Sindre Davidsen Schonhowd
Magnus Lekanger Voll
Øivind Wahlstrøm

Framework for secure data collection and through integration with various APIs

Bachelor's thesis in Digital infrastruktur og cybersikkerhet
Supervisor: Muhammad Mudassar Yamin
May 2023

Bachelor's thesis

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

NTNU
Norwegian University of Science and Technology

Sindre Davidsen Schonhowd
Magnus Lekanger Voll
Øivind Wahlstrøm

Framework for secure data collection and through integration with various APIs

Bachelor’s thesis in Digital infrastruktur og cybersikkerhet
Supervisor: Muhammad Mudassar Yamin
May 2023

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

Abstract

Tussa is a Norwegian company based on the west coast of Norway. Tussa's vision is to create innovative and environmentally friendly solutions. They wanted a better solution for data collection through Microsoft Intune, Cisco Secure Endpoint and Cisco Umbrella to their customers. Therefore this project builds on the assignment of doing research and improve the procedures for SaaS service integration using APIs. The solution should follow Best Practise security strategy for managing scripts, code, API credentials and data. The goal of this project is to develop a solution that collects this data via API calls and displays it in a readable and understandable interface for Tussa's customers.

The project will cover the entire process of developing the solution. All the way from the planning process to implementation. The first couple of chapters covers relevant information needed to understand the solution and the technologies used. Then the report moves over to the more technical part, covering how the solution is built and the technical design with the different microservices. We also conducted a risk assessment for the solution and the work related to the task. Then the report moves over to evaluating and discussing different choices made both on the technical part and how the work was conducted. Wrapping up the report we made a conclusion and consideration for future implementation.

Sammendrag

Tussa er ett norskt selskap lokalisert på vestlandet som arbeider for å skape fremtidsrettet og miljøvennlige løsninger. Tussa ønsket en forbedret løsning på datainnsamlingen fra sine kunder gjennom Microsoft Intune, Cisco Secure Endpoint og Cisco Umbrella. Derfor vil dette prosjektet omhandle å undersøke og forbedre prosedyrene for SaaS servicer med bruk av APIs. Denne løsningen skal følge Best Practise sikkerhetsstrategi når det kommer til behandlingen av script, kode, API credentials og data. Målet med dette prosjektet er å utvikle en løsning som samler inn denne informasjonen ved å bruke API calls, og samle det i en lesbar oversikt for kundene til Tussa.

Oppgaven vil dekke hele prosessen med å utvikle denne applikasjonen. Helt fra startfasen med planleggingsprosessen til implementasjon. De første kapitlene dekker relevant informasjon både på selve løsningen men også ulike teknologier og fagbegreper som er brukt i løsningen. Deretter går rapporten over i teknisk forståelse av applikasjonen, hvor vi beskriver det tekniske designet og hvordan applikasjonen er bygd opp med ulike mikroservicer. Det er også gjennomført en risikoanalyse av den nye løsningen, og arbeidet med dette. Videre går rapporten over i en evaluering og diskusjonsdel hvor vi diskuterer ulike valg som er gjort underveis både med arbeidet og løsningen. Til slutt er en konklusjon av oppgaven samt forslag til fremtidig utvikling.

Preface

We want to share our deepest gratitude to everyone who has been a part of this project. During the entire project there has been multiple participants who has both supported and helped us to get through.

First we would like to thank Md Mujahid Islam Peal and the CyberRange located at NTNU Gjøvik for sharing knowledge about different technologies and providing a highly valuable demonstration of how they work. This remarkable contribution improved our perspective and understanding of available technologies and their significant utility.

Secondly we would like to manifest our profounding appreciation to our supervisor, Muhammad Mudassar Yamin. He has dedicated great effort in helping us achieve a substantial product, and played a crucial role guiding us onto the right path from the very beginning. He has expressed his dedication to our work, by being available from morning to evening throughout the project. He as played a key role by answered questions, providing feedback and helped us reviewing the bachelor thesis to the very end. We truly valued having you as our advisor during this project.

Finally, we would we like to express our gratitude towards our client Tussa, and their hospitality for allowing us to visit them at Ørsta. We cherish the professional relationship we have evolved together. We thank Kjetil L. Sætre and Vigleik Hustadne for their excellent guides, keeping us on track. We highlight a special acknowledgement to Benjamin for providing excellent and persistence availability at all times by helping with technical issues and decisions. A special thank you to Vigleik is also in place for his availability at all times as well answering administrative questions.

Contents

Figures

Tables

Code Listings

Glossary

3-2-1 3-2-1 backup rule is a security measure for backup, where you want to do 3 copies of your data (the production data and 2 copies). 72

AKV AKV stands for Azure key vault. 72

API Application Programming Interface - Mechanism that enables two software solutions to communicate. v, 1, 3, 12, 15, 17, 22, 71, 101, 104, 106

Azure Azure is a cloud platform that gives the user the opportunity to design and manage infrastructure and data with over 200 different products and services. 29

Best Practise Best practise is guidelines and standards that is known to produce good and efficient outcomes. iii, v, 2, 71

Break Glass Expression for emergency access account in Azure AD. 47

ChatGPT ChatGPT is an AI chat robot that offers answers to your questions. 118

Compromised Means that something is made vulnerable by either unauthorized access or exsposure.. 8

Container Image Container Image is a static file with executable code which can create a container on a system.. 9, 90

Data At Transit Data at transit refers to the movement of digital information across networks.. 111

DDOS Distributed Denial of Service - Is a cyberattack that aims to flood the network with traffic so the network wont work properly.. 107

Dependencies dependencies are the name of libraries in JavaScript that an application depend on.. 42

Discord Communication application. 37, 118

DMZ Demilitarized zone (DMZ) is an extra layer of protection on an internal network. The zone restricts the access to sensitive data from external users.. 110

GitHub Code hosting platform for collaboration and version control. 39

Horizontal Horizontal scaling is adding more servers so that the load is distributed across more nodes, making the load on a single node lighter.. 107

IPsec/ESP IPsec/ESP is short for Internet Protocol Security and Encapsulating Security Payload. Network protocol that provides secure communication by encrypting and authenictating data traffic.. 51

LaTeX Is a mark up language specially suited for scientific documents. 38

Libraries libraries are a set of pre-built code which are designed to execute specific tasks and functions.. 42

MFA Multi-Factor Authentication - Authentication method that requires the user to use more than one verification ti gain access.. 26, 59, 72

Microservice Multiple services working together. 23, 24, 29

Middleware Middleware is a software between a application and the operating system its running on. This is used as a translation layer to see the different components and handle data.. 106

OSI model Open Source Interconnection - Is a standard for how protocols work in a network and describes how data transfers between two machines.. 111

Overlay Network Overlay network is a simply a network built on top of another.. 65, 70

Overleaf Online writing software. 38

Pseudo Code Pseudo code is a text based algorithms. The code is written as regulare text to make it more understandable.. 45, 48

Q&A Questions and Answers. 72

RDP Remote desktop protocl (RDP) is a protocol for using a desktop computer remote. . 108

Reverse Proxy Reverse proxy is a security measure that is enabled in front of a web server to forward client requests to those web servers. 17, 24

SaaS Software as a Service - Cloud based software that gives the end user access to applications over the internet. 1, 2, 119

SSH Secure shell (SSH) is a protocol created for network communication.. 108

SSL Secure Sockets Layer - Technology used for keeping an internet connection secure. 22, 103

SSO Single Sign On - Is a identification method that gives the user access to multiple applications with a single sign in.. 26

stateful Stateful means the state is saved on the same server.. 108, 109

Stateless Stateless means the state is saved on an external server.. 108

Sticky Session Sticky sessions is a method that makes it possible for a load balancer to identify requests coming from a client and always send there requests to the same server for the duration of a session.. 108

TLS Transport layer security (TLS) is a protection that encrypts data sent in transport.. 108

Vlan Virtual local area network (vlan) is a virtualized connection which connects devices and nodes from different LANs into one network.. 110

Volume Volume is a storage area with a single file system. 66, 70

Zero-Day zero-day is a security flaw in hardware, firmware or software that is unknown to the responsible parties.. 109

Chapter 1

Introduction

This chapter provide an introduction to the project. Goals and frames as well as the objective of the project. We are also covering the different participates and what background we have to meet the different challenges in the project. The introduction are also covering the problem delamination, meaning the different aspects, carried out through the thesis. At the last stage of the introduction we cover a short summary of each of the chapters.

1.1 Tussa IKT - client

Tussa IKT is a subsidiary of the Tussa Group in Norway. Tussa delivers management IT as well as cloud and security services for different customers. To be able to do this in an efficient way they run their own data center. They utilise a substantial amount of Software as a service (SaaS)- and communications suppliers together with their own fiber network. Tussa IKT is ISO27001-certified [1], which means that they have information security as a high priority.

This is important for us to take into consideration when executing our bachelor thesis. The problem area in our bachelor thesis lays between Tussa IKT and their SaaS-services. Today the reporting between them is not efficient enough, and Tussa wants us to come up with a better way to collect and display valuable data to the customers without losing any security features. They want to be able to give the customers a daily report with relevant information, and that the data collected from different sources is going to be showed in one interface.

1.2 Project Objective

The objective of the assignment is to research and improve the procedures for SaaS service integration using Application programming interface (API)s, with a particular focus on risk assessment and securing the process around handling API credentials [2].

The group is going to develop a better solution for reporting valuable data from the SaaS-services. This data is being collected from Microsoft Intune, Cisco Umbrella and Cisco Secure Endpoint and then given in a daily report, and displayed it in an interface with built-in security features. We are also creating a risk assessment of the different services and the implementation of the application.

1.3 Goals and Frames

1.3.1 Frames

The time frame we have to complete the project is from the 11th of January to the 22th of May 2023. Both our solution and the report need to be finished by this deadline.

1.3.2 Result Goals

- The bachelor group should come up with an efficient way to extract and report relevant information from the SaaS-services.
- There should be some routines used in the development that follows Best Practise, especially regarding API management.
- The reporting and storing of data need to be secure from possible cyber threats.
- The customers should be able to get a daily report with all the relevant information.
- The bachelor group is going to deliver a risk analysis of the new reporting system to make sure that it keeps up with the security standards for both Tussa and their customers.

1.3.3 Effect Goals

- Our solution should greatly increase the efficiency and reliability on the documentation and reporting between Tussa and their clients.
- Our solutions should be secure in a way that the customers can rely on that their information is stored in a secure manure and not in danger to cyberthreats.
- The work we deliver should be modifiable at a later stage, either by Tussa or another group. Meaning that others outside our group should be able to expand, optimize and tweak our solutions at a later stage.

1.3.4 Problem Delamination

Regarding the Problem Delamination, the focus on development of a secure platform that displays logs for dedicated endpoints to respective tenants. We are also using open-source tools in our development. What log fields that are going to be displayed, and how, is decided by the Client, however we are going to implement a solution that gives room for customization for future use. The intention is to make sure that the open-source tools we use are going to fulfill the security requirements sat by Tussa. The group aims to make a risk analysis containing a risk assessment of the different services and the implementation of the application. We intend to also perform security testing of our solution when the product is done.

1.4 Group Background

The client of this project is Tussa IKT. Our main and technical point of contact is Benjamin Yndestad. While Kjetil Sætre and Vigleik Hustadnes are in charge of the administrative regadring this cases. Benjamin's role is to answer questions regarding development, technical issues etc. Kjetil and Vigleik's role is to define our task, and answer the administrative questions regarding the project. Our supervisor from the Norwegian University of Science and Technology (NTNU) in Gjøvik is Muhammad Mudassar Yamin.

Our group consists of Magnus Lekanger Voll, Sindre Davidsen Schonhowd and Øivind Wahlstrøm. We are following the study program Digital Infrastructure and Cybersecurity at NTNU Gjøvik. We applied for this assignment based on our previous experiences in earlier courses such as DCSG2003 - Robust and scalable services, where we learned about container-based infrastructure and orchestration tools such as Docker Swarm and the course DCSG2005 - Risk Management where we learned about risk management and how to perform a risk analysis. Øivind also had an exchange year in Australia where he learned about APIs and cloud computing.

We have no experience regarding multi-tenant environments, nor developing open-source tools for log management so we have to research and learn how to utilize it in a way that satisfies what is required for the task. Nevertheless do we feel that this experience is highly relevant for the future job market.

1.5 Organization

The organizing of the different roles for our project was a part of our project planning at the very start of the project. We specified roles and main responsibility areas in the project plan. For more information regarding this, see Appendix B.

1.6 Thesis Structure

Chapter 1 - Introduction: This chapter provides an overview of the project, including its background, objectives, and the individuals involved. It aims to give the reader a clear understanding of the purpose and scope of the thesis.

Chapter 2 - Background: In this chapter, we dive into the technical details necessary to comprehend the solution. We discuss relevant concepts, technologies, and essential information that lay the foundation for understanding the rest of the thesis.

Chapter 3 - Requirements: This chapter outlines the specific functional and non-functional requirements of the solution. We identify the desired features and functionalities, as well as performance and usability criteria that need to be met.

Chapter 4 - Technical Design: This chapter provides a detailed explanation of the technical design of our infrastructure and the microservices utilized to achieve our goals. We describe the underlying architecture and how different components interact with each other.

Chapter 5 - Development Process: In this chapter, we discuss the process of developing both the solution and the thesis itself. We cover the methodologies, tools, and steps taken during the development.

Chapter 6 - Implementation: Here, we provide a detailed technical description of how the infrastructure is implemented. We explain the coding, configurations, and integration of various components to create a functioning system.

Chapter 7 - Deployment: This chapter focuses on the deployment of the solution into the client's systems. We describe the necessary steps, configurations, and maintenance routines required to successfully deploy and manage the solution in a real-world environment.

Chapter 8 - Security Testing: Security is a critical aspect, and in this chapter, we go through the security testing conducted. We discuss the methodologies, tools, and techniques used to assess the solution's security and present the findings.

Chapter 9 - Risk Assessment: Implementing any solution involves inherent risks. In this chapter, we conduct a risk assessment specific to the customer's context. We identify potential risks, evaluate their impact, and propose mitigation strategies both for the new solution and the development of it.

Chapter 10 - Discussion: Here, we engage in a discussion about the choices made during the development process and the thesis writing. We analyze these choices, provide reasoning behind them, and discuss their implications and potential alternatives for future consideration.

Chapter 11 - Evaluation: This chapter focuses on evaluating the solution and the project. We present the results of surveys or assessments conducted and evaluate how well the solution aligns with the defined requirements.

Chapter 12 - Closing Remarks: Finally, in this concluding chapter, we summarize the key learnings from the project. We provide suggestions for future work and offer our overall conclusion and reflection on the journey undertaken.

Chapter 2

Background

This chapter aims to address two important aspects of the task. First we want to address the importance and benefits of developing a solution like this. The second is the introduction and description of key technologies and core concepts, which is needed to be able to fully understand the product.

2.1 Solution Background

In this section we are going to address the benefits and importance of developing a solution like this one. Tussa has addressed why they want a solution like this. Right now they collect data from three separate products and merge them into one report sheet to give their customers. The three products is Microsoft Intune, Cisco Umbrella and Cisco Secure Endpoint. This is deemed as an inefficient solution, and is considered to be poor time and resource utilization, which could have been delegated to more comprehensive due diligence which again could out put more value. This solution is going to drastically reduce the time used making reports for clients, and is also going to provide the opportunity for daily reports that is visually more understandable.

The solution does not only offer more efficiency, but also looks more professional for Tussa's customers. Reputation is a key aspect in businesses like this, and the professionalism provides better reputation. It gives both Tussa and their customers better overview of computers that might not be compliant or may not even be in use. This might even reduce the cost, because they can take computers not in use out of the system.

2.2 Least Privilege Access Control Principle

The Least privilege access control principle is a concept inside information security which describes how an entity or user should only have access to specific information or resources which are required to complete a task. By limiting access, the attack surface and the risk of malware are being reduced significantly [3]. Imagine if a user who had access to everything inside their organization got Compromised. A potential adversary executing the attack would gain access to everything. With the least privilege access control principle implemented, attackers are "only" going to have access to specific information or resources.

2.3 Auditing and Logging

Audit and logging are two specific terms that are central in terms of dealing with data. When talking about Information Security, auditing is the observation and evaluation of an organizations IT infrastructure, policies, procedures, applications and data management and usage. The goal of the audit is to evaluate if the information technology assets ensure integrity and align with the organizations goals and objectives [4].

Logging in Information Technology is the act of making a log of events that occur in a computer system. Events can be errors, problems or maybe just information on current operations. Events like this may occur in both operating systems or in software [5].

2.4 Containers

Docker's offical definition of a container are the folloiwng: "*A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.*"[6]. Although the definition is short and straight to the point, it does lack explanation.

What it means is that you can deploy code and its dependencies inside a container to run an application. They are designed to be portable, meaning that the applications can run reliably across different environments. A physical container is used for shipping and storing goods. They are standardized, meaning they are built the same way for making it easier for transportation with ships, trains, trailers etc.

The same way can be thought about software containers. They are standardized, portable and can be transported around different environments. The name "container" comes from the idea of a transportable package that can hold code and its dependencies, just as the physical container can hold and deliver its content. Containers are usually managed through container orchestration tools such as Kubernetes and Docker Swarm, which is addressed later.

2.5 Docker Swarm vs Kubernetes

Docker Swarm and Kubernetes are container orchestration tools, which is a software tool that automates administration of containerized applications such as deployment, management, and scaling. You can manage multiple containers across different hosts allowing you to do several changes in a scalable manner[7], [8].

When comparing these two, you find plenty of similarities as mentioned above, but there are certainly some key differences: Kubernetes is more complex to set up compared to Docker Swarm, and has a pretty steep learning curve if you have no experience with it [9]. On the other hand, Kubernetes does provide more features such as more advanced networking, storage options and support for more container-providers. Docker Swarm on the other hand is mainly designed for Docker containers.

When we decided what to use to develop our application, we based our choice on prior experience from earlier subjects during our Bachelor. In the subject DCSG2003 - Robust and scalable services [10] did we learned about Docker Swarm and how to use it. Since Kubernets has a steep learning curve for beginners, we rather want to spend our time on developing our knowledge about Docker Swarm when developing our solution. Docker Swarm justifies the requirements for what is needed to develop the application.

2.5.1 Docker Secrets

Docker Secrets is a security feature inside Docker Swarm that allows you to manage sensitive data such as passwords, and API keys. This feature allows you to encrypt and store data classified as sensitive outside of a Container Image, or source code. The data that is being encrypted is only available to the application at runtime [11].

Since we are using Docker Swarm, we can also use Docker Secrets, which is used to inject the data into the containers at runtime. This means that the secret data are exposed to the application, and gets updated automatically when the secrets are being rotated. With this feature, your application always have access to the latest secrets.

2.5.2 Overlay Network

Overlay network in the context with containerization and Docker Swarm is a type of network that enables communications between containers on different Docker hosts. The idea behind an overlay network is that a virtual network is created on top of the physical network, which allows containers to communicate as they were on the same host, even though they might be running on different places. When using Docker Swarm, an overlay network is being created which let you manage a cluster of Docker host as a single host.

2.6 Azure

Azure is a cloud service platform owned by Microsoft, which offers more than 200 products and cloud services. Inside Azure can you utilize tools and frameworks of your preferences to create, execute and manage applications across various clouds [12]. We intend to walk through services inside Azure that has been relevant to us for the development during the project.

2.6.1 Azure Key Vault

Azure Key Vault is a tool for securely storing and accessing secrets. Secrets is seen as anything you want restricted access to, for example API keys, passwords, certificates, or cryptographic keys. It's a cloud service that supports two types of containers, both vaults and managed hardware security module pools.

With centralized storage of secrets, you can use Azure Key Vault to control their distribution easily. This also greatly reduces the chances of secrets being accidentally leaked. When using a key vault, you no longer need to store security information in the application itself, and Azure Key Vault is a trusted and well recognized tool. Its then easy for the application to access the Key Vault when it needs to use for example a connection string [13].

The vault can be used to solve several problems such as secret management where the Key Vault can be used for storing and controlling access to tokens, passwords, API keys, certificates, and other secrets in a secure manner. It can also be used to handle key management where the key vault can be used to manage the keys. This can be used to easily create and control the encryption keys. At last can it also deal with certificate management. This means that the key Vault can also be used to manage and deploy both public and private Transport Layer security certificates for use with Azure and the applications connected resources.

Azure Key Vault is one of the most known key vaults, and therefore it is seen as a secure platform to store API keys. There are multiple layers of security to protect the keys and other secrets [13]. Some of them are:

Role-Based Access Control: Azure Key Vault allows you to tune the access policies for each key, giving you the power to limit who access and manages them.

Encryption at rest: All the keys stored are encrypted using algorithms.

Auditing and monitoring: Azure Key Vault log every access and usage of the keys. This means it's easier to detect and respond to suspicions activity.

Integration with other Azure services: Azure Key Vault integrates good with other Azure services, which means it's both easy to use and secure.

The Key Vault is designed to seamlessly integrate with other Azure services. This means that it's a secure and easy-to-use tool to use with other services made by Azure. Overall, Azure Key Vault is a secure and good tool to use, especially when using other Azure services. This makes a seamless integration that are built for each other. Azure Key Vault is also compliant with different certifications like SOC 1, SOC 2 and ISO 27001, making it an excellent tool to use for firms relaying on certifications.

Azure Key Vault provides activity logs that makes it easier to trace activity. All the access and usages of the keys and secrets are stored in the activity log. You can then use the Azure Portal to easily access these logs and trace unwanted activity. This is a great tool for better security, and it's easy to use because you can filter the logs after your wanted criteria. Azure Key Vault also provides diagnostic logs. This is a great tool to troubleshoot possible issues with the service, and it also provides insights into service performance and usage. Both the logs and diagnostic logs can be enabled through the Azure Portal.

2.6.2 Azure Container Registry

Azure Container Registry is a fully managed registry where you can manage container images for deployment to different Azure services. With fully managed means that Microsoft takes care of the underlying infrastructure to keep it up, so we can fully focus developing and deploy our container-based application. The registry provides a centralized location for management, and we can store our container-based application safely [14].

2.6.3 Azure Application Registration

Azure Application Registration is the process where Azure Active Directory, also named Azure AD, gets registered with the application in order to get access to the Azure services and the APIs. With this feature can we enable features such as multi-factor authentication and access control. Multi-factor authentication also called MFA, is a second factor of authentication to verify their identity to access the service they attempt.

Usually you sign in with username and password, and then get asked to verify with something else to verify that it is you who is attempting to log in, such as a code on either an email or a SMS on a phone. This process of registration gives us access to configure the application's permission and who can access what, which makes the part as an important step for securing the applications running in Azure [15].

2.6.4 Service Principal

Service principals within Azure are what we call security identities which can be used to authorize and authenticate services or applications to access Azure. This means that a Service Principal can be looked at as an identity which represents the application or the service which need to access Azure resources. Such resources can be storage accounts or Azure Key Vault [16].

It might look like that Azure Application Registration and Service Principal are pretty much the same, but the difference is that the process of registering the application with Azure AD, and configuring its policies when it comes to authentication and authorization. Service Principal on the other hand is what that represents an application or services identity inside Azure AD, which can be used to get access to Azure Resources.

2.6.5 Azure Groups

Azure Groups are a term that allows to you organize and enhance accesses management to resources inside Azure. The groups can be used as a management tool to handle permissions and give access to groups instead of a single persons. By using Azure Groups does it make it a lot easier to handle and manage large numbers of users inside Azure, because you give the permissions to a group instead of a single person.

As people often changes jobs and roles inside an organization, it might be assumed that they are not going to have the access to the same content as they used to, following the least privilege access control. Instead of spending a lot of time editing the permissions and accesses of every single user that changes roles or jobs, you can just edit the Azure Group by removing or adding them to a group [17].

2.7 Microservice Architecture vs Monolith Architecture

2.7.1 Microservice Architecture

Microservices architecture is also known as microservices. This architecture is built with a series of independently deploy-able services. Each of these services have their own database and business logic with a specific goal. Microservices makes it easier to manage different services because the tasks are separated into smaller processes. It does not reduce the complexity but makes it more visible and easier to modify.
One major advantage with microservices is the opportunity to develop, update, deploy and scale each service independently without affecting the other services. You can perform software updates more frequently with improved reliability, uptime, and performance. This is valuable for growing softwares and companies.
There can be some disadvantages of microservices. It can be very complex to set up a microservices architecture. Especially when the architecture gets bigger, there are more services in more places, which makes the whole process more complex. This can again lead to development sprawl. It can also be difficult to see how different components relate to each other [18].

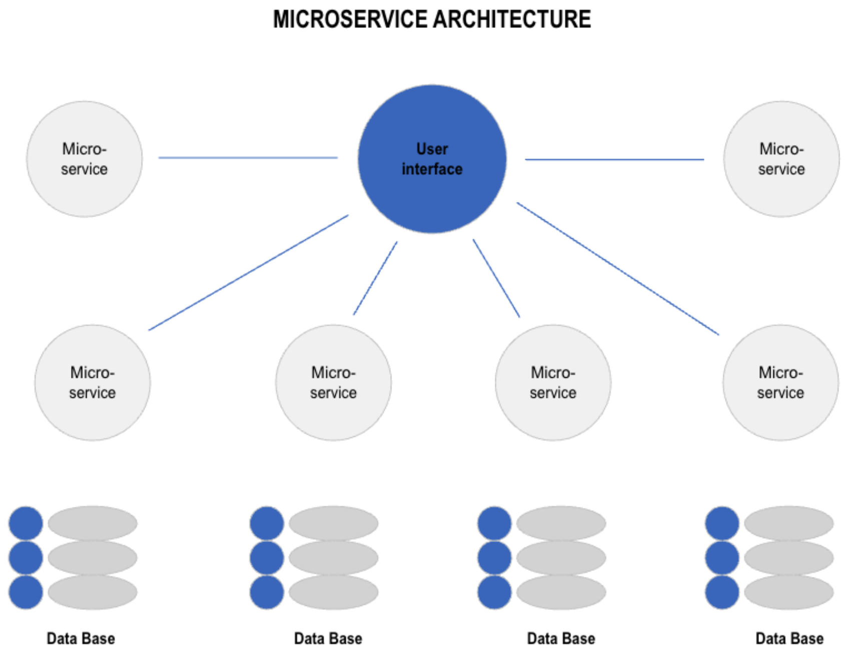


Figure 2.1: This picture shows how Microservices work

2.7.2 Monolith Architecture

The more traditional architecture is the monolithic architecture. This architecture is built as a unified unit which is self-contained. It is called monolithic because it's built as a singular, large computing network with one large code base that ties all the business concerns together. Monolithic architecture has some great benefits, for example that it is easy to deploy. When deployed its usually one executable file or directory. One API can often perform the same function that numerous APIs perform with microservices when it's in a centralized code base or repository. It's also easier to perform end-to-end testing when all the code is in one place.

