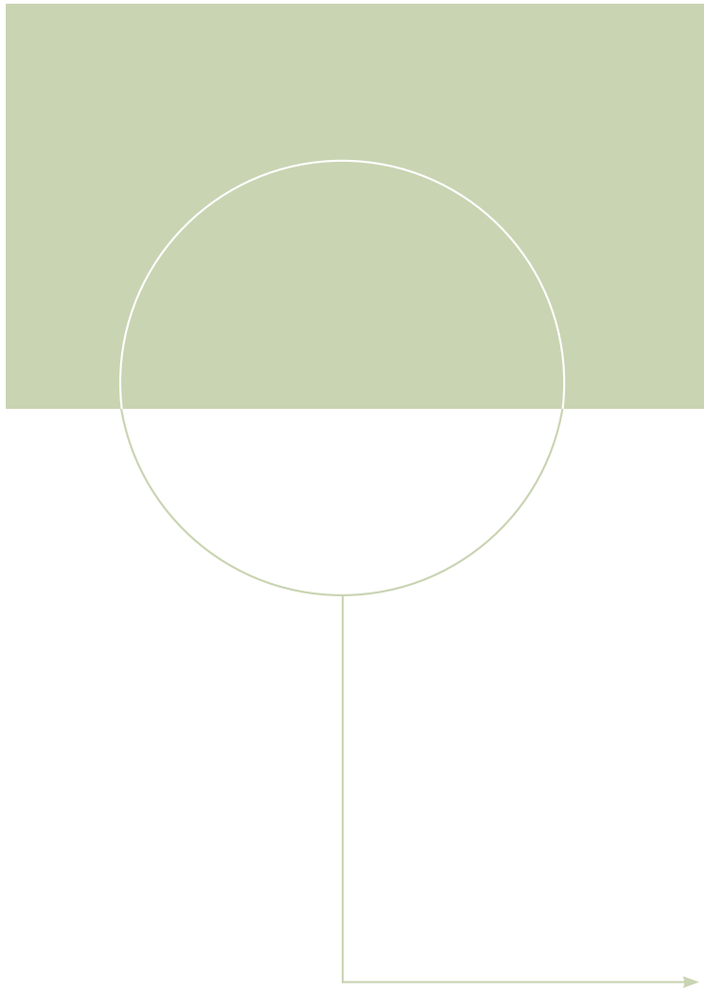

Master's thesis

2021

Emil Myre

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

Emil Myre

PKI and IoT Security: How to choose the most secure implementation?

December 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

PKI and IoT Security: How to choose the most secure implementation?

Emil Myre

Master in Information Security
Submission date: December 2021
Supervisor: Basel Katt

Norwegian University of Science and Technology
Department of Information Security and Communication
Technology

PKI and IoT Security: How to choose the most secure implementation?

Emil Myre

15.12.2021

Abstract

This thesis will look into the combination of the technologies IoT and PKI and how these technologies can be combined to create secure solutions. The enormous variation of products and implementation forms in especially IoT, introduces a risk of implementing vulnerabilities with potential catastrophic outcomes in the event of cyberattacks. The thesis will present an overview and an evaluation of the current solutions and its vulnerabilities. This will also involve practical implementations of PKI technology with associated vulnerability assessments. There will also be a recommendation of PKI solution based on certain criterias for the IoT equipment.

Sammendrag

Denne oppgaven vil fokusere på kombinasjonen av teknologiene IoT og PKI og hvordan disse teknologiene kan brukes sammen for å lage sikre løsninger. Den enorme variasjonen av produkter og implementasjonsformer i spesielt IoT, kan føre til at det implementeres sårbarheter som kan føre til fatale konsekvenser hvis de blir utnyttet under et cyberangrep. Intensjonen med denne oppgaven er å presentere en oversikt og en evaluering av nåværende løsninger og dets sårbarheter. Oppgaven vil også inkludere praktiske implementasjoner av PKI med tilhørende sårbarhetsanalyser, samt en anbefaling av en eller flere PKI løsninger basert på egenskaper til IoT utstyret.

Preface

This thesis was written as part of the Master of Science in Information Security at the Norwegian University of Science and Technology (NTNU) in the Autumn semester of 2021.

I would first of all like to thank my supervisor Basel Katt for guidance and invaluable support through all stages of the thesis. I would also like to thank colleagues for new ideas and discussions surrounding the theme of the thesis as well as my family for providing me with support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without them.

Thank you.

Emil Myre

Contents

Figures

Tables

Code Listings

Acronyms

ARP Address Resolution Protocol. xv

ASN Abstract Syntax Notation One. xv

CA Certificate Authority. xv

CBOR Concise Binary Object Representation. xv

CLI Command Line Interface. xv

CPS Cyber Physical Systems. xv

CRL Certificate Revocation List. xv

CVE Common Vulnerabilities and Exposures. xv

CVSS Common Vulnerability Scoring System. xv

DFD Data Flow Diagram. xv

DNS Domain Name System. xv

DSRP Design Science Research Process. xv

HTTP Hypertext Transfer Protocol. xv

HTTPS Hypertext Transfer Protocol Secure. xv

ICT Information and Communications Technology. xv

IETF Internet Engineering Task Force. xv

IoT Internet of Things. xv

LDAP Lightweight Directory Access Protocol. xv

M2M Machine to Machine. xv

MAC Media Access Control. xv

NIST National Institute of Standards and Technology. xv

NSF National Science Foundation. xv

OCSP Online Certification Status Protocol. xv

PGP Pretty Good Privacy. xv

PKI Public Key Infrastructure. xv

RA Registration Authority. xv

RCE Remote Code Execution. xv

RFC Request for Comments. xv

SaaS Software as a Service. xv

SMTP Simple Mail Transfer Protocol. xv

SQL Structured Query Language. xv

SSH Secure Shell. xv

SSL Secure Sockets Layer. xv

TLS Transport Layer Security. xv

Chapter 1

Introduction

During the last decade, there has been an explosive growth of small physical objects that are equipped with sensors and have the technology to connect and exchange data. The area of use for these objects seem limitless as they can be used in both homes as for instance voice assistants, on body as for instance fitness trackers, in vehicles, at work and as components with specific tasks in todays central infrastructure. These objects are all part of a certain term of network devices which are called the Internet of Things . The expanding volume of IoT devices both in numbers and different variations leads to concerns regarding the ability to protect running applications an data transferred between these products. The Public Key Infrastructure (PKI) is a possible method for securing these devices, however the PKI can have different forms of implementation with unique advantages and disadvantages for each solution. The establishment of a framework with software, policies and procedures is included in the Public Key Infrastructure as a part of establishing security to a system. There are also multiple ways to establish of this kind of infrastructure.

The variety of IoT devices will also have certain characteristics which can make one PKI implementation more secure for one type of IoT, while other types of IoT solutions might be better secured by using another form of PKI implementation. Due to this, there is a risk that implementing the same PKI solution to different types of IoT equipment might create vulnerabilities related to the different variants of IoT devices that are not discovered yet.

1.1 Purpose

The planned contribution of this thesis is to investigate whether how secure PKI solutions that are used in todays IoT devices, and what are their typical weaknesses and vulnerabilities. This will involve practical solutions to be tested in a lab environment. This also involves exposing IoT devices to Cyber attacks. To ensure realistic results, the lab testing should involve relevant IoT devices, relevant PKI solutions and at last relevant attack vectors to be used in the test.

1.2 User scenario

To better understand both the possible advantages and disadvantages of securing IoT equipment with PKI, a user scenario will be presented. This scenario is meant to give the reader an introduction to both IoT and PKI related challenges.

1.3 The case

The user scenario is the following: A hospital is going to invest in new healthcare equipment which include monitoring devices such as remote patient monitoring that collects metrics like heart rate, temperature, glucose levels, but also advanced robots that can be inside a human body and perform complex surgery procedures. In 2018, Norwegian hospitals had to recover from a massive hacker attack with loss of sensitive data and a lot of public attention [1]. Having this in mind, the hospital considers additional security on their new healthcare devices.

If these devices are not properly secured, control over the devices can be lost to unauthorized users who can steal sensitive patient data or manipulate metrics or procedures that can result in loss of human lives.

The IT-department is contacted, and some of the employees recommend using PKI as they have heard that this will enforce the security with encryption of traffic. The employees in the IT-department have only theoretical experience with PKI and never implemented or maintained such a system before. They have however heard that there are several ways to implement such security and are positive to try and fix this themselves.

The hospital are faced with some major decission points. Should they follow the advice of the internal IT-department and trust them to fix this, or should they also contact external experts to validate and improve the security? What should they consider when there are several ways to implement a PKI solution, how much will this cost, and is this truly the best way to secure their systems?

1.4 Research Question

The research questions are the following:

- What are the current PKI solutions for securing IoT devices today?
- What are the vulnerabilities related to securing IoT with PKI?
- Can different PKI solutions be recommended to different IoT solutions and if so, what criteria should be used for this recommendation?

1.5 Methodology

This chapter will present the methodology used in this thesis and how the different processes in this thesis relate to the different parts of the methodology. The

chosen methodology is the Design Science[2] where the objective is to develop and evaluate one or more artefacts as a part of the process of solving a specific problem. The unique artefacts can be a model, a method or a prototype where the goal is to develop knowledge that can be useful for both the understanding and solution of the problem.

1.5.1 Problem Definition

In this thesis, one of the artefacts will be the lab for implementing and evaluating a PKI solution on an IoT device. This lab can be implemented in several ways, and each implementation will include a threat model and a vulnerability analysis that also can be considered artefacts. With these artefacts, the method addresses the unsolved problem of finding the most secure PKI solution for IoT devices. The enormous growth of IoT devices both in numbers and different variations leads to concerns regarding the security of these devices. The Public Key Infrastructure is a possible method for securing these devices, however the PKI can have different forms of implementation with unique advantages and disadvantages for each solution. The variety of IoT devices will also have certain characteristics which can make one PKI implementation suitable for one type of IoT, while other IoT devices might favor another form of PKI implementation. Due to this, there is a risk that implementing the same PKI solution to different types of IoT equipment might create vulnerabilities related to the different variants of IoT devices that are not discovered yet.

1.6 Process model

The Design Science methodology was not that often used in research environments after its publication. Some researchers pointed out that the reason for this could be because the methodology lacked a conceptual model for how the researchers actually could carry out their design science research in a way that would be recognized and evaluated by their readers. To mitigate this, a process model called Design Science Research Process (DSRP) were developed in 2006 [3]. The process model intended to meet the following three objectives:

- To be consistent with prior literature
- To provide a mental model for presenting and appreciating DS research
- To provide a mental model for presenting and appreciating DS research

As shown in figure 1.1, the following six activities are essential in the DSRP model:

A description of both the activities and a how these are applicated to the thesis, is the following:

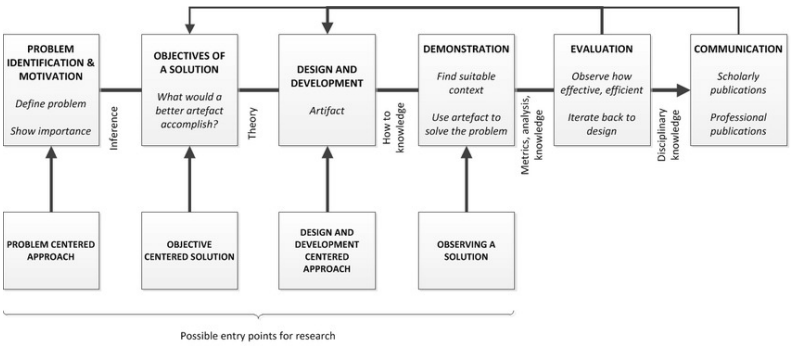

Figure 1.1: DSRP Model [3]

1.6.1 Activity 1: Problem Identification and Motivation

The first activity involves establishing the problem to be addressed and to justify the research based on the perceived benefits of the resulting artefacts. In this thesis, the first activity is applied to the methodology through chapter 1, where the problem is identified as how to find the most secure PKI solution for IoT devices. In this problem establishment, the fictive case study at the hospital plays an important role in both identifying the problem as well as motivating for a solution of the problem.

1.6.2 Activity 2: Objectives of a Solution

This involves creating the a objective of a solution where the researcher is required to define the objectives which will be based on the problem to be solved. In this thesis the objectives will be PKI solutions and IoT devices, while the problem to be solved will be a recommendation of one or more PKI solutions to be implemented on IoT devices.

1.6.3 Activity 3: Design and Development

The activity involves the creation of the artefacts. In this thesis, the third activity will be the design and development of a lab for implementing and evaluating PKI solutions on IoT devices with included threat models and vulnerability analyzis.

1.6.4 Activity 4: Demonstration

This activity includes a demonstration of the artefacts in an appropriate environment which shows that this solves the stated problem. This activity is applicated in this thesis by demonstrating that the implemented PKI solution in lab works as expected.

1.6.5 Activity 5: Evaluation

In this activity the performance of the artefact is reviewed with reference to the one or more stated objectives in activity 2. As figure 1.1 shows, this evaluation may lead the researcher to consider further design and development of the artefact. The researcher may then go back to activity 3 as part of an iterative and improvable process. The vulnerability evaluation in this thesis will relate to this activity, as it is a possibility to implement and evaluate several PKI solutions in this lab.

1.6.6 Activity 5: Communication

The final activity is to publish the researchers work in order to enhance the body of knowledge in the related field.

The communication activity in this thesis can however not be completed by the time of the submission. After the submission there is a possibility of publishing the thesis in relevant forums. Another aspect is that the thesis is regularily discussed with both colleagues at work and supervisors at school which also is a part of enhancing the knowledge in the field.

Chapter 2

Background

In relation to the methodology used in this thesis, this chapter will address activity 2 in DSRP, where the objects in a solution are defined. As earlier mentioned, the objectives in this thesis will be PKI solutions and IoT devices.

To be able to provide the necessary knowledge and background, a literature review was conducted. The aim is to present a background on the Internet of Things followed by the history of the Public Key Infrastructure as well as the possible forms of implementation. At the end of the chapter there will be an overview of related work to this thesis as well as challenges with PKI and IoT.

The sources for this literature review have been identified by using open sources as researchgate.net, scholar.google.com and ieeexplore.ieee.org. To ensure updated research material, most of the relevant publications have been narrowed down to a publish date after 2018. Some sources have also been identified as companies with interests of selling own products, which leads to caution and a need of verifying information from several sources.

The following free-text search terms were used in various combinations and forms: Public Key Infrastructure, Internet of Things, PKI, IoT, challenges, vulnerabilities, X.509, Blockchain

2.1 Internet of Things

In 1982, a Coca Cola vending machine received public attention. The reason was that this machine could actually report its inventory through a network. This is one of the first traces of objects that fit under the description as an Internet of Thing. There is however hard to find a specific definition of the technology described as the Internet of Things. A commonly used description is that IoT is a network of physical objects equipped with software and sensors with the possibility to connect and exchange data with other devices over a network. Similar systems used for describing IoT are Cyber physical systems (CPS), Machine to Machine (M2M), Industrial Internet, Smart cities, Smart Grids, Smart Homes and other. The US independent federal agency named National Science Foundation (NSF), says that

Cyber physical systems integrate sensing, computation, control and networking into physical objects and infrastructure, connecting them to the Internet and each other [4]. This is a broad generalization which means that a very large range of products and technologies are included this description. A common factor of these products is that they consist of an embedded system. An embedded system is a concept used to describe computer hardware which is built into mechanical devices to make the devices capable of digital computation [5].

