and followed at all or maybe only partially.

Cyber exercises are often used to to increase knowledge within the cyber security field. Learning through practical challenges and discussions supplements learning through reading and through lectures. Often Cyber Ranges are used as tools for Cyber Ranges.

Recently, there have been many proposals for Cyber Range designs, however there are not many Cyber Range that have IoT focus. While the existing ranges can emulate IoT devices, there are challenges that can not be solved in the virtual realm and it is not feasible to develop emulation for all devices. IoT testbeds also exist, however their main purpose is not always security or their design is closed with no open publications.

In 2020 Gartner predicted that there will be 21 billion IoT devices in 2025, now in 2022 the prediction for 2025 is already 65 billion IoT devices. In 2020 the estimation of IoT devices where 7 billions, by the end of 2022 the prediction is that

there will be more IoT devices than computers on the Internet [3].
IoT devices are rarely standardized and require different approach for almost every device. With a large number of manufacturers and at least the same many ways to develop firmware and hardware, it is not feasible to have a one size fits all solution. Product life-cycle are often short and end-of-life for developing and patching for security issues and bugs is earlier than when the customer stops using the devices. Devices are also often forgot in an implementation and gets left behind in a network creating vulnerabilities available for a threat actor. Compromised IoT devices can be an issue for the network, the user, other networks or other people. The Mirai botnet [4] alone was a botnet of 600.000 vulnerable IoT devices that were controlled for a massive DDoS attack on services on the Internet.

A combination between virtual and physical cyber range is referred to as a hybrid cyber range and combines the best of both worlds. Virtual devices can be emulated or simulated. Emulating and simulating IoT devices is not always the best option in a Cyber Range, i.e. the diversity in devices makes creating an emulation environment adapted to every device type is time consuming, therefore an IoT Cyber Range should be able to handle physical devices as well as virtual, emulated devices.

Setting up a training environment can be time consuming when configuring every device for a scenario. When one exercise is complete and a new team is to do the same exercise, the administrators of the Cyber Range is to set up the same scenario again.
Reducing the time with human interaction spent on preparing a Cyber Range to be ready for an exercise, or managing other phases of an exercise, is time that can be used in developing new scenarios and exercises, as well as reducing costs. Saving costs and man hours, will in turn result in training more people in IoT security topics. According to Vykopal et al. [5], a cyber exercise is costly, especially in time.

On the assumption that Cyber exercises increase security awareness and Cyber Ranges are useful tools for a Cyber exercise, an IoT Cyber Range could be used to increase IoT security awareness and knowledge. However, a Cyber Range should be as little resource demanding as possible while still providing necessary infrastructure to create realistic exercises and test environments.

1.2 Research questions

Question 1: *What are the requirements for a hybrid IoT Cyber Range for smart home devices?*
The requirements scope is limited to building a hybrid IoT Cyber range for smart home devices and to interact with physical smart home devices.

Question 2: *How to design and implement a hybrid IoT Cyber Range fulfilling the requirements stated in question 1?*
The project is focusing on reducing resources in provisioning the range and scenarios in the Cyber Range.

1.3 Contribution

This thesis plan to design a Cyber Range for IoT, with smart home focus, where the Cyber Range combines virtual emulation and actual physical IoT devices. The Cyber Range also have a large automation focus to reduce human interaction when creating exercises and training scenarios and thereby reducing time spent for managing. The requirements for a Cyber Range, as well as the need for a Cyber Range is researched.

This chapter discusses the problem, the next chapter shows related papers and theories relevant to this project, chapter three discusses the methodology of the project, chapter four shows the results from the study and the discussion chapter evaluates the project. The last chapter show some of the possible future enhancements for this project.

1.4 Keywords

Cyber Range, IoT, smart home, testbed, hybrid cyber range, cyber exercise

Chapter 2

Background and Related Work

This chapter is based on a literature review on why Cyber Ranges are a potential solution for Cyber Security challenges and what relevant Cyber Range proposals are available.

2.1 Background

Touqeer et.al [6] did a study on what the security challenges in IoT smart home might be. They divided the challenges into four IoT layers, application layer, perception layer, network layer and physical layer, each with their challenges. The application layer holds the applications and services for IoT, the sensors and inputs from the environment belongs to the perception layer, network software and devices belong to the network layer and the physical layer holds the "smart" devices. There was also outlined some solutions to the challenges in each layer.

A more recent paper [7], illustrates the need for consumer to increase awareness and have an active participation in their own smart home security and privacy area. They refer to data in a report from GOV UK, that consumers lack awareness on what to look for when buying secure products, and that there is no marking of what products are considered secure, and what their security level is. They propose a platform for IoT security awareness and system hardening advisory, using concepts like crowdsourcing and gamification, a solution that is open and available for both end-users, retailers and manufacturers,

Also Koohang et al. [8] state that end-user IoT security awareness is important, especially privacy, security and trust. Privacy and security are precursors to trust: "IoT awareness is defined as the degree to which users know the basics of growing security/privacy threats of IoT that they may encounter on a routine basis." They found that IoT awareness increases the knowledge in IoT security and IoT privacy, and that this knowledge increases trust in IoT. The trust also increases intentions of IoT usage. Awareness and training programs are suggested as activities to increase IoT awareness.

One method to do training with defined scenarios, resources and participants are cyber exercises. To create an exercise is time and resource consuming. Automating Cyber Ranges reduces the time resetting and restarting an exercise or scenario. Once a setup is created, a Cyber Range should also be able to be set to its initial state with minimal effort. Vykopal et.al. [5] present a cyber exercise life cycle and divide an exercise into 5 parts: preparation, dry run, execution and evaluation. In the preparation phase the learning and training objectives are defined, scenario and background story is developed, scoring system is defined and the technical infrastructure is developed and deployed. A minor test, called Hackathon, is carried out, to test the infrastructure. During the dry run phase, the scenario is completed with testing teams to adjust the products from the preparation phase. The execution phase is the planned exercise as is after the adjustments from the dry run phase. The evaluation phase gathers all information from the previous phases, this information is used for improving the exercises, give learning to the participants, show examples of best practices and discuss other suggestions for solutions. The execution phase can be repeated for several runs, for several teams or several retries, requiring resetting the state of the exercise to the same state as before the exercise.

While the exercise phase is can be days, the preparation phase is longer, possibly months. Also the dry run phase is likely to be longer than the exercise it self. During the preparation and dry run the infrastructure is tuned and adjusted. To reset an infrastructure, especially actual hardware, is a time consuming work. To automate this task could be beneficial to reduce the overall time for an exercise or for infrastructure testing would give an opportunity to use more time to increase training, skills, knowledge and awareness, in stead of using time for manual repeating and possibly tasks.

[9] also show that cyber exercises is a possible way to increase knowledge, and as Koohang et al. [8] state, awareness is based on the knowledge of threats.

Cyber Ranges have many use cases, Päijänen et al. [10] used data from a survey in the CyberSec4Europe project in 2020. Cyber ranges are mainly used for Security education (82%), security research and development (72%), competence building (62%), development of cyber capabilities (51%). A single cyber range could have one or several use cases. The survey listed 11 use cases, only the use cases that covered more than 50% of the cyber ranges are mentioned here. The same paper also shows that 77% of all cyber ranges in the survey support two or more target groups, and 20% of the ranges support four or more target groups, where the largest target groups are companies and enterprises (77%), degree program students (23%) and government organizations (23%).

A similar definition of a cyber range, but broader than NIST's definition, is [11]:

> Cyber ranges, defined as purpose-built testbeds and experimental research infrastructure, are intended to conduct testing and evaluation, training, and exercises.

A Range is a training area, e.g. a shooting range for shooting training. It is an area for training in a controlled environment. A Cyber Range is a training area where cyber is in focus, training with the challenges that can emerge in the cyber domain. While a range is designed for training, a test bed is designed for testing equipment while developing or in manufacturing. A Cyber Range can also be used as a test bed, but a test bed needs more functionality to cover the requirements to be as complex as a cyber range.

2.2 Related work

Cyber ranges has been created since 2008 [12], when DARPA started developing the US National Cyber Range (NCR). The US NCR initiative was started to cover a need for training in cyber network operations. NCR is also used as an abbreviation of the Norwegian Cyber Range.

In 2013 Davis and Magrath from the Australian Department of Defence did a survey on Cyber Ranges and Testbeds [13]. They identified a Cyber Range to cover three main roles. The first role was testing of new devices. The second role is training. This is useful to increase the skills and knowledge for persons participating in operations. The third role is research and development. At that point the training roles was the most popular role for Cyber Ranges. In the time of the survey they noticed a trend where the ranges were moving from simulation to emulation.

Yamin et al. [14] did an IoT Smart Home case study in 2018, with an exercise with physical IoT devices. This was not done in a cyber range, but used as a test bed. The paper showed that an exercise improved skills of the participants, significantly. The study also suggested that automation would improve repeatability and reduce resources spent on an exercise. Pure software platforms do not mimic real world behavior and therefore realistic exercises should be preferred. There was also suggested to use prefabricated IoT devices, which would reduce time to assemble an IoT smart home. Another, more recent, paper from Yamin et.al [15] shows that automation could reduce, or even remove, the need for human interaction in White, green and partially red teams. This under the assumption that the system parts are controllable with automation.

Yamin et al. [2] did a survey on Cyber ranges and security test beds, looking into over 100 publications from 2002 to 2018. Requirements for a Cyber Range are also discussed. The paper developed a taxonomy for Cyber ranges shown in fig.2.1

While developing the Cyber Range KYPO [16], Vykopal et al. also created a list of requirements. KYPO was however not initially designed to handle IoT devices, especially not with a hybrid approach.

Industrial IoT, IIoT, is in focus in the paper discussing design of a cyber-physical cyber range by Kavallieratos et al. [17]. They suggest a reference architecture for Cyber-Physical-Systems (CPS) with a hybrid approach and also do an assessment of existing testbed features to a list of expected features in a testbed developed for security research.

A more recent survey in Cyber Ranges and testbeds is conducted by Chouliaras

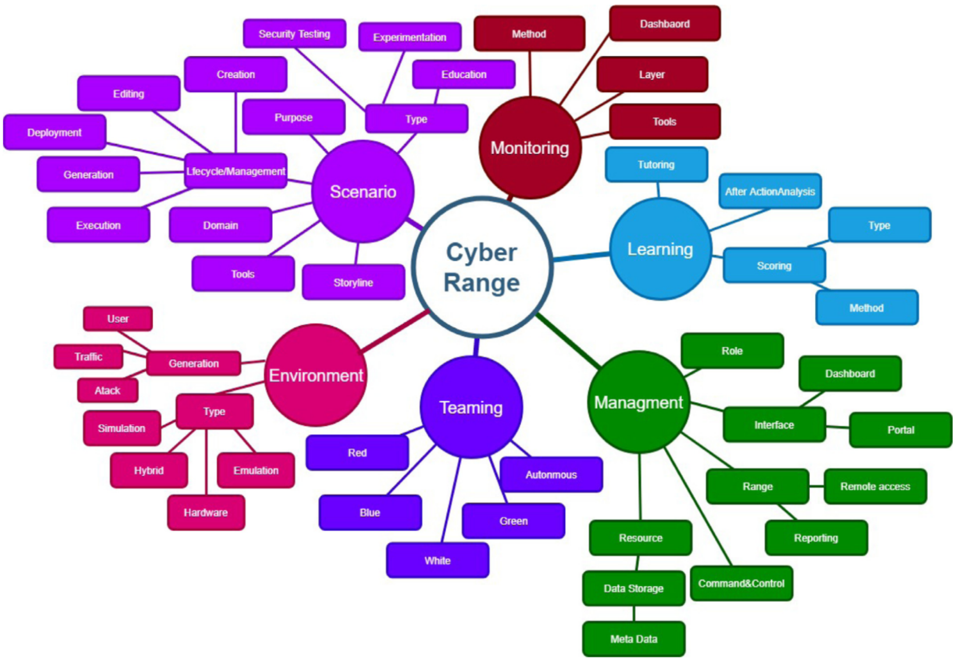


Figure 2.1: Cyber range taxonomy by Yamin et.al [2]

et al. [18]. The authors survey 10 developments and also did interviews with the systems owners to give insights in modern cyber ranges. The paper gives a good insight in what is the state-of-art within the field.

Al-Hawawreh et al. [19] developed a design and implementation of an IIoT testbed for security researchers. The testbed, called Brown-IIoTbed, is a hybrid testbed with virtual servers and physical installations. The IoT devices used in the testbed are used as industrial devices, although not all devices are industrial grade. The approach has relevance for this project as it contains a physical dimension and has several IoT communication channels. However, this testbed design does not have a provisioning focus, nor is it designed as a Cyber Range. Another testbed for industrial IoT is proposed by Lee et al. [20]. As for the Brown-IIoTbed it is not designed as a cyber range, nor does it consider provisioning and automation challenges. While it is a testbed for IoT devices, the test cases could to a large extent be solved in a testbed for conventional network devices.

The "poor man's IoT testbed" by Muños et.al [21] shows a flexible and scalable approach of a testbed using reasonable priced off-the-shelf products to create a test bed with IoT devices and IoT communication protocols. This testbed is a physical installation with a Raspberry Pi as a controller communicating with 4 remote micro-controllers with their sensors, nicely installed in a glass dome. The design could be adapted, at least partially, to a cyber range, but lacks provisioning and automation. It lacks functionality to be a cyber range, and the implementation is physical only.

A cyber range called IoT-CR [22] is a hybrid cyber range with 20 Zolertia devices called RE-MOTE, virtual devices, a resource engine and a front-end engine. The physical devices are controlled by the server via USB interfaces and the virtual devices using are Contiki-NG embedded RTOS executed on top of Cooja. The paper describes a demonstration of a autonomous scenario called "Pass the token".

ForCyRange [23] is an educational IoT Cyber Range for forensics with a design of four blocks: An IoT system simulation, a middleware/database, a learning management system and a forensics workstation. The forensiscs workstation is set up with forensics tool set and visual decision support tool. This cyber range is developed for live digital forensics and training people for those tasks. In IoT forensics is based on digital forensics, and within IoT forensics there are three zones: internal network, middle network and outside/external network. The nature of IoT is a set of several digital forensics sub-types: cloud forensics, network forensics and device forensics.

A Raspberry Pi Cyber Range was created to teach web-attacks in [24]. Raspberry Pi is a mini, single-board-computer often used in IoT projects, because of the easy to use General-Purpose-IO (GPIO) interfaces. While the Cyber Range is not IoT targeted, it uses IoT devices to create an Cyber Range and is able to scale the solution by adding and removing Raspberry Pi’s from the cluster. The project uses low cost cyber range to do security training.

CyberIoT [25]is a conceptual design of a Cyber Range for IoT. However the design is unclear on how the IoT devices interact with the Cyber Range. It is assumed that the IoT devices are emulated. Emulating is not always the best solution to do IoT training.

Cyber ranges have an element of test bed as defined by Schwab and Kline [11]. Several test beds are developed for many purposes in IoT e.g. A network testbed, Testbed@TWISC [26], test bed for eHealth [27], SCADA test bed [28], IoT automated test bed [29] and many more. Testbeds however lack the complete network and administrative infrastructure that Cyber Ranges have to be used as infrastructures for e.g. cyber exercises.