There are some disadvantages with a monolithic architecture. When the application becomes too big, the development speed becomes slower. This is because the development becomes more complex.
The reliability can also become a problem. If there is a problem with one module, it could affect the entire applications availability. It can also become a problem in development because a small change to the application requires the redeployment of the entire monolith, which can be time consuming end expensive.
The flexibility is also greatly decreased, because the monolith is constrained by the technologies currently used in the monolith [19].

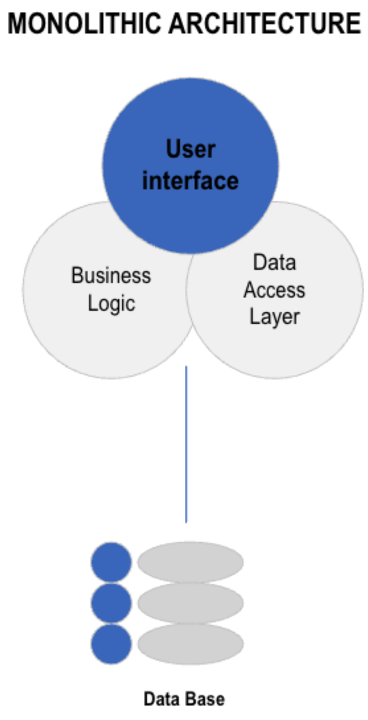


Figure 2.2: This picture shows how Monolithic architecture work

2.7.3 Preferable Architecture

When it comes to our project, we have done research on both architectures to find out what would suit us and our client best. Since our group consist of three bachelor students with little to none experience from real life development, we find the microservice architecture to be the most optimal for us.

There are several reasons to this, one of which is because of our lack of experience. By this we mean that we are most likely to make mistakes in the code at some point. When we are going to fix these mistakes it's easier to have each service separated and not in one single code base.

Another reason for our choice is the continuous deployment. With microservices we can have more frequent and faster release cycles. It's easier to do updates to a single service without effecting all the other services. We also want Tussa to be able to develop the product after we are finished. They may want to develop to newer technology in the future, and this is more doable with microservice. Microservices makes the deployment of new technologies easier, and with a higher reliability.

2.8 Services

In Information Technology a service is referred to as an application a business uses to help them in the creation, management and optimization of their information and business processes. These services can be segmented into three segments; design, build and run. In this section we cover the different services we use in the application. What they are and how they work.

2.8.1 Grafana

Grafana is an open-source platform that is great for visualizing data. The data can be visualised via charts and graphs which are unified into dashboard(s), allowing a better understanding of the data [20]. As in our case, do we visualized log-fields on respective endpoints. Grafana also allows you to filter the current log fields.

As an example, we do have a log-field named "Compliance" showing either non-compliant or compliant. With the filter-method we can decide if we want to only show the endpoints which is either non-compliant or compliant. Another great feature inside Grafana is that we can configure it to be compliant with both Azure and multi-tenancy [21], allowing us to meet the requirements from the client.

2.8.2 ElasticSearch

ElasticSearch is an open-source tool that is being used as a search and analytics engine to handle large volume of data in real-time. Data, or in our case, logs are being stored in a JSON-format, which was one of the requirements from the client, for making it more easy to index and search through the data. This benefit us greatly as Grafana update its information frequently by querying Elasticsearch [22].

2.8.3 Node.js Running Cron Job For API Calls

A cron job is a scheduled job that runs automatically at a specified time. So when talking about Node.js app running a cron job for API calls, we talk about a scheduled job where Node.js make the API-calls. Node.js is an open-source platform that allows users to use JavaScript on server-side of web-applications [23]. It is a widely adopted tool for building scale-able web applications and APIs.

2.8.4 Nginx

Nginx is a powerful open-source software that offers an array of services which is well known and widely used on the real world marked. Such services can be web servers, Reverse Proxy and load balancers. What makes Nginx popular, is that it can handle a high volume of simultaneous connections, and would still deliver content to the site efficiently.

2.8.5 Node.js

Node.js is a great tool for developers to develop both front-end and back-end applications using Javascript. Node.js is open-source, which means the sourcecode is available online and that people all over the world can contribute. It is also cross-platform, which gives the users the opportunity to use Node.js on different operating systems. Node.js can be executed outside a web browser because it runs on Chrome's V8 Javascript engine and it enables developers to make high-performance and scalable web applications [24].

2.8.6 Javascript

Javascript is a programming language primarily used for building dynamic and interactive web applications. The high-level language can be executed on both the client-side and the server-side, making it versatile. Javascript is commonly used by developers all over the world [25].

2.8.7 NPM

NPM (Node Package Manager) is a package manager used for Javascript and Node.js applications. NPM gives developers the opportunity to discover, install and manage dependencies easily. Such as libraries, frameworks and tools used for their Javascript projects. NPM is a bundle with Node.js which allows you to interact with the NPM registry with a command-line interface. The NPM registry is a public repository hosting thousands of open-source Javascript packages [26].

2.9 Related work

As a part of our research, we looked at online solutions. The goal of this section is to look at available solutions online to to get inspiration as well as explore how our applications differs compared to the ones already existing. It is assumed that there might be other solutions which are private that has the same purposes as our application with parsing several log-files into one.

Our research showed that there are multiple solutions of how to display logs in Grafana, such as Splunk and ELK Stack, but there were few to no solutions that was exactly like ours. By that do we mean having multiple log-files from the same endpoint, delivered from different vendors, parsed together into one log-file and then display it in Grafana in a multi-tenancy solution. With that being said, it does not mean that there are not other solutions, as there might be several that are private. We used open source services that are free, and highly customizable to your needs, which means there are likely to be something very similar.

As log fetching is very common, and often specialised to demands, are almost every use case different but with the same purpose of fetching logs and visualize it somehow. It seems that most of the solutions we find online is real time analyzing the logs, while our application takes a snapshot of the current log fields from the vendors, parses it and then displays it in Grafana. A common solution that we mentioned earlier and is widely used and similar to ours is an ELK stack.

2.9.1 ELK stack

An ELK stack is a set of open-source tools such as Elasticsearch, Logstash, and Kibana which are often used for log management:

Logstash: Receives, modifies and transmits the data to desirable location.

Elasticsearch: Indexes the data.

Kibana: Visualizes the data.

Each service provide different jobs that help to collect, store, search and visualize huge amount of data [27].

This means we could have used an ELK stack for our development of our application, however the client wanted to display logs in Grafana and not Kibana. Even though the results could have been the same as both services support visualization of logs, Grafana is more of a general-purpose tool that can visualize logs, but it can not text query, which Kibana Supports [28].

2.9.2 Splunk

The other solution that we looked at was to use Splunk. Splunk is a management platform that allows to collect, index and visualize logs . The platform does also supports integration of visualizing the logs inside Grafana [29]. Based on their documentation does it also seems you can customize it to parse the logs in to one log file, as the client demands [30]. This means we could probably have used Splunk to develop our solution.

An issue that led us to not choose to use this solution is the costs. When researching the prices of Splunk on their website, they do not provide specific pricing information. Instead, they request you to fill in your requirements, and later be contacted by their team in order to obtain pricing details [31]. Splunk offers customized pricing based on the specific needs.

With research, can we make an estimation of how much this are going to cost. The source reports that a licence might start at 65 USD per host, per month [32]. Considering how many hosts/endpoints there are going to be collected logs from, the cost, did we not see this as a viable solution. We knew we could develop a solution with the same output for way less cost.

This means that you can use multiple services online to develop a solution that produces the same result. It depends on what services you want to use, available resources and requirements. As our solution is customized to the clients requirements, available resources and our knowledge of how to utilize the services it makes it somewhat unique.

Chapter 3

Requirements

In this chapter we want to define the requirements for the application. We divide the requirements in to functional and non-functional requirements which each specify aspects of the application. Functional requirements are often defined as something a system must do, and is often related to the technical details of a system. This is the requirements the system needs to be able to operate properly. This can often be reviewed through the systems input and output [33].

Non functional requirements is complementary to the functional requirements, and can be defined as the requirements that describe how a system works. This might sound like less important than the functional requirements, but this is plays a huge part in how good a system operates. This means the non functional requirements does not have an impact on the functionality but it does impacted the performance. Non functional requirements can be seen as the systems usability [33].

3.1 Functional Requirements

F-1: ***The solution should contain access control to ensure confidentiality.***

Description: To ensure confidentiality for each of the clients, we need to use access control. Access control is used to ensure that users only get the information they need, and to ensure that no one access information without the proper authorization.

F-2: ***The solution should be able to fetch information about devices from API calls.***

Description: One important aspect of the task is to be able to fetch information from each clients devices. This information needs to be fetched from all three products (Microsoft Intune, Cisco Secure Endpoint and Cisco Umbrella) and display all the important information in one place.

F-3: ***The solutions needs to use encryption to secure information.***

Description: Another important aspect of the task is to deliver a solution that uses encryption in a secure way. SSL is a great way to secure the internet connections.

F-4: ***The solutions needs to store the information for the devices.***

Description: The data from the different devices needs to be stored in some way, so the clients have access to their data, and not just in real time. This can be done with something like Elasticsearch.

F-5: ***The solutions needs to securely store API credentials and passwords.***

Description: The API credentials and passwords needs to be stored securely to ensure confidentiality and as a security measure. This can be done through a service like Azure Key Vault.

3.2 Non-Functional Requirements

N-1: ***The solution should be easy to use with minimal or no training.***

Description: The solution should be so easy to use that it requires minimal or no training for the users. Therefore its important to make a application that is intuitive.

N-2: ***The solution should be easy to understand and interactive.***

Description: The interface should be easy to understand for the customers and Tussa, so they can use as little time possible on understanding the design.

Chapter 4

Technical Design

This chapter aims to explain the technical design of our infrastructure, and the Microservice utilized to achieve our goals. The microservices being Nginx, Grafana, Node.js and Elasticsearch. It also explains the technologies used to deploy and run the services, as well as explaining how it works.

4.1 System Architecture

The system architecture describes the solutions on how we have solved the problem that Tussa presented to us. To better understand the infrastructure, have created an illustration, Figure 4.1 below. The yellow square represents Virtual machines (VM) and the microservices running within it are represented as the blue squares. The blue frame surrounding the VMs aims to illustrate Azure, and the VMs are running inside it.

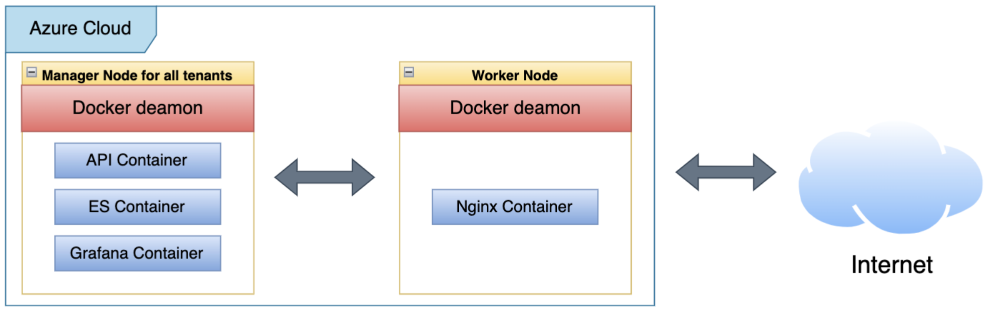


Figure 4.1: Main infrastructure

The infrastructure is hosted on Azure's Virtual machines. It contains several Microservice running inside containers manged by a docker swarm. All of the microservices share one virtual machine except Nginx. This is done because of the role and features Nginx provides. The main purpose of the Nginx is to protect the rest of the infrastructure, act as a Reverse Proxy, and to be a load balancer. This is the most cost effective solution compared to the other options discussed in the next chapter.

The API container, Elasticsearch container and Grafana container are all sharing one virtual machine (VM). They each poses their specific role that is: gather information, store information, visualize information respectively. Having individual roles for each task makes it easier to manage, and to give the possibility to change any of them in the future if deemed necessary, making it highly flexible. The solution is builds upon an easy scaling integration for future implementations i.e more VMs and container replicas can be added if needed, due to the nature of the Nginx load balancer.

4.1.1 Traffic Flow

To get a better understanding on how the traffic flow works within our application, see Figure 4.2. The Figure tend to explain the communication flow for when our service is deployed and when a user tries to log in to Grafana with their Azure AD account, regardless if its an admin or a normal user. Each yellow box symbolise Azure VMs with their respective microservices running on them, each of the nodes are apart of the docker swarm. The blue square symbolise Azure and its services used to power our application.

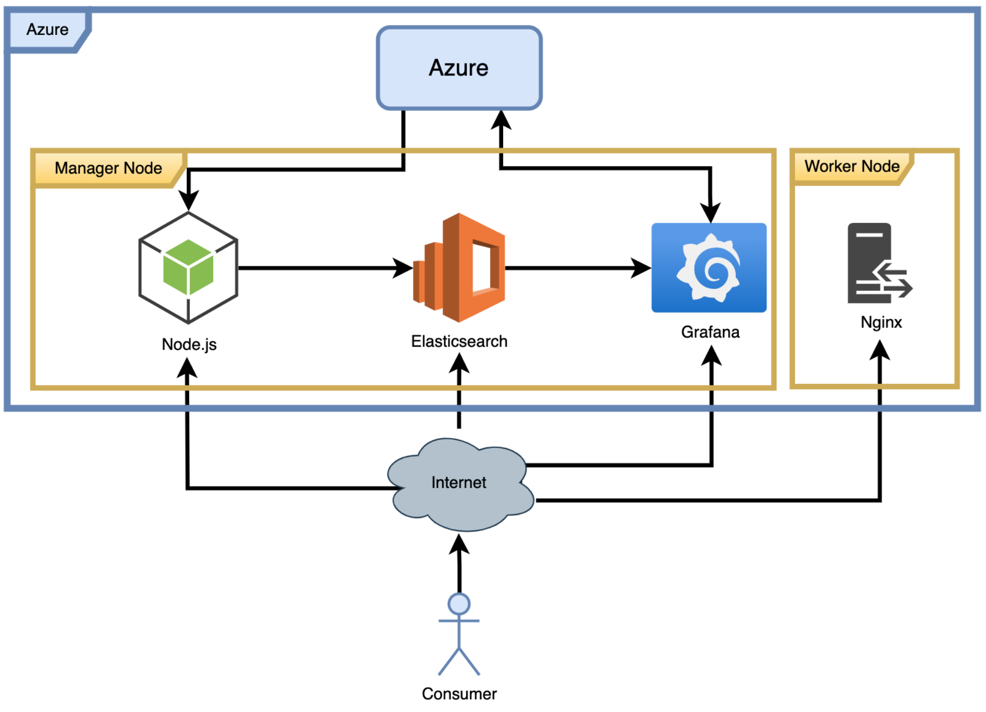


Figure 4.2: Traffic flow

- The first thing that happens when the application is deployed is that the official Elasticsearch image gets bulled from the internet, and the Elasticsearch container gets created from it. Elasticsearch is the first image to be pulled as it has no other services depending on it.
- Then the self made Node.js image is pulled. This image exist in the Azure Container Registry in our Azure environment. The node js app only depends on Elasticsearch to start before it. This is done so that node js app has something to send data to later on.
- Grafana also gets its image from Azure Container Registry. Azures Service Principals are utilized to grant pull access for the manager VM, making it able to pull the images from the Registry.
- The fourth thing to happen is that the official Nginx image gets pulled from the internet. After this, all the microservices should be up and running if there are no unexpected error.
- After all the microservices are deployed, the Node js service contacts Azure to fetch all relevant groups. as well as the securely stored API credentials used in the next step.
- Then it gets API credentials securely stored in Azure Key Vault.
- The Node js app start to fetch information about devices, from the three security vendors. One API call is used per customer to retrieve data.
- After the proper information is collected and process internally in the Node js app. It sends it to Elasticsearch container to be stored and indexed.
- Grafana gets configured based on groups it fetches from Azure.
- In step tenth Grafana pulls information from Elasticsearch for it to be presented for the user.
- After this is done, Grafana stores the password of the Admin user in Azure Key Vault. Since the admin user is dedicated as a Break glass account and not active.
- Over to the user login part. In step twelve, the user access the page through our domain "Securityportal.tikt.no". He/she gets routed to our Nginx reverse proxy.
- It then gets redirected to the Grafana container where they are met with an in login page.
- Regardless if a user is a normal user or an admin, an MFA policy is added to the Single Sign On (SSO) to enhance security. MFA can prevent 99.9‰of attacks to your account and is why this policy is forced for everyone [34].
- After a successful login the users gets redirected back through the Nginx container.
- And than finally in step 16 they get redirected Grafana where the users can view its dedicated information.

The microservices is chosen for their specialised capabilities and efficiency. We leverage these technologies as they are widely adopted and highly relevant in today's tech market. The technologies offers great flexibility as they adapt well with a variety of other real world tools [35].

Instead of building every thing from scratch, do we utilising existing tools to optimizes our time and resource to develop an enhanced a product that both we and Tussa benefits greatly from. By harnessing the interplay of these microservices accurately, do we accomplish a more efficient solution in a shorter time frame. The decision for each microservice is further described later in this chapter.

4.2 Azure

Choosing Azure came down to an easy decision as Tussa was already using the cloud service provider. Both them and us have some prior knowledge about its capabilities, how it works, and the cost that comes with it. Using Azure provide us with a variety of services, which benefited our solution greatly as our whole application is launched in Azure using their VMs. It provides a secure way of storing our images and credentials, as well as giving the flexibility to handle Tussa's customers in separate groups in collective place securely. Furthermore, Azure enables us to manage access control for different services and users, to obtain the principal of least privileged.

4.3 API

We decided to create an Node.js application from the ground up to ensure full control of the traffic when generating API calls. It also allows us to control the security of the API credentials as well as how to structure the data before passing it onto Elasticsearch. Having a self configured Node.js applications gives us a unique capability to construct information based on groups fetched from Azure. This makes up the backbone of the structured data in our infrastructure and was why we chose to do it this way.

Developing a custom solution gives us a fine grained control for the task mentioned above. It also builds upon the principal of high flexibility and ease of integration with other microservices. Having a self built Node.js app is also cost-effective as it is free to use, compared to other options. An alternative to this is to utilize third-party services like Azure, AWS or Google. Here Azure would be the natural option since Tussa is already using the platform. Different Azure services can be used to managing traffic, structuring data, and securing API credentials.

As mentioned, using Azure comes at a higher cost as it is a paid services. It can also posses some personalized limitations. This could be difficult to manage and control an overview, since they provide a wide rage of services. Also choosing Azure as a third party comes with a restrain for future work, where you lock yourself to one vendor and might have to do substantial changes to change this. With this stated, we do recognize the challenges with a self configured Node.js application. It require a enhanced development knowledge and more maintenance. Nevertheless, we do consider the effects of the positive benefits to outweigh these cons.

4.4 Elasticsearch

There are several reason to why we chose to use Elasticsearch, and it comes with several benefits that suits our solution. It offers a free open-source version which is great for a student project. Additionally, generating less negotiates with the customer regarding the budget. Elasticsearch is designed to handle large amounts of data and do quick search queries. It also offers data to be accessed in real time which benefits us greatly as Grafana update its information regularly by querying Elasticsearch. It is a highly flexible tool which offers integration with a variety of other tools such as Beats and Logstash [22].

A competitor to Elasticsearch is Apache Solr which is another search platform that offers similar capabilities. We chose Elasticsearch over Solr due to the ease of nature that Elasticsearch comes with. It uses a RESTful API that supports JSON which integrates well with our Node.js app, while Solr primarily uses XML. Since both Magnus and Øivind where familiar with Elasticsearch from their work place, it was also easier to choose ES. Solr has slightly longer indexing time, making Elasticsearch more suitable for real time applications [36].

4.5 Grafana

Elastic is the company that developed and create Elasticsearch. Even if Elastic has their own visualization tool called Kibana, we chose to go with Grafana for visualizing the data. This decision is mainly based on the request from Tussa, as they wanted to use Grafana since its a program they already use and are familiar with. We agreed to this decision, because Grafana is a powerful free open-source analytic and visualization tool. It is highly functional with management and segregation of organisations, teams, users, data sources, and dashboards. Furthermore, Grafana offers great documentation which makes it easier to configure it to meet the wanted visualisation goals. Grafana is also particularly relevant as it is a real world industry tool.

4.6 Nginx

Nginx is a powerful software that offers an array of services which is widely used and highly appreciated on the real world marked. We therefore chose to use this service, and implement it as one of our Microservices. The specific features that truly captivated our interest were, reverse proxy, DDoS protection, HTTPS termination, Firewall, Load balancer, and API gateway. Harnessing the impeccable advantages these elements present, improves the performance of our service. Furthermore, they fortify a superior solution to provide Tussa, by ensuring enhanced security, reliability and efficiency [37].

To have all these attributes in one place was a huge factor for choosing Nginx. It allowed us to have a centralised place for configurations. Nginx provide a free open-source version which also made it easier to decide to use it. It should be mentioned that the free version comes with some limitation, which is addressed later on. Nginx is a great tool for us since it is designed to handle a high volume of requests simultaneously.

An alternative to Nginx is Apache HTTP Server which is also a widely used tool. Nevertheless, Nginx is know for their high performance of a large amount of requests, due to its event driven architecture. Compared to Apache using a process-based, which may devour more resources [38]. Another factor that persuade our decision is the easy and understandable syntax that Nginx offers, as non of us had relations to it before starting to work with it.

4.7 Architecture Alternatives

When we knew what technologies to use. The next step was to figure out how these would be utilized, to best suit Tussas requirements. We discussed multiple ways to go when building the infrastructure of the application. There are pros and cons with every solution, but it all narrows down to the customer's needs and demands. Some things to take into consideration is the cost of each service and the security they provide. Another important aspect is the scalability of the application. Tussa is a growing firm that aims to get more customers in the future. Therefore, it's important to keep in mind the possibility of expanding the application to even more customers.

For the three alternative infrastructures presented below, we have done a cost estimate. It is important to notice that these are just estimates, and the actual cost depend on each services requirements when launched in a production environment with real data. We have used public information to find system requirements for each Microservice, and then used Azure to find what does requirements cost. This is further described later in this chapter.

4.7.1 Silo

The first option regarding infrastructure is a siloed infrastructure. A silo is defined as an isolated point where some data is stored separated from the rest of the infrastructure[39]. We have made a visual picture of the siloed infrastructure seen below. This infrastructure is built so that every customer has their own virtual machine with all the microservices except Nginx. The microservices being API container, Elasticsearch container and Grafana container.
The siloed infrastructure is heavy resource dependent, and do not optimize cost in an efficient way considering to build the infrastructure. The cost is greatly increased when the number of customers grow.
On the other hand, it's a secure way to isolate every customer to have their vital information stored separately from other customers. It's also a more dependent way to handle a lot of traffic since its separate machines for every customer.

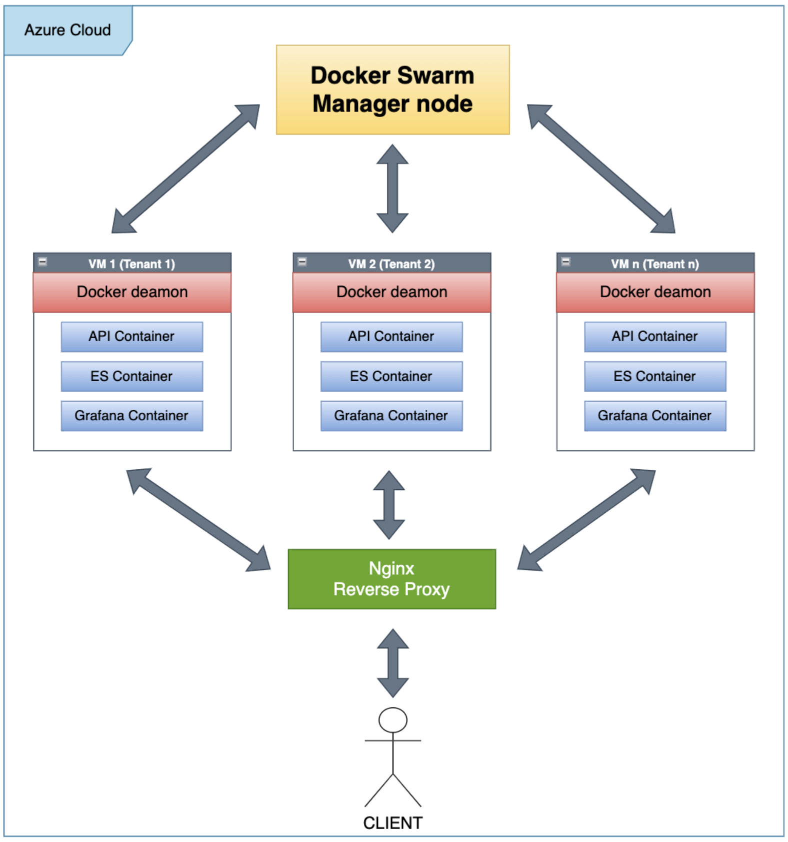


Figure 4.3: Silo infrastructure

4.7.2 Pooled With Silo

This infrastructure is built in a way so that every microservice has its own virtual machine. Each of these microservices are the same as mentioned before, API container, Elasticsearch container, Grafana container and Nginx container. This means that each of the resources are isolated, but unlike the Silo infrastructure, here all the customers share the resources on each VM. But even though they share VM's, each customer has their own container inside each machine. This also makes for easier management and improved scalability. It's also more cost efficient.
We have made a visualization of the infrastructure below, showing how all the customers share VM's. While this infrastructure is more cost efficient and has improved scalability, it needs powerful VM's that can handle a lot of traffic to work.