In order to separate IoT devices from devices from other categories like "Personal Computers" belonging to Information and Communications Technology (ICT), the following characteristics has been listed up [6]:

Interconnectivity; Anything can be interconnected with the global information and communications infrastructure, from an IoT point of view. For IoT devices they do not necessarily need to communicate over IP networks, but also utilize serial protocols like MODBUS and DNP3 or low-powered wireless links with protocols like ZigBee, Bluetooth Low Energy,6LoWPAN and many others.

Things-related services; IoT provides things-related services within the constraints of things. This describes the IoT devices ability to observe and attempt to control specific variables in the physical world. In order to to this, there is often satisfactory to use embedded systems with limited resources. These systems do not need to have an operating system, but can run on only firmware as long as they can perform the tasks they are designed to perform. These devices are also often Real-Time Systems with time-sensitive components and time constraints related to the performance of the device.

Heterogenity; The IoT devices consist of a variety of hardware platforms, firmware, operative systems and network protocols. Due to this diversiness, there are challenges concerning interoperability and standardisation of IoT devices.

Dynamic changes; The status of IoT devices can change dynamically when it comes to being connected or disconnected. The physical location and purpose of such a device can also change several times during its lifetime, due to both technological developments and changing areas of use.

Enormous scale; The number of IoT devices has exploded during the last decade, and the predictions are that this market will continue to grow with around 25 percent the next decade. Even though the Covid-19 virus led to supply chain disruptions in the manufacturing of seminconductors, it also led to higher demands of IoT devices in the health industry [7].

2.2 Public Key Infrastructure

The development and revelation of the public key cryptography to the public crowd started in the the decade following 1970. Prior to this, the preferred choice were symmetric key algorithms which used the same cryptographic keys for both encryption and decryption. Central in the development of both asymmetric key algorithms and the concept of private and public keys, were cryptographers like Diffie, Hellman, Merkle and Rivest. The system consist of pairs of both private and public keys, where the private key is only known to the owner of the key and the public key is known to others and can be distributed through insecure channels. This public key can then be used to provide confidentiality to a message by letting the sender encrypt it with the use of both the receivers public key and the senders own private key. The receiver can then decrypt this message with its own private key. Compared to the use of symmetric keys, this method will unfortunately lead to a greater computational cost and less data throughput. Because of this, the use of symmetric keys in cryptographic systems is not totally excluded. Both symmetric and asymmetric algorithms can coexist in systems where the public key cryptography is used to exchange symmetric keys, and the symmetric keys are used to protect large data transfers. To also provide authentication of messages, the public key cryptography was further developed to let the sender identify himself by encrypting only a small portion of the message with a public key. This portion would then be a digital certificate from the sender and by that prove the senders identification. The use of public key cryptography is also used to encrypt checksums of the original message, which provides the integrity protection of an message. These encrypted checksums are referred to as digital signatures[8].

2.2.1 X.509

The organizations Internet Engineering Task Force (IETF) and National Institute of Standards and Technology (NIST) then developed a series of Request for Comments (RFC) which resulted in the industrial standard X.509 [9]. This insured standardisation and reduction of variety to the public key cryptography. The first version of X.509 was released in 1988. To include further improvements of the certificate, the second and third version of the standard was released in 1993 and 1996. The standard X.509 is also the basis for protocols like TLS, SSL, HTTPS, SMTP, LDAP and several more.

Figure 2.1 shows the structure of the X.509 certificate, where the issuer and subject unique identifiers were added in version 2, while the extensions field were added in the version 3 and gave the possibility to issue certificates with specific purposes.

Information about how lists with invalid or revoked certificates should be distributed in PKI is also defined in the X.509 standard. These certificate revocation lists (CRL) are then used as a part of the verification process of a presented certificate. The Online Certification Status Protocol (OCSP) is a later introduced al-

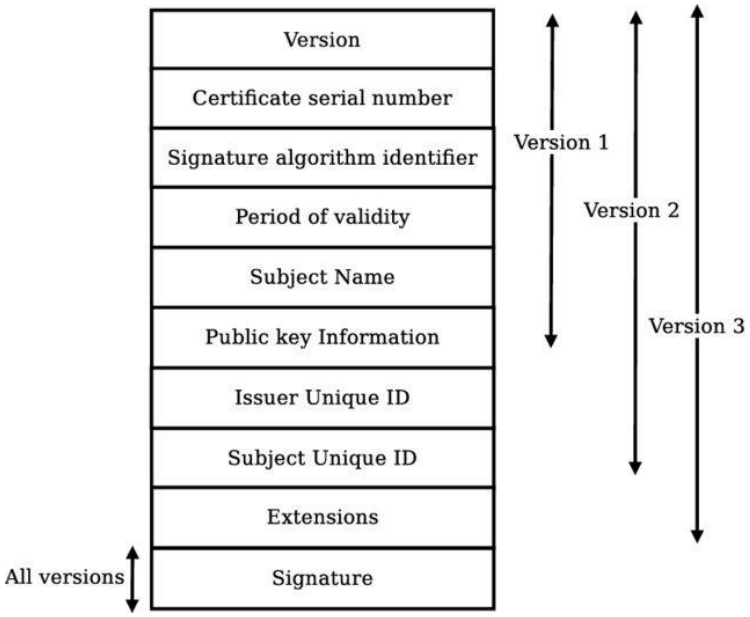

Figure 2.1: The X.509 Certificate

ternative to the CRLs. OCSP is an internet protocol where revocation messages are typically transferred via HTTP and it contains less data than CRL. It is not mandatory to use encryption in OCSP, and it does not send a list over all revocated certificates, only the status over the specific certificate in questioning.

PKI is the system for creating, storing and distributing these certificates, and consist of the following roles [10]:

A certificate authority (CA); The digital certificates are signed, store and issued by a CA

A registration authority (RA); The identity of the users that requests certificates from CA need to be verified from a RA before the CA can issue the certificates.

A central directory: The central directory is a safe location where keys can be stored and indexed.

A certificate management system; The access to stored certificates are managed by the certificate management system.

A certificate policy; To allow third parties to analyze the PKI trustworthiness of the PKI, there is a need for a certificate policy which states requirements and procedures for the specific PKI.

2.2.2 Trust

The certificate policy refers to trustworthiness, where the chain of trust is a concept that needed to be developed in PKI models to make sure the digital certificates could be verified as legit. The root CA is the trust anchor for digital certificates, but there are several digital certificates that cross domains and end up in domains with a new root CA. This development has led to different types of PKI architectures, for instance are Single-CA, hierarchical, mesh and hybrid some of the structures that were developed and used in PKI trust models. The focus in this models is to establish relations between the different trust anchors without loss of performance in the system. It is also typical to differentiate trust levels at a national and governmental level depending on the trust model that is used. These levels are typically ranged from a low to a very high assurance level[11].

2.2.3 PGP

Alongside the development of PKI, Paul Zimmermann developed the protocol Pretty Good Privacy (PGP) in 1991. This protocol combines the use of symmetric and asymmetric keys and is often used for encryption of e-mail messages. Compared to PKI, there is no Central Authority in PGP, and the participants need to both sign and verify each others keys and build a network of trusted connections progressively. This network is called the Web of Trust. [12]

2.2.4 Blockchain

The introduction of the cryptocurrency called Bitcoin in 2008, led to a lot of public attention to the technology known as Blockchain. Since that, this technology has proven to have other areas of use, for instance in establishing smart contracts, securing financial transactions and even as a method to purchase retails in games. Blockchain technology consist of a distributed ledger that maintains a list of order records. These records are part of so-called blocks that, beside the collected transaction records at a certain time period, also contains a timestamp and a link to the previous block. The link to the previous block will also contain a cryptographic hash of the previous block to verify its integrity. In a Blockchain system there will be both users and miners, where the users are the participants who creates transactions, while the miners are the participants who verify and create transaction blocks. The transaction blocks needs to be created in a certain procedure an the miners are often paid each time they make a successfull block. Even though there are hybrid solutions, the traditional Blockchain systems are divided into three main groups:

A public, fully decentralized Blockchain system; This system is permissionless, transparent and without access restrictions. This gives the advantage of being independent of organizations, but also disadvantages concerning the speed and scalability of the system.

A consortium Blockchain system; In this system there are strict rules for how to select a certain group of miners that can participate. This system tends to be more scalable and efficient than a public system, but with less transparancy.

A private Blockchain system; This is a centralized system where there are just one organization that determines the members of the network. This system is often small with the ability to quickly process transactions. On the other hand this network might be less secure due to its centralized nature.

A common factor for all of these groups, is that the first block of the ledger is generated by the creator of the system and is called a genesis block. The following block will contain a link to its previous block, and this will give a linear history of activities that is reliable and traceable all the way back to the genesis block.

2.3 Related work

As IoT devices has made their continuous growth and development in the global market, the research around finding the optimal PKI solution for these devices have been a similar continuous process. The author could however not find research that focuses on comparing different PKI solutions for IoT with regards to related vulnerabilities, nor research that recommends different PKI solutions based on certain characteristics in the IoT solutions.

There has however been published research that focuses on just some of the characteristics. One example is the constrained resources in IoT devices, where a more light-weight X.509 implementation is suggested[13][14]. In this implementation the technique called Concise binary object representation (CBOR) is used instead the standard ASN.1 notation in the certificate profile. In this technique, there is a fixed asymmetric algorithm based on elliptic curves for all certification bodies and certificates. In addition, the certificate can be further compressed by using a DTLS Profile when travelling over constrained networks.

Another example, is the research on decentralization in the PKI structure for IoT devices. In this context, the Blockchain technology have seemed particularly interesting because of the possibility to decentralize the PKI structure for IoT devices with distributed edge devices and nodes. These edge devices and nodes can prevent the infrastructure to suffer from single point of failures when taking the role as a distributed CA for the IoT devices. This system is called IoT-PKI.[15]. There was however not found examples of practical implementations of this system, or a vulnerability assessment that this thesis also will try to answer.

Practical implementations of Blockchain technology in IoT devices was also quite difficult to find in research papers. In 2018, Magnusson tried to validate several Blockchain algorithms in IoT devices, but concluded with a negative answer because of high synchronization times and unstable software[16]. Following this research, there has also been proposed a Blockchain based PKI that is claimed to be optimal for IoT devices. This is the BlockQuick protocol, and according to

the author of this research, this protocol will need less data and time to validate transactions than earlier developed Blockchain technologies[17]. Compared to this thesis, the research found regarding Blockquick is also without focus on possible vulnerabilities related to this protocol.

2.4 Challenges for IoT with PKI security

The company Keyfactor lists the following top challenges in IoT Security[18];

- Lack of standards and retrofitted legacy devices
- Weak passwords, authentication, patches and updating regimes
- Unsigned firmware-versions

The same company recommends using PKI in IoT Security with the Software as a Service (SaaS) model that offers PKI-as-a-service. This is a software distribution model where a third party provider hosts the actual application. Besides Keyfactor, there are several other software companies that offers to sell and host PKI services to their customers. Other challenges related to PKI security are the following:

Trust: A challenge regarding thi PKI-as-a-service model is the trustworthiness of this external software companies. Can they be trusted, and what criteria is this trust based on? There has been published research that, due to unprecise specification documents, some of the trust assumptions used in PKI can create misunderstandings and possibly misuse of trust within systems relying on PKI Security[19].

Resources: Another challenge related to PKI security in IoT, is that size and computational cost of a standard X.509 implementation demands too much resources than what is available in several IoT devices.

Management of certificates: The management of certificates in IoT devices also remains an issue. Several IoT devices may often change state between off- or on-line, which means that the communication channel to the controlling authority will not always be present. In these cases, crucial information regarding updatet CRL's will not always be present for the IoT devices. Instead of placing all the trust in one central certification authority, there has been published several research where the use of Blockchain-based technology is proposed as an alternative [13][20].

Blockchains lacking Proof of Concept: Even though the Blockquick protocol has been developed, there is still uncertainty around the Proof of Concept and practical implementations of this protocol. There is also uncertainty regarding whether the public, consortium or private group is the best alternative for securing IoT devices.

The mentioned challenges in this section will all have to be taken into consideration when the next activity in the DSRP is started.

Chapter 3

Design and threat modeling

In relation to the methodology used in this thesis, this chapter will address activity 3 in DSRP, which involves the creation of the artefact. In this thesis, the third activity will be the design and development of a method for implementing and evaluating a PKI solution on an IoT device in a lab environment.

As mentioned in the introduction of this thesis, the lab testing should involve relevant IoT devices, relevant PKI solutions and at last relevant attack vectors to ensure realistic results.

3.1 IoT Hardware

To find the relevant IoT device for this thesis, the choice was based on both earlier research described in the related work section, as well as searches online for relevant articles on IoT devices. A device that is mentioned and recommended very often, is the series of single board mini computers called Raspberry Pi. Several articles refers to the use this device in IoT solutions[21][22].

The hardware used in this lab is Raspberry Pi 3 Model A+ shown in figure 3.1. This Raspberry model has the following specifications:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 512MB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2/BLE
- Extended 40-pin GPIO header
- Full-size HDMI
- Single USB 2.0 ports
- Micro SD port for loading your operating system and storing data

The Raspberry Pi ran the 32 bit version of the Raspbian OS based on Debian Buster, which is the primary Raspberry Operative System consisting of a Linux distribution composed of free and open-source software. The operating system was downloaded from the official vendor site of Raspberry [23] and installed on an 16 GB micro-SD card by the use of Raspberry Pi Imager v1.6.1.

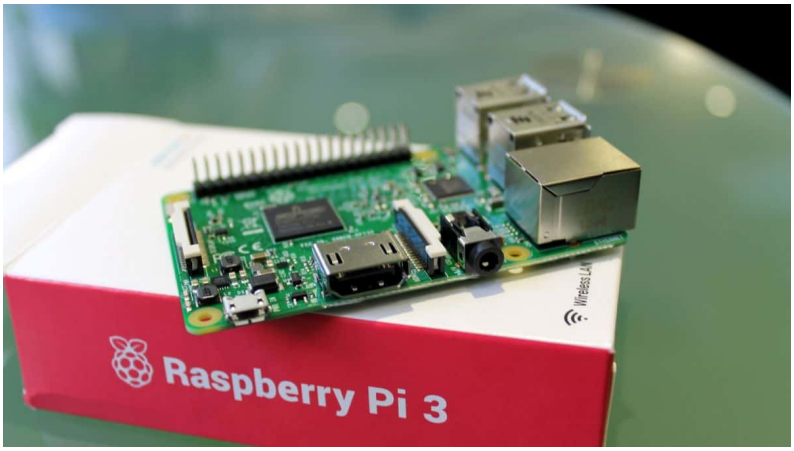

Figure 3.1: Raspberry

When freshly installed, the raspberry performance is a CPU idling around 1 percent and having used 206 of 427 Mb memory.

3.2 Lab setup

In order to implement and evaluate relevant PKI solutions, the choice of PKI solutions were based on both the literature review and searches online. There were established two labs with different PKI technologies; the Blockchain technology with the BlockQuick protocol and the traditional X.509 CA based technology. In both labs there will be on web server host running on one of the raspberries, while the other raspberries will be trying to access this webservice and the data beyond.

3.2.1 Lab 1 Blockchain

The software used for testing the blockchain, was the Diode client written in GO and available at GitHub [24]. This Diode Client runs the Blockquick algorithm and a socks server to transmit data through a diodechain mesh network. The following services can be run with this software[25]:

Website hosting; The client can host secure websites without central services like DNS.