A testbed framework for wearable IoT devices [30] have a design requirement for Data Forensics Analysis where data extraction should be done by side channels like USB and JTAG, by e.g. using ADB. Another requirement is to have an array of simulations to manipulate the IoT sensors, e.g. GPS simulator.

Chapter 3

Method

Empirical research tries to describe and explain. Design research is in addition also interested in changing something in the world. Design research develops artifacts to improve something. Design science is a design research framework developed and used in the information technology world. It is a methodology that can contain several other methods in each of the activities in the framework.

The methodology used for this project is Design Science Research [31]. Design science will in addition to design something also bring new knowledge, use existing and well-accepted scientific knowledge and the knowledge must also made available to other researchers. In design science research an artifact must be defined and this artifact is handled through some defined activities. IT artifacts are broadly defined as constructs, models, methods or instantiations. Artifacts can be in one or more categories.

The five main activities in Design Science Research, as shown in figure 3.1, are:

- Explicate problem: investigate and analyse a practical problem and why is it important
- Define requirements: outline a solution to the problem and elicit requirements
- Design and develop an artifact: fulfill the requirements
- Demonstrate artifact: prove feasibility in one case
- Evaluate artifact: how well do the artifact solve the explicated problem and fulfil the defined requirements

Each activity has an input, from the previous activity, an output, to the next activity, and an activity will also consume controls and resources. Controls in form of strategies and methods, and resources in form om knowledge and workload.

In the **explicate problem** activity there are three sub-activities:

- Define precisely
- Position and justify
- Find root causes

To explicate problem, a problem description needs to be precise and justified. The problem should to be of general interest, while looking into the underlying causes for the problem. For that a literature search for what already was designed in IoT Cyber Ranges and security test beds was necessary. The project starts with a literature search of what is the state-of-art published work in IoT Cyber Range or IoT Security testbed development and designs. Papers were searched for in NTNU Oria, Google Scholar and IEEE Explore. Search strings included IoT, IIoT, industry 4.0, smart home, cyber range, test bed or testbed. It is relevant to study the "future work" chapters in all papers. Studying all papers is interesting to find what the needs for an IoT cyber range, and whether there are solutions for automating and if not, is there room for improving a cyber range setup with more automation. Some cyber range surveys, without IoT focus, were studied, to get more foundation on what the trends in the cyber range, regardless on what they are designed for, communities are. Existing cyber range surveys also show what cyber ranges are used for and the benefits of cyber ranges. The surveys of cyber ranges and test beds also guided towards the existing cyber ranges and test beds in the IoT, and especially smart home, domain.

Since published papers are often published some time after they are written, it is challenging to have an overview of what are the State-of-Art projects and the challenges/needs of, or what is missing in, those projects. Since this project is to develop av cyber range design for IoT, an approach to get the most current best practices and their practical challenges was desired. To get a better understanding of the challenges within the domain a questionnaire was developed for surveying in the Cyber Range, and especially IoT Cyber Range, community. The recipients were selected within the Cyber Range research and users community in Europe, these people were also requested to forward the questionnaire to others that may have contributions to the project. The recipients are selected from the authors in the literature review and also suggested contacts within the supervisors network. The problem is described in the introduction chapter.

In design science, **define the requirements** activity has two sub-activities:

- outline the artifact
- elicit the requirements

The artifact in this project is a model of a IoT Cyber Range, as well a partial instance of the same Cyber Range. As the time for this project is limited, finding automation solutions is prioritized.

To define the requirements for a hybrid IoT cyber range the results from a literature search and the earlier mentioned questionnaire helped in creating the requirements for this project.The requirements are discussed in chapter 4.1. The same papers as in the previous activity are used for base to define the requirements for an IoT Cyber Range. The results from the questionnaire survey were also expected to contribute to a larger extent than it actually did. Requirements are summarized in an qualitative analysis of the information from the literature and the questionnaire.

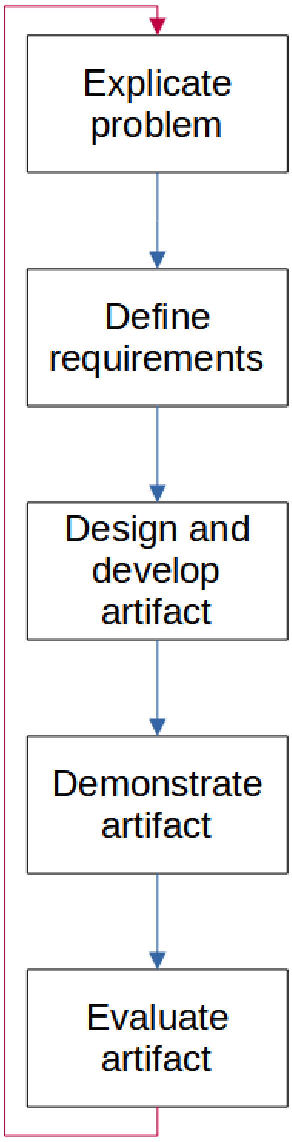


Figure 3.1: Activities in Design Science Research

Design and develop activity in design science has 4 sub-activities:

- Imagine and brainstorm
- Assess and select
- Sketch and build
- Justify and reflect

To design and develop a hybrid IoT Cyber Range, surveys in the Cyber Range domains shows the solutions chosen by others when designing a cyber range. This helps in getting a feel of what technology is available, how others assessed the technology and choosing the best solution for this project. As much of the technology is new for this projects author, a large amount of time is spent on learning the pros and cons of the technical and practical solutions that emerge from the surveys. Learning the capabilities of the technology should give ideas on how the different solutions can be used to solve an IoT Cyber Range. After having a good grasp on the capabilities of the technologies studied, both those used by others and others found by myself, a design should emerge when choosing those functions that seem to fit the best. All sub-activities in design and develop are discussed in the design and implementation chapter. All sub-activities are to some extent overlapping in those sections.

Demonstrating as a main acitivty has two sub-activities:

- Choose and design case
- Apply artifact

Demonstrating the Cyber Range shows a proof-of-concept from the developed solution and showing that the problem described has a feasible solution. This activity includes implementing known attacks for IoT devices and showing that the artifact can be useful. Although Johannesson and Perjons [31] state that the artifact should be demonstrated for one case, this project tests the cyber range for two test cases.

Evaluation has three sub-activies:

- Analyze context
- Select goal and strategy
- Carry out evaluation

The main goal of evaluation is to determine whether the the artifact solved the defined problem, and how well it solves the problem.

The evaluation must show how well the end product covers the requirements set earlier and how well we solve the problems described. An evaluated artifact of a DSR process could be improved as a new problem description, thus the methodology can be an iterative process, a cycle, as shown in the red line in figure 3.1. For this project the DSR activities are only performed in one sequence. The Cyber Range is evaluated to the requirements found in the survey as well as showing the gain in resource usage preparing the scenarios for dry-runs, execution and reruns. The artifact is to be evaluated in the discussion chapter, where the evaluation context, goals and strategy is described, before the evaluation is done.

Chapter 4

Results

This chapter discusses the requirements for a hybrid IoT Cyber Range in the requirements section, suggests a design in the design section, describes a possible implementation as well as demonstrates use cases in the implementation section. The requirements section first shows the requirements found from the literature search, then the results from the survey, before synthesizing the requirements.

IoT awareness is increased by knowledge [8]. Knowledge is increased in cyber exercises [9]. And several papers support that a Cyber Range provide beneficiary tools for carrying out cyber exercises.

4.1 Requirements

According to Yamin et al. [2] a Cyber Range must cover 8 functional architecture components:

- portal
- management
- training and education module
- testing module
- scenario
- monitoring
- run time environment
- data storage.

Yamin et al. also surveyed the future needs to a more efficient exercise lifecycle management and suggested more automation as solution. For scalability SDN and use of containers are suggested technologies. Federation is also pointed out as a future direction, as well as user interaction simulation to reduce human interaction and thereby increasing efficiency.

Kavallieratos et al. [17] developed a reference arcitecture with a control center module, physical components module, virtual components module, cybersecurity defensive mechanisms module. While Yamin et al. has a more general description Kavallieratos et al. has a more hands on approach mostly in the run time environment component of Yamin's description.

The requirements in the Kavallieratos model are flexibility, scalability, isolation, interoperability, cost-effectivenes, built-in monitoring, easy access, adaptability and shareability. While Yamin does not explicitly state requirements for cyber ranges, some components in the architecture cover the same requiremets. When designing KYPO [16], the authors had some important requirements: flexibility, scalability, isolation vs. interoperability, cost-effectiveness, built-in monitoring, easy access, service-based access and open source, which to some degree are the same requirements as Kavallieratos has described.

The tier approach from AL-Hawawreh and Sitnikova [19] with a edge tier, platform tier and enterprise tier can give a new dimension for a IoT Cyber Range design. They suggest 9 features for comparison with other test beds in their implementation of the Security Testbed for IIoT:

- usability
- fidelity
- heterogeneity
- flexibility and scalability
- federation
- safety, reliability and resilience
- user interfacing
- end-to-end testbed

Ukwandu et al. [32] survey 44 Cyber Ranges as well as many cyber attacks used in scenarios. They also show some future trends, technologies and uses of Cyber Ranges which are relevant to look into while designing a Cyber Range. The trends they list are: real-time auto-configurable systems, smart, mobile and integrated technologies and training with augmented reality. The technology trends they list are 5G/6G technologies and more containerisation. The application area trends in the paper are pointing towards smart CPS, smart cities and industry 4.0 and lastly aerospace and satellite industries. Ukwandu et al. also suggested a taxonomy for Cyber Ranges not very different from the taxonomy from Yamin et al.

Siboni et al [30] built an IoT testbed for wearable IoT devices. During design they listed several requirements for an IoT testbed:

- should be able to handle a wide variety of devices from different categories
- should also be able to emulate different testing environments
- should support Security tests, and the paper suggested 13 different security tests to support.
- should simulate actuation's and signaling
- should support of the most common wireless and wire communication channels
- should be able to process and analyze relevant communication protocols
- should be able to extract data from the tested devices, preferably on side channels
- should support management and report mechanisms to control and and

manage test flows
- should support user intervention and automation capabilities
- should be plug-able

4.1.1 Results from questionnaire

The questionnaire, mentioned before, that was distributed to a selected recipients, resulted in a low response count. The 22 recipients are from different universities, research organizations and Cyber Ranges in Europe and totally 4 persons responded answering the questions. One other recipient responded that he/she could not make time to answer the survey. The questionnaire was designed as a multiple choice questions and with the alternatives were chosen after studying literature on cyber ranges and testbeds, both IoT and others. Some questions also were open answer questions. This section describes the questions in the survey as well as the results from the questions. The questions were designed so the respondents would not have to reveal any secrets about their own implementation.

First question asked what type of cyber range the respondent has: virtual, physical or hybrid. 3 of the respondents have a hybrid(virtual in combination with physical) cyber range, 1 is virtual only. The second question was about what category of IoT devices the cyber range supported. Two of them support smart homes, two support smart health, two support smart transportation, two support smart grids and two support IIoT/smart industry. Only one support smart cities. Third question was the usage of the cyber range: The use of the cyber ranges are IRT training, penetration testing and forensics training. The open answer questions about what the functions and interfaces were unanswered.

The next questions were about the technology used in the cyber ranges. The technology used for orchestration is mostly self developed, one uses OpenStack Heat. For management of the cyber range 3 of them use self-developed tools. One uses OpenStack and another uses vCenter in combination with self-developed tools. One flaw in the suggestions in the published questionnaire was that Ansible was not listed as an option, though it probably wouldn't make much of a difference.

Defining scenarios are for 2 of the respondents done in JSON format, and another uses self-developed tools for this.

Emulations are done in Virtual Box, OpenStack, VMWare and in self developed tools. 2 respondents use VirtualBox, while one use VMWare in combination with self-developed tool.

Two of the respondents use tools for simulation, one has a self-developed tool and the other uses OPNET.

For monitoring one uses ELK in combination with WAZUH. WAZUH was a new suggestion to the original listing in the questionnaire. Another combines Wireshark with tcpdump, the third combines ELK, Bro/Zeek, Snort, Wireshark and tcpdump. In retrospect Tshark should be included as an option, but can fall into the Wireshark selection.

Also for data representation JSON is the preferred tool for 3 of the respondents, the fourth uses a self-developed solution. As the project is for IoT the questions should have been more precise that the data representation is for the IoT and physical world representation.
Creating data is, as the previous point about data representation, meant for the simulation of real world and IoT data. The question might not make this clear enough. 3 of the respondent's answer self-developed tools for this and the fourth answer "Klai linux" in the "other" field. This is most likely to be a typing error and meaning Kali Linux.
The comments from the respondents in the question of what the top three functional requirements are for IoT Cyber ranges that differ from conventional/general cyber range designs are:

- mapping digital twin I/O on low level
- cross-platform deterministic human interaction
- integration of advanced techniques such that steganography detection and exercising

The main challenges of cyber range and security testbed implementations, especially IoT focus, are

- diversity of the IoT components
- low-level performance in SW implementation vs high-level portability

When asking for the challenges generating sample data, both benign and malicious there was only one answer: "Cross-platform adaptation" Only one answer came for the question about own IoT Cyber Range drawback is "AI utilization".
The one suggestion on how to solve challenges and functional needs was: "adopt the changes in functional requirements whenever possible".

The questionnaire form, as sent, is available in the appendix.

4.1.2 Requirements summary

Analysing the information of requirements from the literature review and the results from the questionnaire result in these design requirement categorised for this Cyber Range:

- flexibility, scalability, adaptability, interoperability
- shareability, open source
- fidelity
- isolation, safety, resilience, reliability
- cost-effectiveness
- built-in monitoring
- easy access, usability, user interfacing
- service-based access
- heterogenity, handle diversity in IoT, emulating digital twin
- end-to-end testbed

Flexibility and scalability is important to create scenarios as close as possible to real life implementations. The cyber range should give training that resembles real life environments whether training for large or small environments. Adaptability and interoperability are requirements to the system to be able to change with minimal efforts, either to integrate with other systems or to change the internal functions in managing the cyber range or within the scenarios.

As Vykopal et al. state for the KYPO implementation, the platform should reuse suitable open source projects and release its artifacts under open source licence. Use of existing open source projects gives others opportunities to use and reuse this projects artifacts. This is important to make the training for security as manageable as possible for as many as possible, in common available and affordable tools. Others refer to this requirement as shareability.

Fidelity requirement refers to follow standards. Until recently there were not developed standards for cyber ranges. With the work of Yamin et al. and later Ukwandu et al. taxonomies were created. Yamin et al. also developed a list of functional requirements for a Cyber Range. Some standards exist for IoT communication protocols, but the payload is often up to the developer. Zigbee Home Automation Profile is one example of existing standard for communication at an application level. Seljeseth et al. [33] also suggested data formatting framework for IoT devices, UIoT:FMT, the paper also addresses some of the challenges of standardization within IoT.

Isolation, safety, reliabilty and resilience are requirements so that the cyber range it self is not vulnerable to anything happening in the scenario. Also vulnerabilities and malware in the scenarios should not gain access to other networks than intended, e.g Internet. The cyber range infrastructure should not lose any management functions due to resource depleting in the training area, e.g. DoS attacks or process intensive malware and such. When dealing with wireless signals, [30] also isolated the wireless signaling from the surroundings with a special built, shielded room.