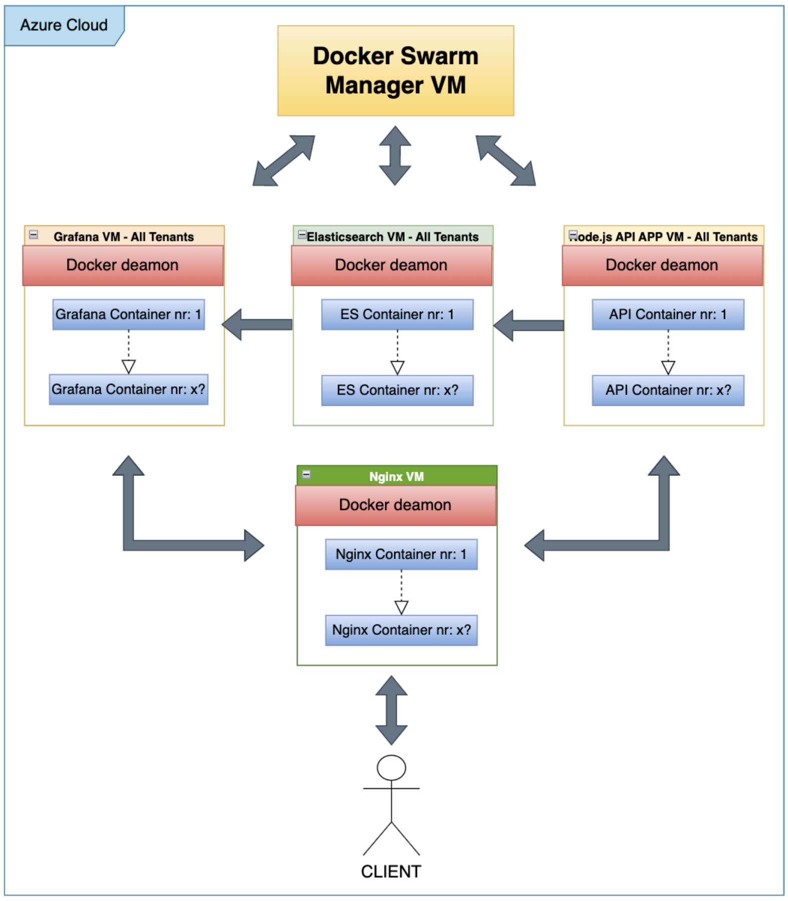


Figure 4.4: Pooled with Silo

4.7.3 Pulled and Partitioned with Separate microservices

This infrastructure is built in a way so that every microservice has its own virtual machine. Each of these microservices are the same as mentioned before, API container, Elasticsearch container, Grafana container and Nginx container. This means that each of the resources are isolated, but unlike the Silo infrastructure, here all the customers share the resources on each VM. The containers inside the VM's are scalable, meaning more containers can be made, if necessary, but all the customers share the containers.
Below you can see a visualization of the infrastructure and how its built. This is a more cost-efficient infrastructure, but it needs more powerful VM's to be able to handle the traffic.

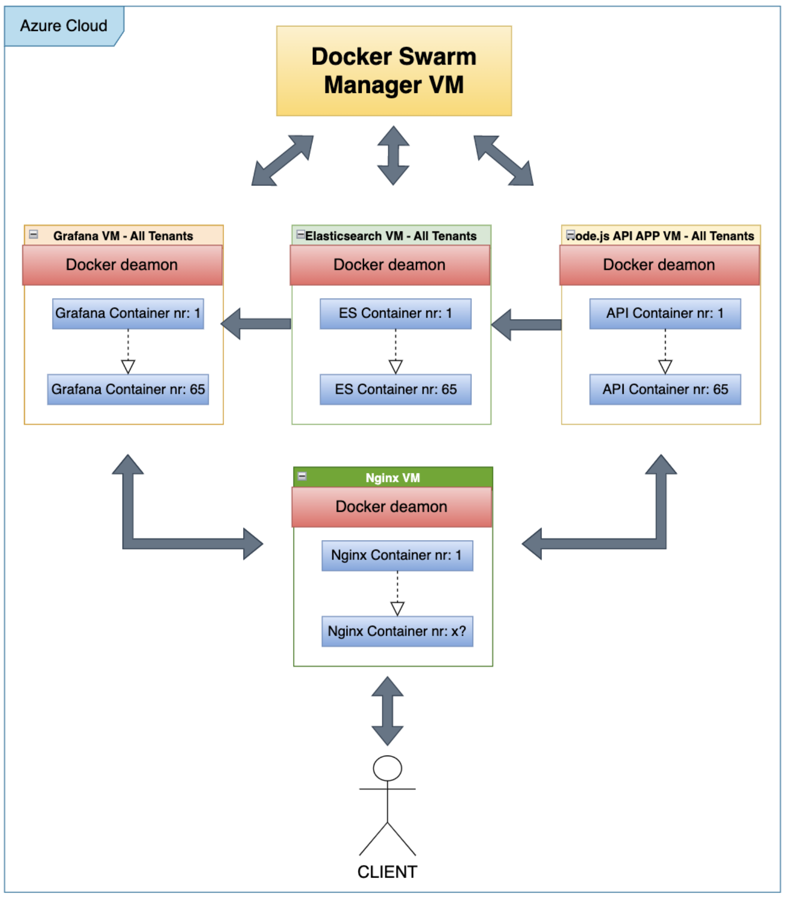


Figure 4.5: polled and partitioned with separate microservices infrastructure

4.7.4 Cost calculation

This section explains how the estimate pricing was calculated. We found the minimum system requirements using open-sources, and then matched it to the pricing table of Azure [40].

Node js is set to have these basic requirements to run in a Ubuntu environment: [41]

- 4 core CPU
- 4 GB RAM

Elasticsearch has the following system requirements: [42]

- Multi core CPU
- 32 GB RAM

Grafana's requirements is as stated: [43]

- 1 core CPU
- 255 MB RAM

Nginx has these system requirements: [44]

- Dual core CPU
- 2 GB RAM

A manager node and a worker node in a docker swarm as the following requirements: [45]

- 4 core CPU
- 8 GB RAM

A worker node in a docker swarm as the following requirements: [45]

- 4 core CPU
- 4 GB RAM

4.8 Pricing Of Each Infrastructure

Table 4.1: Table showing the different prices

Infrastructure	**Monthly cost**	**Yearly cost**
Silo	NOK 103 286	NOK 1 239 000
Pooled with Silo	NOK 8 488	NOK 101 856
Pooled and Partitioned with Separate microservices	NOK 4 588	NOK 55 056
Pooled and Partitioned with Shared microservices	NOK 2 200	NOK 26 400

4.9 Infrastructure Decision

When making a choice of which infrastructure to use there are multiple things to consider. For starters there is the cost of each of the infrastructures. As mentioned before, the numbers above are just estimates, and can vary. On the other hand, there is the security and redundancy. Some of the infrastructures demand a lot of resources, which means a higher cost, but on the other hand, these infrastructures are more reliable and possibly more secure.

These are ultimately things that the client needs to take into consideration and decide on. We presented them with the different infrastructures, as well as the pros and cons. Ultimately Tussa wanted to go with the most cost-efficient structure, the Pooled and portioned with shared microservices. If built correctly, this is the best choice when you take into consideration the importance of uptime and the extreme cost of some of the infrastructures, but this comes of course at the expense of the benefits that isolation provides and what security concerns it can posses.

Other security features were presented to make up for the lack of isolation which is described later. Tussa also has the option to do horizontal scaling if they get more customers that demands more traffic. Horizontal scaling can for example be another virtual machine with the same attributes or a replicated container, and then divide the customers upon available resources.

Chapter 5

Development Process

In this chapter are we going to to describe our development process and the methods we used while working on both our application and for writing the bachelor thesis. In the Project Plan, in Appendix B, have we described how we wanted to organize our work and what development model we wanted to use. This chapter builds further on that plan. We also want to describe our working routines and how we documented our work through the project.

5.1 Development Model

When we started this project we discussed back and forth which development model would suit us best. This can be challenging, especially considering lack of prior experience working with the magnitude of this project. We wanted a development model that would allow us to work with one aspect of the project at a time, preferably in sprints. As mentioned in the Project Plan, which can be found in Appendix B, we went with the Scrum model, and there were several reasons why we chose this model.

The Scrum framework offers flexibility and excellent overview of each task at all times, making it a perfect fit for our project. The process starts off by listing all tasks which are going to arise throughout the project. These tasks are then split into different sprints, depending on which order they need to be finished. The framework is divided into sprints, where we are able to focus on specific tasks. We made five sprints and started implementing tasks we knew we had to do. We also made iterations after every sprint. We did this iterations at our internal meetings, where we discussed what we were happy with and what needed to be improved for the next sprint. These iterations can be read in Appendix K.

What made the Scrum framework a great tool for us was the availability to have control of which tasks were being worked on, which were done and which needed to be started on. This gave us a great overview of the entire project at all times. The system with sprints also worked great, because we had the opportunity to improve our self after each sprint was finished. Scrum also offered us flexible roles, which means we were not limited into a fixed role for the entire project. There was one point that was quite challenging, and that was to be aware of every task we needed when we started a new sprint. Often a new challenge would present itself, and we solved it by simply adding it as a new task, either in the next sprint or the current sprint.

When developing the proof of concept, we used a software development methodology called Rapid Prototyping. This is an agile strategy used when developing a product. The strategy builds on the concept of rapidly making prototypes of the product to test and validate, as well as getting user feedback to improve the next stage. This strategy amplifies an efficient workflow due to its nature, and reveal if chosen build work or not. Ultimately saving yourself from a lot of extra work [46].

When we used this in the development, we ended up with four different stages. After the first stage we generated an extensive document. This stage was presented to our client, to prove the proof of concept without the security measures. This document can be seen in Appendix J.

- Stage 1: Our solution was made on a local environment and not as a cloud solution.
- Stage 2: Deployed in the cloud on Azure virtual machines as a Docker Swarm.
- Stage 3: Implementation Nginx in front of Grafana with its respective security measurements.
- Stage 4: Extensive Azure integration, such as Authentication and group management.

5.2 Routines

5.2.1 Policies

In the Project Plan we set some expectations to each of the team members. We wanted to have some clear policies on how we could conduct our work to optimize efficiency and results. We conducted these ground rules together as fundamental agreementfor future work. These ground rules can be seen in the Project Plan in Appendix B. Our most important points were that every member should conduct a minimum of 30 hours workload per week. Another important point we set was to divide the workload fairly to ensure everyone's contributions.

5.2.2 Communication

The communication within the group have gone according to plan even though we had to add some improvements, acknowledged trough iteration reviews. As Magnus and Øivind have jobs that require night-shifts we did implement a job-log where we could get a better understanding of who worked on what. We were in a unique situation because the three of us were good friends even before we started the project. This has benefited us greatly because we know each other strengths and weaknesses, and it was very easy for us to communicate. Since all three of us values working at home where we have multiple screens to work with, a lot of our communication happened with Discord. Regarding communication with the supervisor and Tussa, we used mostly Microsoft Teams or regular email.

5.3 Meetings

In this section we want to describe how our meetings with both the supervisor, our contact person i Tussa and the internal meetings were organized. The meetings were usually no longer than 30 minutes. Sometimes they were shorter and sometimes longer, depending on the topic of discussion. Every meeting we had with both the supervisor and Tussa can be found in the meeting minutes in the Appendix A. The purpose of the meeting minutes is to act as a summary for each of the meetings, and to make it easier for us to go back and see what was discussed and what feedback we got.

5.3.1 Meetings With Tussa

We had weekly meetings with our contact person from Tussa. This meeting was scheduled every Monday at 9:30am. In this meeting we asked questions regarding their wishes on the application, and gave a short status report regarding the process. This meetings were a great tool to be able to adjust the application to Tussa's wishes. In addition to these meetings we had a couple of meetings with additional people in Tussa to get their point of view on the project.

5.3.2 Meetings With Supervisor

Our meetings with our supervisor was weekly every Monday from 1pm to 1:30pm. Here we would ask eventual questions and give the supervisor a status report on the project process. This meeting were some times rescheduled, and some times cancelled, if we did not have any questions or if either the group or the supervisor was unavailable.

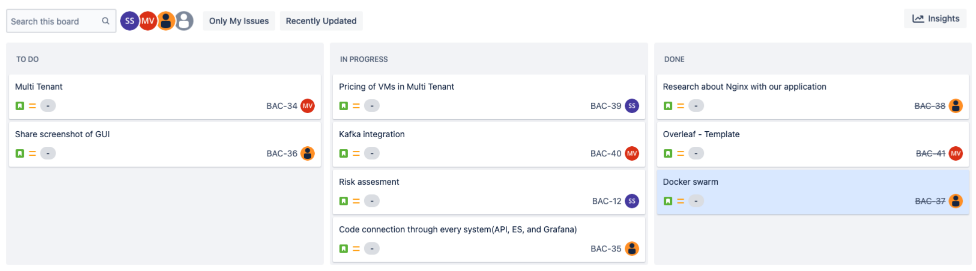
5.3.3 Internal Meetings

The bachelor group had a weekly internal meeting every Friday at 1pm. In this meeting we made plans for the upcoming Monday meetings and also had a evaluation of the week. The internal meetings on Fridays is not the only time during the week the bachelor group met to discuss or work on the project, but this was the formal weekly meeting.

5.4 Documentation

5.4.1 Jira

We used Jira to create a board to keep track of every task that each of the members worked on at what time. Since we used Scrum, the Jira software made it easy for us to make boards to divide the tasks and keep track of tasks that needed to get our attention. Jira is a software built by Atlassian to make seamless project plans. We started by dividing the project into different sprints, giving each of them a description and length. We then started sprint number 1 and began creating tasks. When creating a new task it were automatically placed in the "To do" category. You can then assign it to a team member and move it to the "In progress" category and later to the "Done" category. The Jira board gave us a good overview of everything we needed to work on, and what the other members were working on at any given time. We attached a picture below, Figure 5.1, that shows the tasks in one of the sprints to give a visual picture [47].



TO DO

Multi Tenant — BAC-34 MV

Share screenshot of GUI — BAC-36

IN PROGRESS

Pricing of VMs in Multi Tenant — BAC-39 SS

Kafka integration — BAC-40 MV

Risk assesment — BAC-12 SS

Code connection through every system(API, ES, and Grafana) — BAC-35

DONE

Research about Nginx with our application — ~~BAC-38~~

Overleaf - Template — ~~BAC-41~~ MV

Docker swarm — ~~BAC-37~~

Figure 5.1: The picture shows how the different tasks can be labeled and divided.

5.4.2 Thesis Writing

When writing the thesis we used a software called Overleaf to write in LaTeX. Overleaf gives us the opportunity to write the thesis collaboratively. Overleaf is great for exactly this, because everyone is writing on the correct version of the thesis at all times. Overleaf also ensures the typesetting and formatting is the same through the entire document.
We used the template provided by NTNU, which was created by Community of Practice for Computer Science Education (CoPCSE) [48].

5.4.3 Writing Code

Developing our solution meant we had to utilize a considerable amout of scripts and code to make everything work. For this, we used GitHub. Here, we made a private repository which only the team members had access to, to ensure confidentiality. When developing code we used our private computers, despite some possible security risks. A further consideration supporting this can be found in Appendix H, which is a risk assessment regarding using private computers. While using GitHub, we developed one stage and then pushed the code to our GitHub repository [49].

5.4.4 Time Tracking

We wanted to track all the time used on the project for each of the team members. In the Project Plan we agreed that everyone were going to track their hours worked at the end of each week. For this we just used a simple excel sheet, which can be viewed in Appendix C.

5.4.5 Other Documentation

We also did some other documentation when working on the project. Documents such as status reports, notes, drafts and other documentation were stored in a shared Onedrive folder. This folder was only accessible by the members of the team. Regardless, we did not persist any sensitive information there due to security reasons.

Chapter 6

Implementation

This chapter aims to give a deeper technical understanding of the solutions by magnifying details regarding our implementations and underlying decisions. It introduces repository utilise to persist code as we as overall technical aspects. It emphasize Tussa's guidelines, and how we have strived to integrate them in our practise. Additionally It is address techniques utilise to implement our microservices, except Elasticsearch since we have not done any specific configuration to it. Furthermore, Our script is presented and their use case. Finally, explaining implementation performed in Azure and their utility.

6.1 Repository

The code share a unified repository for all the microservices. The code also share the same branches to increase simplicity instead of having code for each microservice located separately. Scripts and stack files can also be found here. GitHub is enabled with MFA, which a user needs to pass to access the code. The repository is managed and controlled by the own, and is the one delegating access and permissions. This person has an extensive responsibility to managed it correctly to obtain integrity and confidentiality, so no unauthorised personnel can gain access. Pushing code to git is a relative basic knowledge within the IT community, and we are only going to provide a link with a thorough explanation on how this is done: [50]. It should be mentioned that a temporary repository has been made in GitHub for censorship and can be found at [49].

6.2 Technical

We utilised JavaScript as our main programming language. Our api-cron microservice is entirely built using JavaScript. The self configured part of Grafana is also accomplished with using JavaScript. They both use Node.js as their run time environment.

As mentioned in the background chapter, Node.js is built on the V8 JavaScript engine, which makes it high-performance and a great tool for handling a larger number of connections efficiently, and makes it very scalable. Node.js also enables full-stack development, meaning it allows the use of JavaScript both on server-side and client-side development. This allows developers to share code between front-end and back-end, which can result in increased productivity and code reuse [24]. JavaScript is also a great programming language for us to use because of its versatility, meaning it can be used for both front-end and back-end. It's also one of the most used languages out there, making it easy to find information and tips about it on the internet for developers [25].

Nevertheless, our adviser initially recommended us to program in Python as this is another programming language which provides a great frameworks with several benefits. It offers a lot of the same capabilities to achieve our goal as JavaScript. Since each member had little to non experience in Python compared to JavaScript, JavaScript came out on top as the favorite one when choosing which program language to go for. JavaScript is also more similar to other languages like C++ and C which we were familiar with from previous unites, this also helped tipping the scale.

NPM is a great tool because of the large package registry. It provides access to a vast collection of open-source libraries. NPM also makes it easy and convenient to install libraries and manage their versions. It's also seamless to use with Node.js because of it's tight integration, making it efficient to use together [26].

We utilised several Libraries through NPM. Our chosen libraries brought extensive functionalities to our code. We ended up using a total of six libraries that our JavaScript code depends on to function correctly. Each library has their own unique functionality. We utilised the `npm` command line client to install them. These are the following Dependencies:

- **@azure/identity:** Is used to authenticate with Azure services [51].
- **@azure/keyvault-secrets:** Allows for interaction with Azure key vault to store and retrieve secrets [52].
- **@elastic/elasticsearch:** Is used to interact with Elasticsearch, and is used to index information [53].
- **axios:** Is used to handle interaction request for APIs and function as a promise-based HTTP client [54].
- **cron:** Enables scheduled task to be executed at a certain time [55].
- **qs:** Provides utility to format and parse complex nested queries into objects and vice versa [56].

All these libraries play their own specific part and are crucial for our application to function properly, as this is described below.

6.3 Tussa's Guidelines

Tussa provided us with multiple guidelines in the beginning of this project which is located in the Appendix I. These guidelines strive to give an understanding of what Tussa expect regarding different protocols and procedures. The guidelines addressed below explains Tussas mindset towards system development.

Code commenting is stated in Tussa's guidelines regarding DevOps as an essential requirement. Its compulsory with sufficient details to enhance the clarity of whats happening within the code and the intention behind the particular implementation. This procedure aid elicit the meaning about each piece of code and its purpose. Additionally it is an adequate practise to include metadata in the code. It is important to always address the date of code implementation and refer to the author. This practise extends clarification and emphasize development history tracking. We politely refer you to our GitHub repository for a better comprehension on how this is achieved [49].

In addition, the guideline addresses to strictly forbid storing password and API credentials as clear text within the code. Following this security measurement is an essential way of maintaining best practise. Detailed knowledge on how we have achieved this important security safeguard is stated in the next sections. Finally, it is important to highlight that these requirements are universal, regardless of whether developments is done by in-house teams or by outside contractors. We have dedicated us to the best of our ability to comply with these guidelines.

6.4 Self Configured Node.js App

As mentioned did we completely developed our Node.js application from scratch. For future references do we also mention this app by its given service name: api-cron. Its first functionality is to retrieve information about devices from two different vendors using three separate API calls. Its main purpose is to accumulate this information into one data set. This is achieved through multiple functions which is addressed further down. The composed data structure is so forth passed into Elasticsearch, where it is stored for future use.

6.4.1 Azure API

The microservice is configured with system assigned managed identity in Azure, which removes the need of any extra credentials to access Azure resources. This means that access to retrieve and store secrets is configured in Azure as policies instead of generating an additional API to do the same.

6.4.2 Device APIs

The APIs fetching information from three different products supplied by two vendors. Early in the projected presented Tussa us with a method of using a communications service providers (CSP) in the future to enable shared API credentials per product. Shared API credentials authorize information retrieved from all the tenants together by only using one call. They stated this because it is so much easier to managed, instead of having separate API credentials and API calls for each tenant. Azure key vault is configured to simulate this concept. It consist of 3 Secrets holding the credentials to each product for each tenant as a string.

It should be addressed that sharing API credentials across multiple tenants introduces a new security risk and is not seen as best practise. It can potentially increase the risk of a data breach as a single set of credentials can expose sensitive information for all customers, not just one. It can also be more challenging to monitor the use of each one and to ensure compliance of policies and regulations [57]. From a strict security perspective it is considered to be a bad idea, but because of the specific use case scenario, Tussa does accept these risks. The factors accepting this decision is presented below.

The three APIs is configured with least privilege. Meaning they can only read information necessary and nothing else. This gives a potential adversary limited access in case of a breach where the credentials are compromised. Cisco's APIs were pre-configured by Tussa before our projected started, while Microsoft's Graph API giving access to Intune is configured by us. Microsoft provide great documentation on which permissions is needed to delegate their APIs to contact right endpoints. We are contacting the endpoint `List devices` to retrieve information about all the devices. The necessary permissions to access this endpoint is `Device.Read.All` and `Directory.Read.All` [58].

The Secrets stored in Azure follow the least privilege principal, and authorize only people with right access. By default, each of the APIs is configured using SSL/TLS over HTTPS, which adds on another security layer. The code also poses proper error handling to maintain appropriate logging.
Tussa has around 60 different customers, so there would potentially be 60*3=180 APIs to handle. Managing a significant amount of APIs can be prone to errors, and poses a security risk on its own. The complexity of managing is only get more challenging as Tussa gain customers, as well as the likelihood of human error.

Furthermore, proper resource delegation to managing this, be needed, which can potentially take away time from other important tasks. It could lead to a inefficient workflow. It could also increase the attack surface as more APIs are exposed to the world. Finally, since the APIs are located on the same device do a potential breach of the VM, simultaneous is expose, regardless the approach.

6.4.3 Pseudo code for retiving device information

This Pseudo Code aims to explain what the code in the Node js application does. For more reference go to our GitHub repository where the source code is located.

Algorithm 1 Retrieve device information

```
procedure PROCESSAPIS
    groups ← getIntuneGroups()
    API credentials ← getKeyVault()
    for each group in groups do
        Fetch devices from APIs
        Format data into objects
        Merge objects into a single array
        Compar last sync data field from each vendor
        Adds result found
        Adds current date to the array as timefield
        Displays the hole array in the console
        procedure CREATEORUPDATEINDEX(array, groupName)
            Creats elasticsearch client
            Check if it was successful
            Deletes old index
            Create new index (groupName) with the new array data
        end procedure
    end for
    Handle any errors
    Run process immediately
    Schedule process to run daily at 21:00 Oslo time
end procedure
```

6.4.4 Dockerfile

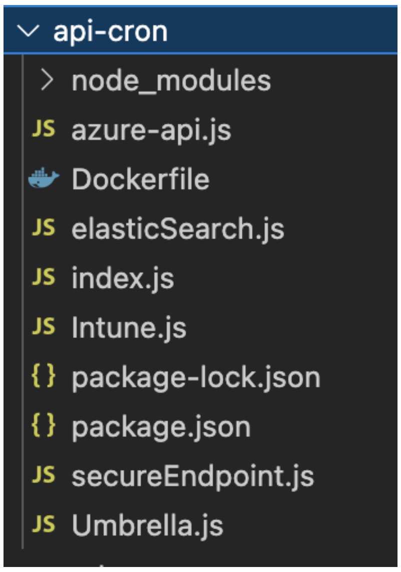
Our docker file contains the necessary commands to assemble our api-cron image. It starts off by fetching the latest node image as its baseline structure. Next is setting the work directory naming it /app. Next copies all the page files into working directory. Then installs all necessary dependencies specified in the page file. Later does it copy all the code into the working directory. Then it creates an environment variable which tells node where to look for all the node modules. Finally its states which command to run when the container is running, and says to wait 10 seconds before running the application. It waits 10 seconds to make sure the container is running properly before running the script, as a safety measure for errors. See Code listing 6.1 below for the Docker file.

Code listing 6.1: Building Docker Image from Node.js

```
FROM node:latest

# Set working directory
WORKDIR /app

# Copy package.json and package-lock.json files into the container
COPY package*.json ./

# Install dependencies
RUN npm install

# Copy the rest of the application code
COPY . .

# Set the environment variable
ENV NODE_PATH=/app/node_modules

# Start the application with a delay
CMD [ "bash", "-c", "sleep␣10␣&&␣node␣index.js" ]
```

6.4.5 File Structure

This is the file structure of the microservice. It is divided into separate files for different purpose to create a better overview.



```
api-cron
  node_modules
  azure-api.js
  Dockerfile
  elasticSearch.js
  index.js
  Intune.js
  package-lock.json
  package.json
  secureEndpoint.js
  Umbrella.js
```

Figure 6.1: api-cron microservice file structure

6.5 Grafana

Grafana utilise a built in Node.js app in its image, to configure it. This scrip run and configure Grafana each time the container is launched or restarted. This means that Grafana gets configured each time the container launches. It uses Grafana's API to configure Users, Teams, data sources, Dashboards, and permissions. The APIs uses basic authentication to authenticate towards Grafana.

6.5.1 Azure AD Authentication

Grafana is integrated with Azure AD authentication enabling users to access Grafana through their Azure account. It allows only users that are part of certain groups in Azure to access it. Synchronising Azure roles is disabled as all permissions is configured using the APIs, except for the admin user. It is configured to allow admin users specified in Azure to be admins in Grafana as well. All these properties are stated in the stack file as environment variables to the Grafana service. An app registration is needed in Azure to enable this which is further be addressed below in the section about Azure.

6.5.2 Break glass

The default admin user in Grafana is configured as a Break Glass account and is only intended to utilised in case of emergencies. The admin account is not supposed be used as a normal account since everyone has their own specific Azure user. Its purpose is to prevent a complete lockout from Grafana in case something is wrong with Azure. It creates a safety measure to always have an alternative way of accessing Grafana for specific users. This was configured as it was one of the requests from Tussa. It provides Tussa with the flexibility and backup to access Grafana in case of an emergency. The code generates a random password consisting of 25 characters, being lower and upper case, as well as including symbols. The password is stored in Azure Key Vault after the admin account is updated with it. An access policy is forced in the Azure Key Vault only letting members of the group `test-admin-GF` have full access to it. An access policy is also created for the application which sets the permission to "set". This mean it can only set a secret in that specific key vault and nothing else.