Remote SSH; The client allows for users to SSH into systems regardless of network topologies inbetween.

Video and data streaming; The client can also publish secure video streams.

The services can be published through the following access levels:

Public; The client can host services from the diode software without being dependant of central services in the Internet like DNS.

Private; For only certain diode clients to access services from the diode software, the publish option -private to publish it so that only a single specified Diode address can access it.

Protected; For only diode clients listed in the same Fleet Contract to access services from the diode software, the option -protected is used.

To manage several clients, a Fleet Contract can be established in the Diode Network where clients can be added or removed from different fleets. The administrator of the fleet will then need to install a crypto wallet in form of an application called Metamask [26].

3.2.2 Lab 2 X.509

For the X.509 implementation, there are several open source implementations to choose between online. A tool that is referred in regards to Linux distributions in several articles[27][28] is the Openssl toolkit, which is a command line tool that can be used for a variety of things related to X.509 implementations[29]. This tool has been under continous development from 1999 and seems to be a very relevant tool for implementing X.509 in this lab.

3.3 Threat model

In order to identify and assess both relevant attack vectors and vulnerabilities related to a system, it is often recommended to use a threat model which also includes countermeasures that could be applied. According to Microsoft, a threat model consist of the following five steps [30]:

Define; First step is the process of defining security requirements to the system

Diagram; The second step is the process of creating applications diagrams of the system

Identify; The next step is to identify the actual threats

Mitigate; The threats that are identified, the need to be mitigated

Validate; The final step is to validate that the threats actually have been mitigated

The threat model should be used repeatedly throughout the lifetime of a system.

For the practical implementation of a threat model, there are different methodologies with unique strengths and weaknesses that can be used. The choice of threat methodology often depends on the focus of an organization. Examples of these methodologies are the following[31]:

Figure 3.2: Threat modelling

OCTAVE; The Operationally Critical Threat, Asset, and Vulnerability Evaluation methodology, which was one of the first models to developed and where the focus is not so technical, but more based on practice and organizational risks awareness.

Trike; The Trike model uses a risk-based approach where the focus is to ensure an acceptable level of risk for certain assets in the system

PASTA; The Process for Attack Simulation and Threat Analysis uses a combination of attacker-centric perspective on potential threats together with risk and mitigation analysis

VAST; The Visual, Agile, and Simple Threat methodology focuses on the whole enterprise where there are own application models for the development teams as well as operational models for the infrastructure teams

STRIDE; This is Microsoft's threat model which is developer focused and aims to fulfill the security requirements of the CIA triangle, Confidentiality, Integrity and Availability. The model also focuses on security requirements of Authorization, Authentication and Non-repudiation.

In 2018, Microsoft developed its own Microsoft Threat Modeling Tool which was released free of charge to the public[32]. This tool will be used in this thesis as a part of the threat modeling. This threat model uses STRIDE in its methodology. STRIDE is also preferred to be used in this thesis because it focuses on security in a system, and that it is recommended for inexperienced threat modelers due to its wide perspective and the ability to see threats from the attackers view[33].

3.3.1 STRIDE

The STRIDE methodology is named out of the acronyms from its following threat categories: [34]:

Spoofing identity; This category violates the confidentiality of a system and involves the use of another user's authentication information to gain illegal access. Username and passwords are typical authentication data, but it can also include MAC, ip adresses and processes performed in the system.

Tampering with data; This category violates the integrity of a system and contains malicious modification of data, which can be altered both during transport over a network, while data is being processed or while data is being stored in a data storage.

Repudiation; This involves users who perform illegal actions and where these actions can not be traced or proven afterwards. The system's ability to avoid and counter such threats are referred to as nonrepudiation.

Information disclosure; This category violates the confidentiality of a system and involves the exposure of information to unauthorized users.

Denial of service; This category violates the availability of a system by denying access for authorized users.

Elevation of privilege; This category can violate both the confidentiality, integrity and availability of a system when priveleged access is gained by an unpriveleged user who can use this access to exploit the system totally .

3.3.2 Implementing the Threat Model

The implementation of this model is done by following the five steps of figure 3.2

Define

First step is the process of defining security requirements to the system. There are three basic security requirements to the system;

Confidentiality; That information is not made available or disclosed to unauthorized users.

Integrity; That data cannot be modified in an unauthorized or undetected way.

Availability; That the information must be available when it is needed

To fulfill both these basic requirements and the purpose of this thesis, there is a specific requirement that Public Key Infrastructure is used in the design. The security object in this system, is the raspberry which hosts the secure website. Another specific requirement is that login to this device shall be password protected.

Diagram

The second step is the process of creating applications diagrams of the system. This is done with the Microsoft Threat Modeling Tool, which is used to draw a Data Flow Diagram (DFD) of the system. A DFD is a graphical representation of source, destination and storage of both the incoming and outcoming data flows in the system. In a DFD, source and destination are described as entities, and represented as a rectangle in the drawing. If there are any tasks performed on the data in the system, these are called processes and represented with a circle in the DFD.

Data storages in the system are represented with a rectangle with missing sides, which also includes eventual databases in the system. Finally the actual data flows in the system are represented with arrows. The Microsoft Threat Modeling Tool also includes Trust Boundaries, where the data changes its trust level and that is represented with dotted red lines.

Figure 3.3 shows the Data Flow Diagram in lab 1 where the raspberries are placed inside a Local Area Network, but the dataflows still have to go through the Diode network and cross several Trust Boundaries in order to get from the client entity to get to the data store entity behind the web service.

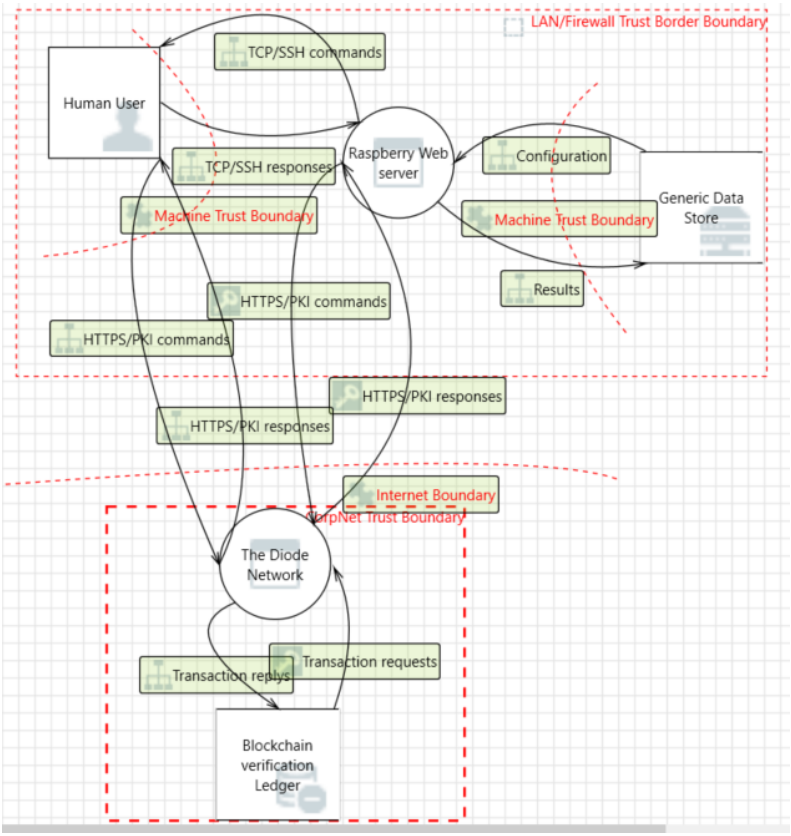

Figure 3.3: Data Flow Diagram Lab 1

Figure 3.3 shows the Data Flow Diagram for lab 2. This Data Flow Diagram is however a lot less complicated since the data flows goes directly from the client entity to the data store entity within the Local Area Network.

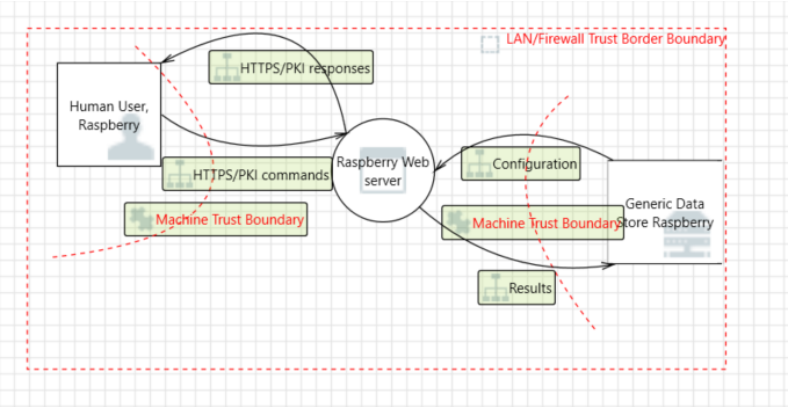

Figure 3.4: Data Flow Diagram Lab 2

Identify

The next step is to identify the actual threats, which also in this case can be done with the use of the Microsoft Threat Modeling Tool. By choosing "Analysis view" in the tool, a list of potential threats to the system is generated based on the inputs in the DFD. The list will sort all potential threats to the system into the categories listed in the STRIDE methodology and also associate a description of each threat as well as a status and modification log.

For Lab 1 the Microsoft Threat Modeling Tool identified a total of 65 potential threats to the system, and for Lab 2 the tool identified 29. This clearly shows that the number of potential threats to the system will increase accordingly to the number of objects in the DFD.

Mitigate

The potential threats that are identified, then need to be mitigated in the design phase as a part of the process of assessing attack vectors. In the Microsoft Threat Modeling Tool, each potential threat in the system can have the following states:

Not Started; Which indicates that the potential threat is not assessed yet

Not Applicable; Which indicates that the potential threat is assessed and found not relevant for this system.

Needs Investigation; Which indicates that the potential threat is found relevant, but that there is still not found a solution for mitigating the threat.

Mitigation Implemented; Which indicates that the potential threat is found relevant, and that there is both found and implemented a solution that will mitigate this threat.

As an example from this lab, the following potential threat is assessed:

Threat ID: 21

Weak Access Control for a Resource; [State: Not Started] [Priority: High]

Category; Information Disclosure

Description; Improper data protection of Generic Data Store Raspberry can allow an attacker to read information not intended for disclosure. Review authorization settings

Justification: No mitigation provided

Short Description: Information disclosure happens when the information can be read by an unauthorized party.

This threat will be mitigated by implementing authentication mechanisms to the resource, for instance by adding password protection to the resource in the design.

When all the potential threats in the system is assessed, it is time to implement the system and proceed to the next step in the threat model.

Validate

The final step is to validate that the threats actually have been mitigated. This is typically done with a vulnerability analysis of the system, which can be done in several ways. If this analysis shows that a threat with the state mitigated is still a valid threat, the state need to be changed in the Threat Modeling Tool, typically to "Needs Investigation", which require further implementation and validation steps to the system.

3.4 Vulnerability analysis

The actual vulnerability analysis will consist of testing that the potential threats identified in the threat model are really mitigated, or if it is possible to prove a presence of vulnerabilities that supposedly should have been mitigated, or not yet discovered in the threat model. This testing will however not prove the absence of vulnerabilities in the system. The analysing equipment will be placed inside the local network of a system, but will still appear to be an external attacker with no knowledge of the system. For each lab, a list of relevant and potential threats from the threat model will be established as well as list for test cases that are used to evaluate the threats.

3.4.1 Test activities

The testing activities will typically be done in the following order, but does not necessarily need to be completed.

Planning and reconnaissance; This involves the planning of goals for the test, target systems, test method as well as intelligence gathering towards the system.

Scanning; This involves identifying open ports and services on a system that can be mapped to ip addresses. It also includes the identification of potential vulnerabilities related to machine configuration.

Gaining Access; Potential vulnerabilities may be exploited in order to gain access to the system. This can be done by cracking passwords, using backdoors or SQL injection.

Maintaining access; Access can further be used to achieve persistent presence in the system and to cover tracks of the initial exploitation.

Analysis; This should include results from the vulnerability analysis, where the focus is on which vulnerabilities that could be expoited and what data could be accessed. It should also include countermeasures that could be used to avoid the unauthorized access.

The earlier developed threat model used in the design part of a system, can be used as a part of all these activities, but it will however require that the performers of this analysis also have access to the original design. As an alternative, a new threat model can be made based upon findings in the analysis.

3.4.2 Software

For the vulnerability analysis, a laptop with a virtual instance of Kali Linux is used. Kali Linux is an open source, Debian based platform that was released in 2013 with intents to be a common toolbox used in Security Research, Computer Forensics and Penetration Testing of systems[35].

The platform contains several hundred penetration testing tools that can be used for vulnerability analysis. Research online show that some of the most popular and relevant tools are the following[36][37][38][39]:

ARP Scan Tool and NMAP; These tools are often used to get information on ARP packets, IP addresses and operating systems used inside a network.

Wireshark; This is a packet analyzing tool with a own GUI to better filter and organize data which go through the network.

Nessus; This is a remote vulnerability scanner that can scan computers and raise alerts if typical vulnerabilities are discovered

Metasploit; This is a framework that can be used to exploit vulnerabilities and inject code to open backdoors in a system.

Chapter 4

Demonstration and evaluation

In relation to the methodology used in this thesis, this chapter will address both activity 4 and 5 in DSRP. This involves a demonstration and evaluation of the artefact that is created. In this thesis, this will mean a demonstration of practical implementations that shows that the implemented PKI solutions in IoT devices works as expected in the lab. This part of the thesis also includes the a vulnerability assessment of various solutions involving PKI and IoT. In addition to activity 4 in the DSRP methodology, this part will also address step 5 in the threat model, where the validation of mitigated threats from the design phase is performed.

4.1 Lab 1 Blockchain

4.1.1 The implementation

The Diode client was implemented by running the following command from a terminal window at the raspberries:

Code listing 4.1: Diode CLI .

```
// The actual installation of software:
# curl -Ssf https://diode.io/install.sh | sh

// Local services on the client can be published with the following commands:
# diode publish -public 80:80
# diode publish -private 80:80,<authorized address>
# diode publish -protected 80:80 -protected 22:22
// The result of a publishment is the following feedback in the terminal window:
# pi@raspberrypi:~ $ diode publish -protected 80:80 -protected 22:22
# INFO Diode Client version : v0.12.5 24 Aug 2021
# INFO Client address       : 0x5ca2d0f14dec39aa0500241980b72caa56944d4d
# INFO Fleet address        : 0x6000000000000000000000000000000000000000
# INFO Network is validated, last valid block: 2606228 0x00003f31044cba9ec6ec1..
# INFO Port      <name>     : <extern>     <mode>    <protocol>     <allowlist>
# INFO Port localhost:80    :       80   protected       any
# INFO Port localhost:22    :       22   protected       any
// 0x6000000000000000000000000000000000000000 is default fleet for new diode clients
// To change the fleet contract for a diode client:
# diode config -set fleet=0x75c2bddbae0efa836138b832b915e6a27488e676
```

The implementation and publishment of a local webserver was tested in several ways. The Diode websites suggests the following configuration to publish static content in a tiny webserver:

Code listing 4.2: Diode Static content .

```
# mkdir project
# cd project
# echo "Hello World" > index.html

# diode publish -http
// The result of a publishment is the following feedback in the terminal window:
# Diode Client version : v0.12.5 24 Aug 2021
# INFO Client address       : 0x035a7d47d7989136676a7863fe590ae6d2586a90
# INFO Fleet address        : 0x6000000000000000000000000000000000000000
# INFO Network is validated, last valid block: 2612139
0x000052283c623b480d0c16e7a9466603f636c887fef5160e8c601955d04cf7bf
# INFO
# INFO HTTP Gateway Enabled :
http://0x035a7d47d7989136676a7863fe590ae6d2586a90.diode.link/
# INFO Port      <name>     : <extern>     <mode>    <protocol>     <allowlist>
# INFO Port        :8080    :       80      public       any
```

The URL in the result can then be accessed from a web browser from any client connected to internet. However, the main drawback with this configuration is that this can not be published at the protected or private access levels where any attempt results in an error message.