IoT is emerging quickly, one of the reasons is that IoT devices are affordable to even normal households. A cyber range should also be cost-effective to give as many as possible the possibility to use the benefits of a cyber range. Using openly available software is one measure to achieve this requirement, emulation is another possibility (with the before mentioned challenges), using devices that require little modification could reduce the time to get the environment set up. Automating tasks that can be automated will reduce the time to get the range ready for exercise, and even reduce total labour time in setting it up. Automating and automatically controlling the physical devices within the exercise zone is a large benefit in reducing labour, and thereby increasing the cost-effectiveness.

Service-based-access requirement is to some extend reasoned for reducing costs and some for providing a cyber range as a service to reduce the needs for implementing many ranges. The interoperability requirement is about consuming services from providers, while this requirement is about providing services.

Since the IoT device implementation diversity is large, the protocols are many and the lack of standards or the lack of will to follow standards the cyber range should require heterogeneity. The system should be able to emulate a large collection of devices as well as be able to support many types of protocols.

Al-Hawawreh and Sitnikova developed an end-to-end test bed and used this feature as comparison. A cyber range should be able to provide the end-to-end communication for IoT devices. Many IoT devices communicate directly to a service provider in the cloud and is managed through external providers. Implementing service like Home Assistant[1] simulate to some extent the service provider of IoT management from a user perspective where the user can operate his or her smart home through a web interface or through an mobile device app.

4.2 Technology

A design depends on the technology available. One of the requirements is to use open source software and to be cost effective. This section discusses the technologies that make the design feasible and some of the challenges that the design has to overcome.

4.2.1 Emulation

Having every possible IoT device available for training or testing is not feasible. The diversity in devices is too large. Running IoT firmware outside its hardware environment requires emulation. Emulating an IoT device seems to be a possible solution to reduce storage and costs. Once developed a virtual infrastructure with digital models [34] of the IoT devices, the devices can be shared with others.

The benefits of emulation are many. Enquiring a large stock of devices require storage space, is expensive and a lot of effort, especially when devices are end-of-production. Running device emulation, with several devices on a commodity computer saves all that and computing power. The benefits are also environmental as the devices at some point has to be disposed.
It is a challenge to emulate a large specter of IoT devices. While the IoT devices are all microcontrollers and computers, they have a large diversity in architecture and composisions. Even within the ARM architecture there are 19 product families, which ARM themselves produce, although lately only Coretex families are developed. Qualcomm also develop CPUs using the ARM instruction set, and several other vendors. To make things even more complicated, the CPUs are often embedded into SoCs (System on Chip). SoCs or CPUs are often part of a system with integrated functions on one board or even with peripherals. Some multipurpose device examples are Arduino boards or Raspberry Pi boards. Also when emulating IoT devices, one must be aware that physical IoT devices have more constraints than emulated devices, e.g. limited processing power, limited power supply, etc.

To emulate a complete system, it is not sufficient to emulate a CPU, all connected peripherals must be emulated. According to Zaddach et.al [35] there are

[1] https://www.home-assistant.io/

three possible emulation solutions: complete hardware emulation, hardware over-approximation and firmware adaptation. Zaddach et.al proposed a system called Avatar^2 where the CPU is emulated with QEMU and JTAG is used to communicate with the actual hardware.

While there are many emulators for each specific architecture, e.g. for AVR: Avora, AVRS, simavr, SimulAVR, atemu, GNU AVR simulator, IMAVR and probably many more, some emulators support more architectures. Most solutions in the literature use QEMU as emulation platform, QEMU has support for ARM, AVR, x86, RISC, SPARC and more architectures and already has support for many SoCs and boards. QEMU is also a hypervisor supported by OpenStack.

4.2.2 Simulation

As IoT devices main task is to represent something in the real world, a digital model of the IoT device is not connected to a real world sensor. This information need either to be simulated with random data, information from a data model or remotely connected to a real world sensor.

4.2.3 IoT communication interfaces

IoT devices can be sensors and actuators installed in locations where there is not desirable to install cables, either because it would be not feasible to install or it is easier or more cost-efficient not to install. Thus IoT devices often have to rely on wireless communication to own control systems. And since the devices are not cabled and need power, the devices must rely on batteries. As the power supplies are constrained there is a wish to reduce power consumption and therefore some protocols are developed for constrained devices. There are wireless protocols like Z-Wave, Zigbee, Bluetooth Low Energy (BLE), LoRaWAN etc. Some protocols are long range and some for short range. Where power supply is not a problem IoT devices often use WiFi since the infrastructure already exist and also reduces the complexity for the user. Where network cables exists IoT devices can also get the power supply through PoE, thus wireless and power challenges are not an issue. Network vulnerabilities still can exist.

The SoCs on the IoT devices often come with some capabilties on the chip. The chip can implement functions as UART, JTAG, SPI and SWD.
Universal Asynchronous Reciever-Transmitter (UART) is also often referred to as serial port. The UART is a hardware device for sending and receiving serial data over 3 wires, one for transmitting (TX), one for receiving (RX) and reference level/ground (GND). Often there is also a fourth wire with constant reference voltage (VCC) or as power supply. What the UART is used for depends on the firmware of the device, often the UART communicates what the console would do if there was a terminal present.

JTAG is an industry standard for verifying and testing PCBs after manufacture. JTAG is named after the Joint Test Action Group which developed the standard. JTAG is a debugging port on a PCB with serial communication to Test Ac-

cess Port(TAP) on the each chip. JTAG provides a possibility to debug CPU's at a machine instruction level, like stopping execution and reading registers. JTAG also allows reading and writing firmware on the non volatile memory, like flash memory. One of the main initial functions of JTAG was boundary scan testing. Boundary scan gives an opportunity to read and set inputs and outputs of each pin on the chip.
The JTAG connector pins are Test Data In (TDI), Test Data Out (TDO), Test Clock (TCK), Test Mode Select(TMS), Test Reset (TRST). The TAPs are daisy chained with TDO on one TAP connected to TDI on the next TAP. The last TAP TDO is connected to the JTAG connectors TDO. This means that when reading the first TAP, the data must pass through all subsequent TAPs before reaching the TDO connector. The TCK speed decides how fast the information flows through the TAPs. The slowest TAP limits the JTAG TCK speed, but typically 10-100MHz is accepted as clock speed [36]. Holding the TMS pin state unchanged also keeps the test mode unchanged. Cycling the TMS changes the test mode on the TAP. The chip can be unaffected by the TAP or be set to behave in a specific way. A System Reset (SRST) pin is also a useful pin that allows resetting the entire system. An alternative specification of JTAG is a reduced pin count JTAG (IEEE1149.7) called cJTAG for compact JTAG. cJTAG has 2 pins with Test clock (TCKC) and Test Serial Data (TMSC) where the TAPs are connected in a star topology.

Serial Wire Debug (SWD) is an also a two pin debugging interface developed by ARM [37]. The SWD pins are Serial Wire Clock (SWCLK) and Serial Wire Input/Output (SWDIO). All SWD operation sequences consist of two or three phases: Packet request, Acknowledge response and possibly a Data transfer phase. The SWD, like JTAG, can give access to registers, memory and internal buses.

SPI is a synchronous serial communication designed by Motorola in 1979. SPI uses a master-slave architecture and is capable of full duplex. Multiple slave devices are possible through a slave-select (SS) wire. E.g. Arduino Pro Mini (Atmega 328P) is possible to program through In-System-Programming (ISP/ICSP) via SPI connections as it is embedded in AVR SoCs.

IoT wireless

IoT devices often use wireless communication. There are several standards for wireless communication. As sensors often are placed in locations without access to constant electricity, devices are powered with batteries. Wireless communication in combination with limited power supply do create some challenges. The amount of data sent over the air must be reduced to reduce power consumption, the amount of awake time is often also reduced. Some manufacturers use proprietary protocols for communication, however as the IoT wireless standards get more mature the developers chose to use the already developed standards to reduce costs and development time. Many microchip vendors also integrate protocols into their products and can offer a single System-on-Chip (SoC). Some wireless protocols used in IoT are standards like Wi-Fi, Zigbee, Z-wave, Bluetooth and many proprietary protocols modulated on top of 433MHz or IEEE 802.15-4.

Zigbee is a well known wireless specification of protocols built on top of IEEE 802.15-4 specification. IEEE 802.15-4 specifies wireless communication for LR-WPAN (Low rate wireless personal area network) and is also the basis for 6LoWPAN. As the name LR-WPAN indicates, the data amount is small, the network is wireless and the scope is at personal area level. Zigbee is built for low power consumption, relatively large networks and to utilize other nodes in the same network to extend the range of the network. Zigbee security is criticised as the network registering process in the basic security model require the network key to be transferred unencrypted over the air. The network key is necessary to join the network and must be shared by all devices. Another layer of security is the link key, which encrypts the traffic before the devices acquires the network key. There is a default global trust center link key defined by the Zigbee Aliance, which normally all new devices know of. The default key is "5A6967426565416C6C69616E63653039". A third key is the master key, which is a key that encrypts traffic between two nodes, and it is a long time key between the devices that is established in a key exchange phase between the devices. A Zigbee device can have the role of a coordinator, which is in charge of the network, a router, which can be an end device, but is also responsible to forward traffic, or an end device, which has no obligations but to it self. A router can not sleep as it must route traffic between end devices and the coordinator, or possibly between end devices and routers, or between routers and routers. Routers and coordinators should not be powered by batteries as they can not sleep to save power.

4.2.4 Linux based IoT boot process

On embedded platforms the boot process is bit different from standard x86 computers. When the devices is powered up, the SoC reads the internal ROM code. The ROM code is hardcoded into the processor. The ROM code instructs the processor to load the next stage, the first stage bootloader. The first stage boot loader does some basic hardware initialization, e.g. memory, and loads the second stage boot loader. The second stage boot loader sets up media like NAND, sdcard, network, file systems, sets up the root file system and start to load the kernel. The kernel handles the rest of the peripherals, does memory and process management and mounts the root file system. Both the first and the second stage boot loaders can be overwritten, as well as the kernel and the root file system. The operating system in the Linux embedded devices is loaded to memory and all changes in the root file system are done in the memory. The devices rarely have functions to write changes done in the file system back to the flash storage, and after rebooting the changes are lost. This is except the configuration changes done to the devices, which is written to a different memory region, and is also the area that is wiped during a reset to factory default settings.

On Raspberry Pi the ROM code is called stage 1, this stage is executed on the GPU. The GPU loads the stage 2, bootcode.bin, from the SD card into the L2 cache.

The stage 2 enables the SDRAM and loads boot stage 3, loader.bin into RAM and runs it. The loader then loads the start.elf to to boot the operating system. Newer version of Pi skips the 3. stage, the bootcode.bin loads the start.elf.

4.3 Design

This design seeks to fulfill the requirements for a complete Cyber Range. The architecture components from Yamin et al. (as discussed before: portal, run-time environment, management, training and education, testing, scenario, monitoring and data storage), are suitable to cover the components used in Cyber Ranges that are mentioned in the previous chapters. As Yamin et al. suggest a unified architecture, and the architecture fits, it is wise to use it for the project. There are some components that can be general for all types of Cyber Ranges, no matter the purpose. As this project focuses on a hybrid IoT solution there are components that must be specific. The components differing from a general Cyber Range to an hybrid IoT Cyber Range are the run-time environment, monitoring and scenario. This chapter discusses how this design is solved in the various components.

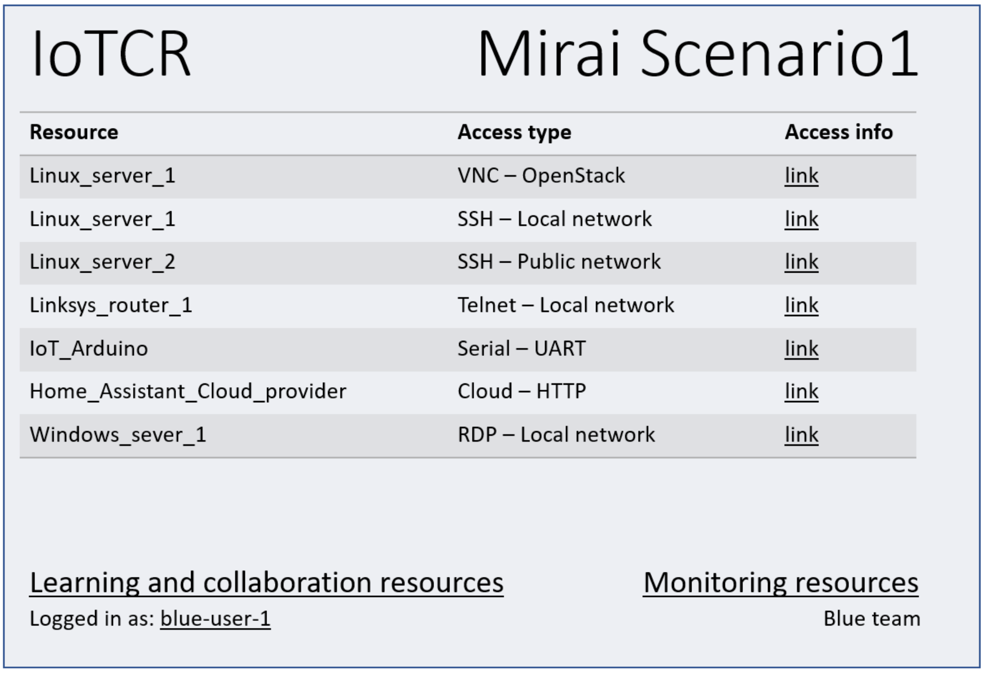
4.3.1 Portal

A portal should give the users in all teams the tools to administer and use the cyber range. This project does not focus on portal, but acknowledges the need for a portal to access the resources needed to be a part of a team in the cyber range. These resources could be scenario provisioning, credentials, network access for management, computer console access for the users etc. The portal also give access further to the network, like management interfaces of the cloud solution and console access to the management servers, access to learning platforms etc. The portal should handle authorization and authentication of the users. The requirements in the previous chapter: Easy access, usability and the possibility for user interfacing should be to a large extent be covered by the portal component. A suggested portal view for one of the exercise participants is shown in figure 4.1. E.g. OpenStack provides console access to instances over VNC.

4.3.2 Training and education

Training and education module should give learning materials to the users, this function should also cover the possibility to give grades and feedback while learning. **Moodle**[2] is an free and open source learning management system with course administration. Moodle is php-based, modular and has a wide variety of plugins. Moodle could serve as a portal for the cyber range and be an identity provider for the other services within the cyber range as well.