6.5.3 Pseudo Code For Configuring Grafana

This Pseudo Code aims to explain what the code in the Grafana application does. For more reference visit our GitHub repository where the source code is located.

Algorithm 2 Configuring Grafana

```
procedure ARROW FUNCTION
    groups ← getIntuneGroups()
    for each group in groups do
        Creat a team in Grafana
        Get all group members
        for each member in groupsMembers do
            create user in Grafana
            Add user to respective team
        end for
        Create data source in Grafana
        Create dashboard in Grafana
        Create dashboard permissions in Grafana
    end for
    procedure CHANGEADMINPWD
        Generates a password consting of 25 random character
        Updates the Grafana admin user with it
        Store the newley created password in Azure Key Vault
    end procedure
end procedure
```

6.5.4 Docker file

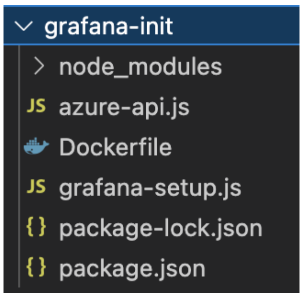
Docker uses Grafana image version 9.4.2 as the baseline of the image. It uses a specific version through out the development face to achieve predictability and consistency. It also offers stability because we know that the image works with our code. Although, it should be mentioned that not utilising the latest version can come as a security risk. Newer version commonly mitigates security vulnerabilities that the prior version poses, as experienced in the chapter regarding security testing. The Docker file sets the working directory as /app, and then copies all the necessary files to it. It installs Node.js and NPM for it to work and runs the code with a 10 second delay. It then makes sure to continue running the script.

Code listing 6.2: Building Docker image for Grafana with additional functionality

```
FROM grafana/grafana:9.4.2

#Set work directory
WORKDIR /app

# Copy the package.json and package-lock.json files into the
    container
COPY package*.json ./

# Copy the node_modules folder into the container
COPY node_modules ./node_modules

# Copy the azure-api.js into the  container
COPY azure-api.js ./azure-api.js

# Copy the grafana-setup.js file into the container
COPY grafana-setup.js ./grafana-setup.js

# Install Node.js and npm
USER root
RUN apk add --update nodejs npm

# Set the entrypoint to run the script
ENTRYPOINT ["sh", "-c", "/run.sh␣&␣sleep␣10␣&&␣node␣
    /app/grafana-setup.js␣&&␣tail␣-f␣/dev/null"]
```

6.5.5 File Structure

Below is Figure 6.2 illustrating the file structure necessary to create our functioning Grafana image. Main code is located in the `grafana-setup.js`, the code connecting to Azure is located in `azure-api-js`.



```
grafana-init
  node_modules
  azure-api.js
  Dockerfile
  grafana-setup.js
  package-lock.json
  package.json
```

Figure 6.2: Grafana microservice file structure

6.6 Nginx

Our Nginx service is configured with two files `nginx.conf` and `default.conf`. `nginx.conf` is the main configuration file for Nginx. It specifies global configurations which applies for everything in Nginx. `default.conf` is used to configure the service side. It is configured as a reverse proxy with multiple security measurements.

6.6.1 Nginx.conf

Here its configured with basic configuration for Nginx to work, as well as specifying some DDOS protection. It is configured with a rate limit where its stated that each client can make up to 10 requests per second, if a client exceeds this limit, a 503 error (Service Unavailable) is issued in return. It also specifies a limit of how many simultaneous connection a single IP can make to the web side, here the limit is 10. It also removes information for the header about which Nginx version that are in place.

6.6.2 Default.conf

The default.conf files states that all HTTP (port 80) traffic shall be redirected to HTTPS (443) as the site only is available through HTTPS. The application achieve HTTPS through SSL certificates from OpenSSL and is valid for a total of 90 days. It is configured to prohibit any traffic towards the /metrics path. This is done because it possessed a security flaw which is further address in the chapter about security testing. It also possesses a IP access list stating it only allows Tussas IP and denies everything else. It disables the browser for trying to guess the MIMI type. It also prevents the page from being framed within another page, mitigating against clickjacking attacks. It specifies a Content Security Policiy, which helps prevent against vulnerabilities like injection. The rate and access limit stated in the `nginx.conf` is applied to the path of the web site.

SSL termination is applied and therefor decrypt the SSL/TLS traffic before the reverse proxy passes it on to Grafana. We have decided to do so because of a few reasons. It makes managing the SSL certificate simpler as it only needs to be done on the reverse proxy. It also decreases the complexity as multiple nodes may be added in the future for enhanced scalability and robustness. The traffic running between Nginx and Grafana is on a private docker overlay network which is generally considered as secure.

The overlay network is configured with IPsec/ESP encryption, which encrypts all network traffic between nodes at the network lay in OSI-model. We have enabled encryption on the docker overlay network to enhance security and integrity, but also to simplify SSL certificate management, as this needs to be done manually as of this moment. However, this encryption does not apply for communication between containers on the same VM, as this is consider to be private traffic, and needs to be handled accordingly. Its also worthy to highlight that it does not provide end to end encryption as the data is decrypted and re-encrypted on the Nginx host.

6.7 Scripts

This section gives a deeper explanation of our two script we made through the use of pseudo code.

The following pseudo code describes how newly created SSL Certificates are saved as Docker Secrets.

Algorithm 3 Update SSL Certificates

```
1: procedure RemoveSecrets
2:     if Docker secrets exist then
3:         Remove existing Docker secret (Public)
4:     end if
5:     if Docker secrets exist then
6:         Remove existing Docker secret (Private)
7:     end if
8: end procedure
9: procedure CreateSecrets
10:     Create new public key
11:     Create new private key
12: end procedure
13: procedure Redeploy
14:     Redeploy the stack
15: end procedure
```

The following pseudo code describes how a new Service principal is issued, and how its credentials is stored as a docker secret for future use.

Algorithm 4 Update Service Principal

```
 1: procedure INSTALL AZURE CLI
 2:     if Azure CLI already installed then
 3:         Azure CLI already exists
 4:     else
 5:         Install Azure CLI
 6:     end if
 7: end procedure
 8: Logging in to Azure
 9: Userinput name of Container Registry
10: Store input in containerRegistry variable
11: Set the value of AcrName variable to the value of containerRegistry variable
12: Log out off Docker registry
13: Obtain the full registry ID
14: procedure DELETE EXISTING SERVICE PRINCIPAL
15:     if Service principal already exists then
16:         Delete already existing
17:     end if
18: end procedure
19: Create service principal with the ServicePrincipalName variable
20: Retrive the appId and print the UserName and Password variables
21: Create a Json file with the Docker authentication
22: Delete excisting Docker secret file
23: Create new Docker secret file with the output from Json file
24: Remove the Json file
25: Docker Secret now created
```

6.8 Azure

This section aims to explain configurations done in Azure which is crucial for the functionality of our application, as well as the security for it. We mostly link configuration instructions to officials sites as we see no purpose of trying to create a better instruction than whats already out there, and would consider it poor time management. Nevertheless, we do instruct what is needed to be done, but the specific method of procedure is found in the link. We also recognise that some of these configuration are quite basic, and is known to Tussa since they are familiar with Azure.

6.8.1 VM

As mentioned before, our application does consist of two virtual machines, one dedicated as a reverse proxy where Nginx runs. The other one has the role as the manager in the docker swarm and is where the rest of the microservices runs. The chosen name of each VM is sensitive as if they are going to be changed, they need to be changed in the stack file as well.

The VMs are both created in Azure within the same resource group, image (Ubuntu Server 20.04 TLS - x64 Gen2) and Region (North Europe), but poses different system configuration since they play different roles. The Nginx VM has the Standard B1s size which comes with 1 vCPU and 1 GiB of memory. Manager VM on the other hand require more processing power, and is fitted with the Standard B2s size consisting of 2 vCPUs and 4 GiB memory. Standard B1s cost 80,71 kr/month while Standard B2s cost 321,43 kr/month, making them quite affordable. This may changed when launched in a production environment with real data as it is described further in chapter about future considerations.

Each of these specifications needs to be configured when creating a virtual machine. Additionally do both bear their own asymmetric SSH key par for authentication. Next step after this is to enable disk encryption, but before this can be done, the feature needs to be switched on. It is done with the following steps:
In the Azure portal select the cloud Shell icon located on the top bar. This generates a separate shell window where you can write the following command to registrar the feature for your subscription:

Code listing 6.3: Registrer feature of your subscription

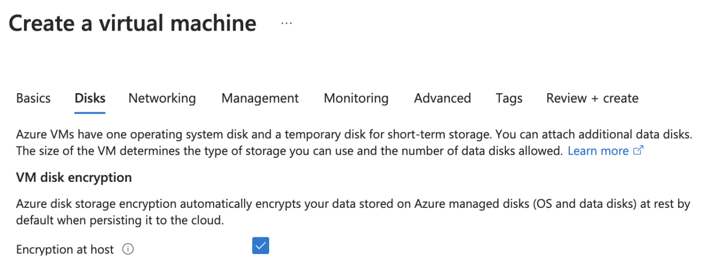
```
Register-AzProviderFeature -FeatureName "EncryptionAtHost"
    -ProviderNamespace "Microsoft.Compute"
```

Then do this command to confirm your acction:

Code listing 6.4: Confirm action

```
Get-AzProviderFeature -FeatureName "EncryptionAtHost"
    -ProviderNamespace "Microsoft.Compute"
```

After this you can enable disk encryption at host, when creating a new VM. The feature is located under the Disks section as shown below:


Create a virtual machine

Azure VMs have one operating system disk and a temporary disk for short-term storage. You can attach additional data disks. The size of the VM determines the type of storage you can use and the number of data disks allowed. Learn more

VM disk encryption

Azure disk storage encryption automatically encrypts your data stored on Azure managed disks (OS and data disks) at rest by default when persisting it to the cloud.

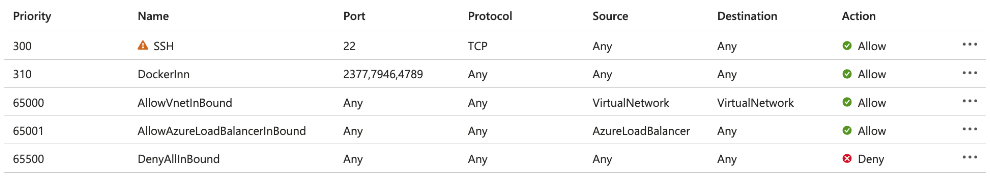
Encryption at host ✓

Figure 6.3: Enabling disk encryption at host

Enabling disk encryption on a host is a critical security aspect as data stored there is encrypts at rest. This features adds an additional security measurement to secure data at rest, which is one of the security requirements of Tussa.

Networking is another crucial security characteristic. It defines what kind of traffic to allow by configuring network security groups. It allows for a fine grained control on what to allow onto the network, potentially reducing the attack surface. This adds another layer of defence which can help against unauthorised access. It accomplish this by specifying what source and destination IP, Port number and Protocol to allow.

Below is a configuration on what we allow as inbound traffic, outbound is the same except without the SSH rule, as you do not need to SSH from the VMs to any place. The rule named "Dockerinn" is required to allow necessary traffic related to docker [59]. The three last rules are basic rules which are pre-configured. They allow all traffic from and to virtual network, as well as from an Azure load balancer, and denies everything that is not specified above. The Priority numbers determine the order in which the access policies apply, where the smallest number has the highest priority. It is important to mention that the SSH access should be turned off when not in use to lower the risk of unauthorised access.



Priority	Name	Port	Protocol	Source	Destination	Action
300	⚠ SSH	22	TCP	Any	Any	✓ Allow
310	DockerInn	2377,7946,4789	Any	Any	Any	✓ Allow
65000	AllowVnetInBound	Any	Any	VirtualNetwork	VirtualNetwork	✓ Allow
65001	AllowAzureLoadBalancerInBound	Any	Any	AzureLoadBalancer	Any	✓ Allow
65500	DenyAllInBound	Any	Any	Any	Any	✗ Deny

Figure 6.4: Inbound security rules

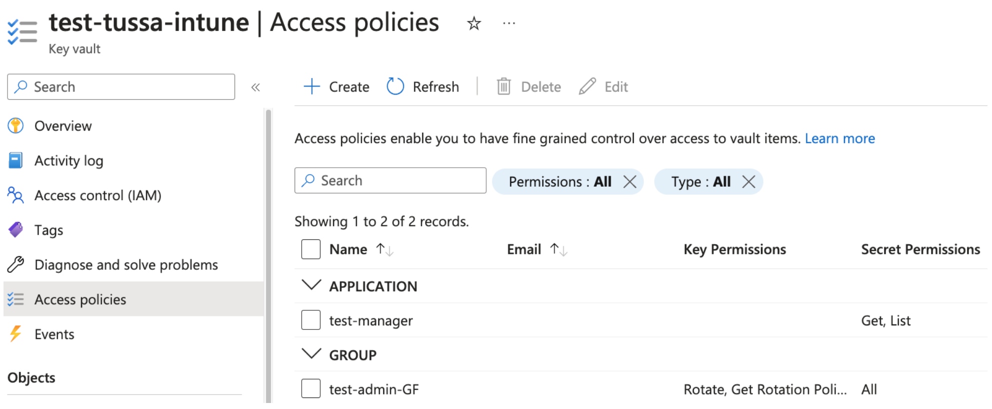
These configurations mentioned above are whats needed to deploy for each VM. Some important software needs to be downloaded, after the creation of each VM. Docker Engine and Azure CLI is needed to run docker swarm and connect to Azure through the terminal. The procedure to best accomplish this can be found on their web sites: Docker [60] - Azure CLI: [61]. Docker engine is needed on both VMs as they are both part of the docker swarm, while Azure CLI is only needed on the manager VM.

6.8.1.1 Access Control

System assigned managed identity is enabled on the manager VM to allow for credential less authentication towards the code. It gives access to Azure resources without the need of credentials. This feature allows us to store no credentials in the code, making it more secure. How this is applied is further shown below regarding Azure key vault.

6.8.2 Azure Key Vault

As mentioned does Azure Key Vault store our credentials. We have 4 different vaults. 3 for API credentials, and 1 for Grafana admin password. Creating an Azure Key Vault is fairly simple and can be done following this guide [62]. One thing to mention is the sensitivity of the names, as if they are to be changed, so does the code. Storing API credentials as secrets can be done following this instruction [63], while Grafana stores the password in Azure Key Vault on its own. The credentials stored each poses their own access policy. The access policy defines who may access it. The access policy for the APIs are the same and is as follows:



Access policies enable you to have fine grained control over access to vault items. Learn more

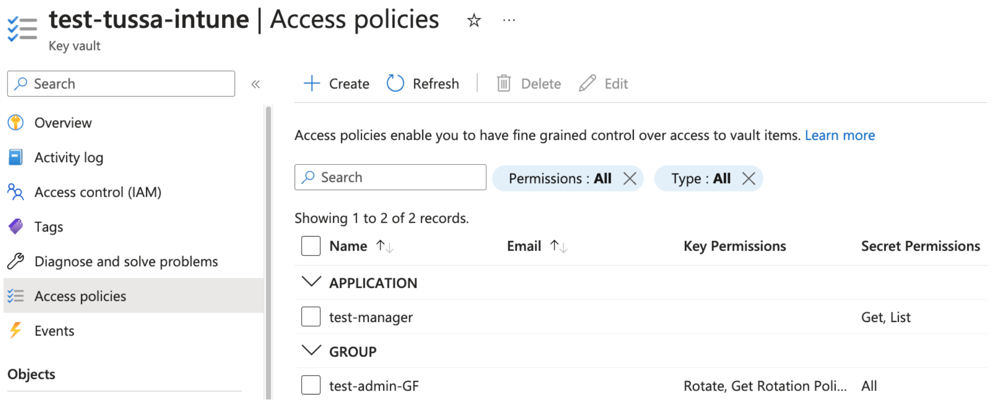
Showing 1 to 2 of 2 records.

Name	Email	Key Permissions	Secret Permissions
APPLICATION			
test-manager			Get, List
GROUP			
test-admin-GF		Rotate, Get Rotation Poli...	All

Figure 6.5: Access policy for AKV storing APIs

The Policy states that member of the "test-admin-GF" group have full access to the secrets as well as rotating privileges while the manager VM only has get and list access. The policy is configured to comply with least privilege principal, where people and services only have necessary access to function.

Access policy holding the Grafana break glass admin password is as follows:



test-tussa-intune | Access policies

Key vault

Access policies enable you to have fine grained control over access to vault items. Learn more

Showing 1 to 2 of 2 records.

Name	Email	Key Permissions	Secret Permissions
APPLICATION			
test-manager			Get, List
GROUP			
test-admin-GF		Rotate, Get Rotation Poli...	All

Figure 6.6: Access policy for AKV storing APIs

Members of "test-admin-GF" also have full access to this secret, but they do not have rotating access, since this is the job of the Grafana application. Grafana gets and set the secret, as well as fully delete it using purge. The price of operating a key vault is measured in number of transactions i.e. actions you take on the secret. The price is 0.319 kr/10.000 transaction [64].

6.8.3 Azure Container Registry

Creating an Azure Container Registry can be done following these steps [65]. It is important to specify that the resource group and location is the same in all configurations. The name of the container registry is not sensitive. The container registry comes in three different versions providing separate features. We have chosen to go with the basic option as we need minimum storage. The premium option provides Geo-replication, storing the images in multiple regions, in case of regional outage.

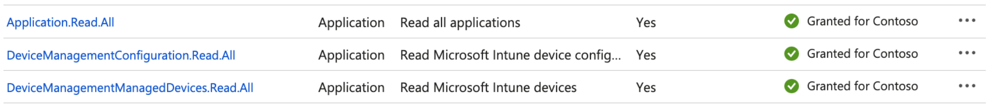
We presented this to Tussa where they decided the cost benefits ratio was not worth is since we store the code other places as well. It contains the two repositories holding the image for api-cron and Grafana application. Having the images stored here allows for automation in the sense that our stack automatically updates based on these images if there is a new version of them. This update can be done while all service are running, and takes them down for a brief moment if it can be updated. Azure automatically encrypts an image before storing it, and decrypts it on-the-fly, enhancing the security around data at rest requirement from Tussa [66].
A service principal is created to authenticate and letting the application access the container registry which is address in the next section.

6.8.4 Azure App Registration

App registration can be utilized to achieve a variety of different tasks and each of our app registrations provide different functionalities. We have a total of 3 app registrations, where one is used to configure the Intune API, another is servicing as the authentication method towards Grafana, and the last one allows access to our container registry as a service principal.

6.8.4.1 Intune API

This app registration allows for the code to retrieve information from Intune through the use of Microsoft Graph API. The APIs are configured with a strict read only access to necessary information complying with least privileged principal. The permissions below illustrate the necessary permission to retrieve data about devices in Intune.



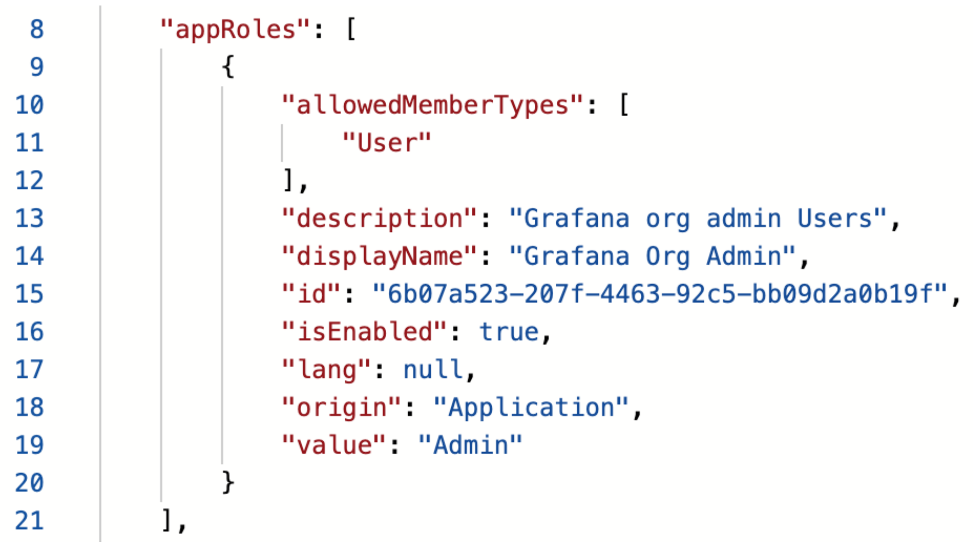
Application.Read.All	Application	Read all applications	Yes	✓ Granted for Contoso	···
DeviceManagementConfiguration.Read.All	Application	Read Microsoft Intune device config...	Yes	✓ Granted for Contoso	···
DeviceManagementManagedDevices.Read.All	Application	Read Microsoft Intune devices	Yes	✓ Granted for Contoso	···

Figure 6.7: Retrieve device information permissions

6.8.4.2 Azure AD OAuth2 Authentication

The app registration named "OAuth-grafana-test" is configured to enable Azure AD OAuth2 authentication towards Grafana. It allows existing Azure AD users to use their account to login to Grafana. This features eliminate the need of each users to create a Grafana account and have an extra set of username and password to manage. The app registration is configured with two redirect rules to enable the user to get back to the application after a successful sign in. The redirect rule contain the URL of the return address that the users gets redirected to. This URL is also required to be HTTPS to enable a secure connection.

A predefined application role for Grafana can be configured under the "manifest" section in the app registration. The admin role for Grafana is enabled here, meaning that users assigned with this role is dedicated to become admins in Grafana. This role is delegated to people that is part of the "test-admin-GF" group. To activate this privilege for the group members, it is necessary to go to "Enterprise applications" in Azure, then search for the app registration in the search bar. Finally under users and groups select "test-admin-GF". Below is the admin configuration in the manifest to create the app role.



```
    "appRoles": [
        {
            "allowedMemberTypes": [
                "User"
            ],
            "description": "Grafana org admin Users",
            "displayName": "Grafana Org Admin",
            "id": "6b07a523-207f-4463-92c5-bb09d2a0b19f",
            "isEnabled": true,
            "lang": null,
            "origin": "Application",
            "value": "Admin"
        }
    ],
```

Figure 6.8: Admin role in Grafana

6.8.4.3 Azure Container Service Principal

An app registration is created as a Service Principal rule. The rule exist to allow access to Azure key vault. The script `update-ACR-auth` is used to configure this service principal to automatically generate a new service principal if it exist. This is done to generate new credentials, in case of a breach or when a user quits. The Service principal is configured to only give pull access to the container registry as stated above in the "Script" section.

6.8.5 Azure Groups

Groups in Azure can be configured following this link [67]. Groups are configured to be individual for each customer Tussa has i.e. one group per customer. Azure has a feature named Azure B2B feature. This feature allows you to invite externals to your organization [68]. Users who are delegated the privilege of inviting externals are required specials roles in Azure to accomplish this task. This prerequisites can be the Privileged Role Administrator. As groups are free of charge to create, do they not poses any additional cost enabling Tussa no limits of creating groups. We have also assigned an Azure Group for the admins who can configure rights and implement changes in Grafana. This group is intended for key personal within Tussa.

6.8.6 Conditional Access Policies

Conditional access is configured to enable MFA for all users. This is an easy best practise rule to follow, especially considering the striking statistic is poses. According to Microsoft does MFA prevent 99.9% of identity attacks. Since every user regardless if its a normal user or admin has access to sensitive information. Therefore we see this mitigation as an absolute necessity. Additionally, Tussa states in their guideline for identity and access management under section 4.2 authentication (Appendix I), that only using password as a authentication mechanism is not considered as sufficient.

There are some prerequisites needed to create an conditional access policy to enable MFA. The user configuration it is required to have either of these accounts: "Conditional Access Administrator", "Security Administrator", or "Global Administrator privileges". For the MFA policy to apply is it demanded to attache it to the app registration mentioned above about Azure AD OAuth authentication. Further instruction on how to configured it can be found here: [69].

Chapter 7

Deployment

This chapter explains how our solution is going to be deployed within Tussas system. It includes prerequisites needed and technical details on how to launch it to a production environment from scratch. The chapter highlights the choices made concerning maintains, routines, scalability and security.

7.1 Prerequisites

Before handling the deployment of the application we need to mention the prerequisite needed to achieve a smooth deployment. Implementation of these are previously descried in the chapter about implementation. Nevertheless, we describe a short recap on how to deploy it in a production environment. This is due to the fact that our application is now running in a test environment with fake data, and everything needs to be moved and built up from scratch into a production environment.

The first thing that needs to be configure is Azure, building the foundations of our cloud based infrastructure. We utilise a variety of Azure features that are mentioned below. The necessity for these features are important for the security measures and access control they provide.
It is important to mention that the commands shown in this chapter are suited to use for VS Codes terminal for Windows. This is because Tussa utilises VS Code as their code editor and Windows is their primary operation system. Required tools: Azure CLI, Git, and Docker, are all anticipated to be installed in advanced as it is necessary for the commands to work.

7.1.1 Azure

Features that are utilized in Azure are mentioned below. All Azure configurations should be done in Tussas own production tenant.

7.1.1.1 Virtual Machine

We intend to start off doing a more thorough walk through of the set-up of the VMs as they are highly significant for the baseline of the infrastructure. First of is to create the VMs in Azure. Follow the description given in the implementation-chapter, in section 7.8.1 VM where comprehensive instructions can be found. As mention in the implementation-chapter, do both VMs needs to have `docker` installed, in addition to `az cli` for the manager VM.

They also need to be initiated as a docker swarm. First decide which VM that is going to be the manager. This is the node that is dedicated to control the Docker swarm and have the following microservices: Node.js, Elasticsearch, and Grafana. This is the command to initialise the manager node:

Code listing 7.1: Initialise manager node

```
sudo docker swarm init --advertise-addr <IP_ADDRESS_OF_VM>
```

In addition generates the command a second command that is used for worker nodes to join the swarm. Copy this command and run it on the desired VM that's going to be the worker. In our case this is the VM where Nginx is going to run. The command should look something like this:

Code listing 7.2: Initialise worker nodes

```
sudo docker swarm join --token <TOKEN>
```

Two scripts are also necessary to be executed. `update-ACR-auth.sh` and `update-SSL-Certificates.sh` provides the right service principal delegation and SSL certificates respectively.

7.1.1.2 Azure Key Vault

Here API keys for each vendor needs to be stored. The Grafana admin password is automatically persisted here when the services are deployed.

7.1.1.3 Azure Container Registry

Images used by the services needs to be stored here i.e. api-cron and Grafana image. These needs to be pushed to the container registry before launching the application. The following steps shows how it is possible to retrieve to code to a local environment. Build the necessary images and push them to the container registry.