The diode websites also suggest the use of the web hosting platform Ghost, which also requires the nodejs and npm packages. The installation of this platform did however fail and the raspberry freezed and needed a hard restart without completing the installation.

As a third option, the lightweight web server application called NGINX was installed with the use of the following configuration:

Code listing 4.3: NGINX .

```
sudo apt-get update
sudo apt-get install nginx
sudo /etc/init.d/nginx start
```

With this webserver installed, it is possible to publish at also the protected and private access levels.

A fleet contract is also established by the use of the METAMASK application, and the diode addresses of the Raspberry clients are added to the fleet via the diode network as shown in figures 4.1 and 4.2.

Each transfer in and out of the fleet requires a transaction confirmation in the Metamask and is shown in figure 4.3.

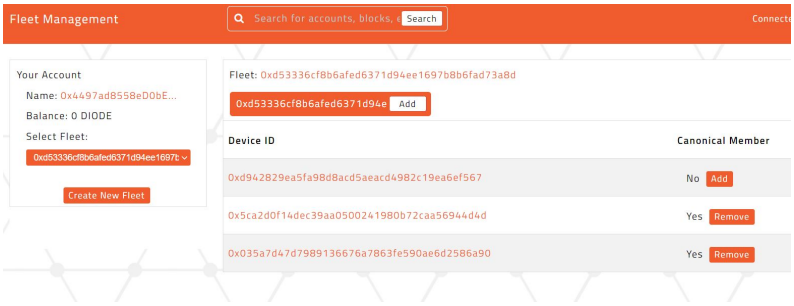

Figure 4.1: Diode Fleet Overview

Figure 4.2: Diode Network Overview

SSH is enabled and published on the raspberry with the following commands:

Code listing 4.4: Diode ssh.

```
//Enable SSH
sudo systemctl enable ssh
systemctl start ssh

// SSH is then published on the diode network with the following command:
diode publish -public 22:22

//This can be accessed through via the Diode network with the following command:
ssh -o "ProxyCommand=nc␣-X␣5␣-x␣diode.link:1080␣%h␣%p"
<user>@<client_address>.diode
// In this lab it will be the follwing
ssh -o "ProxyCommand=nc␣-X␣5␣-x␣diode.link:1080␣%h␣%p"
pi@0x035a7d47d7989136676a7863fe590ae6d2586a90.diode

//The SSH can also be accessed directly between two clients on the network:
Host
diode socksd
diode publish -private 22:22,<authorized_address>
ssh -o "ProxyCommand=nc␣-X␣5␣-x␣localhost:1080␣%h␣%p" <user>@<host_address>.diode

// To change to a different fleet contract:
diode config -set fleet=0xd53336cf8b6afed6371d94ee1697b8b6fad73a8d
```

The results of the implementation is in table 4.1

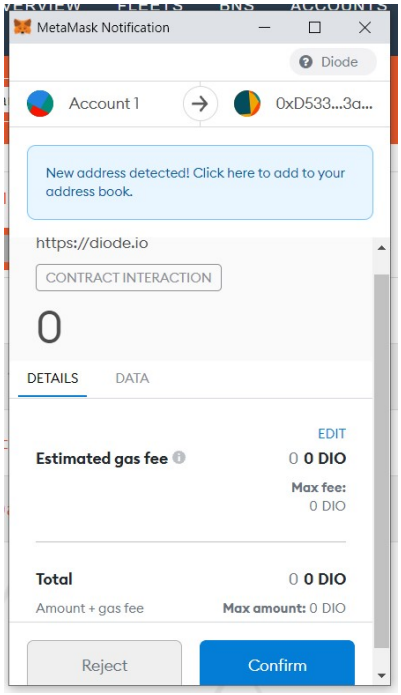

Figure 4.3: METAMASK

Fleet	Publishment	Result
Default	Public	OK
Default	Protected	OK
Default	Private	Broken pipe
Smart Contract	Public	OK
Smart Contract	Protected	Broken pipe
Smart Contract	Private	Broken pipe

Table 4.1: Results implementing Lab 1

4.1.2 Vulnerability assessment

When referring to the Data Flow Diagram established for Lab 1 in figure 3.3, the vulnerability assessment focused primarily on the Generic Data Store that is beyond the Raspberry Web server process. The data flows running to and from this process are also evaluated. This also means that not all potential listed threats listed in the threat modeling tool will be evaluated. The potential threats that are found relevant to evaluate are listed in table 4.2.

Based on this list of potential threats, a set of test cases are established with the goal of evaluating the potential threats. The first case is however not focusing on threats itself, but is more focused on verifying that the Data Flow Diagram is correct. The test cases and the threats they evaluate are listed in table table 4.3, as well as the test activity or type of attack that is used to evaluate the threat.

ID	Category	Title
2, 12, 57	Information Disclosure	Data Flow Sniffing
23, 65	Information Disclosure	Weak Access Control for Resource
19, 20	Denial of Service	Data Store Inaccessible
11, 22, 58	Denial of Service	Process Crash/Stop for Web server
9, 37,50	Elevation Of Privilege	Elevation Using Impersonation
8, 18, 61	Elevation Of Privilege	Web server may be exposed to RCE

Table 4.2: Relevant Vulnerabilities Lab 1

Test Case	Threat	Activity
1	The DFD may be incorrect	Scanning
2	Web server exposed to RCE	Scanning
3	Weak Access control and elevation	Gain access
4	Data Flow sniffing	Analysing traffic
5	Process crash and inaccessible data store	Denial of Service

Table 4.3: Test Cases Lab 1

The test activities listed in section 3.4.1 will be central for each test case, but all test activities are not necessarily used.

Test case 1: Verify Data Flow Diagram

The verification of DFD started with using NMAP and ARP scan Tools at a Computer with Kali installed on the same Local Area Network as the Diode Clients.

The ARP Scan Tool is an ARP packet scanner that shows every active IPv4 device on the subnet. Since ARP is non-routable, this type of scanner only works on the local area network. The following hosts were revealed by using this tool:

Code listing 4.5: ARP scan LAB1 .

```
$ sudo arp-scan --interface=eth0 --localnet

Interface: eth0, type: EN10MB, MAC: 00:0c:29:5b:68:c6, IPv4: 192.168.1.139
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     30:5a:3a:c9:a9:70       ASUSTek COMPUTER INC.
192.168.1.155   b8:27:eb:06:ce:62       Raspberry Pi Foundation
192.168.1.126   b8:27:eb:53:ea:37       Raspberry Pi Foundation
```

This scanning has revealed the IP addresses of the raspberries and verified that there is a router in the network. This test has also verified that the listed entities in the Data Flow Diagram are the same as the devices listed with this tool. The tool NMAP is then used to have a closer look at the device with the web server installed.

Code listing 4.6: NMAP scan LAB1 .

```
$ nmap -v -A 192.168.1.155
Starting Nmap 7.91 ( https://nmap.org ) at 2021-10-21 15:20 EDT
NSE: Loaded 153 scripts for scanning.

Scanning raspberrypi (192.168.1.155) [1000 ports]
Discovered open port 22/tcp on 192.168.1.155
Discovered open port 80/tcp on 192.168.1.155
Completed Connect Scan at 15:20, 2.06s elapsed (1000 total ports)
Initiating Service scan at 15:20
Scanning 2 services on raspberrypi (192.168.1.155)
Completed Service scan at 15:20, 11.12s elapsed (2 services on 1 host)

Nmap scan report for raspberrypi (192.168.1.155)
Host is up (0.0074s latency).
Not shown: 997 closed ports
PORT     STATE SERVICE       VERSION
22/tcp   open  ssh           OpenSSH 7.9p1 Raspbian 10+deb10u2+rpt1 (protocol 2.0)
| ssh-hostkey:
|   2048 65:16:5a:59:7b:cb:a8:2e:fb:b2:9a:17:67:3b:48:69 (RSA)
|   256 b6:1f:36:05:a5:5a:0f:48:89:5b:84:20:34:12:00:3a (ECDSA)
|_  256 49:69:36:6e:47:d5:6f:d8:cd:23:2b:7e:e0:f8:79:67 (ED25519)
80/tcp   open  http          nginx 1.14.2
| http-methods:
|_  Supported Methods: GET HEAD
|_http-server-header: nginx/1.14.2
|_http-title: Welcome to nginx!
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

This portscan verified that the dataflow called TCP/SSH in the DFD only uses port 22 and port 80 which is correct for communication inside the Local Area Network. The test did however not discover the dataflow to the Diode Network.

Test case 2: Web server exposed to RCE

To test whether the Web server could be exposed to Remote Code executions, the scanning tool Nessus were used to scan the raspberries. This tool found 1 critical, 1 high and 3 medium vulnerabilities on the raspberries.

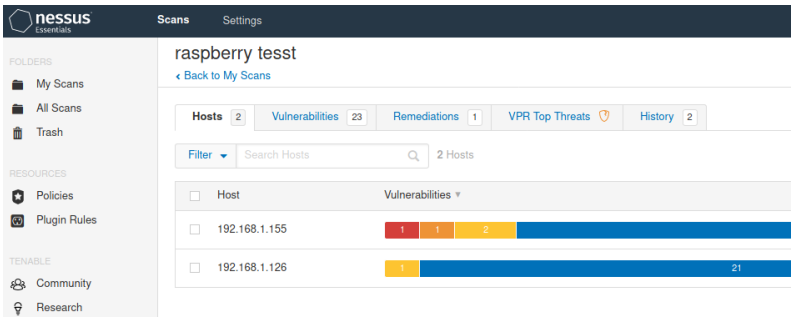

Figure 4.4: Nessus Lab 1

The results are listed in table 4.4.

Host	Severity	Service	Description
192.168.1.155	Critical	Nginx	Byte Memory Overwrite RCE
192.168.1.155	High	Nginx	Multiple Denial of Service vulnerabilities
192.168.1.155	Medium	Nginx	Information Disclosure

Table 4.4: Vulnerabilities Nessus Lab 1

Vulnerability	CVSS Score	Access/Metasploit
CVE-2021-23017	6.8	Medium probability/No
CVE-2019-9511/3/5,	7.8	None
CVE-2019-20372	4.3	Low probability/No
CVE-2005-1794	6.4	Low probability/No

Table 4.5: CVE Scores Nessus Lab 1

The host 192.168.1.155 has both critical and high vulnerabilities related to the Nginx webserver. Searches in vulnerability databases [40] shows the affected areas and are listed in table 4.5.

According to Nessus, there is a vulnerability to Remote Code Execution related to the current version of the Web server. Searches in vulnerability databases or the Metasploit framework did however not reveal how this could be done.

Test case 3: Weak access control and elevation

To evaluate the access control and possible elevation of privileges, the test activity involving both gaining and maintaining access are used. From the scanning and reconnaissance activity in Test Case 1, the tool Nmap reported that the SSH service is open at port 22. The tool ncrack is then used to try a brute force attack at this service, both at the IP address listed in nmap and the diode address to the client. The typical default user of raspberry is pi, and with this credential and a text file with multiple password alternatives the ncrack found the following;

Code listing 4.7: NCRACK scan LAB1 .

```
// The IP address
$ ncrack -p 22 --user pi -P 500-worst-passwords.txt 192.168.1.126

Starting Ncrack 0.7 ( http://ncrack.org ) at 2021-10-22 08:20 EDT

Discovered credentials for ssh on 192.168.1.126 22/tcp:
192.168.1.126 22/tcp ssh: 'pi' 'raspberry'

Ncrack done: 1 service scanned in 141.01 seconds.

Ncrack finished.


// The Diode address
$ ncrack -p 22 --user pi -P 500-worst-passwords.txt
https://0x035a7d47d7989136676a7863fe590ae6d2586a90.diode.link/

Starting Ncrack 0.7 ( http://ncrack.org ) at 2021-10-22 08:27 EDT

// Ncrack freezes after this command and never finishes the scan.
```

This shows that the password to the diode client can be revealed in a brute force attack and that access can be gained. A login with these credentials also show that the user has root access, which means privileges can be elevated to whatever level wanted. It is however the ip address that is available inside a local network that could be exploited, and not the diode address that is reachable from the Internet.

Test case 4: Data flow sniffing

To verify that the data flow can not be viewed by unauthorized users, both the certificates and traffic are analyzed. The TCP/SSH data flows within the Local Area Network are unencrypted HTTP-traffic where the NMAP tool in Test Case 1 could identify both the content in the Data Store behind the Web server, as well as the version of the Web server. The data flows to the Web server via the Diode Network is however classified as a secure connection with encrypted traffic. HTTPS, short for Hyper Text Transfer Protocol Secure, appears in the browser when a website is secured by an SSL or TSL certificate. By clicking on the lock symbol on the browser bar, the details of the certificate, such as issuing authority and the corporate name of the website owner, can be viewed. For the Web server in the diode client, the certificate is shown in figure 4.5.

The encrypted traffic can also be shown by using the tool Wireshark and applying a ssl filter. Even though the traffic is encrypted, it is, according to figure 4.6,possible to see the ip addresses it communicates with, 172.105.85.69, the server name, prenet.diode.io and some information about the key exchange.

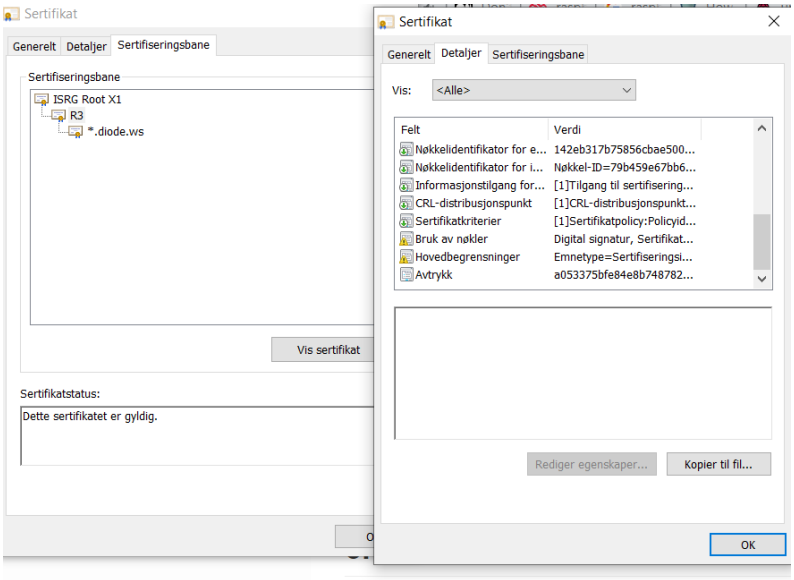

Figure 4.5: Diode Certificate

Test Case 5: Process crash and inaccessible data store

To test whether the Web server can experience process crash leading to inaccessible data store, a combination of Denial of Service attacks as well as spoofing attacks is used. The attacks were performed at the open ports of both the ip address and the diode address of the raspberries.

Code listing 4.8: Hping LAB1 .