[2] https://moodle.org/



IoTCR Mirai Scenario1

Resource	Access type	Access info
Linux_server_1	VNC – OpenStack	link
Linux_server_1	SSH – Local network	link
Linux_server_2	SSH – Public network	link
Linksys_router_1	Telnet – Local network	link
IoT_Arduino	Serial – UART	link
Home_Assistant_Cloud_provider	Cloud – HTTP	link
Windows_sever_1	RDP – Local network	link

Learning and collaboration resources

Monitoring resources

Logged in as: blue-user-1

Blue team

Figure 4.1: Portal view example

4.3.3 Run time environment

This component is the main focus of this project. The design should able to both emulate and run actual hardware. The goal is to automate the provisioning and orchestration as much as possible. The provisioning is to be done in the virtual and physical platform. **Ansible**[3] is an open source software for automation and provisioning, configuring and installation tool. Ansible supports Linux, Windows and Mac. It is an agent-less software, using remote connections to the target for running commands. Ansible is used as the base software for the provisioning and Ansible supports and can control many other solutions. Ansible controls the orchestration of the cloud and on-site system. For provisioning the system Ansible controls Terraform which in turn manages the OpenStack functions.
Terraform[4] is an open source orchestration, infrastructure as a code software developed by Hashicorp. Terraform support several cloud platforms like AWS, Google Cloud, Azure and OpenStack. Terraform provisioning is done by declaring the infrastructure and its providers through HashiCorp Declaration Language. The definitions are generated by Ansible with templates bases created by the cyber range administrators. Terraform does much as OpenStack Heat also does, but Terraform has plugins for many more systems.
OpenStack[5] is a open source cloud computing platform for managing private and

[3] https://www.ansible.com/
[4] https://www.terraform.io/
[5] https://www.openstack.org/

public clouds. OpenStack has modular design providing many components. OpenStack Ironic is a component for bare metal provisioning of physical hardware as opposed to virtual machines. For this project an OpenStack cloud is used, however since we also need a hardware platform then OpenStack Ironic is used. The cloud solution used in this project is managed by an external party, so integration with Ironic is not an option, also to be able to have flexibility to use the solution with other cloud providers it was chosen to install a separate Ironic platform and orchestrate this through Ansible and Terraform. An cloud installation on an existing cloud is referred to as a cloud-in-cloud solution.

Linking the virtual world with the physical installation should be fast, reliable and secure. Wireguard[6] is a free open-source VPN software which shows good results[38]. Wireguard is designed to be easy to use, have good performance, reduce overhead and reduce attack surfaces. Wireguard communicates over UDP.

Networking

A Cyber Range must have one or more networks to exercise in. As stated in [2] teaming is an important part of a cyber range, and the teams in training must have their own network(s) to defend or to attack from. Depending on the needs for the scenario and the what and how the teams organize, some possible network needs are: Internet, red team network, blue team network, provisioning and management networks. Provisioning network is for provisioning the devices over network, management network is for administering the devices as well as giving the automation processes access to the infrastructure nodes. Networks need to be separated to cover the isolation requirement and to create realism; to ensure separation, VLANs are used. Orchestration of network is possible through OpenStack network component, Neutron.

Wireguard is fast and secure, and reliable considering the limitations. The VPN is established over Internet and Wireguard uses UDP, UDP does not guarantee delivery, and Internet can be an unstable network. However to ensure delivery a tunneling protocol is used. The tunneling protocol also ensures delivery VLAN to the right network. A VLAN in the cloud should also be present in the physical installation. To tunnel network traffic from the cloud, an **Open vSwitch**[7] is installed in the cloud and another on-premise. Those Open vSwitch installations are then connected over Wireguard with VXLAN protocol. VXLAN packages has a disadvantage that it can not be fragmented, therefore we must ensure that all packages are within Maximum transmission Unit (MTU). The MTU from Ethernet standard is 1500 bytes without using Jumbo Frames. As our transmission is over Internet without MTU guaranties we must ensure that packets are within MTUs. VXLAN has a 50 byte overhead and Wireguard has some overhead, 80 bytes for IPv6 and 60 bytes for IPv4. We must ensure that when network nodes transfer that they don't exceed the MTU boundaries. The network adapters in the solution

[6] https://www.wireguard.com/

[7] `https://www.openvswitch.org/`

must obey lower MTU requirements, these settings can often be set in the DHCP server. The use of the protocol GRE in stead of VXLAN might solve some of these fragmentation and MTU issues. Still the issue of 1500 byte MTU limit on the network out of our control may exist, fragmenting and assembling frames require processing time and power, the optimal solution is to keep the MTU value below the threshold for fragmentation.

Monitoring most of the network traffic in the Cyber Range is possible through the Open vSwitch as it is a central component forwarding traffic between the physical and virtual environment.

OpenStack Ironic has plugins to manage physical switches as well as virtual switches. Provisioning, running, managing and cleaning can be done with the support of Ironic.

Emulating IoT devices

One possibility for emulating IoT devices is through **FirmAE** [39]. FirmAE shows high degree successful emulation even without creating and adapting the hardware peripheral emulation. FirmAE is based on Firmadyne [40] and is designed for Linux embedded devices. Support for other IoT device emulation is possible on QEMU, however hardware emulation is challenging. Feng et.al [41] suggests a solution that looks promising for emulating purposes, but complete system emulation often require developing the peripheral components outside the CPU like Osman [42] did in his thesis. Osman's solution could be used to transfer input and output signals from the emulated world to the physical, but the delay must be accepted. Some SoC and board implementations exist, but the emulation development can not cope with the speed that developers create new hardware.
QEMU is the emulator used by most of the literature studied in this project, however it has shortcomings as stated earlier. Emulating the Atmega 328 board used in this project completely was not possible since the 8-bit timer for AVR boards was not developed in QEMU. This resulted in use of timing functions, like measuring time and delay, would not work. The firmware would run, but tests based on time measurements would fail and the program would not behave as expected. Many other emulations, also in FirmAE, will run, but behaviour can be unpredictable. To have exact behaviour in emulation, all peripheral functions need to be developed and it does not seem feasible. Future developments in AI will possibly overcome these challenges, as also stated from one of the respondents in the survey, but this is out of scope for this project.

Managing physical components

Provisioning the physical IoT devices for an exercise require a different approach. During an exercise a devices can be altered or die. For emulated devices this is probably not a big challenge as a fresh image can be restarted. Many IoT devices and their architectures have implemented some debugging capabilities. Serial UART, JTAG, SWD and SPI are some capabilities often used in rescuing bricked devices. Serial interfacing gives a large flexibility when the system installed has tools and commands to do tasks. An optimal solution would be to re-flash every devices to make them ready for a new exercise. There are some challenges with flashing devices, NAND flash technology devices wear out with several new writes. Transferring images to flash over UART can be time consuming.

Generally one has to be careful connecting electrical interfaces to others. Voltage differences creates electrical currents which can damage equipment. In general voltage level shifter should be used, but as the components used in this installation all had 3,3V interfaces this was not necessary. However, electrical separating the devices reduces the chances of propagating damage if one device has failure. How one device is reset, is depending on the device. Normally all devices have a reset button to go back to factory defaults, but if the factory defaults are compromised it has no value. The reset buttons are often connected to an input on the controller, and is therefore dependent of the system to be running the firmware that handles the button press. Resetting through UART interface requires the device to be ready for receiving and processing commands. At a more low level, if using JTAG, JTAG needs to have the correct capabilites and to be enabled. JTAG can be disabled by the manufacturer in many different ways: it can be disabled in software, only available during a specific time e.g. during boot or fuses deliberately blown after production test to disable JTAG or SWD functions. SoC with multifunction pins can also combine JTAG pins with other pins and that the selection is done e.g. during boot.

To handle hardware controlling scripting in Python for controlling the GPIO interfaces in Raspberry Pi was chosen. Raspberry Pi is an affordable one-board computer with very flexible use.
To make the Raspberry pi provisioning automated and fast the Raspberry Pi has to be reprogrammed to boot from network. For this pipxe[8] is used to give Ironic better control when booting the Raspberry Pi. Neutron has to control the network to switch the network to provisioning during boot. Controlling the power cycle of the devices is possible through Ironic power control, but for this project those features were not available. One could use power control through Raspberry pi GPIO ports and relays and IPMI scripts and interfaces like diy-ipmi [43]. For this project the use of consumer laptops no managing interfaces on the computer is available. When using relay managed power through e.g SNMP[9] supplies the batteries of the laptops need to be removed to ensure that the device is powered off before restoring the power.
Raspberry pi has only one hardware UART compatible port through GPIO pins 14 and 15. To handle this limitation for several serial connections it is possible to use other GPIO pins for software serial communication, also called bit banging. As the GPIO ports aren't that fast, and don't provide hardware buffering, it is a challenge to communicate with other devices that have preset communication speed

[8] https://github.com/ipxe/pipxe

[9] https://docs.openstack.org/ironic/latest/admin/drivers/snmp.html

that exceeds the speed of the Raspberry Pi. The soft_uart[10] project recommends speeds below 4 800 bps which is very low considering that console communication speed through e.g. Cisco and Linksys routers is at 115 200 bps. Using a real-time operating system (RTOS) could possibly reduce the challenges of timing with bit banging, but it is not tested in this project. Another solution is to add more USB to serial port (RS-232), but the differences in voltage levels must be handled as discussed before with voltage level shifters. RS-232 standard was developed 60 years ago and naturally do not follow to days TTL and CMOS standard voltage. Low level on RS-232 is between -5v and -15v, while high level is between +5v and +15v. RS-232 has a 2v fault tolerance, resulting in accepting low state with voltage lower than -3v and high state with voltage higher than +3v. The circuits used in this project uses 0v as low and 3.3v as high, the voltage difference must be handled to ensure the survival of the devices. USB to UART devices are also available cheaply, the voltage level is specified and are provided for 5v, 3.3v, 1.8v. There even exists USB UART devices with voltage selectors and adaptable voltages and USB devices with several UART channels. The Raspberry Pi can also be extended with more UARTs using the i2c communication channel from the UARTs to Raspberry Pi, these extension boards are a cheap alternative while saving USB ports for some other uses.

Linux Boot loader

Boot loaders of Linux embedded devices might support booting altenative boot loaders uploaded through UART, network or in attached storage. E.g. Das U-boot[11] boot loader can load the image through tftp protocol and write data to memory on the device. Since the boot loader runs before the installed operating system this gives an opportunity to change the system before it is running and also write desired firmware or configuration before booting. To ensure isolation between administration, provisioning and exercise areas, the devices must be moved to a different network segment before restarting, this ensures separation of the devices at the physical layer in the OSI network model. Resetting devices can be accomplished with power controlling a relay(s) through Raspberry Pi GPIO pins. Monitoring and controlling the behaviour of the boot loader can be controlled by scripting the serial communication using the devices UART. Controlling the serial communication can be accomplished with the use of the **expect** library for python and/or bash. Moving the device to the correct networks can be accomplished by moving physical NIC ports on Open vSwitch bridges via OpenStack Neutron, or in telnet/cli to program managed switches. And the managing instances can provide TFTP service to the boot loader.

1. Connect and monitor the serial interface
2. Power off and on the device with relay connected to Raspberry GPIO pin
3. Stop the boot process of the target through serial interface

[10] https://github.com/adrianomarto/soft_uart

[11] https://www.denx.de/wiki/U-Boot/

4. Move the target device to the provisioning network by managing the connected switch.
5. Provide TFTP service to the target, configure IP settings and move the TFTP server to provision network.
6. Load image from TFTP server to the device using serial commands.
7. Boot the target device with the new firmware
8. Move devices to the correct network segments by managing the switches.

4.3.4 Monitoring

Monitoring events and traffic gives valuable information for evaluating an exercise. Information that can be used for creating a timeline of events, timing of events for scores, proof of claims from the participants etc. Events can also be triggers for dynamic events in a scenario, e.g an attack is launched when a user logs into a specific server. The Elastic stack[12] is a combination of Elasticsearch, a search engine, Kibana, a user interface for searching and visualization of data, and Logstash, a log collector. Logs can be collected with Beats from Elastic as well. Collecting logs from the management infrastructure is a straight forward task as the management servers have full access to all resources. The network traffic is also possible to duplicate through the central network components. Collecting information from the exercise nodes(red/blue team) is more challenging. The nodes could be configured to push traffic, but this raises some challenges: during the exercise the nodes configuration change resulting in not pushing information, the traffic can be subject to manipulation or blocking and the nodes receiving the information can be subject to attacks. Pulling information can also raise challenges: the pull access is blocked by participants, the pull itself generates log entries in nodes, traffic managing through firewalls are prone to configuration errors and so forth. The monitoring information should preferably be collected out-of-band of the exercise. The cloud solution gives some opportunities to use shared storage and objects which can be used, but this is not a possibility with the physical devices. When the devices log via the UART, one can use that capability, however the exercise participant might manipulate the devices so that this function is out of play. The manipulation might not be intentional, but inadvertently. Some policies for the cyber range use could solve this challenge to some extent.

4.3.5 Management

Managing the cyber range should be done as a part of the portal, to ease the administrative tasks. The use of OpenStack gives e.g. the possibility to use Horizon as a management tool, logging to a ELK give data presentation via Kibana. As this Cyber Range project is mainly focusing on the run time environment, this part is not implemented.

[12] `https://www.elastic.co/elastic-stack/`

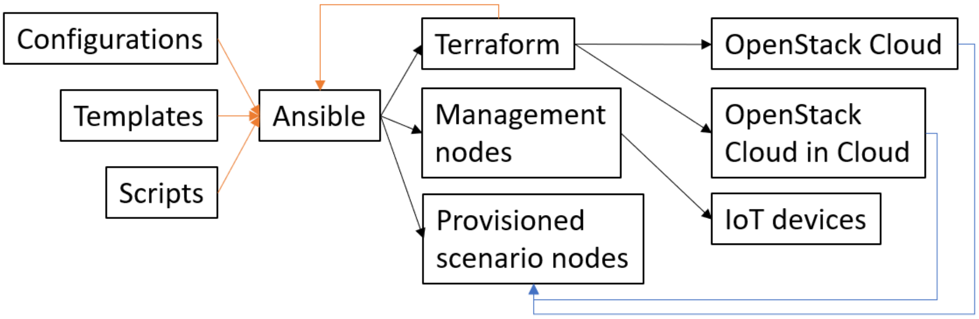


Figure 4.2: Work flow in creating a scenario

Although not implemented in this project, Moodle is suggested as a portal. Moodle is modular and gives possibilities to create modules for uploading and running scripts for executing tasks on the Cyber Range. Moodle can be used for access control, and thereby separating the teams, and give resource access to instances, like console, ssh, etc. for the resources allocated. The implementation in this project is done through scripting in Ansible, Terraform, bash and python.

4.3.6 Scenario

Scenarios can be developed with the combination of Ansible, Terraform templates and scripts, utilizing the infrastructure available as shown in figure 4.2. The Ansible playbooks, Terraform templates and scripts are grouped into two collections. One for management (instantiating the Cyber Range with management resources) and one for scenario. The scripts must be adapted to the Cyber Range, e.g. using networks available in the OpenStack infrastructure to provision resources to the correct networks.

Ansible calls on Terraform to provision nodes for the scenario, using defined templates. Ansible polls Terraform for information on how to reach the nodes, waits for the nodes to be ready and runs playbooks to configure the nodes. The management nodes are responsible for resetting and configuring the IoT nodes that can not be controlled via OpenStack. After Ansible has completed the configuration of each node, the node must be moved to its respective network. When all nodes are installed, configured and placed in the right network segment, the scripts and playbook complete and the Cyber Range is ready for an exercise.

In a management host, OpenStack2, FirmAE with a Linux IoT image, is executed inside a docker container. This docker container has network connection to the blue team's network, thus having access to an emulated IoT device through docker with network address translation (NAT) within the docker container.

If a portal was to be implemented, a descriptive file would be necessary to give present the resources in the scenario, give access to the resources, describe the scenario, and provide assignments to the participants.