The first thing that needs to be done is creating a container registry in Azure as shown in the implementation-chapter. The name of the registry is optional.
Next step is to pull code from GitHub that docker uses to create an image. The command is shown below. Follow the steps presented after running it to log in to GitHub. After successful authentication and passing MFA, go into the directory of the newly cloned repository.

Code listing 7.3: Clone Github repository

```
git clone <GitHub_REPO_URL>
```

It is required to log into Azure and its container registry to be able to communicate with it. The following commands shows you have to do it, and are necessary for completion of the final step:
Login to Azure, which takes you to a login page in the browser

Code listing 7.4: Log into Azure

```
az login
```

Login to the container registry.

Code listing 7.5: Log into Container registry

```
az acr login --name <ACR_NAME>
```

Final step is to push the images to the container registry. It is important to notice that the names of the images can not be changed if it is not also changed in the stack file. The following steps needs to be done in the directories where you have the code for both the api-cron service and the Grafana service. Navigate to the api-cron directory and create the api-cron image with necessary tag:

Code listing 7.6: Create api-cron image

```
docker build -t <ACR_NAME>.azurecr.io/api-cron:latest .
```

Push the image to the repository:

Code listing 7.7: Push api-cron image to repository

```
docker push <ACR_NAME>.azurecr.io/api-cron:latest
```

Navigate to the Grafana directory and create the Grafana image with necessary tag:

Code listing 7.8: Create Grafana image

```
docker build -t <ACR_NAME>.azurecr.io/grafana:latest .
```

Push the image to the repository:

Code listing 7.9: Push Grafana image to repository

```
docker push <ACR_NAME>.azurecr.io/grafana:latest
```

7.1.1.4 Azure App Registration

Azure consists of several app registrations which are significant for the application. As mentioned previously, app registration can consist of different functionalities. In our case for Service principals, Azure authentication in Grafana, and two for APIs. This is paragraph exist as a remainder of what app registrations that needs to be configured. While the instructions on how to implement them can be found in implementation-chapter. The app registrations is as follows:

- API to fetch information from Intune.
- API to fetch information about groups, and group members in Azure AD
- Enable Grafana authentication with Azure AD, and where to redirect the user after a. Successful authentication.
- Service principal to allow the application to gain access to the Container registry

7.1.1.5 Azure Groups

Configuring Azure groups is an important part for segregating tenants and managing access control. Creating the groups require precise work to avoid giving the wrong person unauthorised access by mistake. Each client that Tussa has poses their own group where designated personnel is invited to. The invitation proceed to give them necessary access for their correlating resources. Creating groups and inviting people is explain previously in implementation-chapter.

7.1.1.6 Conditional Access Policies

How to implement conditional access is mentioned in the implementation-chapter. The functionality of this feature is to enable MFA for users that are part of the admin group as previously stated.

7.1.1.7 System Assigned Managed Identity

Configuring System assigned managed identity is described previously in the chapter about implementation. As mentioned gives this feature the capabilities of allowing resource access to the virtual machines.

7.2 Deployment-Process

After all the prerequisites are arrange the application is ready to be deployed. Our goal was to find a way to automate the deployment process. In addition for all the microservices to deploy together instead of manually starting one by one. Docker swarm enables us to do just that through a stack file shown in the figures below. We create a stack file that contains all the commands needed to start each microservice. This features is one of the reason why we chose to use a container orchestration tool like Docker swarm.

We are going to explain each service configured in the stack file step by step. First line in the stack file explain which version we utilize. Then under `services:` comes all the services. Tab-delimited-format is used to indicate blocks of code belonging together.
api-cron is the first service to be mentioned. `image:` is used to specify which image the service is going to use. Here its specified to use api-cron:latest located in Azure container registry. One replica is to be made an is dedicated to run on the node named `test-manager`. If the container goes down unexpected under any conditions, the container try to restart for a total of 3 times with a 5 second gap between each attempt.

These specifications are universal for all the microservices. The services is also part of an Overlay Network making all service capable of communicating with each other across nodes. The purpose of the docker secret is further descried below. Lastly it depends on the Elasticsearch service to start before it. This is because Elasticsearch needs to be up and running before data can be sent to it from the api-cron service.

Code listing 7.10: Set up Docker Service 'api-cron'

```yaml
version: '3.7'
services:
  api-cron:
    image: testtussaacr.azurecr.io/api-cron:latest
    deploy:
      replicas: 1
      placement:
        constraints:
          - node.hostname==test-manager
      restart_policy:
        condition: any
        delay: 5s
        max_attempts: 3
        window: 120s
    networks:
      - tussa-network
    secrets:
      - acr-auth-config
    depends_on:
      - elasticsearch
```

The next service to be specified is Elasticsearch. The image is fetched from the Elastic docker registry, and uses version 7.17 of Elasticsearch. An older version is specified to rely on a fixed and tested version throughout the project. Newest versions may introduce new security vulnerabilities, as well as changing some key features used during the project. The default port Elasticsearch listens to is 9200 which is also stated in the stack file [70]. Environment variables added tells Elasticsearch to name the node, only one instance is to be made, and disable auto indexing. Line 42 and 43 shows that it has a Volume which allows it for information to be persisted even if the container goes down. It has no dependencies meaning it is the first one to start.

Code listing 7.11: Adding service for running Elasticsearch

```yaml
elasticsearch:
  image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0
  deploy:
    replicas: 1
    placement:
      constraints:
        - node.hostname==test-manager
    restart_policy:
      condition: any
      delay: 5s
      max_attempts: 3
      window: 120s
  ports:
    - "9200:9200"
  environment:
    - node.name=elasticsearch
    - discovery.type=single-node
    - action.auto_create_index=false
  networks:
    - tussa-network
  volumes:
    - elasticsearch_data:/usr/share/elasticsearch/data
```

Grafana has a bit longer configuration, due to the Azure AD authentication configurations is implemented here. Grafana also fetches its image from Azure container registry. It listens on port 3000 as it is the default port of Grafana [71]. Grafana depends on both Elasticsearch and api-cron to be started, in order to visualize the data. The environment variables Specified explains how the Azure ad authentication is configured, further explanation about the configuration can be found on Grafana's official page [72]. Nevertheless is it important to notice the `GF_AUTH_AZUREAD_ALLOW_ASSIGN_GRAFANA_ADMIN: "true"` field allowing Azure AD users to be assigned Grafana admin role. Meaning if a certain Azure group is assign the admin privileged, it is transferred to Grafana to be assigned the admin role there as well.

Code listing 7.12: Deploying Grafana Service in Docker Swarm

```
grafana:
  image: testtussaacr.azurecr.io/grafana:latest
  deploy:
    replicas: 1
    placement:
      constraints:
        - node.hostname==test-manager
    restart_policy:
      condition: any
      delay: 5s
      max_attempts: 3
      window: 120s
  ports:
    - "3000:3000"
  networks:
    - tussa-network
  secrets:
    - acr-auth-config
  depends_on:
    - elasticsearch
    - api-cron
  environment:
    GF_AUTH_AZUREAD_NAME: "Azure␣AD"
    GF_AUTH_AZUREAD_ENABLED: "true"
    GF_AUTH_AZUREAD_ALLOW_SIGN_UP: "true"
    GF_AUTH_AZUREAD_AUTO_LOGIN: "true"
    GF_AUTH_AZUREAD_CLIENT_ID: "${GF_CLIENT_ID}"
    GF_AUTH_AZUREAD_CLIENT_SECRET: "${GF_CLIENT_SECRET}"
    GF_AUTH_AZUREAD_SCOPES: "openid␣email␣profile"
    GF_AUTH_AZUREAD_AUTH_URL:
    "https://login.microsoftonline.com/${TUSSA_AZURE_TENANT}
/oauth2/v2.0/authorize"
    GF_AUTH_AZUREAD_TOKEN_URL:
    "https://login.microsoftonline.com/${TUSSA_AZURE_TENANT}
/oauth2/v2.0/token"
    GF_AUTH_AZUREAD_ALLOWED_GROUPS: "5e48474c-9f67-40d3-81e6-
f9ec6298aa1b,2bc480c5-4437-4463-849c-87cc2be788ec,c750cecf-
6be0-429b-93d3-1973b76945a7"
    GF_AUTH_AZUREAD_ROLE_ATTRIBUTE_STRICT: "false"
    GF_AUTH_AZUREAD_ALLOW_ASSIGN_GRAFANA_ADMIN: "true"
    GF_AUTH_AZUREAD_SKIP_ORG_ROLE_SYNC: "true"
    GF_SERVER_ROOT_URL: "https://securityportal.tikt.no"
```

Nginx utilize its stable version. It is a more tested in regards to compatibility, security and stability. Find support for this version is also more likely because it is more used. It expose port 80 and 433 witch is the normal ports for web traffic. The two docker secrets is utilised on line 19 and 20 represents the SSL certificates. Two configuration files are copied into the container to configure Nginx. Nginx rely on Grafana to start, making it the last microservice to deploy.

Code listing 7.13: Deploying Nginx Service in Docker Swarm

```
 1 nginx:
 2   image: nginx:stable
 3   deploy:
 4     replicas: 1
 5     placement:
 6       constraints:
 7         - node.hostname==test-ngnix
 8     restart_policy:
 9       condition: any
10       delay: 5s
11       max_attempts: 3
12       window: 120s
13   ports:
14     - "80:80"
15     - "443:443"
16   networks:
17     - tussa-network
18   secrets:
19     - fullchain_pem
20     - privkey_pem
21   volumes:
22     - /home/test-nginx/default.conf:/etc/nginx/conf.d/default.conf
23     - /home/test-nginx/nginx.conf:/etc/nginx/nginx.conf
24   depends_on:
25     - grafana
```

The last figure illustrates how the Overlay Network, docker secrets, and Volume is created.

Code listing 7.14: Defining the network, secrets, and volumes configuration for Docker Swarm

```
networks:
  tussa-network:
    driver: overlay
    driver_opts:
      encrypted: "true"

secrets:
  acr-auth-config:
    external: true
  fullchain_pem:
    external: true
  privkey_pem:
    external: true

volumes:
  elasticsearch_data:
    external: true
```

The services are simply deployed using the following command:

Code listing 7.15: Launch/Redeploy the stack

```
sudo docker stack deploy --with-registry-auth -c stack.yml tussa-app
```

This command is launched from the VM, which is the manager in the docker swarm. This part is important to execute, because the manager node is the only one that is allow to carry out this action. It is also noteworthy to specify that the command needs to be typed in the same folder as the `stack.yml` file is located. Different part of the command have their own aspect and necessity.

As mentioned, the command is all you need in order to deploy the application. The command is built up by different parts. `sudo` enables the command to run as a super user on the Ubuntu VM. `docker stack deploy` This is the command that deploys a set of services. `-with-registry-auth` is needed to allow the services to contact a private container registry to pull images from. `-c stack.yml` specifies the which stack file to use and where to find it. `tussa-app` defines the name of the stack, and can be optional in general. But in our case it needs to have this exact value to correlate with whats specified in the code.
In short, the command tells docker to deploy a stack, based on the stack.yml file containing the microservices specifications, and then calls the stack: "tussa-app".

7.3 Routines

The routines stated below explains how to maintain the application, why these are important to follow, and what kind of benefit they provide. This are routines based on Best Practise methods, so these routines can actually be used on all work with integration and API.

Tussa has been clear throughout the project about the importance of proper routines regarding sensitive information. This section aims to explain just that, what routines are necessary, how to follow them and the importance of them. The routines are based on Best Practise principals, and aims to create a guideline for Tussa to follow. It is important to mention that even when implementing these best practise routines, that they can not mitigate all the risk associated with this application. In a changing society is there always some level of risk one have to accept.

7.3.1 Non-Disclosure Agreement

A basic routine that covers a broad aspect, allowing people to have access to the sensitive information such as code and credentials opens up for a vulnerability where people can pass it on and share it with other people. It is difficult to avoid this vulnerability, and to be completely secured against it. It is therefore up most important that employees who has access to this information signs a Non-Disclosure Agreements (NDA). The NDA should explicitly mention that employees are not allow to share sensitive information about this application with unauthorized personnel. Its important to mention that this does not mitigate the vulnerability completely, but helps on keeping it secure.

7.3.2 Code / Scripts

Developers who wants to edit the source code needs to pull the code to their machine from GitHub, and push the newly updated code back to the Repository. Because of this, users stores a copy of the code locally on the PC where they pulled it to. No code should be stored anywhere else than on a work laptop where proper security measurements have already been implemented [73]. Only using a work laptop to edit code and scripts enables for a clear separation between work and private activity, creating less likelihood of data leak or mixed data.

In combination of whats stated in the last two sections, we recommend an access policy for devices as well. The purpose of the policy is to only grant access to the code and let a user login, if they are doing it from a verified device. This is to create a multi layer security strategy, thereby enhancing the safety of the scripts and code.
It should be implement a routine to review code, to ensure code quality, and protect against potential vulnerabilities in the code. The policy should include what is to be reviewed and by whom. It should be integrated as normal procedures in the developing phase, i.e. in the form of Q&A before pushing the code to GitHub and into production.

Storing the code in a second location in case of outage creates higher redundancy and helps increases the availability of the code. We recommend using familiar backup services and methods for Tussa. These services and methods, including Veeam Backup, Azure, Locally and Tape, offers a variety of solutions which can benefit Tussa against an outage, but also in case of a cyber-attack. Backup should ideally be made after every change. For enhances security should the best practise 3-2-1 rule be followed. the 3-2-1 backup role takes the security one step further with a total of 3 copies [74].

7.3.3 API Credentials And Passwords

All API credentials are always stored in Azure Key Vault and no place else. As mentioned before, is it favorable to have them securely collected in one place. It makes it easier to manage and to provide access control as well as monitoring the credentials. The same goes for passwords stored there.

Only people in the designated admin group have full access to credentials stored in Azure Key vault. implementing least privilege access for the application to gain access to AKV is important to sustain the integrity. This means that the application only have read access as it is all it needs. MFA policy should also be implemented to ensure the validation of the user. Enabling MFA complies with Tussa's guidelines regarding access control.

A policy in Azure should also be implemented stating that only certain pre-validated machines can access the key-vault. A designated person should be notified when either a user logs-in or when the machine gets validated. This routine enhances monitoring and adds an additional security measure to prevent unauthorised access.

To preserve the integrity of the API credentials stored in Azure Key Vault it is important to scheduled a rotation of them. This means that they are periodical changed. It is recommended to rotate API credentials annually [75] [76]. API credentials should also be rotated in case of a security incident and should be implement in Tussa's incident response plan. Rotating API credentials should also be implement in the off boarding process of an employee.

Actions to automate the process of renewing API credentials is recommended to be done to create minimal down time. Changing the API credentials should be done in a different time slot of the one time a day when they are used i.e. 9 pm in the evening. It can be easy to track older API credentials as it is possible to view older secrets in Azure Key Vault if they are not deleted.
It should be added as a routine to implement regularly updates and security patching for the code. It can protect against know vulnerabilities [77].

Comprehensive security training should be compulsory for personnel managing APIs, and integrate it as a duty for their role. Topics should consist of how they are handled securely in code, understand ordinary API threats like broken object level authorization and broken authentication, and how to prevent them with proper mitigation.

An API rate limit should be implemented as it controls how frequently request can be made by the application. This is configured and managed on the vendors endpoint to allow how many times an API can be requested, to retrieve information. Therefore, not directly affecting our application development. Regardless is it a good practise to follow, and it can help prevent against DoS attack towards the vendors. What it can do is increasing the data availability for Tussa in case API credentials are compromised and this is being exploited.

Configuring an API gateway on Nginx is a native capability. Its recommended to implement this as it can enhance the management of the API traffic flow. It should be configured to sit in front of our api-cron microservice to restrict API calls to only be allowed to pass through when they are scheduled to. A notification to an administrator should be issued if an API request is made to one of the endpoints outside the scheduled time frame. Implementing an API gateway and its benefits is further described in future considerations.

7.3.4 Docker Secrets

As mentioned previously, one of the docker secrets stored in the docker swarm contains the credentials to access the service principal, allowing the applications to gain access to the Container registry. The other two is the private and public SSL Certificates that Nginx uses to enable HTTPS. A great feature with docker secret is that they are encrypted during transit and rest, which is a great option considering its one of Tussas requirements.

7.3.4.1 Service Principal

The Service principal and its credentials does not have the same necessity as API credentials to be rotated as frequent. As the APIs contains far more sensitive and delicate information. The service principal only allow for pulling images from the Container registry, while the APIs have full access to all the devices from the three products. Therefore should it be sufficient to rotate the service principal and its credentials primarily in case of a security incident or when an employee with access resigns.
You can simply run the script "update-ACR-auth.sh" to renew the service principal and its credentials when necessary. This script is made to automate and simplify the process.

7.3.4.2 SSL Certificate

SSL certificates from Let's Encrypt holds a relatively short life cycle of 90 days, and tus they need to be changes accordingly. As described earlier Certbot does not support an auto renewal DNS plugin for the DNS provider that Tussa use. The certificates needs to be generated manually. You can the command: "certbot –manual –preferred-challenges http-01 -d Securityportal.tikt.no". Follow so the guidelines that are presented by certbot [78]. After this run the script "update-SSL-Certificates.sh" to automatically deploy the application with the new SSL Certificates.
For further work we recommend creating a authentication hook script with the "–manual-auth-hook" option added to command above to automatically update SSL certificates [78].

7.3.5 Azure

It should be added to the off boarding process for when a user quits to revoke the user of their Azure account, so that they no longer has the correlating access that comes with it. This align well with Tussas existing guidelines.

It is essential to perform a compressing identity validation before granting personnel access to this group. This implies that any Azure account seeking to join this group must undergo a detailed security review to safeguard any potential security breaches. It also essentials for personnel to manifest full control over their respective account. Furthermore, it is indispensable for the personnel to conduct necessary security training, making them eligible to handle sensitive information. This route makes sure to follow Tussa's guidelines regarding identity management.

We deemed it as important to inspect the Azure groups related to this application periodically because of the resources they grant access to. We recommend doing this annually at the same time as the API Credentials are rotated. A designated person from each customer should also be delegated the responsibility to manage their own group. The person should control whom to invite and have the overall authority of the participants.

Auditing user access should also be done in the same period, as employees might accumulate extra access over time than needed to comply with the least privileged principal.
It is considered a best practise in DevOps to Scan docker images stored in Azure container registry for vulnerabilities [79]. It adds another layer of security which helps ensure the integrity. It also help minimizing the risk of exploitation.

Allowing SSH traffic in network security groups should be turn off, when direct access to the VMs is not necessary. This precaution can reduce the attack surface by minimizing when access is authorized. Configuring a bastion as authentication to access the virtual machines can be implemented to avoiding this cumbersome method of turning on and off SSH. further explanation about this is addressed in further considerations.

7.3.6 Rotation Of Employees

We repeat the steps for when a person quits, as this was important for Tussa. We have created an easy highlight process to follow: • Run Update script to update service principal to gain access to ACR • Update / renew SSL certificates • Update SSH keys • Restart Grafana service to generate a new admin break glass password.

7.4 Scalability

The application has no functioning auto scaling as this was not stated in the assignment as a requirement. Nevertheless are scalability crucial functionality for a modern web application due to several reason.

There are two different types of scaling, Vertical and Horizontal scaling. Vertical scaling involves increasing the resources on the already existing resource e.g. increasing the CPU power. This approach is simple, but can come with some limitations to the resource capacity on the machine. Horizontal scaling involves adding more services of the same service to the infrastructure to increased the load capabilities. i.e. add one more containers running Grafana, running alongside the already existing one. Horizontal scaling is more complex since it also require a load balancing mechanism, but in return offer far greater flexibility, and possess no limitation but cost [80].

A web application should be able to handle different kind of load during the day. The web site may slow down or even crash, creating a reduced user experience, if its not designed to handle increased load. A more redundant application is achieved by having multiple instances of the services. This increase the availability of the application, ensuring it is accessible when needed. It also prevent the service from going down if something one of the instances are having problems or are hijacked. Scalability also allows for reduced cost as you only pay for resources used.

Due to these reasons, is scaling options further addressed in the chapter regarding future considerations. Here it is discussed how Tussa can achieve auto scaling for the application.

7.5 Final Solution With User Guide

This section aims to display the final solution and how a user with access can access the dashboard.

1. **Accessing the domain**
   - Description: Visit https://securityportal.tikt.no/ which redirect you to https://securityportal.tikt.no/login which display the login page of the portal.

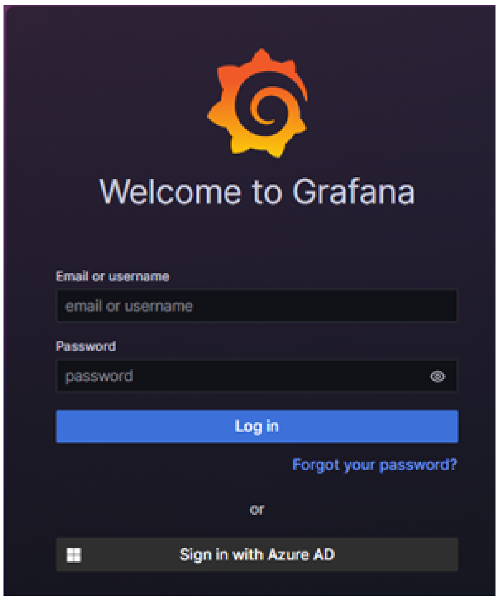
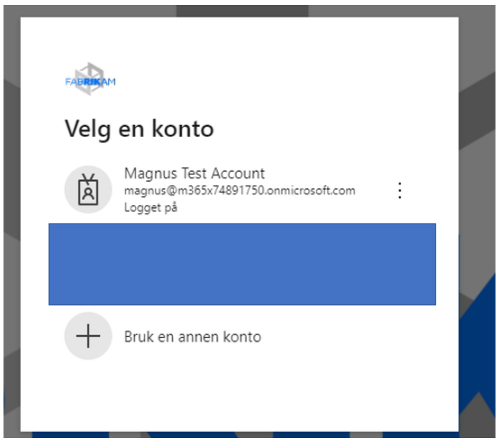


Figure 7.1: What you see when visiting the portal

2. **Signing in with Azure AD**
   - Description: As you see on Figure 7.1 there is a button that displays "Sign in with Azure AD". Click on the button to start the process of authenticating to access the dashboard.


Velg en konto

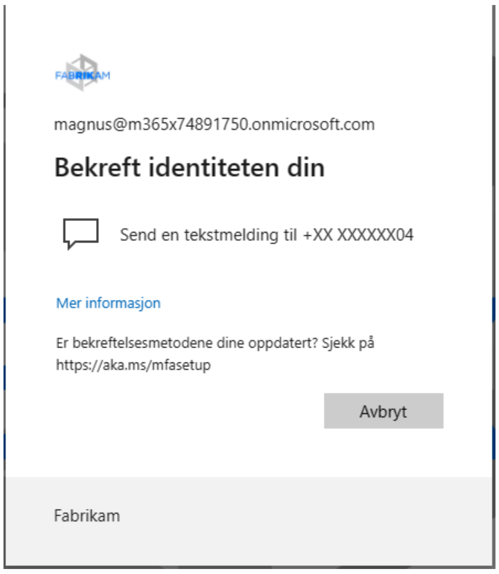
Magnus Test Account
magnus@m365x74891750.onmicrosoft.com
Logget på

[redacted]

Bruk en annen konto

Figure 7.2: What you see when visiting the portal

   - MFA is required after single sign in is succeeded. In this case of the user case was mobile phone registered as MFA-method. Other methods can be added.


magnus@m365x74891750.onmicrosoft.com

Bekreft identiteten din

Send en tekstmelding til +XX XXXXXX04

Mer informasjon

Er bekreftelsesmetodene dine oppdatert? Sjekk på https://aka.ms/mfasetup

Avbryt

Fabrikam

Figure 7.3: MFA challenge

3. **Visit your respective dashboard**
   - Description: You get access to your respective dashboard according your tenant, after a successful authentication.

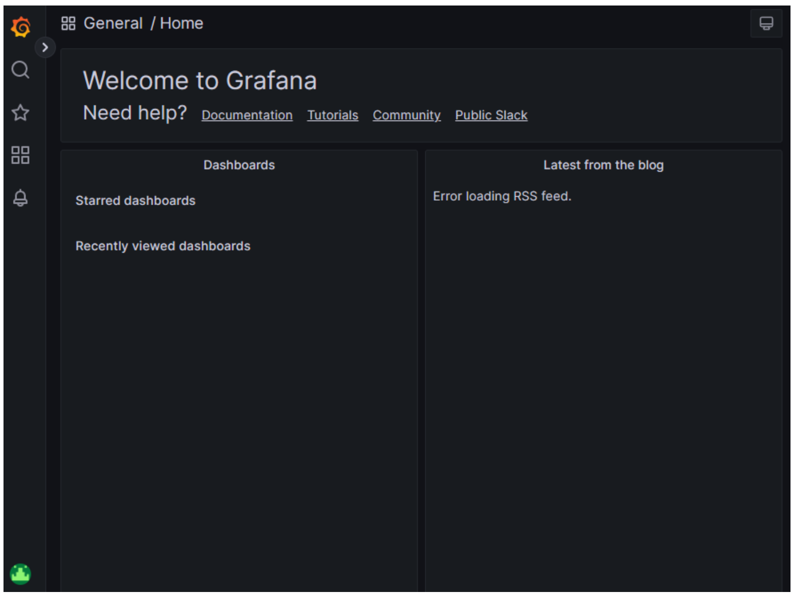


Figure 7.4: The view after a successful authentication

   - Later you have to click on the corresponding icon to Figure 7.5 in the left menu:

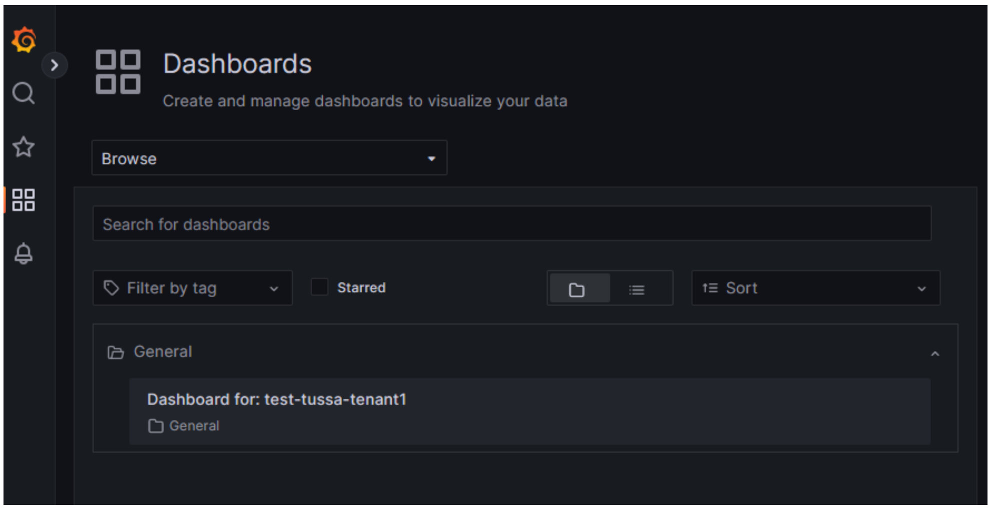
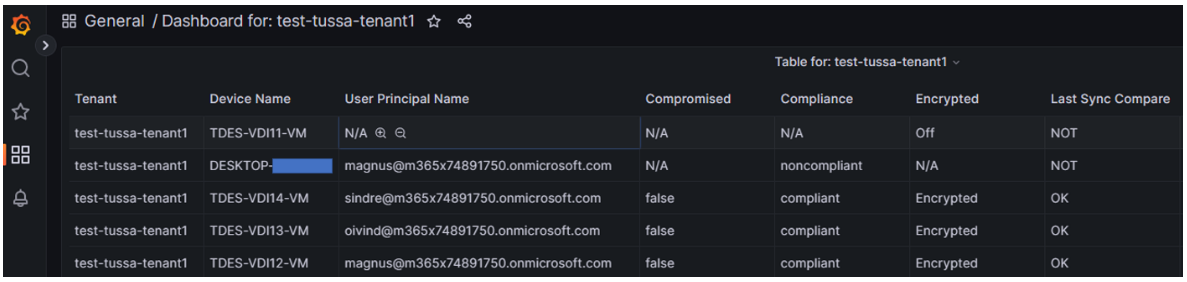

Dashboards
Create and manage dashboards to visualize your data
Browse
Search for dashboards
Filter by tag
Starred
Sort
General
Dashboard for: test-tussa-tenant1
General

Figure 7.5: Dashboard icon displaying which dashboard you can access

   - "Test-tussa-tenant1" represent a customer/tenant. To progress further click on the "Dashboard for <tenant> as displayed in Figure 7.5.