```
$ sudo hping3 -S --flood -V -p 80 192.168.1.126
--- 192.168.1.126 hping statistic ---
15525135 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

$ sudo hping3 -S --flood -V -p 22 192.168.1.126
--- 192.168.1.126 hping statistic ---
14024011 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

$ sudo hping3 0x035a7d47d7989136676a7863fe590ae6d2586a90.diode.link
-q -n -d 120 -S -p 80 --flood --rand-source
--- 0x035a7d47d7989136676a7863fe590ae6d2586a90.diode.link hping statistic ---
2603356 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

//Spoofing the 192.168.1.155
$ sudo hping3 -1 --flood 192.168.1.126 -a 192.168.1.155

$ sudo hping3 -1 --flood 192.168.1.126 -a
0x035a7d47d7989136676a7863fe590ae6d2586a90.diode.link
```

The attack at port 80 at the raspberry led to a slower response while attacking port 22 led to a non-responsive Web server and no ping response as well as

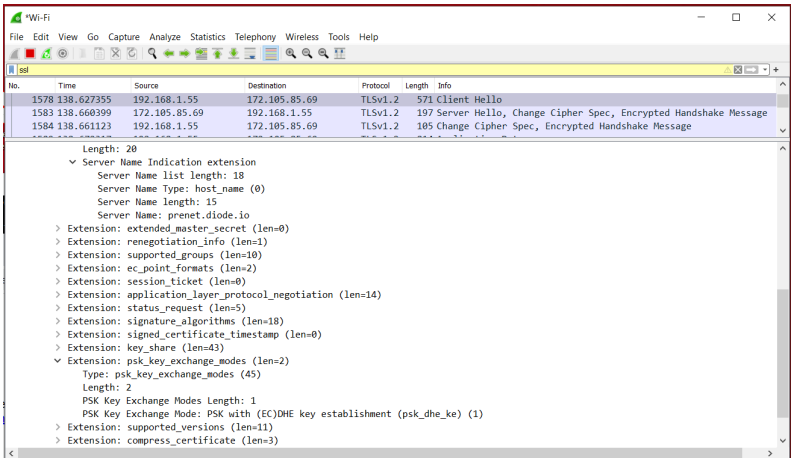

Figure 4.6: Wireshark Lab 1

existing SSH connections were broken. The attack at the diode address led to a slower reponse from the web server, while ping response as well as existing SSH connections were nonresponsive or broken.

The spoofing attack led to the target ip being nonresponsive, while the spoofed ip had around 50 percent packet loss. The attack could be performed with both the ip address and the diode address.

Test Case	Threat	LAN	Internet
1	Incorrect DFD		
2	Exposed Web Server	(X)	(X)
3	Weak Access Control	X	
4	Data Flow sniffing	X	
5	Process crash	X	X

Table 4.6: Results Lab 1

Vulnerability summary

The results of the various vulnerability test cases can be summarized in table 4.6. There has been various results depending on whether tests were done inside the Local Area Network towards the ip address, or from the Internet towards the diode address. Table 4.6 shows where the potential threat was verified as a real threat with the mark X. A comment to this table is that in Test Case 2 the potential threat could only be partially verified, marked with (X), as the Nessus scan tool reported this vulnerability without referring to vulnerability databases where a possible exploitation method and CVSS rating could be found.

The main difference between LAN and Internet results come from the fact that the open SSH port was vulnerable from a brute force password attack from inside the LAN which gave full privileges to an attacker.

As a part of the threat model methodology it is now important to go back to step 3, Identify, and update the threat model with the discovered vulnerabilities. If an already listed threat with the state mitigated is still a valid threat, the state need to be changed in the Threat Modeling Tool, typically to "Needs Investigation", or if the threat is not yet listed, this will need to be added manually. Remember the example in section 3.3.6, where the threat with ID 21 and description "Weak Access Control for a Resource" was mitigated by implementing a password? This vulnerability evaluation shows that the Access Control is still too weak which require further implementation and validation steps to the system. Implementing Access Control with multifactor authentication could be the next mitigation step, but this would also have to be evaluated after the implementation.

4.2 Lab 2 CA X.509

The implementation of a PKI Lab is based on on a traditional CA structure with X.509 certificates. Due to various results a number of potential solutions were tested.

4.2.1 X.509 implementation with root CA

The first implementation included creating an own Root Certificate Authority as well as Sub CA designed to sign server and user certificates. The certificates were

based on openssl listed in Appendix and created on an freshly installed Kali image and then exported to the raspberries. [41]. Details of the implementation are shown in section 4.2.4

Verifying certificate from web browser

The user certificate is also implemented to the users of the webservice, but the results is not promising as the web browsers report the certificate to the web site to not be valid because the issuer of the certificate is not found.

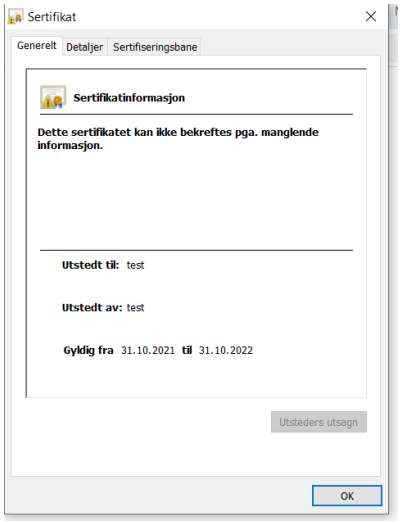

Figure 4.7: X.509 rootsigned certificate part 1

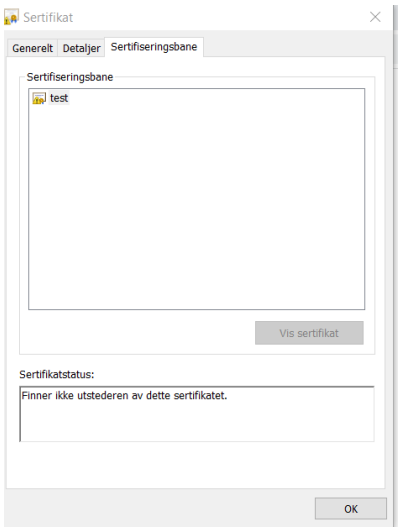

Figure 4.8: X.509 rootsigned certificate part 2

4.2.2 X.509 implementation with self signed certificate

The second implementation involved creating a self signed certificate on the raspberry which was then implemented in the web server Bloggerbrothers [42]. This implementation was also based on the openssh toolkit.

Creating and implementing certificates

Code listing 4.9: CA X.509 self-signed implementation

```
//Generating self signed certificate and key
pi@raspberrypi:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
Generating a RSA private key
..............+++++
.............+++++
writing new private key to '/etc/ssl/private/nginx-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NO
State or Province Name (full name) [Some-State]:Oslo
Locality Name (eg, city) []:oslo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NTNU
Organizational Unit Name (eg, section) []:Test
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.155
Email Address []:
pi@raspberrypi:~$ sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.................................
//Updating the Nginx server
pi@raspberrypi:/etc/nginx/snippets$ sudo vi self-signed.conf
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

pi@raspberrypi:/etc/nginx/snippets$ sudo vi ssl-params.conf
add: openssl_dhparam /etc/ssl/certs/dhparam.pem;

pi@raspberrypi:/etc/nginx$ sudo nginx -t
pi@raspberrypi:~$ sudo nginx -t
nginx: [warn] "ssl_stapling" ignored, issuer certificate not found for certificate
"/etc/ssl/certs/nginx-selfsigned.crt"
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
pi@raspberrypi:~$ sudo systemctl restart nginx
pi@raspberrypi:~$
```

Verifying certificate from web browser

This implementation of a self signed certificate changed the web browsers report. The new certificate is now valid, however it is not trusted since it is not stored in the web site.

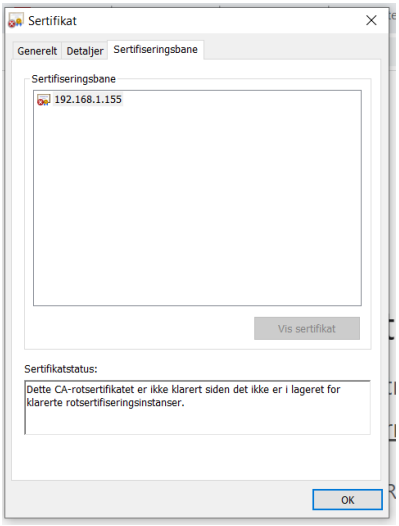

Figure 4.9: X.509 selfsigned certificate

The certificate was then exported to the user and imported to the web browser as a root certificate.

Code listing 4.10: CA X.509 self-signed verification

```
sudo openssl pkcs12 -export -in certs/nginx-selfsigned.crt -inkey
private/nginx-selfsigned.key \
 -out selfsigned.pfx -name "Selfsigned␣Certificate"
 pi@raspberrypi:/etc/ssl$ sudo openssl pkcs12 -export -in certs/nginx-selfsigned.crt
 -inkey private/nginx-selfsigned.key \
>  -out selfsigned.pfx -name "Selfsigned␣Certificate"
Enter Export Password:
Verifying - Enter Export Password:
pi@raspberrypi:/etc/ssl$
```

The import of this certificate finally made the browser report this as a valid certificate.

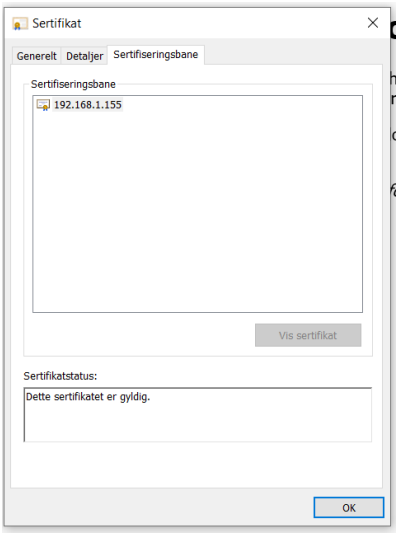

Figure 4.10: X.509 valid selfsigned certificate

4.2.3 X.509 implementation with automated certificate authority

The third implementation involved using an automated and open source tool called Certbot to generate a certificate from the certificate authority named Let's Encrypt [43].This implementation can also be classified as a free of charge PKI-as-a-Service installation.

Code listing 4.11: CA X.509 automated certificate authority

```
root@raspberrypi:/etc/ssl# sudo apt-get install certbot
Leser pakkelister ... Ferdig
Skaper oversikt over avhengighetsforhold
Leser tilstandsinformasjon ... Ferdig
certbot is already the newest version (0.31.0-1+deb10u1).
Følgende pakker ble automatisk installert og er ikke lenger påkrevet:
  gyp libc-ares2 libjs-inherits libjs-is-typedarray libssl-dev libuv1
  libuv1-dev nodejs-doc python-colorzero
Use 'sudo␣apt␣autoremove' to remove them.
0 oppgraderte, 0 nylig installerte, 0 å fjerne og 0 ikke oppgradert.
root@raspberrypi:/etc/ssl# sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx
No names were found in your configuration files. Please enter in your domain
name(s) (comma and/or space separated)  (Enter 'c' to cancel): donothaveadomain.com
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for donothaveadomain.com

Failed authorization procedure. donothaveadomain.com (http-01):
urn:ietf:params:acme:error:dns ::
DNS problem: NXDOMAIN looking up A for donothaveadomain.com -
check that a DNS record exists for this domain

IMPORTANT NOTES:
 - The following errors were reported by the server:
```

```
   Domain: donothaveadomain.com
   Type:   None
   Detail: DNS problem: NXDOMAIN looking up A for donothaveadomain.com
   - check that a DNS record exists for this domain
root@raspberrypi:/etc/ssl#
```

Since the web server was set up without a domain name visible from the Internet, this implementation did not work.

4.2.4 Detailed implementation with root CA

Creating a self signed root certificate

Code listing 4.12: CA X.509 root

```
// creates a key for the certificate
$ sudo openssl rand -out private/.randRootCA 8192
$ sudo openssl genrsa -out rootca.key -rand private/.randRootCA
Generating RSA private key, 2048 bit long modulus (2 primes)

// Create the actual certificate
$ sudo openssl req -new -x509 -days 3650 -key cakey.pem -out cacert.pem

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NO
State or Province Name (full name) [Some-State]:oslo
Locality Name (eg, city) []:oslo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NTNU
Organizational Unit Name (eg, section) []:test
Common Name (e.g. server FQDN or YOUR name) []:emil
Email Address []:

 // To show the certificate
openssl x509 -text -in cacert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            7e:2a:7f:92:cc:79:be:cc:ba:9b:d7:fa:25:e6:8d:cf:5a:97:20:56
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = NO, ST = oslo, L = oslo, O = NTNU, OU = test, CN = Emil
        Validity
            Not Before: Oct 30 12:11:27 2021 GMT
            Not After : Oct 28 12:11:27 2031 GMT
        Subject: C = NO, ST = oslo, L = oslo, O = NTNU, OU = test, CN = Emil
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
```

Create a sub ca certificate signed by the root ca

Code listing 4.13: CA X.509 sub

```
// creates a key for the certificate
$ sudo openssl rand -out .randSubCA 8192
$ sudo openssl genrsa -out private/subca.key -aes256 -rand .randSubCA 2048

Enter pass phrase for private/subca.key:
Verifying - Enter pass phrase for private/subca.key:

// Create the actual certificate
$ sudo openssl req -new -key private/subca.key -out subca.csr -config openssl.cnf
Enter pass phrase for private/subca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NO
State or Province Name (full name) [Some-State]:Oslo
Locality Name (eg, city) []:NTNU
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NTNU
Organizational Unit Name (eg, section) []:emil
Common Name (e.g. server FQDN or YOUR name) []:test
Email Address []:


Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123456
An optional company name []:

// Signs the actual certificate
$ sudo openssl ca -name CA_default -in subca.csr -out certs/subca.crt \
  -config openssl.cnf
Using configuration from openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 1 (0x1)
        Validity
            Not Before: Oct 31 16:37:08 2021 GMT
            Not After : Oct 31 16:37:08 2022 GMT

Certificate is to be certified until Oct 31 16:37:08 2022 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Creating a server certificate signed by sub ca

Code listing 4.14: CA X.509 server

```
// creates a key for the certificate

$ sudo openssl rand -out .randServer 8192
[sudo] password for kali:

$ sudo openssl genrsa -out private/server.key -aes256 -rand .randServer 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.................................+++++
.............................................................+++++
e is 65537 (0x010001)
Enter pass phrase for private/server.key:
Verifying - Enter pass phrase for private/server.key:

// Create the actual certificate
$ sudo openssl req -new -key private/server.key -out server.csr -config openssl.cnf
Enter pass phrase for private/server.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:NO
State or Province Name (full name) [Some-State]:Oslo
Locality Name (eg, city) []:oslo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NTNU
Organizational Unit Name (eg, section) []:emil
Common Name (e.g. server FQDN or YOUR name) []:test
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123456
An optional company name []:

// Signs the actual certificate with the SubCa
sudo openssl ca -name CA_SubCA -in server.csr -out certs/server.crt
 -extensions server_cert -config openssl.cnf
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./private/subca.key:
Error Loading extension section user_cert
140212158022976:error:0E06D06C:configuration file routines:NCONF_get_string:
no value:../crypto/conf/conf_lib.c:273:group=CA_SubCA name=email_in_dn
140212158022976:error:0E06D06C:configuration file routines:NCONF_get_string:
no value:../crypto/conf/conf_lib.c:273:group=CA_SubCA name=rand_serial
-extensions: command not found

// ERROR, tries to sign with root CA instead

sudo openssl ca -name CA_default -in server.csr -out certs/server.crt \
  -config openssl.cnf
Using configuration from openssl.cnf
Check that the request matches the signature
Signature ok
```

```
Certificate Details:
        Serial Number: 2 (0x2)
        Validity
            Not Before: Oct 31 20:28:17 2021 GMT
            Not After : Oct 31 20:28:17 2022 GMT
        Subject:
            countryName               = NO
            stateOrProvinceName       = Oslo
            organizationName          = NTNU
            organizationalUnitName    = emil
            commonName                = test
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                E3:FE:78:37:B1:BA:8B:E2:6C:80:9B:F2:83:F8:A2:31:87:12:F4:D7
            X509v3 Authority Key Identifier:
                keyid:13:56:71:0D:1A:C7:B8:E3:43:F8:89:E3:E3:53:4D:9F:82:0C:38:ED

Certificate is to be certified until Oct 31 20:28:17 2022 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Creating a user certificate signed by sub ca

Code listing 4.15: CA X.509 user