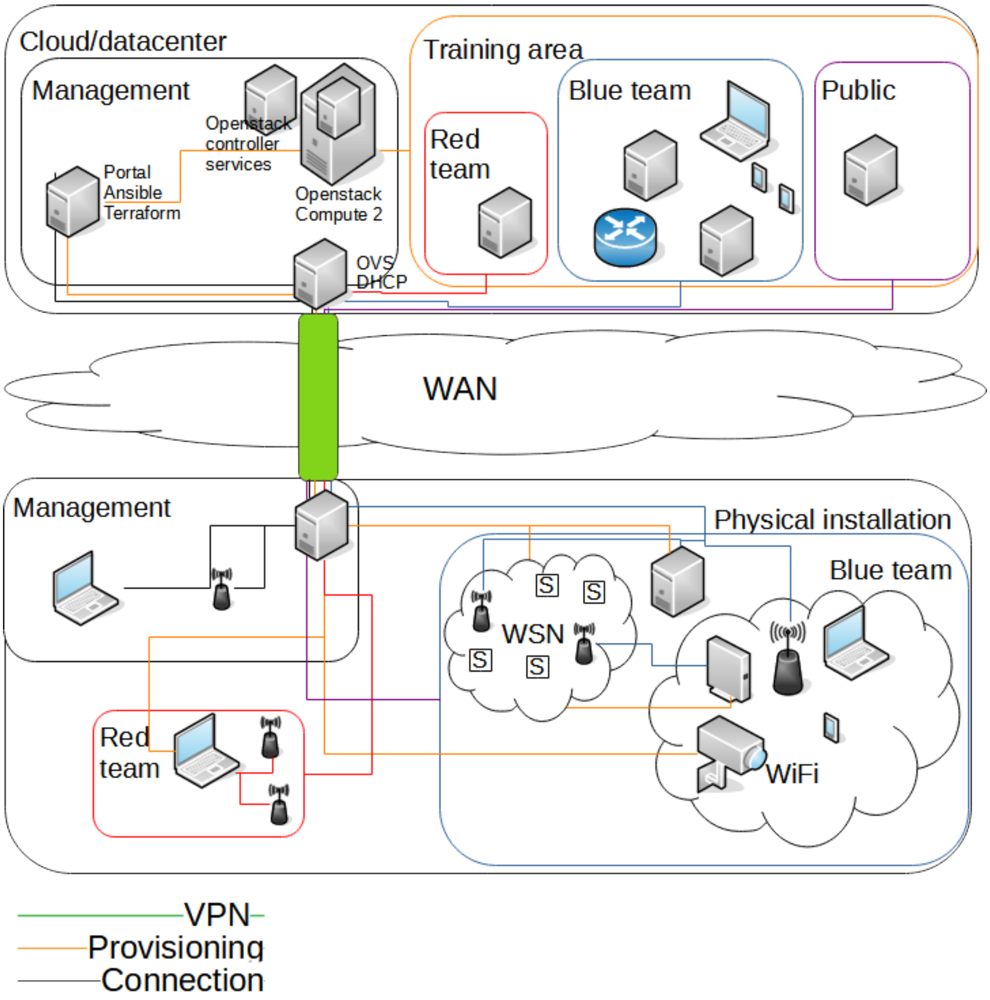


Figure 4.3: Design of IoT Cyber Range

4.3.7 Design Discussion

The figure 4.3 shows the design proposal. Both the physical installation and the virtual installation is divided into at least 3 areas, red team, blue team and management. The areas for the red and the blue team can be further divided if the scenario requires more network zones, like DMZ or further segmentation of networks etc. To ensure the isolation requirement so that malware or attack traffic is not inadvertently propagated to the Internet the Internet services on the public zone is emulated by inetsim[13].

OpenStack Neutron provides virtual routers and virtual networks to route traffic between the network segments. As the cloud solution does not guarantee to provide bare metal services, a separate cloud is established within the cloud. The internal cloud is also based on OpenStack and provides services like iden-

[13] https://www.inetsim.org/

tity management(Keystone), image management(Glance), bare metal provisioning (Ironic), networking (Neutron) and virtualization (Nova). OpenStack Ironic provides possibilities to provision hardware through orchestrating network ports through Neutron, images through Glance, powering on/off devices and network booting. The OpenStack controller contains Neutron controller, Nova controller, Ironic controller, Keystone, Glance, as well as a Nova compute installation to manage the bare metal nodes (compute 1). In the management zone another OpenStack compute node, compute 2, is provisioned to be controlled by the internal cloud to provision images not available in the "outer" cloud. The compute 2 node purpose is also to emulate any systems with a different architecture than which the cloud is built upon. E.g. OpenStack can not run ARM processor architecture on a x86 hardware, although the hypervisor could have handled such case. Therefore a virtual instance of Linux is deployed so QEMU can do system emulation of the supported architectures. QEMU then runs a i.e. ARM emulation which installs another OpenStack hypervisor(Compute 3) providing ARM capabilities to the cyber range. Alternatively the compute 2 node can run FirmAE to emulate Linux-based firmware's. The Neutron network between the OpenStack components is handled by a separate VXLAN network via the switch server. The switch server (switch1) has a Open vSwitch installation to manage the networks, team networks, management network, provisioning network, public network, and provides VXLAN capabilities. The same server is also available on Internet to provide VPN access using WireGuard to the physical installation. Depending on the needs for the scenario, nodes to the scenario can be provisioned on the provider cloud or the internal cloud compute node 2 or 3. Compute node 1 is only to provide bare metal services to hardware in the physical part of the cyber range.

On the physical side another Open vSwitch (switch2) installation handles the networks tunneled over VXLAN over VPN. All networks available on the cloud is also available on the physical side. Open vSwitch is able to attach physical network adapters to the desired network, the switch2 can also be patched to an external managed switch. The Ironic-Neutron combination on the OpenStack controller can control the network adapters on Open vSwitch and e.g. another managed switch, Cisco IOS is supported through the same plugin.

To ensure monitoring what happens on the physical air, i.e. wireless communication, the switch2 also has wireless sniffer(s) installed. As it is not possible to listen to multiple frequencies at the same time, one radio must be installed for each frequency that has to be monitored. A possibility is to scan several frequencies to map what frequencies are in use, but while jumping frequencies information in another frequency might be missed. The red team in the physical installation should also be given possibility to monitor the wireless communication as well as sending signals. Monitoring and sending at the same time is not possible either with only a single radio, multiple radios are required or at least recommended.

4.4 Implementation

This section discusses the practical scenario implementation and the hardware used.

4.4.1 Openstack services

Openstack Neutron provides DHCP capabilities, however the DHCP service only offers addresses to nodes provisioned via Nova. This might be configurable, but since the cloud is out of our control we chose to provide own DHCP services for the public network and for the management network. DHCP services for the red and blue team network is to be provided by the team it self or via the scenario definitions. In general, not only for DHCP, it can be a good idea not to be dependant of any special configuration from other service providers.

4.4.2 Physical installation

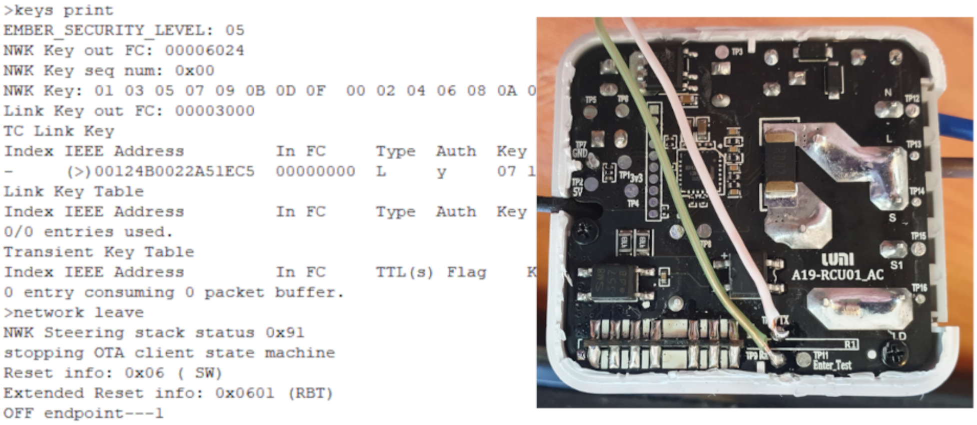
The devices used in the implementation are diverse. These are devices are used in the project to illustrate all the different challenges and opporunities that can arise. This selection show some of the diversity in IoT devices, and to some extent the flexibility of the Cyber Range. Devices used and tested in this implementation:

- Linksys E900 N300
- Vivotek FE9180-H
- Raspberry Pi 3b+ with additional USB-UART adapters
- Aqara SSM-U01
- Cisco 2960
- Arduino Pro Mini with Digi XBee S2
- Laptop with extra USB network adapters
- CC2531 USB Zigbee traffic sniffer
- Sonoff Zigbee 3.0 USB dongle
- Nedis WIFIP130FWT
- Cleverio Smart switch 51701

The **Aqara SSM-U01** is a Zigbee switch module. Looking inside the SSM-U01 reveals already marked testpoints for ground (GND), RX and TX. The switch is powered through mains and handling it requires caution to not touch parts that can have high voltages. Ensuring the RX and TX voltage level to be the same as the testing device it is time to communicate with the device. The SSM-U01 seems to follow the Silicon Labs ZigBee Application Framework CLI language[14]. Some testing gives indication on what commands can be used for resetting, connecting etc. Useful commands in this device are:

- plugin network-steering start 0
  Starts scanning and joining a network. The device has no interface for provid-

[14] https://docs.silabs.com/d/zigbee-af-api/6.9/cli



```
>keys print
EMBER_SECURITY_LEVEL: 05
NWK Key out FC: 00006024
NWK Key seq num: 0x00
NWK Key: 01 03 05 07 09 0B 0D 0F  00 02 04 06 08 0A 0
Link Key out FC: 00003000
TC Link Key
Index IEEE Address         In FC     Type  Auth  Key
-     (>)00124B0022A51EC5  00000000  L     y     07 1
Link Key Table
Index IEEE Address         In FC     Type  Auth  Key
0/0 entries used.
Transient Key Table
Index IEEE Address         In FC     TTL(s) Flag    K
0 entry consuming 0 packet buffer.
>network leave
NWK Steering stack status 0x91
stopping OTA client state machine
Reset info: 0x06 ( SW)
Extended Reset info: 0x0601 (RBT)
OFF endpoint---1
```

Figure 4.4: Aqara SSM-U01 switch screenshot and wiring

ing keys, therefore the network must be open for joining and providing the network key. This window of opportunity is where an adversary should listen for traffic.

- network leave
  Disconnects from the Zigbee network
- reset
  Restarts the device
- keys print
  Prints the network and the link key.

Figure 4.4 shows the connections to the Aqara sensor as well as the output of some commands.

Some Tuya devices with WB2S modules from Nedis and Cleverio were also tested. The WB2S[15] is a module with wifi and bluetooth integrated and 2 UART devices. According to [44] the first UART (1RX and 1TX) is for programming, and the second UART (2RX and 2TX) is for logging to serial from the SoC. Connecting to the first UART gave no communication on the Nedis WIFIP130P nor the Cleverio 51701. It is possible that the firmware on these devices are programmed not to use the first UART for any communication. There are no other peripheral controllers in the device that require serial communication. The devices have other test points that indicate I/O interfaces, but since the firmware is not available for flashing the device there were no further exploration of the devices.

After soldering a wire on the test point for the second UART unfortunately on the Nedis device, a careless move, resulted in pulling off the 2TX test point on the device, making the device useless for further tesing and implementation.

The WB2S board on the Cleverio device is located with its back towards the relay, and mounted inwards into the frame, making it difficult to use the test points. The WB2S module was removed from the device to get to the test/soldering points, se figure 4.5. The devices PCB also has high voltage connected and special caution

[15] https://developer.tuya.com/en/docs/iot/wb2s-module-datasheet?id=K9ghecl7kc479

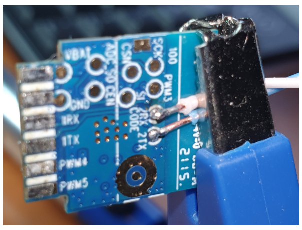


Figure 4.5: WB2S board dismounted from the Cleverio 51701

is required. Connecting to the 2. UART (2RX/2TX) on the device shows the log of the device, but the devices does not seem to accept any commands.

Earlier version of Tuya devices were possible to reflash with custom firmware over-the-air, but with a new PSK format that is not possible at the moment[16].

Vivotek FE9180-H is a 180°fisheye camera. The camera is powered over the network cable with Power-over-Ethernet (PoE). When opening the camera cover it reveals test points fairly available, many of them are marked. There is a 4-pin connector on the PCB which seems to be a good candidate for a serial connection. The serial connector often has a VCC, RX, TX and GND. Measuring the pins with a multimeter shows that is has pin with 0Ωto testpin marked ground on the PCB and one pin with 0Ωto testpoint marked 3.3v on the PCB. The two other pins have at least 1kΩresistance to both 3.3v and to ground. Measuring the voltage shows 3.3v on all pins exept the GND pin. The 2 other pins, the middle ones, are likely to be TX and RX: one of them fluctuates during boot and is possibly TX as it is likely to do this when writing information from the boot process.

Arduino Pro Mini is a programmable micro-controller with at Atmega 328p SoC. The SoC supports digital and analog inputs, digital outputs, has UART and SPI interfaces and more. For this project this controller is programmed through SPI. The Arduino is used in combination with **Digi XBee S2** as a ZigBee interface for the Arduino as an IoT device. The XBee has data input and output with serial communication, the same I/O can be used to configure the XBee. Programming the XBee is done by sending a configuration software to the Arduino, where as the Arduino programs the XBee. When the Arduino has programmed the XBee, the Arduino is programmed with the software for the scenario. To program the Arduino, the SPI interface is used through Raspberry Pi with bit banging using the avrdude[17] software. An alternative to program the XBee is to intercept the serial communication line between the Arduino and the XBee while ensuring that the Arduino does not communicate on the same line (e.g. by pulling the Arduino reset pin to low/gnd). The XBee is configured with the Digi XBee supported AT commands. Flashing new firmware and/or changing the operation mode (coordinator/router/end device) is only possible through a XBee Explorer board.

A 12 pin dual-in-line test points (DJ1) and a 4 pin test test point(DJ2) are available on the **Linksys E900 N300**, see figure 4.6. The 4 pin is likely to be serial

[16] `https://github.com/ct-Open-Source/tuya-convert/issues/483`

[17] `https://github.com/avrdudes/avrdude`

Pin	R VCC	R GND	V	Pin	R VCC	R GND	V
1 - nTRST	630 Ω	640 Ω	3.3 v	2	100 Ω	0 Ω	0 v
3 - TDI	630 Ω	570 Ω	3.3 v	4	100 Ω	0 Ω	0 v
5 - TDO	630 Ω	640 Ω	3.3 v	6	100 Ω	0 Ω	0 v
7 - TMS	630 Ω	640 Ω	3.3 v	8	100 Ω	0 Ω	0 v
9 - TCK	630 Ω	640 Ω	3.3 v	10	100 Ω	0 Ω	0 v
11 - nSRST	∞ Ω	∞ Ω	3.3v	12	100 Ω	0 Ω	0 v

Table 4.1: Measurements for 12-pin connector DJ1 on Linksys E900

connection and the 12 pin could possibly be JTAG. Measurements show half of the pins to be ground/0v and others to have some function. JTagEnum[45] is a JTAG scanner for Arduino and Raspberry Pi. Connecting these pins to the Raspberry and doing the scan reports the pin functions. Measurements and results are in table 4.1. Pins 2, 4, 6, 8, 10, 12 are all most likely to be ground.