General / Dashboard for: test-tussa-tenant1

Table for: test-tussa-tenant1

Tenant	Device Name	User Principal Name	Compromised	Compliance	Encrypted	Last Sync Compare
test-tussa-tenant1	TDES-VDI11-VM	N/A	N/A	N/A	Off	NOT
test-tussa-tenant1	DESKTOP-[redacted]	magnus@m365x74891750.onmicrosoft.com	N/A	noncompliant	N/A	NOT
test-tussa-tenant1	TDES-VDI14-VM	sindre@m365x74891750.onmicrosoft.com	false	compliant	Encrypted	OK
test-tussa-tenant1	TDES-VDI13-VM	oivind@m365x74891750.onmicrosoft.com	false	compliant	Encrypted	OK
test-tussa-tenant1	TDES-VDI12-VM	magnus@m365x74891750.onmicrosoft.com	false	compliant	Encrypted	OK

Figure 7.6: Dashboard icon displaying the endpoints inside the tenant

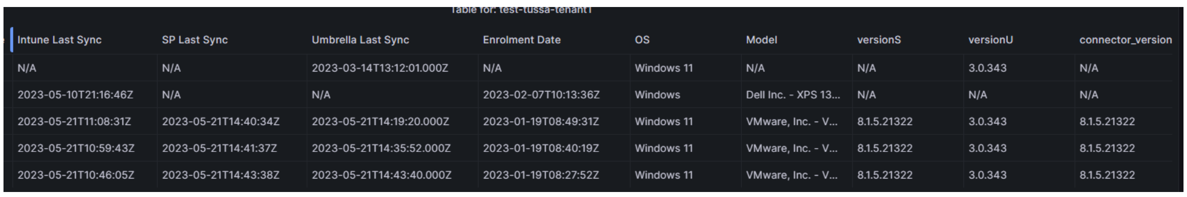


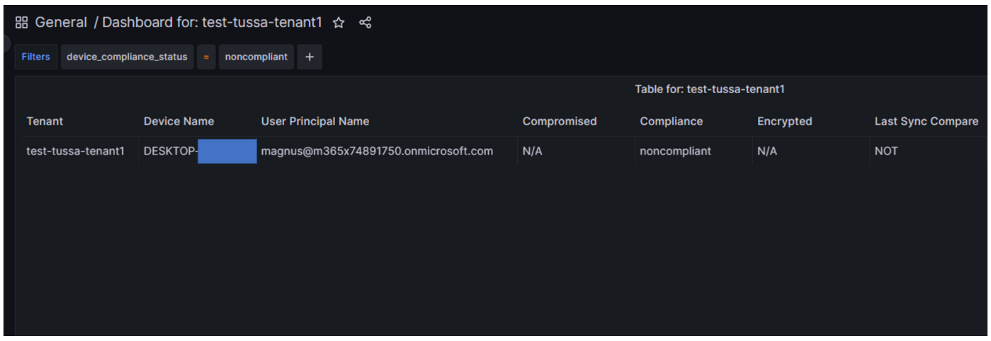
Table for: test-tussa-tenant1

Intune Last Sync	SP Last Sync	Umbrella Last Sync	Enrolment Date	OS	Model	versionS	versionU	connector_version
N/A	N/A	2023-03-14T13:12:01.000Z	N/A	Windows 11	N/A	N/A	3.0.343	N/A
2023-05-10T21:16:46Z	N/A	N/A	2023-02-07T10:13:36Z	Windows	Dell Inc. - XPS 13...	N/A	N/A	N/A
2023-05-21T11:08:31Z	2023-05-21T14:40:34Z	2023-05-21T14:19:20.000Z	2023-01-19T08:49:31Z	Windows 11	VMware, Inc. - V...	8.1.5.21322	3.0.343	8.1.5.21322
2023-05-21T10:59:43Z	2023-05-21T14:41:37Z	2023-05-21T14:35:52.000Z	2023-01-19T08:40:19Z	Windows 11	VMware, Inc. - V...	8.1.5.21322	3.0.343	8.1.5.21322
2023-05-21T10:46:05Z	2023-05-21T14:43:38Z	2023-05-21T14:43:40.000Z	2023-01-19T08:27:52Z	Windows 11	VMware, Inc. - V...	8.1.5.21322	3.0.343	8.1.5.21322

Figure 7.7: More of the current log fields that are being displayed

- The current data that are being displayed in the current dashboard are the endpoints inside the tenant and the log fields that have been requested by the client. It is easy to change the present, and might be changed in the future if the client want so in the future. To make sure we got the required functionality, has a non-compliant endpoint been added to make sure we can filter.

4. **Filter on log fields**
   - Description: If you want to filter on the current log fields are the multiple ways to do it. The easiest method is to click on the log field where you get the option to filter on that respective field.



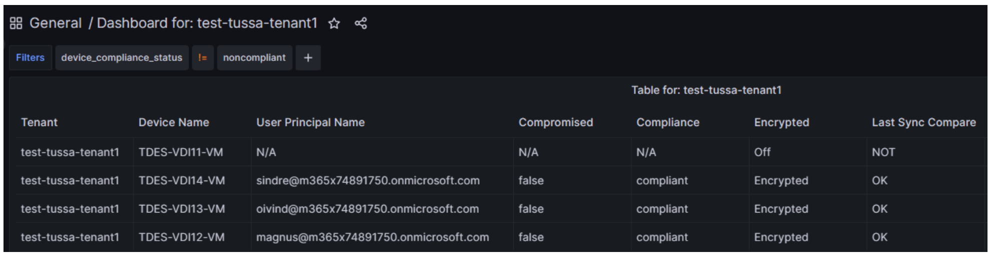
General / Dashboard for: test-tussa-tenant1

Filters device_compliance_status = noncompliant

Table for: test-tussa-tenant1

Tenant	Device Name	User Principal Name	Compromised	Compliance	Encrypted	Last Sync Compare
test-tussa-tenant1	DESKTOP-[redacted]	magnus@m365x74891750.onmicrosoft.com	N/A	noncompliant	N/A	NOT

Figure 7.8: Filtered on non-compliant devices


General / Dashboard for: test-tussa-tenant1

Filters device_compliance_status != noncompliant +

Table for: test-tussa-tenant1

Tenant	Device Name	User Principal Name	Compromised	Compliance	Encrypted	Last Sync Compare
test-tussa-tenant1	TDES-VDI11-VM	N/A	N/A	N/A	Off	NOT
test-tussa-tenant1	TDES-VDI14-VM	sindre@m365x74891750.onmicrosoft.com	false	compliant	Encrypted	OK
test-tussa-tenant1	TDES-VDI13-VM	oivind@m365x74891750.onmicrosoft.com	false	compliant	Encrypted	OK
test-tussa-tenant1	TDES-VDI12-VM	magnus@m365x74891750.onmicrosoft.com	false	compliant	Encrypted	OK

Figure 7.9: Filtered on compliant devices

5. **Sign out**
   - Description: After a finish session inside the portal can you sign out .

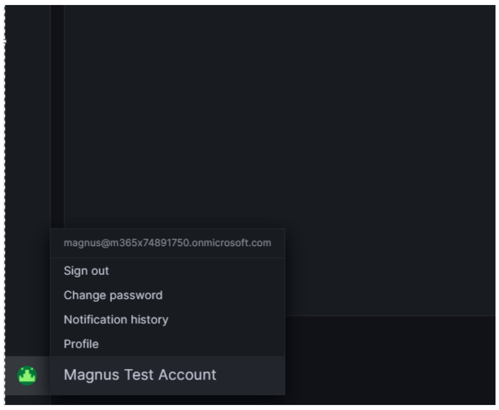


Figure 7.10: Where to sign out

Chapter 8

Security Testing

One of the security requirements from the client was to perform security testing. Sindre and Magnus have followed the course IIK3100 - Ethical Hacking and Penetration Testing, where we were introduced to ethical hacking and security of IT systems by trying to find and exploit security vulnerabilities [81]. This chapter descries how we did it, and what we found.

8.1 Testing Of The Web Application

For security testing of the web application, Kali Linux was used, and the tools that follows this system. Kali Linux is an open-source distribution of the operating system Linux, which aims at security testing with more. The system contains over 600 tools which can be used to perform testing [82].

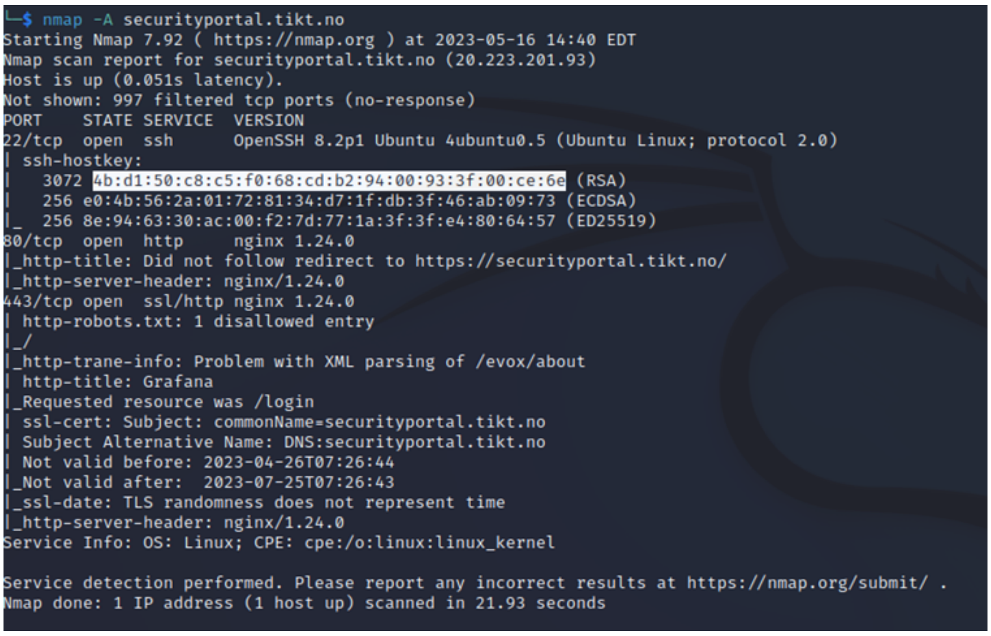
8.1.1 Nmap

Nmap is short for Network mapper and is a tool used for performing reconnaissance scanning to explore networks, open ports and discover information about services running [83]. A nmap scan with the following command was ran:

Code listing 8.1: command for nmap scan

```
nmap -A securityportal.tikt.no
```

When running this command the option -A was included. This option includes several other options which includes operating system detection, if a host is up or down, and can enables a default scan which enables users to execute pre-define scripts. The following was information was gathered:



```
└─$ nmap -A securityportal.tikt.no
Starting Nmap 7.92 ( https://nmap.org ) at 2023-05-16 14:40 EDT
Nmap scan report for securityportal.tikt.no (20.223.201.93)
Host is up (0.051s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 8.2p1 Ubuntu 4ubuntu0.5 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 4b:d1:50:c8:c5:f0:68:cd:b2:94:00:93:3f:00:ce:6e (RSA)
|   256 e0:4b:56:2a:01:72:81:34:d7:1f:db:3f:46:ab:09:73 (ECDSA)
|_  256 8e:94:63:30:ac:00:f2:7d:77:1a:3f:3f:e4:80:64:57 (ED25519)
80/tcp  open  http     nginx 1.24.0
|_http-title: Did not follow redirect to https://securityportal.tikt.no/
|_http-server-header: nginx/1.24.0
443/tcp open  ssl/http nginx 1.24.0
| http-robots.txt: 1 disallowed entry
|_/
|_http-trane-info: Problem with XML parsing of /evox/about
| http-title: Grafana
|_Requested resource was /login
| ssl-cert: Subject: commonName=securityportal.tikt.no
| Subject Alternative Name: DNS:securityportal.tikt.no
| Not valid before: 2023-04-26T07:26:44
|_Not valid after:  2023-07-25T07:26:43
|_ssl-date: TLS randomness does not represent time
|_http-server-header: nginx/1.24.0
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.93 seconds
```

Figure 8.1: Print from executed Nmap command

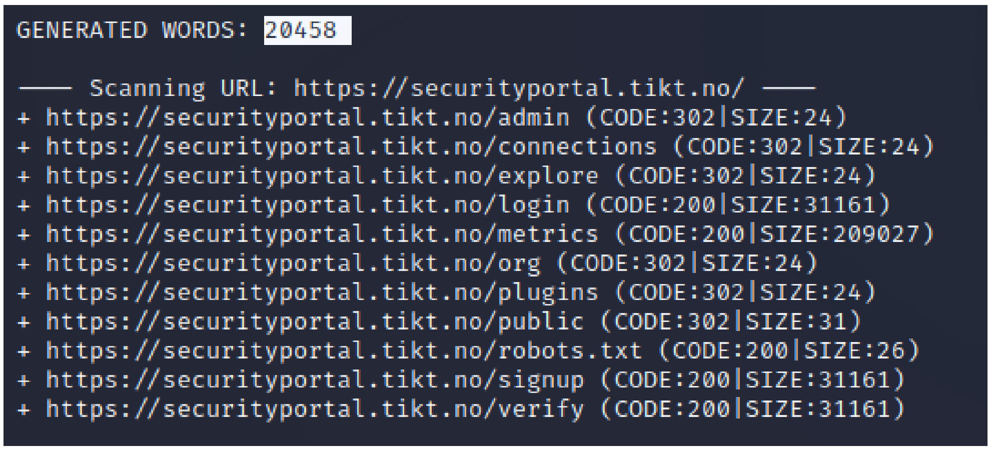
As we can see, we get information regarding which ports thats open, and what services that are running on the ports. What's interesting is what we see on port 22 regarding SSH. We can see the ssh-hostkeys. The ssh-hostkey is a crypto-key used SSH-protocol for authentication. The hostkey is being presented when a SSh connection establishes as a verification, however by walking through nmap's documentation, it does seem like the script only collects the fingerprint of a public key [84]. As it is public by default does information like this serve no vulnerability or threat.

8.1.2 Dirb

Dirb is a tool inside Kali that allows to scan for directories with a predefined list of domains [85]. Kali does also contain a word list of the most common directories. By using predefined wordlist named big.txt which contains 20458 names which is further used in searching for directories. We ran the following command:

Code listing 8.2: command for dirb scan

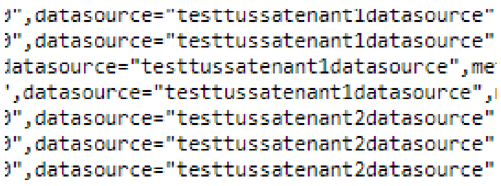
```
dirb https://securityportal.tikt.no
    /usr/share/dirb/wordlists/big.txt
```



```
GENERATED WORDS: 20458

---- Scanning URL: https://securityportal.tikt.no/ ----
+ https://securityportal.tikt.no/admin (CODE:302|SIZE:24)
+ https://securityportal.tikt.no/connections (CODE:302|SIZE:24)
+ https://securityportal.tikt.no/explore (CODE:302|SIZE:24)
+ https://securityportal.tikt.no/login (CODE:200|SIZE:31161)
+ https://securityportal.tikt.no/metrics (CODE:200|SIZE:209027)
+ https://securityportal.tikt.no/org (CODE:302|SIZE:24)
+ https://securityportal.tikt.no/plugins (CODE:302|SIZE:24)
+ https://securityportal.tikt.no/public (CODE:302|SIZE:31)
+ https://securityportal.tikt.no/robots.txt (CODE:200|SIZE:26)
+ https://securityportal.tikt.no/signup (CODE:200|SIZE:31161)
+ https://securityportal.tikt.no/verify (CODE:200|SIZE:31161)
```

Figure 8.2: What directories we found after running the dirb-command

Code 302 tells that it found a domain, but gets redirected. After checking all code 302 domains, do you get redirected to the home page where you can login. After also checking the code 200 domains we did find a critical finding which needs to be fixed instant. By checking the metrics directory we did find the following:



```
)",datasource="testtussatenant1datasource"
)",datasource="testtussatenant1datasource"
datasource="testtussatenant1datasource",me
',datasource="testtussatenant1datasource",
)",datasource="testtussatenant2datasource"
)",datasource="testtussatenant2datasource"
)",datasource="testtussatenant2datasource"
```

Figure 8.3: Print from the Metrics directory displaying tenat-data

It can be hard to read, but what is listed are several customer names.When working with test data, the customers have been called test<customer>1 and test<customer>2. Whats more interesting about this observation is that you do not even have to be signed in to access this data. This is a pretty bad information leakage and should be fixed instantly. The solution to this was to edit the configuration file for the reverse proxy, where a block rule had to be added for the /metrics domain.

8.1.3 SQL Injections

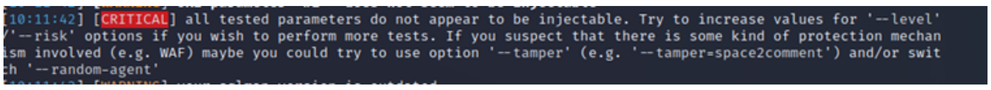
SQL injection is a common hacking technique that allows a threat actor to send queries to a database, with intentions to either view more data than they are supposed to or destroy the database [86]. In our case is the focus on SQLmap. SQLmap is a tool where the goal is to detect SQL injection vulnerabilities inside the application [87]. We ran the following commands below. What these commands do is to search for databases and extract them to check if I can dump them later. As Figure 8.4 displays, we did not manage to find any databases with the SQLmap.

Code listing 8.3: SQLmap of portal

```
sudo sqlmap -u "https://securityportal.tikt.no/login" --dbs
```

Code listing 8.4: SQLmap of portal when signed in

```
sudo sqlmap -u "https://securityportal.tikt.no/?orgId=1" --dbs
```



```
[10:11:42] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'
/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechan
ism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or swit
ch '--random-agent'
```

Figure 8.4: Print from sqlmap. Both codes gave the same output

8.1.4 Brute Force

Even though the web application is designed to be authenticated through Azure AD with the button below in Gigure 8.5 with the "Sign in with Azure AD". A brute force attack is when an attacker sends multiple login-requests for a user hoping that one the requests authenticates. The test aims to get access through username and password-fields.

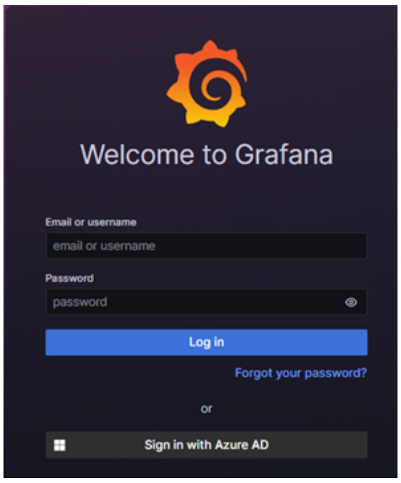
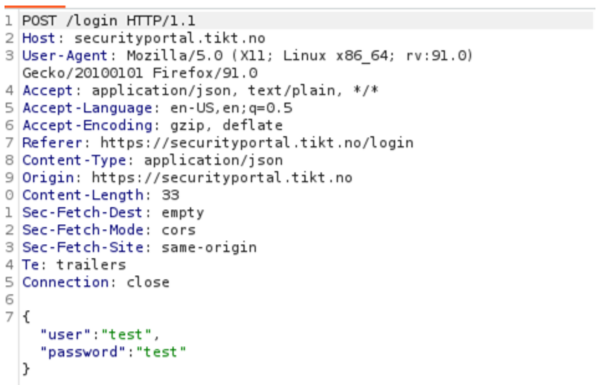


Figure 8.5: Our sign in page to Grafana

To do this did we used Burp Suite to capture packets to identify the name of the fields we are supposed to brute force. Displayed in the picture below you can see two fields named user and password. This is the post request we send by trying to login to the page. We tried to sign it with test as username and password.



```
POST /login HTTP/1.1
Host: securityportal.tikt.no
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0)
Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://securityportal.tikt.no/login
Content-Type: application/json
Origin: https://securityportal.tikt.no
Content-Length: 33
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers
Connection: close

{
  "user":"test",
  "password":"test"
}
```

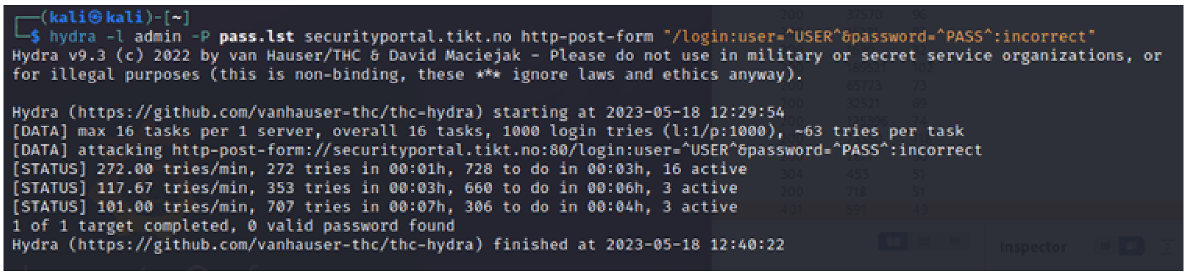
Figure 8.6: Post capture when attempting to sign in

Hydra is a tool inside Kali that allows the user to perform several attacks regarding login cracking [88]. After performing open-source intelligence, we found that the default user that is being created has the username “admin” [89]. Therefore is the "admin"-user our target. We ran the following command

Code listing 8.5: Command for brute force with most common passwords

```
hydra -l admin -P pass.lst securityportal.tikt.no http-post-form
    "/login:user=^USER^&password=^PASS^:incorrect"
```

In the first command we did use a list containing the 1000 most common passwords according to open source intelligence [90], and named the list pass.lst. Pay attention to that we define the user and password to match the following fields above. We received 0 hits on this try.



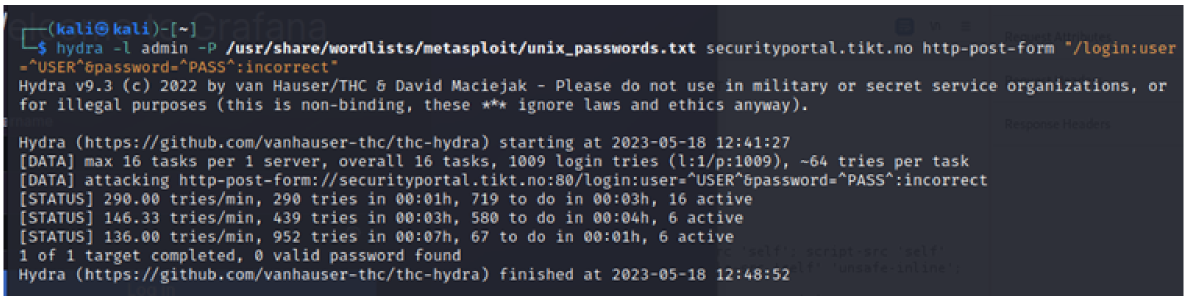
```
┌──(kali㉿kali)-[~]
└─$ hydra -l admin -P pass.lst securityportal.tikt.no http-post-form "/login:user=^USER^&password=^PASS^:incorrect"
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or
for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-18 12:29:54
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1000 login tries (l:1/p:1000), ~63 tries per task
[DATA] attacking http-post-form://securityportal.tikt.no:80/login:user=^USER^&password=^PASS^:incorrect
[STATUS] 272.00 tries/min, 272 tries in 00:01h, 728 to do in 00:03h, 16 active
[STATUS] 117.67 tries/min, 353 tries in 00:03h, 660 to do in 00:06h, 3 active
[STATUS] 101.00 tries/min, 707 tries in 00:07h, 306 to do in 00:04h, 3 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-18 12:40:22
```

Figure 8.7: Output from first brute force attempt

To get a wider range of attempting to brute force, we did also try the unix passwordlist from Metasploit [91], which also gave 0 hits.