```
// creates a key for the certificate

$ sudo openssl rand -out .randServer 8192


$ sudo openssl genrsa -out private/user.key -aes256 -rand .randServer 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
..............................+++++
.............................................................+++++
e is 65537 (0x010001)
Enter pass phrase for private/user.key:
Verifying - Enter pass phrase for private/user.key:

// Create the actual certificate
$ sudo openssl req -new -key private/user.key -out user.csr -extensions usr_cert \
 -config openssl.cnf
Enter pass phrase for private/user.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
```

```
Country Name (2 letter code) [AU]:NO
State or Province Name (full name) [Some-State]:Oslo
Locality Name (eg, city) []:oslo
Organization Name (eg, company) [Internet Widgits Pty Ltd]:NTNU
Organizational Unit Name (eg, section) []:emil
Common Name (e.g. server FQDN or YOUR name) []:test2
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:123456
An optional company name []:

// Signs the actual certificate with the RootCa (instead of SubCa)

sudo openssl ca -name CA_default -in user.csr -out certs/user.crt \
 -extensions usr_cert -config openssl.cnf
Using configuration from openssl.cnf
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 3 (0x3)
        Validity
            Not Before: Oct 31 20:48:31 2021 GMT
            Not After : Oct 31 20:48:31 2022 GMT
        Subject:
            countryName               = NO
            stateOrProvinceName       = Oslo
            organizationName          = NTNU
            organizationalUnitName    = emil
            commonName                = test2
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                04:17:D1:13:D2:00:31:F5:DC:64:74:9C:97:76:E4:0E:E2:00:64:06
            X509v3 Authority Key Identifier:
                keyid:13:56:71:0D:1A:C7:B8:E3:43:F8:89:E3:E3:53:4D:9F:82:0C:38:ED

Certificate is to be certified until Oct 31 20:48:31 2022 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
```

Export certificate to users

Code listing 4.16: CA X.509 export

```
// Change server and user certificate to PKCS#12 format
$ sudo openssl pkcs12 -export -in certs/user.crt -inkey private/user.key \
 -out user.pfx -name "User Certificate"
Enter pass phrase for private/user.key:
Enter Export Password:
Verifying - Enter Export Password:


$ sudo openssl pkcs12 -export -in certs/server.crt -inkey private/server.key \
 -out server.pfx -name "Server Certificate"
[sudo] password for kali:
Enter pass phrase for private/server.key:
Enter Export Password:
Verifying - Enter Export Password:

 sudo openssl pkcs12 -export -in cacert.pem -inkey cakey.pem \
 -out Root.pfx -name "Root Certificate"
[sudo] password for kali:
Enter Export Password:
Verifying - Enter Export Password:


// Transfer certificates to raspberries by using ssh/scp
 $ sudo scp server.pfx pi@192.168.1.155:/home/pi
pi@192.168.1.155s password:
server.pfx                                    100% 2666   403.8KB/s   00:00

$ sudo scp user.pfx pi@192.168.1.126:/home/pi
pi@192.168.1.126s password:
user.pfx                                      100% 2662    52.4KB/s   00:00
sudo scp user.pfx pi@192.168.1.126

sudo scp Root.pfx pi@192.168.1.155:/home/pi
pi@192.168.1.155s password:
Root.pfx                                      100% 2694    47.2KB/s   00:00

// Export and implement server certificates in raspberries

pi@raspberrypi:~$ openssl pkcs12 -in server.pfx -nocerts -out server.key -nodes
Enter Import Password:
pi@raspberrypi:~$ openssl pkcs12 -in server.pfx -nokeys -out server.crt -nodes
Enter Import Password:
pi@raspberrypi:~$
pi@raspberrypi:~$ sudo openssl pkcs12 -in Root.pfx -nocerts -out Root.crt -nodes
Enter Import Password:
pi@raspberrypi:~$ sudo cp Root.crt /usr/local/share/ca-certificates/Root.crt

pi@raspberrypi:~$ sudo cp server.crt /usr/local/share/ca-certificates/server.crt
pi@raspberrypi:~$ sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
2 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
```

Implement certificate and key in the NGINX web server

Code listing 4.17: CA X.509 implementation

```
// Generating two new files in the directory /etc/nginx/snippets
pi@raspberrypi:~$ sudo vi /etc/nginx/snippets/self-signed.conf
ssl_certificate /usr/local/share/ca-certificates/server.crt;
ssl_certificate_key /etc/ssl/private/server.key;
pi@raspberrypi:~$ sudo vi /etc/nginx/snippets/ssl-params.conf
ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
ssl_ecdh_curve secp384r1;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
// Editing the configuration file of NGINX by adding the following
pi@raspberrypi:~$ sudo /etc/nginx/nginx.conf
http
 server
  # SSL configuration
  listen 443 ssl http2 default_server;
  listen :443 ssl http2 default_server;
  include snippets/self-signed.conf;
  include snippets/ssl-params.conf;
// Checking configuration file and restarting service
root@raspberrypi:/etc/nginx# nginx -t
nginx: warn "ssl_stapling" ignored, issuer certificate not found for certificate
"/usr/local/share/ca-certificates/server.crt"
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
pi@raspberrypi:/etc/nginx# sudo systemctl restart nginx
```

ID	Category	Title
2, 12	Information Disclosure	Data Flow Sniffing
23	Information Disclosure	Weak Access Control for Resource
19, 20	Denial of Service	Data Store Inaccessible
11, 22	Denial of Service	Process Crash/Stop for Web server
9	Elevation Of Privilege	Elevation Using Impersonation
8, 18	Elevation Of Privilege	Web server may be exposed to RCE

Table 4.7: Relevant Vulnerabilities Lab 2

4.2.5 Vulnerability assessment

When referring to the Data Flow Diagram established for Lab 2 in figure 3.4, this vulnerability assessment also focused primarily on the Generic Data Store that is beyond the Raspberry Web server process, as well as the connected data flows.

Two of the implementations could not provide a valid certificate to the web server, which made the second implementation with the self signed certificate a natural candidate for the vulnerability assessment.

Even though the threat modeling tool listed fewer potential threats in Lab 2 versus Lab 1, the potential threats that were found relevant to evaluate were the same as in Lab 1 and listet in table 4.7. The difference was that there were fewer ID's connected to each threat.

Based on this list of potential threats, the same list of test cases as in Lab 1 were used.

Test case 1: Verify Data Flow Diagram

The verification of DFD were performed by the use of NMAP and ARP scan tools

The ARP Scan Tool gave the same results as in lab 1, but the NMAP revealed that in addition to the open ports in lab 1, one could now see that port 443 is open. NMAP also revealed information about the certificate that is used on the raspberry with web server.

Code listing 4.18: NMAP scan LAB2

```
$ nmap -v -A 192.168.1.155
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-13 06:57 EST

Initiating Connect Scan at 06:57
Scanning raspberrypi (192.168.1.155) [1000 ports]
Discovered open port 22/tcp on 192.168.1.155
Discovered open port 443/tcp on 192.168.1.155
Discovered open port 80/tcp on 192.168.1.155
Completed Connect Scan at 06:57, 0.18s elapsed (1000 total ports)
Initiating Service scan at 06:57
Scanning 3 services on raspberrypi (192.168.1.155)
Completed Service scan at 06:57, 12.32s elapsed (3 services on 1 host)

PORT     STATE SERVICE       VERSION
```

```
22/tcp   open  ssh          OpenSSH 7.9p1 Raspbian 10+deb10u2+rpt1 (protocol 2.0)
| ssh-hostkey:
|   2048 65:16:5a:59:7b:cb:a8:2e:fb:b2:9a:17:67:3b:48:69 (RSA)
|   256 b6:1f:36:05:a5:5a:0f:48:89:5b:84:20:34:12:00:3a (ECDSA)
|_  256 49:69:36:6e:47:d5:6f:d8:cd:23:2b:7e:e0:f8:79:67 (ED25519)
80/tcp   open  http         nginx 1.14.2
| http-methods:
|_  Supported Methods: GET HEAD
|_http-server-header: nginx/1.14.2
|_http-title: Welcome to nginx!
443/tcp  open  ssl/http     nginx 1.14.2
| http-methods:
|_  Supported Methods: GET HEAD
|_http-server-header: nginx/1.14.2
|_http-title: Welcome to nginx!
| ssl-cert: Subject: commonName=192.168.1.155/organizationName=
NTNU/stateOrProvinceName=Oslo/countryName=NO
| Issuer: commonName=192.168.1.155/organizationName=
NTNU/stateOrProvinceName=Oslo/countryName=NO
| Public Key type: rsa
| Public Key bits: 2048
| Signature Algorithm: sha256WithRSAEncryption
| Not valid before: 2021-11-06T20:05:01
| Not valid after:  2022-11-06T20:05:01
| MD5:   0a01 26b7 9a92 b096 351f a745 3440 9c28
|_SHA-1: 349d e769 69a9 2f9a 128a ca59 647b 1544 0f49 5e31
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|   h2
|_  http/1.1
| tls-nextprotoneg:
|   h2
|_  http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

This portscan verified that the data flow called HTTPS/SSH in the DFD uses port 22, 80 and 443. This means that the data flow also uses HTTP, which is not in correspondance for the Data Flow Diagram for this lab.

Test case 2: Web server exposed to RCE

In the same way as Lab 1, the scanning tool Nessus were used to test whether the Web server could be exposed to Remote Code executions

This tool now found 2 critical, 2 high and 11 medium vulnerabilities on the raspberries.

The results are listed in table 4.8.

Even though more vulnerabilities were reported in lab 2, there is still the same affected areas as in lab 1 that can be found in a vulnerability databases [40]. This means that the result of this test case regarding the Remote Code Execution will be the same as Lab 1.

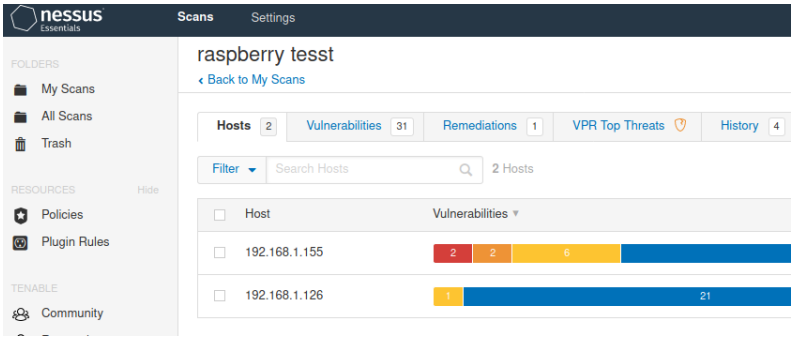

Figure 4.11: Nessus Lab 2

Host	Severity	Service	Description
192.168.1.155	Critical x2	Nginx	Byte Memory Overwrite RCE
192.168.1.155	High x2	Nginx	Multiple Denial of Service vuln
192.168.1.155	Medium	Nginx	Information Disclosure
192.168.1.155	Medium	SSL	Certificate cannot be trusted
192.168.1.155	Medium	SSL	Self-Signed Certificate
192.168.1.155	Medium	TLS	Version 1.0 Protocol Detection

Table 4.8: Vulnerabilities Nessus Lab 2

Test case 3: Weak access control and elevation

In order to gain access, the method from lab 1 with brute force attacking SSH is still a valid method to both gaining access and maintaining access.

Test case 4: Data flow sniffing

To verify that also the data flows in this lab can not be viewed by unauthorized users, both the certificates and traffic are analyzed.

The connection to the webserver with the selfsigned certificate is classified as a secure connection from some web browsers, like Internet Explorer, while other browsers, like Chrome, still reports this site as not secure due to missing Subject Alternative Names in the certificate.

This traffic can also be shown by using the tool Wireshark and applying a ssl filter. In this lab it is also possible to see information about the certificate and key exchange.

In Test case 1 the NMAP tool showed that the data flow also uses HTTP, which was not in correspondance for the Data Flow Diagram for this lab. This also means that it is possible for unauthorized users within the local network to view parts of the data flow.

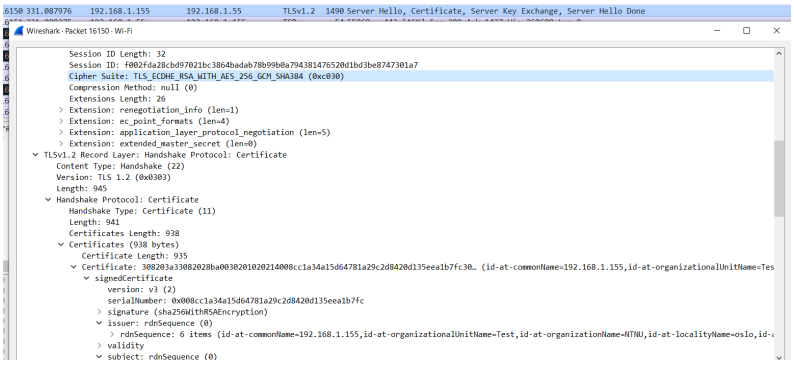

Figure 4.12: Wireshark Lab 2

Test Case 5: Process crash and inaccessible data store

To test whether the Web server in this lab also can experience process crash leading to inaccessible data store, the same combination of Denial of Service attacks as well as spoofing atacks were used. The tool hping3 is also used in this lab to test whether the web server can withstand denial of service attacks. The attack were now performed at the new open port 443 of the raspberry.