After identifying the test pins to control the JTAG TAP **Open On Chip Debugger** (OpenOCD) is used. OpenOCD[18] is an open source software for providing debugging, in-system programming and boundary-scan testing for embedded target devices. OpenOCD can debug devices through debug adapters connected to debug port, like JTAG or SWD, on target devices. The configuration of OpenOCD has to configure the debug adapter, which in this case is Raspberry Pi, and configure the target. OpenOCD is capable to reset devices, read and write to input/outputs, to memory and to registers. By this we can rewrite the IoT device memory with the desired firmware or settings and reboot the device so that the cyber range can restart at a desired state. The Linksys router, and many other Linux-based embedded devices, that use boot loaders with tftp capabilities, also have the capability to flash the firmware while only starting the boot loader. For devices without flashing capabilities with JTAG or SWD, using the bootloader, UART and tftp solves the problem just the same. The Raspberry Pi controls a relay, through GPIO output, the relay is connected to the 12 v power supply of the router.

For connection to the cyber range in the sky a commodity laptop is used. The laptop must be preinstalled with a Linux operating system, in this case Ubuntu, with WireGuard to connect with the rest of the installation in the virtual realm. To distribute the networks Open VSwitch is used. For full utilization of the Ironic functions for provisioning each device should have their own port in a switch, either on the laptop with Open VSwitch or as in this case a Cisco 2960. Ironic supports both with the networking-generic-switch plugin available for Ironic. As the laptop is not connected to the installation server it has to be installed manually, scripts are provided. VXLAN does not allow fragmenting, other tunnelling protocols should possibly also be considered.

Texas Instruments CC2531 is a SoC with Zigbee capabilities. This is on a USB dongle is used as a sniffer to log Zigbee traffic. The CC2531 must be flashed with

[18] https://openocd.org/

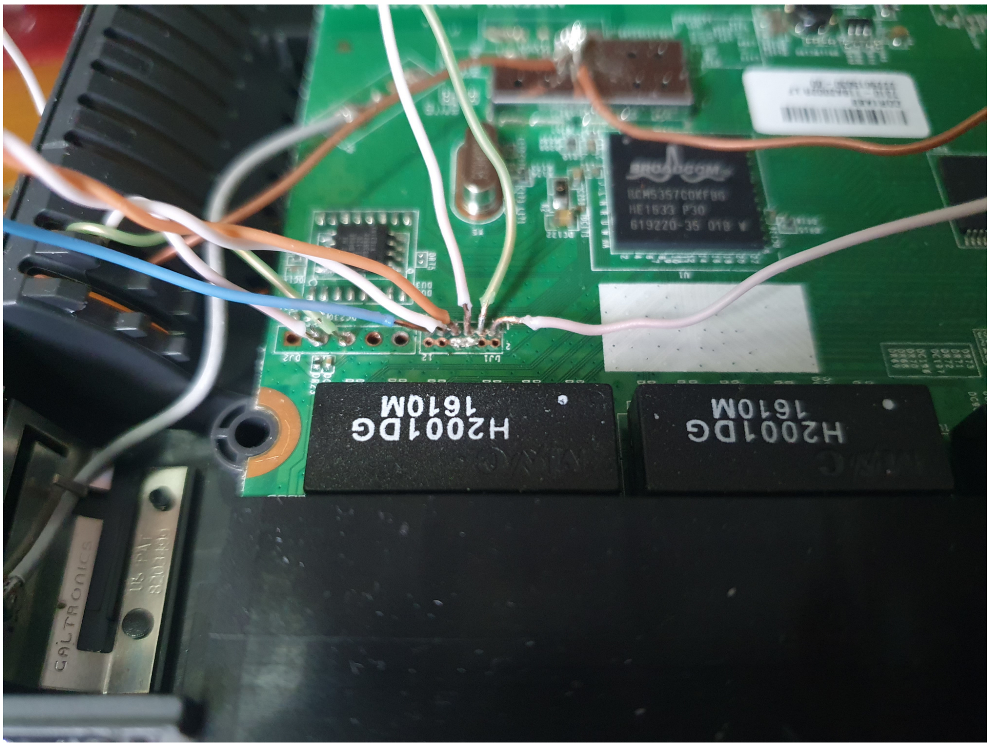


Figure 4.6: Connection on Linksys E900 N300

new firmware to be a sniffer. The flash _cc2531[19] software is used for flashing the sniffer via Raspberry Pi. Since the Raspberry Pi only has one UART as default, the implementation also is limited to controlling one device at the time. More UARTs must be installed when creating scenarios with more IoT devices as discussed earlier.

Figure 4.7 shows the switch with 3 ports, first port on the management network, the second on the red team network and the third on the blue team network, installed on the laptop in the lower left corner in the picture. The same computer has the CC2531 installed for logging Zigbee traffic. The other laptop is installed with the Sonoff Zigbee USB dongle as a Zigbee coordinator in the red team segment.

4.5 Scenarios

The fourth activity in Design Science Research is to demonstrate the artifact. This section describes two attacks towards IoT devices and shows how this hybrid IoT cyber range can implement scenarios with the attacks. The Cyber Range must be able to implement attacks with relevance to be used in demonstrations, education

[19] https://www.zigbee2mqtt.io/guide/adapters/flashing/alternative_flashing_methods.html

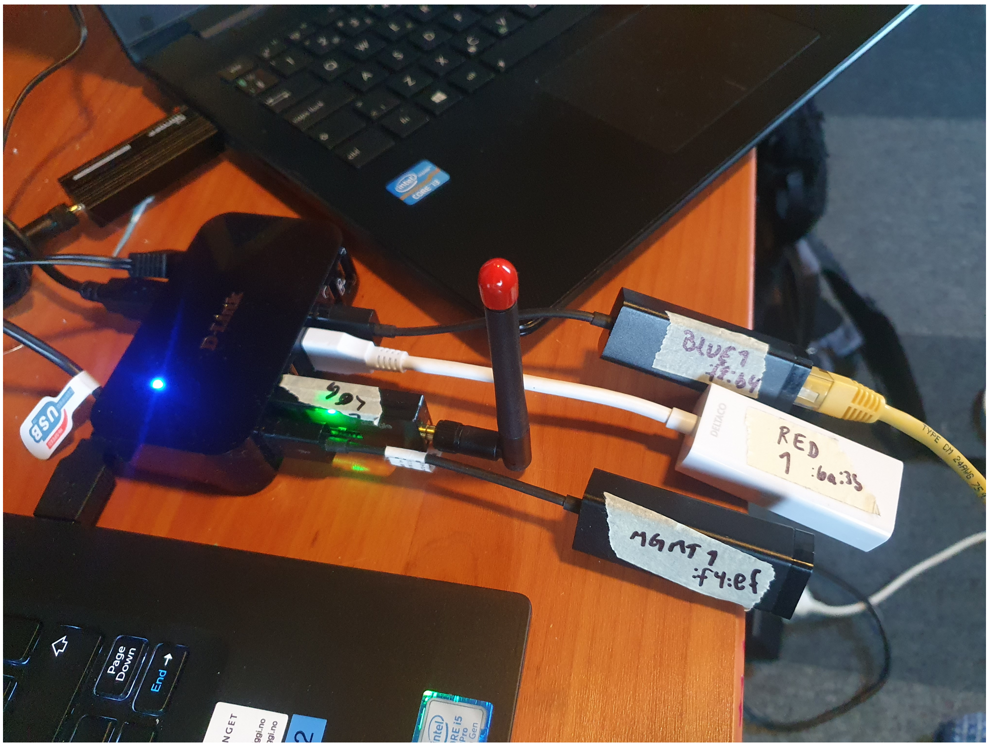


Figure 4.7: Switch

etc.

4.5.1 Attack 1: Mirai botnet

The Mirai botnet was a botnet used to attack among others the DNS provider Dyn. When Dyn was attacked with DDoS the attack resulted in many large sites being unavailable, sites like Github, Twitter, Netflix and others. The Mirai botnet is a malware that was designed to target IoT devices. The Mirai botnet uses IoT devices on Internet with available ports and default password to do attack. An already infected node scans the network to find devices and upon finding devices it tries credentials from a list of factory default passwords. If it malware gets access then the malware contacts a Command and Control (C&C) server to wait for instructions. While waiting the malware continues to search for new nodes to expand the botnet [4]. The ports scanned by the Mirai bot are port 23 and 2323, and the protocol is telnet. Telnet is a clear text protocol similar to serial communication, but used over network.

The Mirai malware affects Linux hosts and is designed to propagate over networks. The Mirai malware, at least in it original design, was also designed to be run in memory only. When rebooting the device the Mirai code would disappear from the device. If the device still is vulnerable after starting up, it would likely be infected again. The malware have capabilities for DDoD. The volatility of the

Mirai infection makes discovering indicators from a Mirai attack difficult on the device it self if the device is rebooted.

The source code for Mirai malware was made publicly available, also on Github. The Mirai botnet has serveral elements, a C&C server , a database server, scan reciever, DNS server and a loading server. The domain name for the Mirai botnet is registered in a DNS system and coded in to the bot executable. An attacker connects to the C&C server through telnet for control. The C&C server sends commands to infected IoT devices. If an infected devices finds new vulnerable devices, it is reported to the scan receiver. The loading server will act upon the new information and infect the vulnerable device. The new bot then registers itself to the C&C server. The IP address to the C&C server is resolved from the DNS server. The availability of the source code has resulted in many versions of Mirai malware.

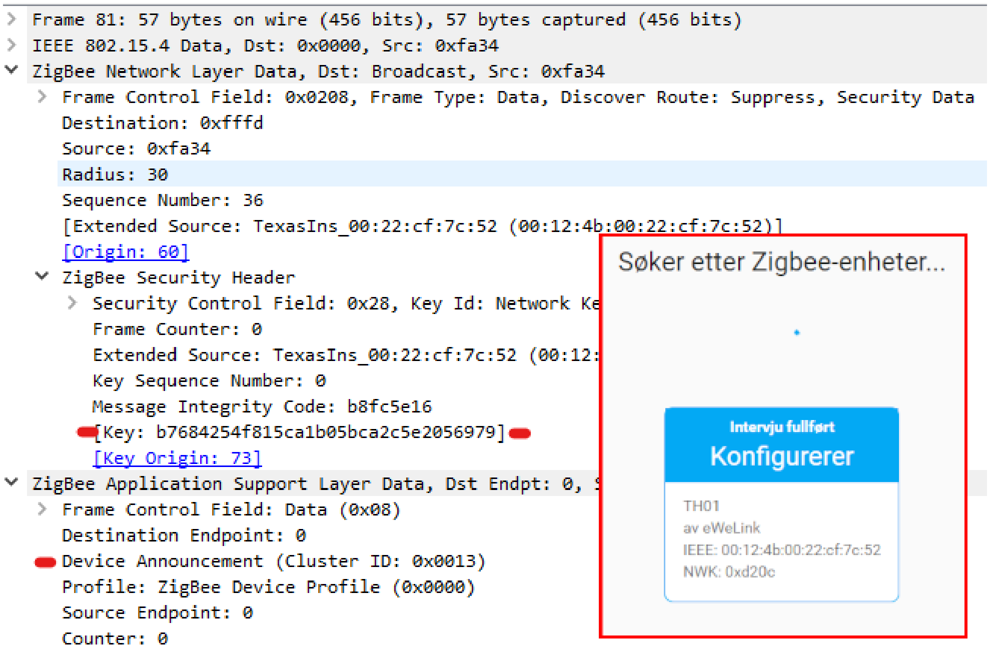
For this setup the C&C, database/mysql, scan reciever and dns server were installed on Ubuntu servers in the cloud installation. Also router and camera firmware was running on the cloud as well as on the physical installation. Ansible, with the help of Terraform and some scripts, was responsible for the orchestration to create, destroy and recreate the scenario. To make the Mirai network spread one device had to be infected as part of the scenario. The infected and a vulnerable device was started with a FirmAE installation. As we are interested in the network access and the operating system of the IoT device, FirmAE should be able to provide the functions needed for creating a scenario.

Mirai C&C server is programmed in Go. The bot is written in C and has to be precompiled to each architecture is needs to be executed on, the IoT devices is not likely to contain compilers, nor having the process power to compile within reasonably time. Compiling on the target on the IoT device would also affect the IoT device performance and thereby increase the chance of being discovered. To cover most architectures for IoT devices the Mirai code was cross-compiled for 10 [46] different architectures. Creating cross-compilers is described at osdev.org.

Files can be inserted to the scenario within images. For instances within the cloud environment cloud-init[20] is a possibility. Cloud-init is a method for cloud instance initialization, also supported by Openstack. For devices outside the Openstack environment configuration is possible through Ansible. Some systems may have user data limit, like AWS has a 16KB size limit, making Ansible the more prominent solution even for cloud instances.

The purpose of this test is to confirm botnet propagation from the cloud infrastructure an to the physical side. Also to revert the changes the botnet may do.
FirmAE establishes a serial device as a unix socket. This unix socket can be used as a serial port to the emulated device and act as a side channel for configuring the IoT device. On the host FirmAE creates network devices via QEMU, the devices are avalable as tap ethernet devices on the host and in the same network segment as the emulated device. The ethernet device can be added as a port to the Open

[20] `https://cloudinit.readthedocs.io/en/latest/topics/examples.html#writing-out-arbitrary-files`



```
> Frame 81: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)
> IEEE 802.15.4 Data, Dst: 0x0000, Src: 0xfa34
v ZigBee Network Layer Data, Dst: Broadcast, Src: 0xfa34
  > Frame Control Field: 0x0208, Frame Type: Data, Discover Route: Suppress, Security Data
    Destination: 0xfffd
    Source: 0xfa34
    Radius: 30
    Sequence Number: 36
    [Extended Source: TexasIns_00:22:cf:7c:52 (00:12:4b:00:22:cf:7c:52)]
    [Origin: 60]
  v ZigBee Security Header
    > Security Control Field: 0x28, Key Id: Network Ke
      Frame Counter: 0
      Extended Source: TexasIns_00:22:cf:7c:52 (00:12:
      Key Sequence Number: 0
      Message Integrity Code: b8fc5e16
      [Key: b7684254f815ca1b05bca2c5e2056979]
      [Key Origin: 73]
v ZigBee Application Support Layer Data, Dst Endpt: 0, S
  > Frame Control Field: Data (0x08)
    Destination Endpoint: 0
    Device Announcement (Cluster ID: 0x0013)
    Profile: ZigBee Device Profile (0x0000)
    Source Endpoint: 0
    Counter: 0
```

Søker etter Zigbee-enheter...

Intervju fullført

Konfigurerer

TH01
av eWeLink
IEEE: 00:12:4b:00:22:cf:7c:52
NWK: 0xd20c

Figure 4.8: Screenshot Wireshark - Zigbee network key while configuring the device in Home Assistant

vSwitch installation with a tunnel to the cyber range network. Using the correct vlan id when adding the port will make the emulated device be available on the correct network in both the virtual and the physical part of the infrastructure.