Code listing 8.6: Command for brute force with metasploit’s list

```
hydra -l admin -P
    /usr/share/wordlists/metasploit/unix_passwords.txt
    securityportal.tikt.no http-post-form
    "/login:user=^USER^&password=^PASS^:incorrect"
```



```
┌──(kali㉿kali)-[~]
└─$ hydra -l admin -P /usr/share/wordlists/metasploit/unix_passwords.txt securityportal.tikt.no http-post-form "/login:user
=^USER^&password=^PASS^:incorrect"
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or
for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-05-18 12:41:27
[DATA] max 16 tasks per 1 server, overall 16 tasks, 1009 login tries (l:1/p:1009), ~64 tries per task
[DATA] attacking http-post-form://securityportal.tikt.no:80/login:user=^USER^&password=^PASS^:incorrect
[STATUS] 290.00 tries/min, 290 tries in 00:01h, 719 to do in 00:03h, 16 active
[STATUS] 146.33 tries/min, 439 tries in 00:03h, 580 to do in 00:04h, 6 active
[STATUS] 136.00 tries/min, 952 tries in 00:07h, 67 to do in 00:01h, 6 active
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-05-18 12:48:52
```

Figure 8.8: Output from second brute force attempt

8.1.5 OWASP ZAP

To perform more advanced scanning, OWASP ZAP has been used, which is an open-source web scanner, that identifies potential vulnerabilities inside the application [92]. This tool do we also find inside Kali [93]. When starting an automated scan of our web link the following happened:

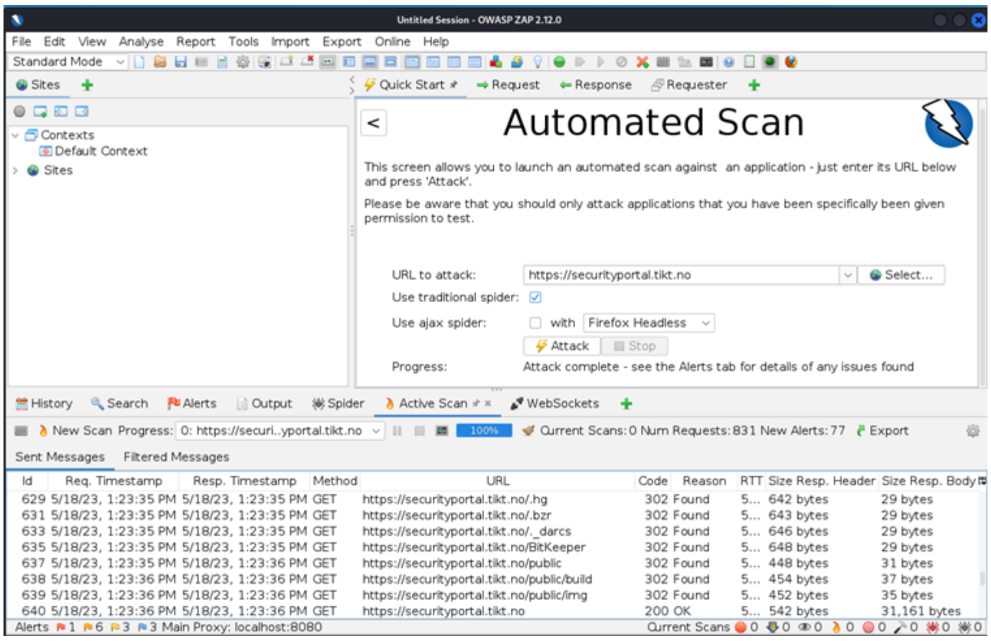
Target Discovery: Identifies domain which are going to be scanned. In this case do we enter it by our self.

Spidering: The tool manages to use spidering process to crawl through the application to check of the links and mapping. It crawls through what it finds, which is for to identify and map the pages.

Passive scanning: Goes through the requests and responses to look for security vulnerabilities in the background without sending any payloads, making it harder to detect.

Active scanning: Opposite of the passive. Crafted requests are being sent to the application and analyzes the response to identify vulnerabilities. It has a predefined set of checks that simulate various type of attacks and aims to discover weaknesses that require interaction with the application.

Generate a report: Gives the option to make a report to give a summary of scan and the findings. Our report of the scan is added as an attachment.



Automated Scan

This screen allows you to launch an automated scan against an application - just enter its URL below and press 'Attack'.

Please be aware that you should only attack applications that you have been specifically been given permission to test.

URL to attack: https://securityportal.tikt.no

Use traditional spider: ✓

Use ajax spider: ☐ with Firefox Headless

Progress: Attack complete - see the Alerts tab for details of any issues found

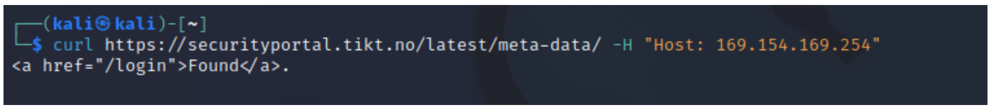
New Scan Progress: 0: https://securi..yportal.tikt.no 100% Current Scans: 0 Num Requests: 831 New Alerts: 77 Export

Id	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
629	5/18/23, 1:23:35 PM	5/18/23, 1:23:35 PM	GET	https://securityportal.tikt.no/.hg	302	Found	5...	642 bytes	29 bytes
631	5/18/23, 1:23:35 PM	5/18/23, 1:23:35 PM	GET	https://securityportal.tikt.no/.bzr	302	Found	5...	643 bytes	29 bytes
633	5/18/23, 1:23:35 PM	5/18/23, 1:23:35 PM	GET	https://securityportal.tikt.no/._darcs	302	Found	5...	646 bytes	29 bytes
635	5/18/23, 1:23:35 PM	5/18/23, 1:23:35 PM	GET	https://securityportal.tikt.no/BitKeeper	302	Found	5...	648 bytes	29 bytes
637	5/18/23, 1:23:35 PM	5/18/23, 1:23:36 PM	GET	https://securityportal.tikt.no/public	302	Found	5...	448 bytes	31 bytes
638	5/18/23, 1:23:36 PM	5/18/23, 1:23:36 PM	GET	https://securityportal.tikt.no/public/build	302	Found	5...	454 bytes	37 bytes
639	5/18/23, 1:23:36 PM	5/18/23, 1:23:36 PM	GET	https://securityportal.tikt.no/public/img	302	Found	5...	452 bytes	35 bytes
640	5/18/23, 1:23:36 PM	5/18/23, 1:23:36 PM	GET	https://securityportal.tikt.no	200	OK	5...	542 bytes	31,161 bytes

Alerts 1 6 3 3 Main Proxy: localhost:8080 Current Scans 0 0 0 0 0 0 0 0

Figure 8.9: OWASP ZAP

A short summary of the report is that it found a total of 13 alerts. Where they are divided into 4 levels informational, low, medium and high. We had only 1 high alert, and it was regarding cloud meta data. The description of the alerts says that the meta data maintained by cloud service could be reachable for attackers. By reading how to exploit this vulnerability on their documentation [94], and testing our self, we were not vulnerable (see Figure 8.10). As mentioned is the full report be attached as an Appendix F.



```
┌──(kali㉿kali)-[~]
└─$ curl https://securityportal.tikt.no/latest/meta-data/ -H "Host: 169.154.169.254"
<a href="/login">Found</a>.
```

Figure 8.10: An attempt to curl the meta data

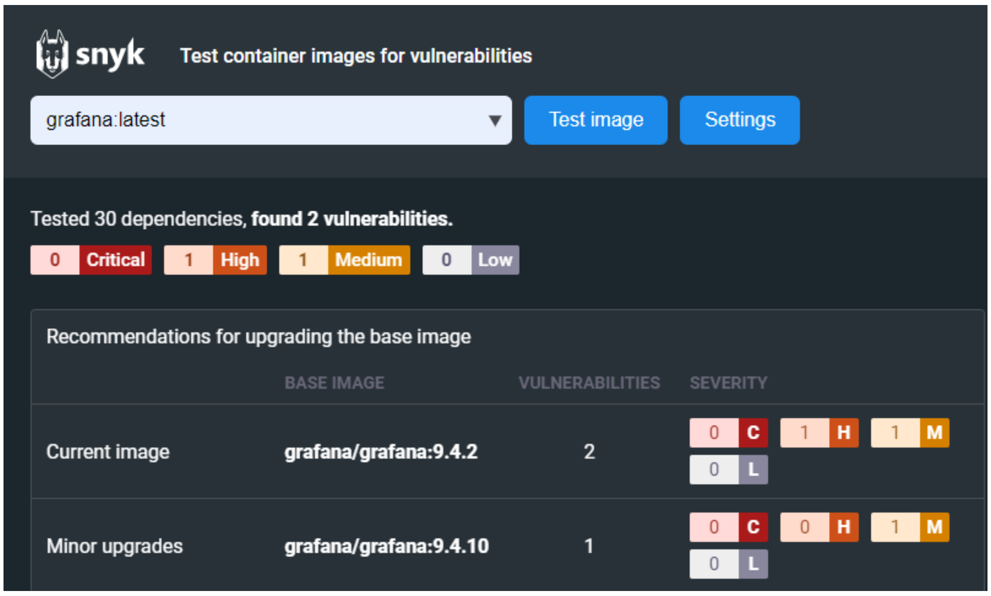
8.2 Container Images

8.2.1 Snyk

We also want to scan the Container Images. To do so a tool named Snyk was used. Snyk is an open-source platform that provides tools to help developers find vulnerabilities inside their code. Snyk first scans, and then analyzes the code. Later it identify known vulnerabilities and advices of how it can be patched. The tool can be added as extension in Docker desktop, which is a desktop version for Docker that allows you keep track of volumes, images and containers [95]. We have two images named api-cron and Grafana-init which is going to be scanned.

8.2.2 Grafana-init

As figure 8.11 displays below, were there two vulnerabilities found inside this image. Both of these vulnerabilities were regarding improper certificate validation in OpenSSL, which could lead to a Denial-of-service on affected systems [96]. Snyk reports that a minor upgrade would be to upgrade the base image from the current Grafana/Grafana:9.4.2 to Grafana/Grafana:9.4.10, which is a newer version of the base image.



grafana:latest

Tested 30 dependencies, **found 2 vulnerabilities.**

0 Critical | 1 High | 1 Medium | 0 Low

Recommendations for upgrading the base image

	BASE IMAGE	VULNERABILITIES	SEVERITY
Current image	**grafana/grafana:9.4.2**	2	0 C, 1 H, 1 M, 0 L
Minor upgrades	**grafana/grafana:9.4.10**	1	0 C, 0 H, 1 M, 0 L

Figure 8.11: Results of the scan of Grafana-init

8.2.3 Api-Cron

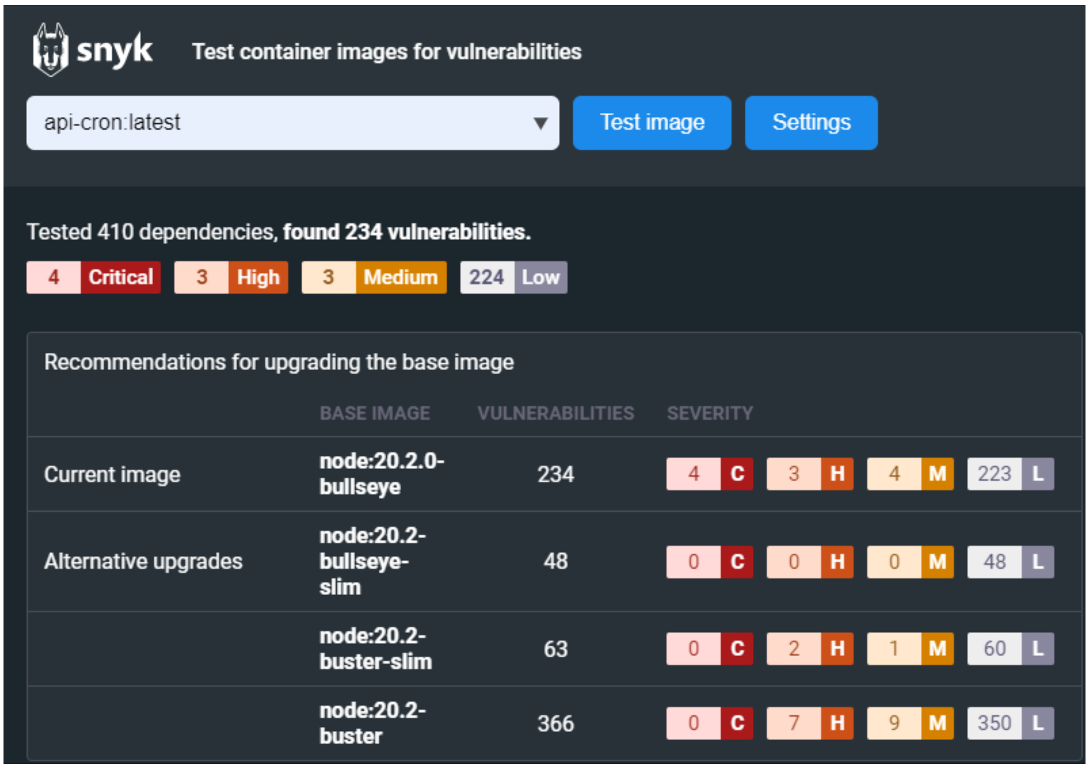
As Figure 8.12 displays below, did this image contain more vulnerabilities. It counted 243 vulnerabilities where 4 were categorized as critical and 3 as high. Multiple of those vulnerabilities are found inside the same library. The libraries that are repeated are aom/libaom0 and curl/libcurl3-gnutls. The following critical and high vulnerabilities were found inside aom/libaom0:

1. **Name: Release of Invalid Pointer or Reference**
   - **Vulnerability Level:** Critical
   - **Reference:** CVE-2021-30473
   - **Description:** This vulnerability describes memory that has not been allocated to the heap, which can allow attackers to cause memory-related problems [97].
2. **Name: Use After Free**
   - **Vulnerability Level:** Critical
   - **Reference:** CVE-2021-30474
   - **Description:** This vulnerability is a "use-after-free", which describes memory that has been freed, the software still references to it. Attackers who exploits this vulnerability can potenically cause memory-related problems and security breaches such as running arbitrary code execution or information disclosure [98].
3. **Name: Buffer Overflow**
   - **Vulnerability Level:** Critical
   - **Reference:** CVE-2021-30475
   - **Description:** This vulnerability is classified as a buffer overflow vulnerability, which could allows an attacker to write beyond the set limit of a buffer which could cause memory corruption or crashes [99].
4. **Name: Out-of-bounds Write**
   - **Vulnerability Level:** High
   - **Reference:** CVE-2020-0478
   - **Description:** This vulnerability is found inside the Android operating system. Attackers to obtain elevated privileges on affected system due the vulnerability allows for an out of bounds write because of a missing bounds check [100].

The following critical and high vulnerabilities were found inside curl/libcurl3-gnutls:

1. **Name: Cleartext Transmission of Sensitive Information**
   - **Vulnerability Level:** Critical
   - **Reference:** CVE-2023-23914
   - **Description:** This vulnerability is about clear text transmission of data classified as sensitive. An error that might occur is when multiple URLs are requested serially, and the HTTP Strict transportation also named HSTS might fail. HSTS is a function forcing the use of HTTPS instead of HTTP, so when the HSTS fails, data can be transferred over HTTP instead [101].
2. **Name: Cleartext Transmission of Sensitive Information**
   - **Vulnerability Level:** High
   - **Reference:** CVE-2022-42916
   - **Description:** This vulnerability is also about bypassing the HSTS check. The mechanism can be bypassed when a host named in the URL contains Internationalized Domain Name (IDN) characters that are replaced with ASCII during IDN conversion [102].
3. **Name: Cleartext Transmission of Sensitive Information**
   - **Vulnerability Level:** High
   - **Reference:** CVE-2022-43551
   - **Description:** This vulnerability is also about bypassing the HSTS check with the usage of IDN. The mechanism is being bypassed when the stored information is encoded in IDN, while the HSTS check are being done with decoded information allowing it to be bypassed [103].

An alternative upgrade that Snyk refers to is to upgrade the base image to node:20.2-bullseye-slim which contains "only" 48 vulnerabilities classified as low compared to the current solution which is node:20.2.0-bullseye. As the security testing was performed at the very end of the project, we simply did not have time to patch all of the vulnerabilities. We managed to patch the threats we found inside the web platform, however the vulnerabilities we found inside the containers was simply no time to patch and re-test.



Tested 410 dependencies, **found 234 vulnerabilities.**

4 Critical 3 High 3 Medium 224 Low

Recommendations for upgrading the base image

	BASE IMAGE	VULNERABILITIES	SEVERITY
Current image	**node:20.2.0-bullseye**	234	4 C 3 H 4 M 223 L
Alternative upgrades	**node:20.2-bullseye-slim**	48	0 C 0 H 0 M 48 L
	node:20.2-buster-slim	63	0 C 2 H 1 M 60 L
	node:20.2-buster	366	0 C 7 H 9 M 350 L

Figure 8.12: Results of the scan of api-cron

Chapter 9

Risk Assessment

9.1 Risk Assessment of New Implementation

One important part of the assignment from Tussa was to actively use risk assessment to secure the process around our work and the solution. This is also something they told us during the weekly meetings. Tussa wanted us to use a template they usually use when conducting a risk assessment. This template was not so easy to implement into latex, and therefore we had to split up the tables into different pages.

Following each risk there is a short description and an initial consequence and probability. The risk level of the consequence and probability also had to be justified in a short description.
We then came up with some measures that can help lower the risk levels of the different risks. These measures are specified in another page. Then the table moves over to the expected risk levels after measures, before the table ends with the current risk levels. Current meaning the risk level currently, and not effected by some of the measures that might not have been set into action yet.

The assessment of each risk builds upon the matrix seen below. This matrix divides the risk levels into:

High - High risk that can cause severe damage to both Tussa's systems and their reputation to their clients.

Serious - Serious risk that may cause damage to both Tussa's system and their reputation.

Medium - Medium risk that is notable and may cause damage in some way.

Low - Low risk that is insignificant and can be easily dealt with. The likelihood is low and the impact does not cause severe damage.

Probability

High	Medium	Seroius	High	High
Serious	Low	Medium	Seroius	High
Medium	Low	Low	Medium	Seroius
Low	Low	Low	Low	Medium
	Low	Medium	Serious	High

Consequence

Table 9.1: Table showing the different measures

Measure ID	**Description**	**Status**
1	Secure network (VPN or Proxy)	Planned
2	Private Github for sharing scripts	Established
3	Use a secure API manager	Established
4	Test everything with testdata before implementing	Established
5	Password manager for everything connected to the project	Suggestion
6	Enrolled into SOC overwatch	Suggestion
7	Induvidual API keys for every customer	Established
8	Continually testing scripts after updates and changes before implemented into vital networks	Established

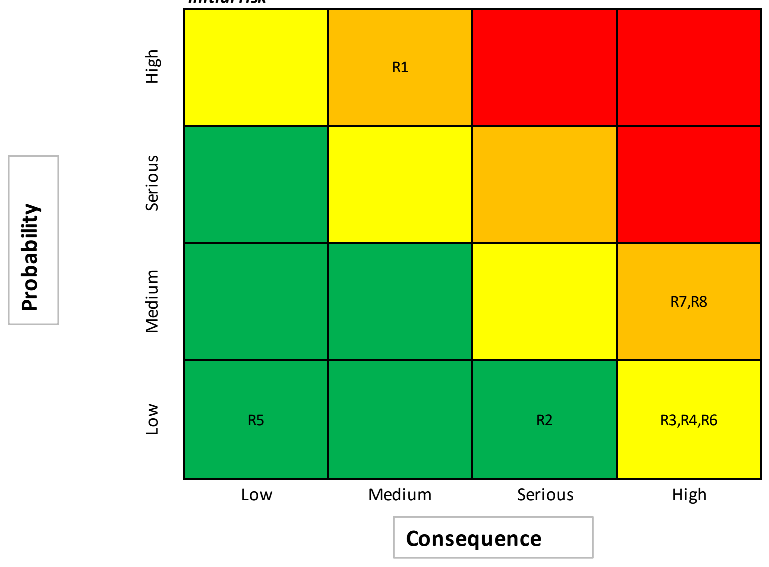
This is a table with the different measures we have conducted. The measures are in some way made to be able to tackle the different risks in the best way possible with the resources available. Most of the measures are established, but both SOC overwatch and password manager is just a suggestion. To use a secure network (VPN or Proxy) was initially planned but was not used.
As you can see on the matrix maps on page 100, we have made some of the risks go down from the higher risk levels. We used our knowledge from 3 years at NTNU to come up with good measures to handle the risks.

Assessment of new implementation			First assessment date: 07.03.2023			
ID	NR	**Risk description** Bullet points about: (1) Initial event(s) (2) Informationsecurity breach (3) The unwanted consequences that can happen	**Justification for consequence assessment**	**Initial consequence**	**Initial probability**	**Justification for assessment of probability**
R1	1	1. The students still have access to scripts and data after the delivery of the task 2. The students still possess valuable information regarding scripts and company data that can be exploited 3. Scripts and valuable data can end up in the hands of attackers and make it easy for them to see how the arcitechture is built	The probability that these students will use their information to exploit anything is very small. They have also signed a disclosure agreement.	Medium	High	It is very probable that the students will still have knowledge and access to scripts they built when they developed the system. Even though they wont have access to the live structured used by Tussa, they will still have information on how they made the scripts
R2	2	1. The students share information or scripts regarding the task on unsecure platforms 2. Information and scripts can be easily exploited from attackers if shared on unsecure platforms 3. Attackers gain access to vulnerable information that can be used to exploit Tussa and their customers	In a worst case scenario, the scripts and information can be exploited through the unsecure plaforms and used by attackers to gain critical information	Serious	Low	The students are cyber security students on their last year of a bachelor. They have good knowledge on how to store and share data and sensitive information in a secure way
R3	3	1. API credentials gets compromised 2. Attackers gain access to API credentials because of lack in security when storing the credentials 3. When attackers have access to API's, they can attack a service and it will be logged as if the developer were responsible. They can also be used to access sensitive information	In a worst case scenario, attackers can gain access to the API keys and then use that information to make API calls on your behaf, meaning it wil be logged as if the developer were responsible. The attacerks will also be able to access sensitive information. In addition can a adversary use this as leverage for extortion of money or other forms of goods from Tussa or there costumers.	High	Low	The students are using Microsoft Azure key vault to store credentials, which is a secure and trusted way to store and manage API keys.
R4	4	1. There is a fault in one of the scripts 2. Fault in scripts can be vital for the services delivered 3. Customer A can get information regarding customer B and vice versa. This is critical because each customer expects privacy regarding their own devices and services. Fault in scripts can also lead to that the services does not work propely, which is critical for the different customers.	A fault or error in one of the scripts would be vital for the security of each customer. If one customer got information regarding another customer, this would be a critical breach of the customers privacy and GDPR. This can also cause reputation damange for Tussa, which can lead to a sequence of consequences for the future of the company.	High	Low	The chances that there will be any fault in the scripts are very small. This is because the scripts are tested on testdata before being implemented into the network. Therefore we would find errors and faults before they are implemented to real data.
R5	5	1. Limited avaliability to application and downtime due to restricted resources 2. The application doesn't work because of the restricted resorces, meaning that customers wont be able to use the application to the full advantage 3. Customers won't be able to use the application the way they want, and worst case they expierience downtime	In a worst case scenario, the customer wont be able to use the application due to downtime. This can cause trouble because important information wont be available when desired. The security information can be retrived from the original source, i.e. the vendors however, but will take extra unnecessary time.	Low	Low	It is probable that the application won't be able to work properly, due to lack of resources, and properly managed infrastructure.
R6	6	1. Every customer share one API key 2. Makes it easier for an attacker to gain access to every customer 3. If an attacker gets hold of one API key, they have access to every customer and all their data. It only takes one security breach for all of the customer data to be compromised	If an attacker gets hold of one API key, they then have access to every customer Tussa got, which is 50+ customers. The attacker can then make API calls on your behalf and gain access to sensitive information. They can also use this sensitive information as leverage for extortion of money both from Tussa and their customers.	High	Low	Since we use Microsoft Azure Key Vault to store the credentials, it is not realy probable that there will be a breach. Azure Key Vault is a trusted and secure way to store credentials, and is used by big companies all over the world.
R7	7	1. Multi tenant is configured wrong 2. Fault in configuration can breach the GDPR and confidential agreements Tussa has with their customers 3. Configuration fault can end up giving one customer information or even access to other customers data	If one customers gets hold of another customers data, we would have a serious GDPR breach, and the agreements between Tussa and their customers would be broken. Therefore the consequences from a fault in the Multi tenant service is vital for the customers.	High	Medium	Multi tenant can be a complex to set up and run. It is important to use multiple tests before implementing the solution with live customer data.
R8	8	1. Multi tenant is configured wrong 2. Fault in configuration can breach the GDPR and confidential agreements Tussa has with their customers 3. Configuration fault can end up giving one customer information or even access to other customers data	If one customers gets hold of another customers data, we would have a serious GDPR breach, and the agreements between Tussa and their customers would be broken. Therefore the consequences from a fault in the Multi tenant service is vital for the customers.	High	Medium	Multi tenant can be a complex to set up and run. It is important to use multiple tests before implementing the solution with live customer data.