Code listing 4.19: Hping LAB2

```
$ sudo hping3 -S --flood -V -p 443 192.168.1.155
[sudo] password for emyre:
using eth0, addr: 192.168.1.139, MTU: 1500
HPING 192.168.1.155 (eth0 192.168.1.155): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

$ sudo hping3 -S --flood -V -p 80 192.168.1.155
[sudo] password for emyre:
using eth0, addr: 192.168.1.139, MTU: 1500
HPING 192.168.1.155 (eth0 192.168.1.155): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown

$ sudo hping3 -S --flood -V -p 22 192.168.1.155
[sudo] password for emyre:
using eth0, addr: 192.168.1.139, MTU: 1500
HPING 192.168.1.155 (eth0 192.168.1.155): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown


//Spoofing the 192.168.1.155
$ sudo hping3 -1 --flood 192.168.1.126 -a 192.168.1.155
HPING 192.168.1.126 (eth0 192.168.1.126): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

The attack to port 443, 80 and 22 at the raspberry led to a slower response where figure 4.13 shows that the ping time went from typical 3 milliseconds to around 3 seconds.

The spoofing attack also led to the target ip being non-responsive to ping, while the spoofed ip had around 50 percent packet loss.

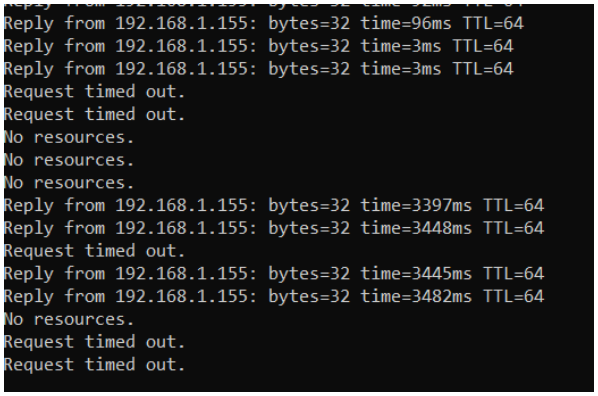

Figure 4.13: Denial of Service Lab 2

Test Case	Threat	LAN	Internet
1	Incorrect DFD	X	
2	Exposed Web Server	(X)	
3	Weak Access Control	X	
4	Data Flow sniffing	X	
5	Process crash	X	

Table 4.9: Results Lab 2

Vulnerability summary

The results of the various vulnerability tests can also in this lab be summarized in a table. The difference from lab 1 is that all test have been done inside the Local Area Network and that there has not been possible to do the tests from the Internet.

Table 4.9 shows where the potential threat was verified as a real threat with the mark X.

Also in this lab the open SSH port was vulnerable from a brute force password attack from inside the LAN which gave full privileges to an attacker.

As a part of the threat model methodology, it is also for this lab important to go back to step 3, Identify, and update the threat model with the discovered vulnerabilities.

Chapter 5

Discussion

This chapter continues on activity 5 in the DSRP with a continued evaluation of the artefact. The chapter will also address the final activity, which is communication.

The chapter start by evaluating aspects of both the implementation and vulnerability assessment of the artefact in the thesis. There will also be an evaluation of the trust levels needed for each implementation, followed by a presentation of limitations that might have affected the result of the thesis.

At the end of the chapter, there will be an attempt to answer the research questions to this thesis, as well as relating these to the earlier presented user scenario. This will address the related topics from the introducion of the thesis, and be essential in the activity of communicating the presented work to the public.

5.1 Aspects of the implementation method

The implementation in lab 1 was the blockchain technology called Diode. The diode client was easy to install in the raspberries where just a few command lines were needed to install software and certificates and to publish services world wide over the Internet without administrating proxy or firewall settings at routers between the Local Area Network and Internet. There is also no need to buy domain names in order to publish the services. The services could easily be accessed at the public access level, where especially the SSH service was easy to verify from other clients. Web services on the other hand, proved a bit more difficult to verify as the raspberry slows down and sometimes freezes when opening its web browser. The secure connections were also broken from time to time when running the software where the diode system reported that it could not verify transaction blocks.

To use the private and protected access levels, it is a bit more complicated when it comes to installing the Metamask application and generate fleet accounts. There is often a need for several attempts to move a client from the default account to the fleet account, and if more than one account is generated, the client, from the diode network perspective, seem to be added to all the accounts when it is only attempted to do this at only one. It also seems that the fleet accounts are

in a different server, US1, than the defalt account, EU1, and it does not seem to be a way for the fleet account administrator to choose which server to use. This might affect the outcome of the result when the attempts to publish web services at these access levels failed. The Diode Client also need to use socks proxy server to communicate at these access levels, and this service slows down the raspberry. Another disadvantage with this implementation is that it can not be installed without the clients having access to the Internet.

While the implementation in lab 1 was quite easy to install, the implementations in lab 2 were far more complicated. Creating keys and certificates were highly dependant on the configuration and settings in the Openssl config file. This configuration file also seems to be lacking standardization, as it will have different content depending on the operating system and its version and system policies installed. The error messages received when trying to sign with CA Sub were not intuitive, and made it difficult to correct. In addition, the Nginx server needed several manual changes in the configuration settings to include server certificates. The export and implementation of user certificates also needed a lot of manual installation steps. The advantage with this solution is that it can be implemented in a network without access to Internet, and that the local administrators will have full access to every configuration setting in the system. The PKI-as-a-service alternative with Certbot and Lets Encrypt could be an alternative that is quite easier to install, however it is also depending on Internet access and domain details.

5.2 Aspects of the vulnerability results

The vulnerability assessments in both lab implementations have common factors. In both implementations an attacker could gain full access and exploit the web server and data store of the raspberry with the open SSH port and weak password protection. This shows the importance of not believing a system is secure as soon as it is secured with PKI. One has to consider all weaknesses and continuously evaluate the system to keep it as secure as possible. Another common factor for both labs is the outdated Nginx installation with several vulnerability issues. Even though the Raspberry updated all apt packages, the newest installation of Nginx has to be installed manually. Another vulnerability that is not shown in the results, is the lack of secure storage for the keys. Neither the raspberries nor the local root certificate server in lab 2 use Trusted Platform Modules and this makes the keys in the PKI vulnerable for exposure and exploitation to an attacker. For the implementation in lab 1 the installation of Metamask is also dependant on secure storage, and it is uncertain how the Diode Network secures the needed keys.

What separates the results in the labs is that in Lab 1 it is possible to attack the system from outside the Local Area Network. A comparison on results from the test cases listed in table 5.1 shows that in Lab 2 the DFD was incorrect since it could still go unprotected HTTP traffic via port 80 that was not described in the DFD. Having an implementation that deviates from the design introduces a great

Test Case	Threat	Lab 1	Lab2
1	Incorrect DFD		X
2	Exposed Web Server	(X)	(X)
3	Weak Access Control	X	X
4	Data Flow sniffing	X	X
5	Process crash	X	X

Table 5.1: Results comparison

risk of not being able to identify the all the correct potential vulnerabilities with the threat modelling tool.

In lab 2 there is also an extra vulnerability associated to an outdated version of the TLS certificate which again is depending on which openssl version the certificate was generated with. There are also issues with the self signed certificate where some web browsers report this as an security issue, while other browsers validates the certificate as safe.

5.3 Trust

An important aspect to consider when implementing PKI is the level of trust needed for the system. Blockchain implementations need a consensus of miners to generate trust, where the public blockchain system with several miners might lead to a slow network with little scalability. As a contrast to this, the smaller the networks are, the more vulnerable they will be to both centralization and 51 percent attacks [44] where a majority of rogue miners can compromise the trust mechanism.

PKI-as-a-service also introduces a third party provider that need to be trusted. How do a customer of this service decide the level of trust that this provider should have, and how can a customer control that the trust level is maintained.

As a contrast, a customer can have full control over the network and PKI system when implementing its own Certification Authority. This will however depend on hiring resources with the right competence to implement this. Without this competence, the result can be a trusted system with lots of flaws and vulnerabilities.

5.4 Limitations

This thesis had some limitations that made research, testing and validation challenging. The freezing of raspberries in lab 1 indicated that the chosen hardware were too constrained to fully test and validate all the functions in the blockchain implementation. The lack of standardization in the Openssl configuration in lab 2 also made testing and validation difficult. To compensate for this, three different implementation methods with the same purpose were used. A function that

could not be tested in the lab implementations, due to both hardware restrictions and necessary time, is how the certificate revocation lists would behave in the lab environment, and if there are vulnerabilities related to the distribution of these lists. This is very unfortunate, especially since recent studies suggests that almost one third of the organizations that implement PKI, does not include technology to revoke certain certificates[45].

Another limitation is that some of the research information is retrieved from web sites and blogs which are community driven without official authors. This is typical for open source driven technologies with releases in software repositories like Github [45]. Some of the sources have also been identified as companies with interests of selling own product, which can be challenging for validating the the credibility of the sources. As a compensation, the information have often been verified at several sources to enhance the credibility.

5.5 The case

It is time to take a look at the user scenario presented in the introduction of the thesis. The hospital were faced with some major decission points regarding the choice of PKI solution and whether this should be implemented by internal IT-department or external experts. Based on suggested criterias, the following assessments can be done:

Availability to Internet There was not mentioned whether the health devices would be installed in a private network without access to the Internet, or if they would be accessible from Internet. To lower the attack surface, it is recommended that these devices are on a private network. This also means the only current alternative is to use the standard and lightweight X.509 certificates with a central Certification Authority installed inside the Network.

Size The internal resources of the health devices were also not specified in the case. Considering the fact that blockchain solutions in some settings demanded too much resources from the raspberries in the lab also favours the use of lightweight certificate.

Numbers The amount of health devices to be installed in the hospital were also not specified. It is assumed that for just one hospital, there will not be several thousand devices, which also mean that the scalability advantages offered through PKI-as-a-service or blockchain are not that necessary.

Importance The healthcare devices are however considered very important, since failure or misuse can lead to loss of sensitive pasient data or in worst case loss of life. For these important devices there need to determined a high level of trust where all variables of the PKI implementation are controlled.

So far, all recommendations point to implementing a standard or lightweight X.509 certificates with a central Certification Authority at the hospital. This thesis has however shown that such an installation can be very complex which demands specific expert knowledge. The high level of trust that is needed, combined with the assumption that this hospital is not the only hospital that considers implementing the same kind of health devices, leads to the following recommandation:

The hospital should reach out to its neighboring hospitals and also to the administration levels above, which might be all the way up to the national Ministry of Health and Care Services. This Ministry will probably have access to more resources and can establish a task group with expert knowledge that can be used to develop a national implementation method with the right level of trust. With this method, all of the hospitals in the nation can get help to secure their health devices with the same methodology. It is also expected that this national task group will be well educated in the use of threat modeling tools, and that they can help and train the local IT departments at the hospitals with how to use these tools at the most efficient way.

The disadvantage with this recommendation is however that the implementation will have a centralized form of nature which introduces the possibility for single points of failures. The more hospitals this task group supports, the larger the need for scalability will be. When all the hospitals in a nation is secured with the same solution, this could be a possibility for introducing blockchain technology that will work inside a private network and where there could be decentralization in for of a consortium system with one miner, or trusted validating member, for each hospital.

5.6 Research Questions

The aim of this thesis is to answer three research questions involving PKI solutions for IoT devices. The two first questions are what the current PKI solution for securing IoT devices today are, and what the vulnerabilities for these solutions are. The answer to these two questions can be found from a combination of extracted research literature as well as findings in lab results.

Standard and Lightweight X.509 Certificates This is the most common and implemented PKI solution where infrastructure is dependant on a hierarchical CA structure. The perhaps greatest vulnerability with these solutions is that the system depends on one CA which introduces single points of failures that can be exposed to Denial of Service Attacks. Another vulnerability can be related to the complexity and diversity of various implementations, especially if there are less experienced personell that do the implementation.

PKI-as-a-service There are several companies that offer to host PKI services on constrained devices to customers all over the world. Allthough this solution probably ensures qualified personell to implement and maintain the security of the IoT devices, there is still a considerable risk and vulnerability in

placing the trust of the system to an external company. This solution often demands that the IoT device have to be accessible from outside the Local Area Network, which can introduce a higher probability of being attacked.

Decentralized PKI like blockchain This is a relatively new solution with advantages like scalability and not being dependant of single point of failures. The trust is in the consensus of several miners, which introduces the vulnerability of being exposed to a 51 Percent attack. The current implementations in the lab in this thesis also demands that the IoT device have to be accessible from outside the Local Area Network, and that this, similar to PKI-as-a-service introduces a larger attack surface.

There are also some common vulnerabilities that exist for all PKI solutions. The most dangerous vulnerability is perhaps to not implement and follow a threat model methodology, which can lead to the misconception of believing the IoT device is secure as soon as a PKI solution is implemented. The vulnerability is then to ignore or neglect other basic security mechanisms as changing password or implementing multifactor-authentication, not updating to the latest software patches or to skip antivirus solutions. These are all threat mitigations that could have been discovered when following the steps of a threat model methodology.

Another vulnerability related to the PKI solutions is to store keys on devices without investing in secure storage devices and by that introduce the risk of losing keys with following security breaches. There are also vulnerabilities related to PKI security and the availability of the IoT devices. Implementing too resource demanding PKI solutions can lead to lower performance from the IoT devices, or in worst case make them non-responsive similar to a successful Denial of Service attack.

For the third question of the thesis, which is if different PKI solutions can be recommended to different IoT solutions, the answer is yes. The heterogenity and various areas of use makes it unlikely to recommend just one solution that should cover everyones need. When it comes to the different criterias that should be used for this recommendation, the author of the thesis suggests the following:

Availability to Internet A large portion of todays IoT devices are installed in private networks without any connection to Internet. For these devices, the only current alternative seem to be standard and lightweight X.509 certificates with a central Certification Authority installed inside the Network.

Size The internal resources of a IoT device will determine what PKI solution it should use. Findings from the lab showed that blockchain solutions in some settings demanded too much resources from the raspberries and slowed down the performance considerably. A typical lightweight certificate would be preferred in these settings.

Numbers The amount of PKI devices to be included in a PKI solution is very relevant. For a large amount of several thousand devices one will prefer

a solution that is scalable without too much administration of the unique devices. Blockchain and PKI-as-a-Service are preferred in these settings.

Importance Some IoT devices are more important than others. For instance can failure or misuse of IoT devices in a hospital or power plant result in loss of life or significant damages related to environment. Other IoT devices are however only meant to be low cost and used for entertainment where failure or misuse leads to minimal consequences. For the more important devices one will need to determine a high level of trust and possibly control all variables by implementing a internal Certification Authority which favours the a standard X.509 implementation.

Chapter 6

Conclusion

This thesis have focused on PKI solutions that can secure IoT devices, with both theoretical research and practical implementations with following vulnerability assessments. There are several ways to implement PKI solutions in IoT devices and the security of these solutions will depend on the individual characteristics and the purpose and criticality of each IoT device. Even though this thesis has presented and evaluated some of the current PKI solutions that exist for IoT devices today, these solutions are continuously developing. It is especially interesting to see the development of the blockchain technology, which earlier has seemed promising, but have had implementation challenges related to synchronization time and storage. It now seems like a lot of these challenges have been overcome, but also that there remains some challenges to be solved. This thesis has also tried to make a recommendation for how a decision maker should choose the most secure PKI solution for his IoT devices. This recommendation should be based on a certain set of criterias for the IoT devices. The thesis also shows that in order to have a secure system, a threat modeling methodology to identify and mitigate all potential vulnerabilities in a system, should be implemented and used during the whole lifetime of the system.

6.1 Future work

To mitigate the vulnerabilites related to a centralized Certificate Authority which can end up as a single point of failure, the decentralized infrastructure of blockchain should be further researched. A big disadvantage with the Diode Network that was implemented in this thesis, is the dependancy to Internet. To also make this a realistic solution for IoT devices that are not connected to the Internet, there could be research on how to make smaller and autonomous versions that could be installed inside a local network. The users of these versions could also determine themselves who should be trusted miners to further mitigate the consensus challenge with the related 51 percent vulnerability.

Bibliography

[1] Digi.no, *Datainnbruddet hos helse sør-øst:* Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.digi.no/artikler/datainnbruddet-hos-helse-sor-ost-jaktet-pa-pasientjournaler-og-militaer-informasjon/426197.

[2] M. Hevner Salvatore and Park, *Design science in information systems research*, 2004.

[3] G. Peffers Tuunanen and R. et al, *The design science research process: A model for producing and presenting information systems research*, 2006.

[4] NSF, *Cyber-physical systems*, Web Page, Last accessed 11 November 2021, 2020. [Online]. Available: https://www.nsf.gov/news/special_reports/cyber-physical/.

[5] B. Group, *Embedded systems glossary*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://barrgroup.%20com/Embedded-Systems/Glossary.

[6] Y. K. M Sain and H. Lee, *Survey on security in internet of things: State of the art and challenges*, 2017.

[7] F. B. insights, *Internet of things (iot) market*, Web Page, Last accessed 11 November 2021, 2020. [Online]. Available: https://www.fortunebusinessinsights.com/industry-reports/internet-of-things-iot-market-100307.

[8] Keyfactor, *Know the difference of a digital signature vs. digital certificate*, Web Page, Last accessed 11 November 2021, 2020. [Online]. Available: https://blog.keyfactor.com/digital-signature-vs-digital-certificate.

[9] IETF, *Rfc 5280 - internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile*, Web Page, Last accessed 11 November 2021, 2008. [Online]. Available: https://tools.ietf.org/html/rfc5280.

[10] J. Vaccha, *Public key infrastructure: Buildingtrusted applications and web services*, 2004.

[11] D. o. d. Australian Government, *Public key infrastructure defence public key infrastructure levels of assurance requirements certificate policy object identifiers (oids)*, 2016.

[12] H. Z. Z Li X Yin Z Geng P Li Y Sun and L. Li, *Research on pki-like protocol for internet of things*, 2013.

[13] M. V. T. U. Z. M. Simic and M. Stankovic, *Entity identification and security solutions in iot based on pki and blockchain technology*, 2020.

[14] M. P. D Vranics and Z. Bottyanl, *Electronic administration of unmanned aviation with public key infrastructure*, 2019.

[15] E. B. J Won A Singla and G. Bollella, *Decentralized public key infrastructure for internet-of-things*, 2018.

[16] S. Magnusson, *Evaluation of decentralized alternatives to pki for iot devices*, 2018.

[17] D. Letz, *Blockquick: Super-light client protocol for blockchain validation on constrained devices*, 2019.

[18] Keyfactor, *Iot device security*, Web Page, Last accessed 11 November 2021, 2020. [Online]. Available: `https://info.keyfactor.com/iot-device-security`.

[19] J. Huang and D. Nicol, *An anatomy of trust in public key infrastructure*, 2017.

[20] A. Singla and E. Bertino, *Blockchain-based pki solutions for iot*, 2018.

[21] 2004.

[22] K. S, *Why is raspberry pi 4 a good choice for your iot project?* Web Page, Last accessed 11 November 2021. [Online]. Available: `https://rootsaid.com/raspberry-pi-iot/`.

[23] T. R. P. Foundation, *Raspberry pi documentation*, Web Page, Last accessed 11 November 2021. [Online]. Available: `https://www.raspberrypi.org/documentation/computers`.

[24] Github, *Diode*, Web Page, Last accessed 11 November 2021. [Online]. Available: `https://github.com/diodechain/diode_client`.

[25] Diode, Web Page, Last accessed 11 November 2021. [Online]. Available: `https://diode.io/products/diode-cli/`.

[26] Metamask, *Metamask*, Web Page, Last accessed 11 November 2021. [Online]. Available: `https://metamask.io/`.

[27] O. S. GmbH, *Generating x.509 certificates*, Web Page, Last accessed 11 November 2021. [Online]. Available: `http://www.ipsec-howto.org/x595.html`.

[28] J. Woods, *Understanding public key infrastructure and x.509 certificates*, Web Page, Last accessed 11 November 2021. [Online]. Available: `https://www.linuxjournal.com/content/understanding-public-key-infrastructure-and-x509-certificates`.

[29] T. O. P. Authors, *Welcome to openssl!* Web Page, Last accessed 11 November 2021. [Online]. Available: `https://www.openssl.org/`.

[30] Microsoft, *Threat modeling*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling.

[31] Threatmodeler, *Which threat modeling methodology is right for your organization?* Web Page, Last accessed 11 November 2021. [Online]. Available: https://threatmodeler.com/threat-modeling-methodologies-overview-for-your-business/.

[32] Microsoft, *Getting started with the threat modeling tool*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://docs.microsoft.com/nb-no/azure/security/develop/threat-modeling-tool-getting-started.

[33] K. Poniatowski, *Is stride still relevant for threat modeling?* Web Page, Last accessed 11 November 2021. [Online]. Available: https://blog.securityinnovation.com/stride.

[34] Microsoft, *The stride threat model*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20).

[35] O. S. L. 2021, *The most advanced penetration testing distribution*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.kali.org/.

[36] SoftwareTestingHelp, *Top 10 most popular ethical hacking tools (2021 rankings)*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.softwaretestinghelp.com/ethical-hacking-tools/.

[37] P. at Edureka, *Top 10 ethical hacking tools in 2021*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.edureka.co/blog/ethical-hacking-tools/.

[38] T. Blake, *Best hacking tools of 2017: Nessus vulnerability scanner*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://seniordba.wordpress.com/2017/06/26/best-hacking-tools-of-2017-nessus-vulnerability-scanner/.

[39] K. May, *Metasploit, nessus, nmap more: The hacking tools it pros need to know about*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.comptia.org/blog/metasploit-nessus-nmap-more-the-hacking-tools-it-pros-need-to-know-about.

[40] M. Corporation, *Cve details*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://www.cvedetails.com/.

[41] b. a. w. funkym0nk3y blogger at wordpress.com, *How to create your own pki with openssl*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://evilshit.wordpress.com/2013/06/19/how-to-create-your-own-pki-with-openssl/.

[42] Bloggerbrothers, *Nginx on a raspeberry pi*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://bloggerbrothers.com/2019/03/10/nginx-on-a-raspberry-pi/.

[43] I. S. R. Group, *Let's encrypt*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://letsencrypt.org/.

[44] C. blog, *How blockchain can be hacked*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://cipher.com/blog/how-blockchain-can-be-hacked-the-51-rule-and-more/.

[45] Github, *Github*, Web Page, Last accessed 11 November 2021. [Online]. Available: https://github.com/.

Appendix A

Additional Material

OPENSSL FOR LAB 2.1