4.5.2 Attack 2: Zigbee

As described earlier, Zigbee is vulnerable for eavesdropping of the network key while linking new devices to the network. An installation with the Sonoff Zigbee 3.0 dongle integrated into Home Assistant will make Home Assistant be able to link to new Zigbee nodes. The default setting of the Sonoff Zigbee dongle is to allow linking for all devices, meaning that the network key is also available for all nodes within reach.Trying to link a new device to the network will transmit the network key over the air and be available for sniffing with whsniff[21] using the CC2531 USB device and viewing the packets in Wireshark/tshark. The default global trust center link key must be inserted into Wireshark to decrypt the traffic when the network key is transferred, and the Network key is revealed as shown in figure 4.8.

With the network key available the traffic content is available for listening, interception and inserting. Controlling a relay for light might not have a big impact, but for devices that control devices with higher energy potential, like heating elements etc., the devices could cause fire if overheating, water heaters might ex-

[21] https://github.com/homewsn/whsniff

plode if the water reaches boiling temperatures and so forth, where the result might be fatal.

The unsecure approach for network key exchange is a result of that the installation must be as easy as possible to the end user while pairing the devices, and since the devices often don't have an interface other than one single button. The simplicity increases the usability and reduces cost, but comes with a cost of reduced security.

Chapter 5

Evaluation and Discussion

This chapter evaluates the Cyber Range and discusses the limitation found and the ethical challenges that can arise.

5.1 Evaluation

The last activity in Design Science Research is evaluation. The main goal of the evaluation activity is to assess whether, and how well, the designed artifact solves the problem. However, the product is also compared to the requirements as defined in section 4.1.2, on page 18, these are used as guidelines when developing the cyber range in this project. Some requirements are evaluated *ex ante*, by informed argument, as they are not part of the development, opposed to the requirements that are developed, is evaluated *ex post*, by doing an experiment.

The main problem to be solved in this project is to reduce the resource usage for doing IoT Cyber exercises, mainly resources in form of man hours. Evaluating this project in large scale is difficult as there are not that many IoT devices available at hand for deploying. One can safely assume that several processes can be executed in parallel depending on the managing channel count available. Time measuring evaluation is done by testing the artifact, running the scripts developed in the project, and comparing it to the time it would take to manually set up the same exercise environment.

In [15], a Cyber Range with 75 virtual machines and 5 networks were deployed in 5 minutes after the scenario files were prepared and uploaded. The assumption for measuring time is that all files and connections are prepared. Deployment requires timing and orchestration as some components can not be deployed or prepared before some other components. Physical components in this hybrid environment has other limitations that virtual systems do not have.

This project has used of-the-shelf IoT devices as well as self-built devices. The IoT world has a plethora of brands, devices and solutions. The validity in the tests are considered to be good as the Cyber Range has support for both handling emulation as well as many different types of physical components. The setup can therefore be complex, the report discusses some of those challenges, but to handle as many types as possible the setup has to be complex.

While some papers in the related work chapter present testbeds and cyber ranges, also for IoT, none of the papers have a complete description on how it could be implemented, testing and finding possible technologies, nor have an approach for using IoT test access points for preparing a cyber range and monitoring the components.

5.1.1 Requirements

The requirements and their fulfillment for this IoT Hybrid Cyber Range are listed in table 5.1. A requirement can be fulfilled in design or in implementation. When covering a requirement in implementation, it is implied that the requirement is also fulfilled in design. To recap the requirements:

- flexibility, scalability, adaptability, interoperability
- shareability, open source
- fidelity
- isolation, safety, resilience, reliability
- cost-effectiveness
- built-in monitoring
- easy access, usability, user interfacing
- service-based access
- heterogeneity, handle diversity in IoT, emulating digital twin
- end-to-end testbed

Flexibility, scalability, adaptability, interoperability

Combining the flexibility in OpenStack services with the physical components, and linking these with Open vSwitch, WireGuard and VXLAN functionality gives a large flexibility in how to design scenario networks, how the networks are linked and how they can be combined.

Shareability, open source

To cover the requirement for shareability, the project has focused on using open source software. Implementations on this Cyber Range will be available published on github. The software used in the implementation is all open source.

Fidelity

The suggested taxonomy and unified functional component architecture from Yamin et.al [2] is used as a reference for this cyber range. Well known large software projects are used as base for the infrastructure. By this the fidelity requirement is covered.

Cost-effectiveness

The cost for implementing this cyber range is reasonable. The devices used are low-budget devices. Cost-effectiveness is not only about investing devices, reducing work hours for administering the cyber range will also reduce costs. Using provisioning, scripting and interfacing with the IoT devices will automate tasks and thereby reduce time spent resetting already developed scenarios.

Isolation, safety, resilience, reliability

Using side-channels, such as JTAG and serial interfaces, to communicate with the IoT devices ensures isolating management segment from the exercise/training/testing segment of the cyber range. Simulating own Internet services, providing services on own cloud and isolating the exercise segment from other networks and Internet. Any attack happening in the training area should not affect the managing segments nor be spread outside the exercise area provided the provisioning steps are carefully considered. It has also been suggested to use specially built rooms or areas to ensure that wireless signals from outside are not present in the Cyber Range, and that the wireless signals whitin Cyber Range are not propagated to the outside world.

Monitoring

This design suggests methods and solutions for monitoring, but it is not implemented nor tested in this project. Monitoring through the side-channels in IoT gives an opportunity to log what happens in every device, and also use the same timestamps as the rest of the Cyber Range. One alternative to extract the information live, directly from an IoT device is to collect the logs after the exercise, which can result in only partial logs due to storage limitations, or compromised logs due to an action by the training teams in the exercise.

Easy access, usability, user interfacing

A portal was outside the scope of this project, the design suggests the requirements for user interfacing, usability and easy access to be a part of the portal. Finding and accessing the side-channel interfaces on the IoT devices can also be a challenge due to the design and lack of design documents of the devices, but the approach and used tools for this tasks are referred to in the report.

Selecting IoT devices for use in a cyber range for exercises must be done with care. To be able to automate a Cyber Range provisioning the IoT devices must have side-channel interfaces which can be used in automation and monitoring. JTAG/UART/SWD/GPIO/SPI is used in this project. If the Cyber range is used as a firmware testbed, emulating with FirmAE can be sufficient. Extracting from or writing to memory can be done with test access ports. Writing new firmware to the physical router used in this project was possible since the router starts a

TFTP server for a brief period during boot. This was a boot loader feature in this firmware. Features in a firmware depends on the developer.

Service-based access

Service-based access is out of scope for this project, all though most of the components used provide service-based access. The overall design does not limit it, but no component was given the role for this.

Heterogeneity, handle diversity in IoT, emulating digital twin

FirmAE adds the Cyber Range a capability to emulate a large library of Linux-based IoT firmware's. Because of the challenges in emulating all peripherals and integration's of an IoT device, this cyber range also has a physical infrastructure which is capable of handling real IoT devices provided they have interfaces for managing the devices.

End-to-end testbed

The cyber range do not have any limitations that prevents creating scenarios or environments that can simulate end-to-end connectivity, from IoT device to cloud services. The infrastructure has support for implementation from local network, where the home IoT devices operate, to a simulated Internet or public network, where the cloud services reside.

Requirement	Fulfillment
Flexibility, scalability, adaptability, interoperability	Design: yes Implementation: yes
Shareability, open source	Design: yes Implementation: yes
Fidelity	Design: yes Implementation: partially
Isolation, safety, resilience, reliability	Design: yes Implementation: partially
Cost-effectiveness	Design: yes Implementation: yes
Built-in monitoring	Design: yes Implementation: partially
Easy access, usability, user interfacing	Design: partially Implementation: partially
Service-based access	Design: partially Implementation: no
Heterogeneity, handle diversity in IoT, emulating digital twin	Design: yes Implementation: yes
End-to-end testbed	Design: yes Implementation: yes

Table 5.1: Requirements evaluation

5.1.2 Efficiency

Even with the knowledge on how the devices work, a manual setup with 10 devices, installed for the networks, logging in and installing configuration and software, it is difficult to complete all tasks for this implementation in under one hour, even for this small setup. The total time would of course be reduced if the preparation is done by several persons doing parallel tasks, however, some time must be set to coordination. Using the scripts and design developed in this project, the same tasks are done in less than 40 minutes. Time measurement is not used as an exact measurement, since factors like network speed, processing power, wireless scanning algorithms and so forth can affect the time usage. To ensure reliability in the efficiency, the Cyber Range is reset to the same scenario several times and reporting the average.

The single most time consuming task is to provision the bare metal computer via OpenStack Ironic. Transferring the image to the boot image on the computer takes time. The time consumption is depending of the network speed, disk speed and processing speed in both ends. Even if the time difference in doing it manually vs automation is not that large, the automation ensures that the setup is the same for all reruns if the same scenario is to be executed again. From the deployment start, until Ironic reports the node active, it takes about 30 minutes on a laptop computer with Intel Core i3 3217u with 4GB RAM and 500GB SATA 5400 rpm HDD, to install a Ubuntu Bionic with an image size of 642 MB. It is first at this point that the client is ready to have additional software installed, unless software is already prepared in the image. Doing software installation as a separate part, after the image installation, gives more flexibility. As WireGuard and VXLAN both use UDP, and as UDP is an unreliable protocol, network congestion in some part of the transmission chain could affect performance.

5.2 Ethical considerations

Software used in this project is used within the license requirements to the best of my knowledge. Anyone using the same software is responsible to accept the license of the software. Care must be taken when implementing a Cyber Range that the range is isolated from other systems to ensure that no harm is propagated outside the range. Using a Cyber Range to learn vulnerability exploitation gives an ethical and moral responsibility to report zero-day vulnerabilities to the manufacturer or maintainer, and also not use the vulnerabilities to compromise systems outside the Cyber Range.

While the software for malware is available on the Internet, and there exists description on how to use them, anyone working with malware must handle them in a controlled environment. A misconfigured Cyber Range can potentially get Internet access and spread malware, even if it is not intentionally.

5.3 Limitations

The participation on the questionnaire is limited, with only 18% received of the 22 initial recipients. The total number of recipients is also limited. There was a request to forward the survey to others as well, and it was in deed forwarded by some. There was also one respondent who replied about having to busy schedule. The community in Cyber Range development is small, and people focusing on IoT Cyber Ranges are even fewer. While the count may not give a full empirical picture of the world, the result do give some valuable information that this project has chosen an approach that others find useful as well, both in design and choice of tools. At least, it also confirms that the choices in this design are relevant. Since the size of the community is limited, it is likely that they have busy schedules and are not available to participate in surveys in every level. As Yamin et al. [15] also pointed out, these are persons with a very specific skill set, and the selection is considered to be small.

During the implementation there were some challenges with the electronics of the devices. On the UART of the Vivotek FE9180-H camera, the RX pin stopped working. The TX pin still sent data from the camera, but thee terminal was not able to send commands to the camera. On the Nedis WIFIP130FWT, a soldering point was loosened from the PCB, making the TX pin unavailable on the device.

A soldered wire loosened from the ground soldering point on the Aqara SSM-U01 when moving it around, resulting in a short circuit and that the entire device did not start anymore. These examples shows the requirement for caution when working with electronics and how sensitive the components are. When soldering on the test wires, it is recommended to fasten the wires to the device with e.g. a glue gun, this can ensure insulation as well. Isolating the devices electrically from each other with e.g. optocouplers[1] should be considered in a production implementation. Limiting potential large currents with resistors is also good safety measure to have components survive unfortunate incidents.

[1] `https://en.wikipedia.org/wiki/Opto-isolator`

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The Hybrid IoT Cyber Range designed in this project fulfill the functional architecture components and the requirements as discussed in chapter 4.3. The project was limited in implementation to the scenario and run-time environment components. The requirements, which are within the scope of this project, surveyed for the cyber range have been discussed and found to be covered by the design.

Selecting the physical IoT devices used in a cyber range must be done with care. Although the device can be instructed to do tasks via e.g the serial/UART side channel, it might not be enough if the device is unmanageable due to corrupt firmware or parts of firmware is changed. Using alternative communications with the devices gives an opportunity to automate behaviour of and log activity on the devices while still being able to use the same devices as part of training, testing and exercises.

Creating the entire scenario, finding the IoT alternative communication capabilities, creating scripts etc. are time consuming. If the exercise is a one time doing, it might be a better option to manually set up an environment, at least the physical IoT part. This must be a consideration before using the approach in this project.

Automating scenario creation does not only save time, it also ensures that the same exercise has the same setup and configuration for every run.

6.2 Future work

This project focused on the integration between a virtual and a physical cyber range and how to administer the IoT devices out-of-band of the exercise. The other components of a cyber range are not implemented, as they are more general for cyber ranges. The next step is to implement a complete cyber range with all components.

The cyber range should also include an approach for emulating IoT devices that are not Linux-based, possibly using p^{2}im [41].

A more generalized approach for creating device libraries would make it easier to add more devices to the cyber range, however the diversity in IoT devices makes a general approach difficult. The common elements like power control, serial communication over UART, JTAG or SWD usage could be elements for function prototyping/interfaces.

Bibliography

[1] NIST National Initiative for Cybersecurity Education (NICE), *NICE One Pager for Cyber Ranges*. [Online]. Available: `https://www.nist.gov/document/cyberrangespdf`.

[2] M. M. Yamin, B. Katt and V. Gkioulos, 'Cyber ranges and security testbeds: Scenarios, functions, tools and architecture,' en, *Computers & Security*, vol. 88, p. 101 636, Jan. 2020, ISSN: 01674048. DOI: `10.1016/j.cose.2019.101636`. [Online]. Available: `https://linkinghub.elsevier.com/retrieve/pii/S0167404819301804` (visited on 13/10/2020).

[3] *49 Stunning Internet of Things Statistics 2022 [The Rise Of IoT]*. [Online]. Available: `https://techjury.net/blog/internet-of-things-statistics/` (visited on 04/05/2022).

[4] A. G. Eustis, 'The Mirai Botnet and the Importance of IoT Device Security,' en, in *16th International Conference on Information Technology-New Generations (ITNG 2019)*, S. Latifi, Ed., vol. 800, Series Title: Advances in Intelligent Systems and Computing, Cham: Springer International Publishing, 2019, pp. 85–89, ISBN: 978-3-030-14069-4 978-3-030-14070-0. DOI: `10.1007/978-3-030-14070-0_13`. [Online]. Available: `http://link.springer.com/10.1007/978-3-030-14070-0_13` (visited on 08/11/2020).

[5] J. Vykopal, M. Vizvary, R. Oslejsek, P. Celeda and D. Tovarnak, 'Lessons learned from complex hands-on defence exercises in a cyber range,' en, in *2017 IEEE Frontiers in Education Conference (FIE)*, Indianapolis, IN: IEEE, Oct. 2017, pp. 1–8, ISBN: 978-1-5090-5920-1. DOI: `10.1109/FIE.2017.8190713`. [Online]. Available: `http://ieeexplore.ieee.org/document/8190713/` (visited on 04/10/2022).