```
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
#

# Note that you can include other files from the main configuration
# file using the .include directive.
#.include filename

# This definition stops the following lines choking if HOME isn't
# defined.
HOME                    = .

# Extra OBJECT IDENTIFIER info:
#oid_file               = $ENV::HOME/.oid
oid_section             = new_oids

# System default
openssl_conf = default_conf

# To use this configuration file with the "-extfile" option of the
# "openssl x509" utility, name here the section containing the
# X.509v3 extensions to use:
# extensions            =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)

[ new_oids ]

# We can add new OIDs in here for use by 'ca', 'req' and 'ts'.
```

```
# Add a simple OID like this:
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

# Policies used by the TSA examples.
tsa_policy1 = 1.2.3.4.1
tsa_policy2 = 1.2.3.4.5.6
tsa_policy3 = 1.2.3.4.5.7

####################################################################
[ ca ]
default_ca      = CA_default            # The default ca section

####################################################################
[ CA_default ]

dir             = .             # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt # database index file.
#unique_subject         = no                    # Set to 'no' to allow creation of
                                        # several certs with same subject.
new_certs_dir   = $dir/certs            # default place for new certs.

certificate     = $dir/cacert.pem       # The CA certificate
serial          = $dir/serial           # The current serial number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out to leave a V1 CRL
crl             = $dir/crl.pem          # The current CRL
private_key     = $dir/cakey.pem# The private key
```

```
x509_extensions     = usr_cert          # The extensions to add to the cert

# Comment out the following two lines for the "traditional"
# (and highly broken) format.
name_opt    = ca_default        # Subject Name options
cert_opt    = ca_default        # Certificate field options

# Extension copying option: use with caution.
# copy_extensions = copy

# Extensions to add to a CRL. Note: Netscape communicator chokes on V2 CRLs
# so this is commented out by default to leave a V1 CRL.
# crlnumber must also be commented out to leave a V1 CRL.
# crl_extensions    = crl_ext

default_days    = 365           # how long to certify for
default_crl_days= 30            # how long before next CRL
default_md  = default           # use public key default MD
preserve    = no                # keep passed DN ordering

# A few difference way of specifying how similar the request should look
# For type CA, the listed attributes must be the same, and the optional
# and supplied fields are just that :-)
policy      = policy_match

# For the CA policy
[ policy_match ]
countryName         = match
stateOrProvinceName = match
organizationName    = match
```

```
organizationalUnitName          = optional
commonName              = supplied
emailAddress            = optional

# For the 'anything' policy
# At this point in time, you must list all acceptable 'object'
# types.
[ policy_anything ]
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName          = optional
commonName              = supplied
emailAddress            = optional


####################################################################
[ req ]
default_bits            = 2048
default_keyfile  = privkey.pem
distinguished_name      = req_distinguished_name
attributes              = req_attributes
x509_extensions         = v3_ca # The extensions to add to the self signed cert

# Passwords for private keys if not present they will be prompted for
# input_password = secret
# output_password = secret

# This sets a mask for permitted string types. There are several options.
# default: PrintableString, T61String, BMPString.
```

```
# pkix     : PrintableString, BMPString (PKIX recommendation before 2004)
# utf8only: only UTF8Strings (PKIX recommendation after 2004).
# nombstr : PrintableString, T61String (no BMPStrings or UTF8Strings).
# MASK:XXXX a literal mask value.
# WARNING: ancient versions of Netscape crash on BMPStrings or UTF8Strings.
string_mask = utf8only

# req_extensions = v3_req # The extensions to add to a certificate request

[ req_distinguished_name ]
countryName                     = Country Name (2 letter code)
countryName_default             = AU
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = Some-State

localityName                    = Locality Name (eg, city)

0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = Internet Widgits Pty Ltd

# we can do this but it is not needed normally :-)
#1.organizationName             = Second Organization Name (eg, company)
#1.organizationName_default     = World Wide Web Pty Ltd

organizationalUnitName          = Organizational Unit Name (eg, section)
#organizationalUnitName_default =

commonName                      = Common Name (e.g. server FQDN or YOUR name)
```

```
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 64

# SET-ex3                       = SET extension number 3

[ req_attributes ]
challengePassword               = A challenge password
challengePassword_min                   = 4
challengePassword_max                   = 20

unstructuredName                = An optional company name

[ usr_cert ]

# These extensions are added when 'ca' signs a request.

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE

# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType                    = server

# For an object signing certificate this would be used.
# nsCertType = objsign
```

```
# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment                       = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move

# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl              = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
```

```
#nsCaPolicyUrl

#nsSslServerName


# This is required for TSA certificates.

# extendedKeyUsage = critical,timeStamping


[ v3_req ]


# Extensions to add to a certificate request


basicConstraints = CA:FALSE

keyUsage = nonRepudiation, digitalSignature, keyEncipherment


[ v3_ca ]



# Extensions for a typical CA



# PKIX recommendation.


subjectKeyIdentifier=hash


authorityKeyIdentifier=keyid:always,issuer


basicConstraints = critical,CA:true


# Key usage: this is typical for a CA certificate. However since it will

# prevent it being used as an test self-signed certificate it is best

# left out by default.

# keyUsage = cRLSign, keyCertSign
```

```
# Some might want this also
# nsCertType = sslCA, emailCA

# Include email address in subject alt name: another PKIX recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy

# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where 'obj' is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF

[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a CRL.

# issuerAltName=issuer:copy
authorityKeyIdentifier=keyid:always

[ proxy_cert_ext ]
# These extensions should be added when creating a proxy certificate

# This goes against PKIX guidelines but some CAs do it and some software
# requires this to avoid interpreting an end user certificate as a CA.

basicConstraints=CA:FALSE
```

```
# Here are some examples of the usage of nsCertType. If it is omitted
# the certificate can be used for anything *except* object signing.

# This is OK for an SSL server.
# nsCertType                    = server

# For an object signing certificate this would be used.
# nsCertType = objsign

# For normal client use this is typical
# nsCertType = client, email

# and for everything including object signing:
# nsCertType = client, email, objsign

# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment

# This will be displayed in Netscape's comment listbox.
nsComment                     = "OpenSSL Generated Certificate"

# PKIX recommendations harmless if included in all certificates.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer

# This stuff is for subjectAltName and issuerAltname.
# Import the email address.
# subjectAltName=email:copy
# An alternative to produce certificates that aren't
# deprecated according to PKIX.
# subjectAltName=email:move
```

```
# Copy subject details
# issuerAltName=issuer:copy

#nsCaRevocationUrl              = http://www.domain.dom/ca-crl.pem
#nsBaseUrl
#nsRevocationUrl
#nsRenewalUrl
#nsCaPolicyUrl
#nsSslServerName

# This really needs to be in place for it to be a proxy certificate.
proxyCertInfo=critical,language:id-ppl-anyLanguage,pathlen:3,policy:foo

####################################################################
####################################################################
[ CA_SubCA ]
dir             = .             # Where everything is kept
certs           = $dir/certs            # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt # database index file.
new_certs_dir   = $dir/newcerts                 # default place for new certs.
certificate     = $dir/certs/subca.crt   # The CA certificate
serial          = $dir/serial           # The current serial number
crlnumber       = $dir/crlnumber        # the current crl number
                                        # must be commented out to leave a V1 CRL
crl         = $dir/crl.pem        # The current CRL
private_key     = $dir/private/subca.key    # The private key
RANDFILE        = $dir/private/.subca.rand  # private random number file
x509_extensions         = user_cert             # The extentions to add to the cert
name_opt        = ca_default     # Subject Name options
```

```
cert_opt        = ca_default    # Certificate field options
default_days    = 3650          # how long to certify for
default_crl_days= 30            # how long before next CRL
default_md      = sha512        # use public key default MD
preserve        = no            # keep passed DN ordering
policy          = policy_match
[ tsa ]

default_tsa = tsa_config1       # the default TSA section

[ tsa_config1 ]

# These are used by the TSA reply generation only.
dir             = ./demoCA              # TSA root directory
serial          = $dir/tsaserial # The current serial number (mandatory)
crypto_device   = builtin               # OpenSSL engine to use for signing
signer_cert     = $dir/tsacert.pem      # The TSA signing certificate
                                        # (optional)
certs           = $dir/cacert.pem       # Certificate chain to include in reply
                                        # (optional)
signer_key      = $dir/private/tsakey.pem # The TSA private key (optional)
signer_digest  = sha256                 # Signing digest to use. (Optional)
default_policy  = tsa_policy1           # Policy if request did not specify it
                                        # (optional)
other_policies  = tsa_policy2, tsa_policy3      # acceptable policies (optional)
digests     = sha1, sha256, sha384, sha512  # Acceptable message digests (mandatory)
accuracy        = secs:1, millisecs:500, microsecs:100    # (optional)
clock_precision_digits  = 0     # number of digits after dot. (optional)
ordering                = yes   # Is ordering defined for timestamps?
                                # (optional, default: no)
tsa_name                = yes   # Must the TSA name be included in the reply?
```

```
                                # (optional, default: no)
ess_cert_id_chain       = no    # Must the ESS cert id chain be included?
                                # (optional, default: no)
ess_cert_id_alg         = sha1  # algorithm to compute certificate
                                # identifier (optional, default: sha1)
[default_conf]
ssl_conf = ssl_sect

[ssl_sect]
system_default = system_default_sect

[system_default_sect]
MinProtocol = TLSv1.2
CipherString = DEFAULT@SECLEVEL=2
```