[6] H. Touqeer, S. Zaman, R. Amin, M. Hussain, F. Al-Turjman and M. Bilal, 'Smart home security: Challenges, issues and solutions at different IoT layers,' en, *The Journal of Supercomputing*, vol. 77, no. 12, pp. 14 053–14 089, Dec. 2021, ISSN: 0920-8542, 1573-0484. DOI: `10.1007/s11227-021-03825-1`. [Online]. Available: `https://link.springer.com/10.1007/s11227-021-03825-1` (visited on 19/10/2022).

[7] A. Shepherd and E. Apeh, 'An IOT Security Awareness and System Hardening Advisory Platform for Smart Home Devices,' en, in *HCI International 2021 - Posters*, C. Stephanidis, M. Antona and S. Ntoa, Eds., vol. 1420, Series Title: Communications in Computer and Information Science, Cham: Springer International Publishing, 2021, pp. 439–446, ISBN: 978-3-030-78641-0 978-3-030-78642-7. DOI: 10.1007/978-3-030-78642-7_59. [Online]. Available: https://link.springer.com/10.1007/978-3-030-78642-7_59 (visited on 25/10/2022).

[8] A. Koohang, C. S. Sargent, J. H. Nord and J. Paliszkiewicz, 'Internet of Things (IoT): From awareness to continued use,' en, *International Journal of Information Management*, vol. 62, p. 102 442, Feb. 2022, ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2021.102442. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0268401221001353 (visited on 25/10/2022).

[9] M. Karjalainen, S. Puuska and T. Kokkonen, 'Measuring Learning in a Cyber Security Exercise,' en, in *2020 12th International Conference on Education Technology and Computers*, London United Kingdom: ACM, Oct. 2020, pp. 205–209, ISBN: 978-1-4503-8827-6. DOI: 10.1145/3436756.3437046. [Online]. Available: https://dl.acm.org/doi/10.1145/3436756.3437046 (visited on 10/11/2022).

[10] J. Päijänen, K. Saharinen, J. Salonen, T. Sipola and J. Vykopal, 'Cyber Range: Preparing for Crisis or Something Just for Technical People?' en, p. 12,

[11] S. Schwab and E. Kline, 'Cybersecurity Experimentation at Program Scale: Guidelines and Principles for Future Testbeds,' en, in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Stockholm, Sweden: IEEE, Jun. 2019, pp. 94–102, ISBN: 978-1-72813-026-2. DOI: 10.1109/EuroSPW.2019.00017. [Online]. Available: https://ieeexplore.ieee.org/document/8802402/ (visited on 09/11/2020).

[12] J. Keggler and T. Mahon, 'Preparing for cybergeddon,' eng, *Armada International*, vol. 33, no. 2, pp. 34–36, 2009, Place: Gurgaon Publisher: Media Transasia Group tex.copyright: COPYRIGHT 2009 Media Transasia Group, ISSN: 0252-9793.

[13] J. Davis and S. Magrath, 'A Survey of Cyber Ranges and Testbeds,' en, p. 38,

[14] M. M. Yamin, B. Katt, E. Torseth, V. Gkioulos and S. J. Kowalski, 'Make it and Break it: An IoT Smart Home Testbed Case Study,' en, in *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control*, Stockholm Sweden: ACM, Sep. 2018, pp. 1–6, ISBN: 978-1-4503-6628-1. DOI: 10.1145/3284557.3284743. [Online]. Available: https://dl.acm.org/doi/10.1145/3284557.3284743 (visited on 17/10/2022).

[15] M. M. Yamin and B. Katt, 'Modeling and executing cyber security exercise scenarios in cyber ranges,' en, *Computers & Security*, vol. 116, p. 102 635, May 2022, ISSN: 01674048. DOI: 10.1016/j.cose.2022.102635. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404822000347 (visited on 17/10/2022).

[16] J. Vykopal and R. Ošlejšek, 'KYPO Cyber Range: Design and Use Cases,' en, p. 12,

[17] G. Kavallieratos, S. K. Katsikas and V. Gkioulos, 'Towards a cyber-physical range,' en, *New Zealand*, p. 10, 2019.

[18] N. Chouliaras, G. Kittes, I. Kantzavelou, L. Maglaras, G. Pantziou and M. A. Ferrag, 'Cyber Ranges and TestBeds for Education, Training,and Research,' en, p. 23, 2021.

[19] M. AL-Hawawreh and E. Sitnikova, 'Developing a Security Testbed for Industrial Internet of Things,' en, *IEEE INTERNET OF THINGS JOURNAL*, vol. 8, no. 7, p. 16, 2021.

[20] S. Lee, S. Lee, H. Yoo, S. Kwon and T. Shon, 'Design and implementation of cybersecurity testbed for industrial IoT systems,' en, *The Journal of Supercomputing*, vol. 74, no. 9, pp. 4506–4520, Sep. 2018, ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-017-2219-z. [Online]. Available: http://link.springer.com/10.1007/s11227-017-2219-z (visited on 08/11/2020).

[21] J. Munoz, F. Rincon, T. Chang, X. Vilajosana, B. Vermeulen, T. Walcarius, W. van de Meerssche and T. Watteyne, 'OpenTestBed: Poor Man's IoT Testbed,' en, in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France: IEEE, Apr. 2019, pp. 467–471, ISBN: 978-1-72811-878-9. DOI: 10.1109/INFCOMW.2019.8845269. [Online]. Available: https://ieeexplore.ieee.org/document/8845269/ (visited on 13/10/2020).

[22] O. Nock, J. Starkey and C. M. Angelopoulos, 'Addressing the Security Gap in IoT: Towards an IoT Cyber Range,' en, *Sensors*, vol. 20, no. 18, p. 5439, Sep. 2020, ISSN: 1424-8220. DOI: 10.3390/s20185439. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5439 (visited on 01/10/2020).

[23] S. Friedl, M. Glas, L. Englbrecht, F. Böhm and G. Pernul, 'ForCyRange: An Educational IoT Cyber Range for Live Digital Forensics,' en, in *Information Security Education - Adapting to the Fourth Industrial Revolution*, L. Drevin, N. Miloslavskaya, W. S. Leung and S. von Solms, Eds., vol. 650, Series Title: IFIP Advances in Information and Communication Technology, Cham: Springer International Publishing, 2022, pp. 77–91, ISBN: 978-3-031-08171-2 978-3-031-08172-9. DOI: 10.1007/978-3-031-08172-9_6. [Online]. Available: https://link.springer.com/10.1007/978-3-031-08172-9_6 (visited on 13/10/2022).

[24] S. K. Oh, N. Stickney, D. Hawthorne and S. J. Matthews, 'Teaching Web-Attacks on a Raspberry Pi Cyber Range,' en, in *Proceedings of the 21st Annual Conference on Information Technology Education*, Virtual Event USA: ACM, Oct. 2020, pp. 324–329, ISBN: 978-1-4503-7045-5. DOI: 10.1145/3368308.3415364. [Online]. Available: https://dl.acm.org/doi/10.1145/3368308.3415364 (visited on 07/11/2020).

[25] A. Z. Sharifi, V. Vanijja, D. Pal and W. Anantasabkit, 'CyberIoT: An Initial Conceptualization of a Web-based Cyber Range for IoT,' en, in *2021 International Conference on Computational Performance Evaluation (ComPE)*, Shillong, India: IEEE, Dec. 2021, pp. 091–096, ISBN: 978-1-66543-656-4. DOI: 10.1109/ComPE53109.2021.9752401. [Online]. Available: https://ieeexplore.ieee.org/document/9752401/ (visited on 04/10/2022).

[26] P.-W. Tsai and C.-S. Yang, 'Testbed@TWISC: A network security experiment platform,' en, *International Journal of Communication Systems*, vol. 31, no. 2, e3446, Jan. 2018, ISSN: 10745351. DOI: 10.1002/dac.3446. [Online]. Available: http://doi.wiley.com/10.1002/dac.3446 (visited on 09/11/2020).

[27] Y. Berhanu, H. Abie and M. Hamdi, 'A testbed for adaptive security for IoT in eHealth,' en, in *Proceedings of the International Workshop on Adaptive Security - ASPI '13*, Zurich, Switzerland: ACM Press, 2013, pp. 1–8, ISBN: 978-1-4503-2543-1. DOI: 10.1145/2523501.2523506. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2523501.2523506 (visited on 07/11/2020).

[28] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga and S. Hariri, 'A testbed for analyzing security of SCADA control systems (TASSCS),' en, in *ISGT 2011*, Anaheim, CA, USA: IEEE, Jan. 2011, pp. 1–7, ISBN: 978-1-61284-218-9. DOI: 10.1109/ISGT.2011.5759169. [Online]. Available: http://ieeexplore.ieee.org/document/5759169/ (visited on 04/11/2020).

[29] O. Abu Waraga, M. Bettayeb, Q. Nasir and M. Abu Talib, 'Design and implementation of automated IoT security testbed,' en, *Computers & Security*, vol. 88, p. 101 648, Jan. 2020, ISSN: 01674048. DOI: 10.1016/j.cose.2019.101648. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167404819301920 (visited on 21/04/2022).

[30] S. Siboni, A. Shabtai, N. O. Tippenhauer, J. Lee and Y. Elovici, 'Advanced Security Testbed Framework for Wearable IoT Devices,' en, *ACM Transactions on Internet Technology*, vol. 16, no. 4, pp. 1–25, Dec. 2016, ISSN: 1533-5399, 1557-6051. DOI: 10.1145/2981546. [Online]. Available: https://dl.acm.org/doi/10.1145/2981546 (visited on 07/11/2020).

[31] P. Johannesson and E. Perjons, *An Introduction to Design Science*, en, 1st ed. 2014. Cham: Springer International Publishing : Imprint: Springer, 2014, ISBN: 978-3-319-10632-8. DOI: 10.1007/978-3-319-10632-8.

[32] E. Ukwandu, M. A. B. Farah, H. Hindy, D. Brosset, D. Kavallieros, R. Atkinson, C. Tachtatzis, M. Bures, I. Andonovic and X. Bellekens, 'A Review of Cyber-Ranges and Test-Beds: Current and Future Trends,' en, *Sensors*, vol. 20, no. 24, p. 7148, Dec. 2020, ISSN: 1424-8220. DOI: 10.3390/s20247148. [Online]. Available: https://www.mdpi.com/1424-8220/20/24/7148 (visited on 21/04/2022).

[33] M. Seljeseth, M. M. Yamin and B. Katt, 'UIOT-FMT: A Universal Format for Collection and Aggregation of Data from Smart Devices,' en, *Sensors*, vol. 20, no. 22, p. 6662, Nov. 2020, ISSN: 1424-8220. DOI: 10.3390/s20226662. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6662 (visited on 29/11/2022).

[34] A. Fuller, Z. Fan, C. Day and C. Barlow, 'Digital Twin: Enabling Technologies, Challenges and Open Research,' en, *IEEE Access*, vol. 8, pp. 108 952–108 971, 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2998358. [Online]. Available: https://ieeexplore.ieee.org/document/9103025/ (visited on 21/04/2022).

[35] J. Zaddach, L. Bruno, A. Francillon and D. Balzarotti, 'Avatar: A Framework to Support Dynamic Security Analysis of Embedded Systems' Firmwares,' en, in *Proceedings 2014 Network and Distributed System Security Symposium*, San Diego, CA: Internet Society, 2014, ISBN: 978-1-891562-35-8. DOI: 10.14722/ndss.2014.23229. [Online]. Available: https://www.ndss-symposium.org/ndss2014/programme/avatar-framework-support-dynamic-security-analysis-embedded-systems%E2%80%99-firmwares/ (visited on 03/05/2022).

[36] *JTAG - Wikipedia*. [Online]. Available: https://en.wikipedia.org/wiki/JTAG (visited on 23/05/2022).

[37] *The Serial Wire Debug Port - Documentation – Arm Developer*. [Online]. Available: https://developer.arm.com/documentation/ihi0031/a/The-Serial-Wire-Debug-Port--SW-DP-/Introduction-to-the-ARM-Serial-Wire-Debug--SWD--protocol (visited on 23/05/2022).

[38] S. Mackey, I. Mihov, A. Nosenko, F. Vega and Y. Cheng, 'A Performance Comparison of WireGuard and OpenVPN,' en, in *Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy*, New Orleans LA USA: ACM, Mar. 2020, pp. 162–164, ISBN: 978-1-4503-7107-0. DOI: 10.1145/3374664.3379532. [Online]. Available: https://dl.acm.org/doi/10.1145/3374664.3379532 (visited on 24/05/2022).

[39] M. Kim, D. Kim, E. Kim, S. Kim, Y. Jang and Y. Kim, 'FirmAE: Towards Large-Scale Emulation of IoT Firmware for Dynamic Analysis,' en, in *Annual Computer Security Applications Conference*, Austin USA: ACM, Dec. 2020, pp. 733–745, ISBN: 978-1-4503-8858-0. DOI: 10.1145/3427228.3427294. [Online]. Available: https://dl.acm.org/doi/10.1145/3427228.3427294 (visited on 02/05/2022).

[40] D. D. Chen, M. Egele, M. Woo and D. Brumley, 'Towards Automated Dynamic Analysis for Linux-based Embedded Firmware,' en, in *Proceedings 2016 Network and Distributed System Security Symposium*, San Diego, CA: Internet Society, 2016, ISBN: 978-1-891562-41-9. DOI: `10.14722/ndss.2016.23415`. [Online]. Available: `https://www.ndss-symposium.org/wp-content/uploads/2017/09/towards-automated-dynamic-analysis-linux-based-embedded-firmware.pdf` (visited on 02/05/2022).

[41] B. Feng, A. Mera and L. Lu, 'P2IM: Scalable and Hardware-independent Firmware Testing via Automatic Peripheral Interface Modeling,' en, p. 19,

[42] Gyokan O. Osman, 'Emulating the Internet of Things with QEMU,' M.S. thesis, Chalmers University of Technology, Sep. 2019.

[43] Ben Curtis, *GitHub - Fmstrat/diy-ipmi: A DIY IPMI / IP KVM system utilizing the Raspberry Pi*, 2018. [Online]. Available: `https://github.com/Fmstrat/diy-ipmi` (visited on 03/05/2022).

[44] *WB2S/BK7231 Tutorial - writing custom firmware - UDP/TCP/HTTP/MQTT*. [Online]. Available: `https://www.elektroda.com/rtvforum/topic3850712.html` (visited on 29/05/2022).

[45] Nathan Fain, *GitHub - cyphunk/JTAGenum: Given an Arduino compatible microcontroller or Raspberry PI (experimental), JTAGenum scans pins[] for basic JTAG functionality and can be used to enumerate the Instruction Register for undocumented instructions. Props to JTAG scanner and Arduinull which came before JTAGenum and forwhich much of the code and logic is based on. Feel free to branch and modify religiously (readme, credits, whatever)*. [Online]. Available: `https://github.com/cyphunk/JTAGenum` (visited on 18/05/2022).

[46] *Mirai-Source-Code/ForumPost.md at master · jgamblin/Mirai-Source-Code · GitHub*. [Online]. Available: `https://github.com/jgamblin/Mirai-Source-Code/blob/master/ForumPost.md` (visited on 08/11/2020).

Appendix A

Attachments

Additional materials are delivered as attachments to the project report.

- Survey questionnaire form - as distributed
- Survey results - anonymized
- Code - Configuration files
- Code - Scripts
- Code - Templates
- Code - Arduino software

NTNU

Norwegian University of
Science and Technology