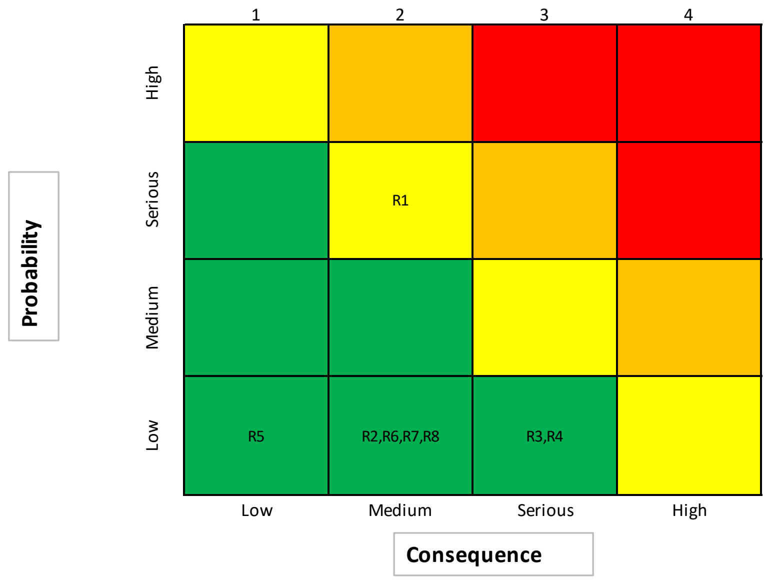
Updated date: 08.05.2023

ID	Initial risklevel	Handling - specified on new page: Measures	Expected consequence after measures	Expected probability after measures	Planed risklevel	Risik responsible	Current consequence	Current probability	Current risk
R1	Seroius	5	Medium	Serious	Medium		Medium	Serious	Medium
R2	Low	1, 2	Medium	Low	Low		Medium	Low	Low
R3	Medium	3, 6	Serious	Low	Low		High	Low	Medium
R4	Medium	4, 8	Serious	Low	Low		Serious	Low	Low
R5	Low	4, 8	Low	Low	Low		Low	Low	Low
R6	Medium	6, 7	Medium	Low	Low		High	Low	Medium
R7	Seroius	4, 7, 8	Medium	Low	Low		Medium	Low	Low
R8	Seroius	4, 7, 8	Medium	Low	Low		Medium	Low	Low

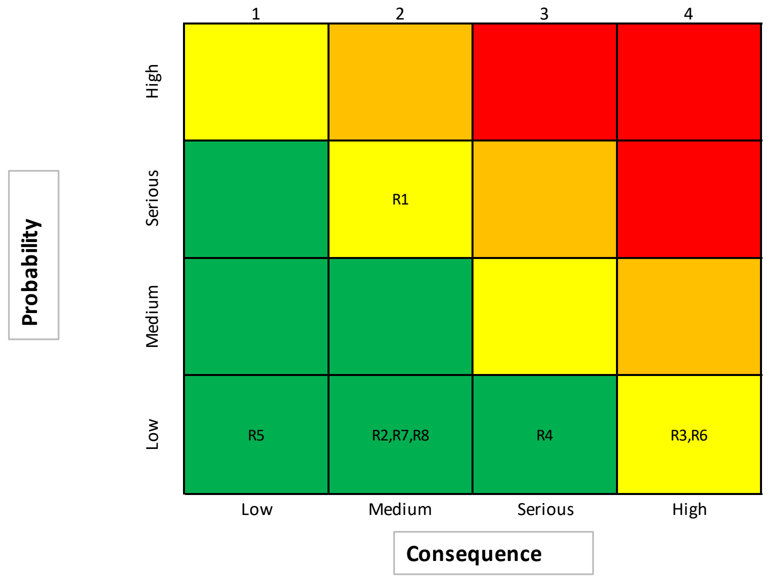
Initial risk



Planned risk



Current risk



Chapter 10

Evaluation

In this chapter, we discuss and walk through the survey and its results. Finally, we evaluate our performance regarding requirements that was defined in the requirements-chapter. This chapter focus on presenting an evaluation of our effort considering this project.

10.1 Survey

As a part of the project, and a quality control, we conducted a survey to the client to assess the outcome of the project as a whole, and the product it self. We sent the survey in our common Microsoft Teams channel with our client, and told the client to distribute the survey to colleagues who were familiar with the project. The survey included a set of questions to help us evaluate the outcome of the project, and the value of the product it self. The goal is to see if we met the clients requirements and if the product we made has been helpful, and something they want set in production.

10.1.1 Results From The Survey

Throughout the project, we have mainly been in contact with two persons, which also resulted in a corresponding number of responses on our survey. It is a little unfortunate, but also understandable as it is most likely those from the client side who has designed the task who participated in the survey. The results of the survey showed that the customer is very satisfied.

The first question we asked was how satisfied the client are with the final outcome of the project. One user reporting very satisfied, and another reporting somewhat satisfied with the following comment: "Looking for some specific recommendations on risk mitigation when using APIs, to improve our procedures and developer training". When receiving this, we took action to add specific recommendations written in the deployment-chapter.

The next questions we asked was if the solution would be usable in the client's production environment, and if they knew any challenges regarding putting in it in production. The following question was if the solution would need any user training when it is in production. Based on the answers, did One of user report that this be usable in production, and another is unsure. The same patterns follows for if user training is needed. One of the comments was "Don't know of any big challenges, of course we must do some adaption work", which means there are possibilities for taking the solution in action. To help with the user training did we create a user guide the final work section in the deployment-chapter.

The last questions that we asked was if the client had any feedback regarding what we could improve, if we met their initial goal and requirements they sat for the project and for last and if they any last comments. As there were no comments for improvements, and we met their goal and requirements does it seems that they are satisfied with the current solution and the project as a whole. The last comment was the following: "It seems to me that the students are skilled, well organized and hard-working.", which is also a satisfying comment to receive from the client. For a more visual version, see Appendix G.

10.2 Evaluation Of Requirements

In the requirement-chapter, we sat some requirements, and in this section do we evaluate our application to see if we managed to meet the requirements sat.

10.2.1 Functional Requirements

F-1: "***The solution should contain access control to ensure confidentiality.***"

Description: To ensure confidentiality for each of the clients, we need to use access control. Access control is used to ensure that users only get the information they need, and to ensure that no one access information without the proper authorization.

Evaluation: To ensure confidentiality for each of the clients have we implemented Azure Groups to ensure confidentiality and access control. We have also enabled several security mechanisms such as Nginx and MFA to ensure the confidentiality. Our solution satisfies the current requirements. By dividing the client's subcustomers in Groups, ensuring the logs that are being displayed only is available for the respective group. Another featured that was added to ensure the confidentiality was to enable MFA for authentication, minimizing the chances of unwanted access.

F-2: "***The solution should be able to fetch information about devices from API calls.***"

Description: One important aspect of the task is to be able to fetch information from each clients devices. This information needs to be fetched from all three products (Microsoft Intune, Cisco Secure Endpoint and Cisco Umbrella) and display all the important information in one place.

Evaluation: The solution fetches the relevant information from the products with API calls. The Node.js application fetches information with API calls and sends that information to Elasticsearch. Grafana then pulls the information from Elasticsearch for visualization. One API call is used per customer to retrieve data. The API calls fetches a lot of information, and the interface can be changed to display what is relevant for the customer. Therefore the solution satisfies the current requirement.

F-3: "***The solutions needs to use encryption to secure information.***"

Description: Another important aspect of the task is to deliver a solution that uses encryption in a secure way. SSL is a great way to secure the internet connections.

Evaluation: The internet connection coming from the client to Nginx is encrypted with SSL which gives the clients and information some security when connecting. Another valuable information that is encrypted is both the API credentials and the admin passwords in Azure.

F-4: "***The solutions needs to store the information for the devices.***"

Description: The data from the different devices needs to be stored in some way, so the clients have access to their data, and not just in real time. This can be done with something like Elasticsearch.

Evaluation: Partially. Device information are being collected from different vendors and acquire a persistent data solution before visualized in Grafana which replaces the current data inside Grafana. This means that the old data does not get stored, but only poses the newest data available. This is how the client wanted it. As this does not quite meet the requirement we sat, does it meets the requirement from Tussa.

F-5: "***The solutions needs to securely store API credentials and passwords.***"

Description: The API credentials and passwords needs to be stored securely to ensure confidentiality and as a security measure. This can be done through a service like Azure Key Vault.

Evaluation: The solution uses Azure Key Vault to store the API credentials, which is a trusted and reliable way to store them. Also the admin passwords are stored in Azure Key Vault. This means that the solution satisfies the current requirement.

10.2.2 Non-Functional Requirements

N-1: ***The solution should be easy to use with minimal or no training.***"

Description: The solution should be so easy to use that it requires minimal or no training for the users. Therefore its important to make a application that is intuitive.

Evaluation: The solution is easy to use and requires no directly training to use. To be able to understand how everything works, you might need some form of instructions or training, but overall the solution meets the current requirement.

N-2: "***The solution should be easy to understand and interactive.***"

Description: The interface should be easy to understand for the customers and Tussa, so they can use as little time possible on understanding the design.

Evaluation: The solution is easy to understand and requires very little technology background. Therefore the current requirement is fulfilled.

Chapter 11

Discussion

This chapter aims to describe the aspects of the project as a whole, walk through what challenges and limitations we encountered along the project and what choices we had to make based on this. We further discuss future consideration Tussa may consider which we mean is going to enhance and evolve the product. Finally do we address the possibility of being presented with the opportunity to redo the project, and what we could have done differently?

11.1 Limitations/Challenges

Challenging decisions done during our development face brought some limitations to our application. This section aims to explain these limitations, as well as discussing what could have been handled differently. We present limitations around our chosen Grafana solution and an alternative we viewed as relevant at a certain point. We also discuss restriction our Nginx service poses. In addition do we address limitations regarding managing access control for the VMs. Finally do we debate our interpretation of the assignment and its correlating limitations.

11.1.1 Grafana

As mentioned previously does Grafana integrate well with Azure AD. Our intention was to take advantage of Grafana's built in Azure AD Authentication to authenticate users and let their Azure groups automatically be integrated in Grafana. Unfortunately this feature is only offered by Grafana Enterprise which is a paid version of Grafana. The feature automatically synchronize teams i.e. groups in Azure AD is also going be teams in Grafana, which is exactly the capability required by our service.

This solutions comes with great management utility as users are created by first time they sign-in and are delegated to their respective teams also named tenant. Additionally combining this with pre-configured data sources and dashboards utilizing Grafana APIs does this streamline the onboarding process. It also ensure dynamic user management and scaling. This can help Tussa in the potential scenario if they expands their business.

Anticipating Tussas potential hesitation to bear additional cost of upgrading to Enterprise. We conceptualized alternative solutions to work around this issue to present Tussa.

We began the process of developing an individual Node.js authentication service that where to be hosted along side Nginx at the same VM. The idea was to utilize Microsoft Azure AD Passport.js Plug-in as a Middleware to handle the authentication since Passport.js is designed to be a highly flexible and modular tool [104]. Azure provides an authentication token upon successful authentication.

The token is exchanged by the Node.js application for an ID Token and access token. The It Token contains information about the Azure user [105]. Appropriate configurations in Grafana can be made by leveraging this information. The application could utilize Grafana APIs to manage their Permissions. Additionally can users be assigned to their respective teams. This streamline the process of integrating Azure groups with Grafana teams.

Another feature passport.js poses is their approach to handle sessions data. Cookies are normally this data, making it vulnerable to Cross-site scripting (XSS) [106]. Passport.js store session data on the server side in session and not in the browser, reducing the probability of potential adversaries levering this vulnerability [107].

An alternative solution was to configure Grafana completely through APIs, disregarding the team functionality Grafana Enterprise offers. This method requires to pre-configure every element to meet our desired functionalities when the service is launched. A consequence is limiting Grafana's dynamism, as the service needs to be relaunched when a new user or group is added in Azure.

After spending two weeks developing and individual Nods.js authentication with slow progress and confusing errors. Errors were related handling session information interplay with Nginx, as all traffic where to go through Nginx. We consulted our advisor with how to proceed, knowing the deadline was closing up. We talked about these three paths to our solutions where we concluded to present them to Tussa and tell them about their up and down side. Letting them make a decision about it, instead of us. We highlighted that Grafana would poses an additional cost. Prioritising an individual Node.js authentication would take additional time at the cost of other security measurements, where there were some uncertainty if we would be able to finish it. Finally pre-configure Grafana, having dynamic limitations.

The meeting with Tussa concluded in proceeding with a pre-configured Grafana, eliminating Grafana enterprise and passport.js. Going with this presented more benefits, as it freed up time to develop other security features with a lower cost.

Reflecting on this experience made it clear for everyone in the group, that communication at an early stage is vital in a time sensitive project. It also shows how crucial it is to be clear about a decisions before engaging in development. Accomplishing this properly, result in improved efficiency, less confusion and miss delegation of resources.

11.1.2 Nginx

This section describe why Nginx open-source (free version) was chosen instead of the paid Nginx plus version and its limitations that follows by using the free version. Tussa stated they wanted DDOS protection, which Nginx provides several features to improve protection against. This was also descried earlier in the implementation chapter.

The feature `GeoIP2 dynamic module` in Nginx restricts access to a certain geographical location, as an additional method to mitigate DDOS threats. This feature is only available in Nginx plus version [108]. By presenting this for Tussa with this capability and the staggering price of $2500 a year, they concluded that the cost outweighed the benefit [109].

Close to the end of the development process did we plan to make the application more scalable. We thought of creating more replicas of the Grafana-image to improve performance and redundancy. If we were to scale Grafana Horizontal, Nginx needed to work as a load balancer which is an embedded feature. Nevertheless, we stumbled upon some problems achieving this.

After a user successfully authenticate they get redirected back to Grafana through Nginx. For Nginx to know which Grafana container that originally initiated the authentication and to send the user back to the same container, a Sticky Session needs to be enabled. This is needed due to Grafana is stateful and not Stateless, meaning the state of the session is stored on the container. Sticky session, also know as session affinity and session persistence are features only available for Nginx plus subscription.

Since we already had disregarded Nginx plus as an option, a different solution needed to be developed to work around this limitation. Having a centralised persistence solution to store the state of Grafana, making it Stateless could solve this issue. By the time we figured this out, sufficient amount of time to develop this was not to our disposal. This is therefore further descried in the next section regarding future considerations.

11.1.3 Virtual Machine Access

The group deemed it necessary to handle access to each VM different then the current traditional way. The conventional method of assigning each user with their own unique key par, can potentially entail some challenges. As the required number of peoples access is increased, so does the complexity of managing individual key pairs. This not only makes it cumbersome to handle access management. It also poses a security risk if overview of access control is impaired.

In the scenario where user access should be revoked, its manually required to remove their respective public key from each virtual machine. This method leaves an absent of a centralized control mechanism for managing access to VMs. A certificate authority (CA) can address the centralized limitation associated with traditional key pair. It issues and revoke signed certificates to enable SSH access. It stream lines the managing process as it does not need every public key configured on each VM.

Furthermore can a certificate be implemented with an expiry date, ensuring that access is revoked automatically after a certain period of time. This feature helps mitigate the problem of stolen or lost keys [110]. Such solution were attempted to be constructed, but did not succeed before the development face was due.

Discussing this limitation in the aftermath of the development face with the client, we concluded that a their solution would suit a bastions server to play the role as a centralised and secure way of managing user access to VMs. Bastion enables RDP and SSH directly in the Azure portal eliminating the necessity for a key pair. The session is secured using TLS on port 443. It also dismantle the need of a public IP-address, making it more secure.

Additional information about the extended features bastion provide like protection against Zero-Day exploits can be found on: [111]. Since bastion has an expensive operational cost of 3.081 kr per hour, only activating the service when necessary to optimize financial expenses were desired by Tussa [112].

11.2 Assignment Interpretation

This section presents our retrospective interpretation of the assignment. Our assignment introduced diverse and complex security requirements for enhancing procedures for SaaS services with a particular focus on risk assessment and secure API management. The assignment presented a vast majority of security components to meet the specific details of the project. Each component has their own comprehensive and unique entanglement, meaning pro founded knowledge is necessary to utilise their full capacity. Given the comprehensive nature of this assignment we felt that we lacked the time to explore the entirety of each topic in its complete aspect.

Our philosophy is that there is always room for improvement. Recognizing that the security landscape is dynamic and in constant change. Due to these factors have we prioritised to follow best practice and widely adopted solutions over specialized approaches. This has lead to some limitations along the way such as the security testing revealed. Our application poses several security flaws that would take extensive time, resources and knowledge to mitigate. some of these threats where discovered during the development process revealing the importance of keeping up to date with pro founding knowledge about our technologies.

11.3 Future Considerations

11.3.1 Grafana Scalability

As mentioned previously is Grafana configured as a stateful application, presenting some challenges if we were to replicate Grafana. Specially when a user gets redirected back to a separate Grafana instance after a successful authentication. There are multiple techniques of changing Grafana to a stateless architecture, solving this issue. A stateless solution allows for individual request handling, making them more scalable and robust. One approach could be to create an external database where the state is stored. This would allow Grafana to maintain the state externally making it stateless.

Another strategy to accomplish this is to utilize Redis. Redis is cache mechanism known for its utility to be low latency, due to its persistence solution, storing information in memory. A possible draw back from this, is the significant consummation of RAM it absorbs. This naturally depends on how many sessions it needs to handle and store at the same time. An attempt was made to utilize Redis for this purpose, and Redis was chosen due to its specialized capabilities and efficiency.

Redis is also a highly relevant and frequently used tool in to days real worlds marked [113]. It solves the state limitation issue without further cost since its open-source, compared to Nginx plus. One of the group members were familiar with the service from a previous unit which also helped persuade the decision. Unfortunately in this case as well, was there not sufficient amount of time to finish developing this and integrate it with our current solution for it to work. Nevertheless do we highly recommend considering this for future implementations.

11.3.2 Azure Virtual Network

While we utilise an overlay network in docker for node communication, We only utilize the default Vlan virtual network created in Azure for our VMs. Applying a better virtual network strategy in Azure, could resolve in numerous benefits. Placing Nginx in a DMZ facing the internet could help protecting the microservices located outside of it. If the Server is comprised can a potential adversary not gain direct access to whats outside of the DMZ.

Segregating the network into separate Vlans could also increase the security. It is an additional security measurement that creates a clear separation between services. it has the same functionality as the DMZ i.e. to not gain access of different parts of a network if one is compromised. creating clear separations in our network is fundamental principal of achieving defence in depth. It would help protect against lateral moment in case of a security breach as it is more challenging to move across the network.

This improvement was brought to light upon reviewing our solution and was discussed internally in the group. Each member agreed upon the value these configurations brings, but also acknowledges the limited time to fully research these implementations. We therefore recommend future investigation on how this can be executed to add the most value.

11.3.3 Load Balancer / Firewall

We disregarded to possibility of adding an Azures application Gateway with Web application firewall (WAF) in front of Nginx during our development face, as a additional security measurements. This oversight heavily relied on keeping the cost within the original budget given by Tussa, as each services exceeded the initial financial plan of their own. This causing us to overlook the possibilities of implementing these payed services. By looking back we realised that we missed out on this opportunity. Tussa expressed excitement towards this solution and were willing to bear the additional cost after this proposal were presented. However was the development face concluded, when this was brought to light, making it to late to implement.

The benefits of these two services bring for a potential future implementation. Application gateway is a web traffic load balancer that operates at the application level of the OSI model i.e. layer 7. As normal load balancers operate on the transport layer i.e. layer 4. This makes it do decisions based on additional attributes of an HTTP request, making it capable of URL-based routing [114]. Azure application gateway v2 also provide WAF capabilities. It comes with autoscaling which can automate resource utilize by demand, ensuring Tussa only pay for resources used. It also makes it highly flexible as it dynamically matches the demand of the website. It also enhances the protection against common web exploits and vulnerabilities. Additional information can be found on Microsoft web site [115].

11.3.4 Securing Data At Transit

Securing Data At Transit was a part of the assignment which was not fulfilled to its full potential. Data Moving between Node.js, Elasticsearch and Grafana are not encrypted. This requirement was neglected throughout the developing process. As the assignment states " Outline a solution that secures the data during transit". This does not mean that everyone can access the data by any mean. The Traffic takes place with in a closed environment as it runs inside a single VM on a designated docker network.

We have implemented security features such as SSH which means they need the right set of key pairs to be able to access. Access to the manager VM is need to even attempt to sniff out the communication flow between the microservices, to fetch the sensitive data. As we see the current solution to be sufficient, is layered security always be recommended. Enabling encryption for data in transit also follows the best practise principal.

With this statement do we mean that having more layers of security gives a wider depth of protection against several types of threats and attacks. Multiple layers of security can lower the chances of multiple breaches, if a breach first would occur. This means that if an unauthorized actor first gains foothold on the manager VM, extended measures would be needed to be able to capture the traffic in transit.

11.3.5 API Gateway

Open source version of Nginx offers a capability called API gateway. This capability was involved in deciding to use Nginx. However did we not poses adequate time to develop it. We prioritised enabling HTTPS to our domain instead. This decision was made because our APIs already constitute the capability of encrypted traffic. We recommend implementing an API Gateway in the feature as it offers several advantages. It simplify management by creating a centralized point of entry for all APIs. This unified entry can be configured to consist of several security measurements like IP filtering, rate limit and authentication checks, instead of each API is configured with their own specifications. It can also enable scalability for the Node.js container as it can work as a load balancer.

Playing the devils advocate could this implementation also offer some negative effects. Latency cloud be increased is the network traffic gains an additional hop. We view this as acceptable since reducing information from APIs is not time sensitive, and is scheduled to be done at nine in the evening. It also introduce a single point of failure if the Nginx instance were to fail.

11.3.6 VM Resources

Its hard to predict the actual VM resources needed in a production environment as the virtual machines only have been tested with fake data. When launching the VM with real data we recommend doing a thorough testing and monitoring of hardware used to optimize resource utilisation. Doing this properly ensure efficient use of resources as well as cost. The advisor recommendation feature in Azure can help with this process. The feature gives advice on how to optimize the VM [116].

11.3.7 Azure AD Authentication Integration With GitHub

We recommend looking at the possibilities to integrate Azure AD authentication with GitHub to allow access control based on groups and roles in Azure. This ensure a secure and easy way of managing access to the source code and scripts. These users is also required to pass a MFA to gain access, this is done to improve the security. Having GitHub integrated with Azure AD authentication allows for a centralised place to manage access control, and giving better overview over the access list.

Allowing people to already use their existing Azure AD account for authentication towards GitHub, eliminates the need to create an additional GitHub account where more credentials needs to be handled. Having Azure Ad integrated automates the process of allowing already configured groups in Azure to be created in GitHub and let their access be controlled based on that. Azure also enables a centralised place to view login activity regarding the application.

11.3.8 Security Tests And Patching Before Putting It In Production

As mentioned in security testing-chapter, did we conduct our testing at the very end of the project due to time spent on developing, which caused too little time for patching and re-testing. Based on the results from our scan of the web application did we find only a few leakages that we were able to patch, but regarding the container images did it turn out to be multiple vulnerabilities regarding one of the images which should be patched before putting in in production. As we also have less experience with deep security testing do we recommend to conduct more scanning of both web application and the current images and patch the vulnerabilities making it more secure before putting it out in a production environment. This is going to make the entire platform more secure and minimize the risk of unauthorized access or information leakage.

11.4 Why We Developed This Application

When we initially got the task from Tussa we soon found out that the ground principles of the solution was fairly easy to develop. Making an application that collects the data from three different products into one interface was not the biggest task in the world. When we dug deeper into the proof of concept, we soon found out that the hardest part of the task was to implement all the security measures. The first stage of the solution is an application that does the basics, but is misses some key security and automation features. These features were later implemented into the solution. Some of these features is no hardcoded credentials and groups. Images are saved in the cloud, and Nginx is used as a reverse proxy. These are all features making the solution more robust and secure against possible attackers.

For Tussa a solution like this can help them optimize their workflow and lower time used making reports for customers. The application also give them a more professional look to their customers, which is a important aspect in maintaining them as customers and possibly gain new ones. As mentioned earlier in the thesis, there is also an important aspect that Tussa gets better overview of compliant computers. The computers that are not compliant can also be taken out of the system. All in all this is a great solution which optimize the time usage and security handling of customer information.

11.5 If we were to redo the project, what would we have done differently?

The review of the chapter has so far been linked to challenges, limitations and future considerations, but not what we would have done differently if we were going to conduct the project again. This section aims to discuss and reflect what we would a done differently. The topics we want to cover are development, security testing, planning and reviewing the task.

11.5.1 Development

The development phase has been a steep learning curve with a lot of new material to learn in a short amount of time. Knowing which tools and services to use and how to utilize them was very challenging, but still rewarding and enjoyable to see how they functioned in practice. We managed to develop a solution that met our requirements, but the journey was not easy.

In terms of what we would have done differently, is to gather more fundamental information before attempting to develop it in the solution. When testing with different solutions did we face several limitations which caused the either significant amount of time was lost in attempting to develop something that did not fit or function properly according to our solution. We would have wanted to sketch out the desired final version of the solution before starting the development process.

11.5.2 Security Testing

The security testing was conducted at the very end of the project, which resulted in us having slightly less time than we planned to complete the testing and patch the vulnerabilities with re-testing. Upon discovering numerous vulnerabilities associated with one of the container images, we would have preferred to patch as many of them as possible before delivering the code to the client.

In terms of what we would have done differently, would we likely have initiated the testing phase and related work a bit earlier. This would probably have allowed us to allocate more time to identify vulnerabilities and patch them, ultimately making the solution more secure. This phase does heavily rely on being happy with one of the current solution allowing it to be tested at a earlier stage.

11.5.3 Reviewing Of The Task

Throughout our weekly meetings with our advisor, client, and internally, there was a strong emphasis on the development aspect. As a result, a significant amount of time was dedicated to developing a solution that met the client's needs, which resulted in overshadowing the theoretical aspects defined in the assignment. This led to an interpretation of the task where a significant emphasis was placed on the overall solution and its associated mechanisms.

In terms of what we would have done differently, we should have allocated more time regularly to read through and understand the task better as a part of its heavily focused on researching solutions that are not a part of the development process. We managed to do so, and based survey described in the evaluation-chapter did it seem that the client was satisfied. As a retrospective assessment, we should have discussed the theoretical aspects of the task in a earlier stage.

11.5.4 Planning

We sat a plan early in the project on how we want to work throughout the project. As we learned in the Military, you got to have a plan and be able to adapt the plan to the current circumstances. We made our best effort to adhere to our plan, but we realized early on, that changes had to be made along the way. That is why we divided the work into sprints to assess what worked and did not work at the end of each sprint.

In terms of what we would have done differently is to be aware of the amount of work each part requires, before planning. It was very easy to underestimate the amount of work actually required, which led to incorrect time estimations such as the entire development phase. We learned along the way that unfortunately, we could not implement everything we wanted, and this was something we had to accept.

Chapter 12

Closing Remarks

12.1 Learning outcome

In this section, we are going to walk through the Learning Outcome during the project. It is going to address the project as a hole. Additionally, our communication and work process is shortly summarized afterwords. Finally, we present a highlight about writing the thesis.

12.1.1 Project As A Whole

Throughout the project we have managed to develop our experience and knowledge of project work as a team. We have received a better understanding of the importance of planning, prioritizing, and communication between all participants inside the project. We have also gained a better understanding of how we got to limit our scope to manage to finish what we plan, even though we naturally wish for additional time to develop the product.

12.1.2 Teamwork, Communication and Working process

As a group, we are satisfied with how the project has been conduct from the beginning to the final result. As mentioned in the development process-chapter, the group already got to know each other in the early stages of education for the bachelor's degree, as well as collaborating in several projects throughout the education. Magnus and Øivind also share an apartment, meaning there was a short way to reach each other when necessary.

We constructed a plan early in the project on how desired communication was to be conducted. After each iteration, we gathered to discuss what went well and what could be improved. Through out the project there were an issue stated regarding miscommunication. It was a mutual acknowledgement that each individual lacked an overview about their piers work and what they where doing. A dedicated text channel in Discord was implemented to address this subject. Is was an important learning outcome, as it demonstrated our ability to adapt to an unforeseen challenge in the middle of the project and thrive with a solution that worked well.

As mentioned above did we divided the length of the project into iterations. After each iteration we did walk through the iteration addressing what we think went well and what we could improve for future iterations. By doing this, did we achieve a continuous adoption to unexpected topic, as well as boosting moral by highlighting what we did well.

12.1.3 Writing The Thesis

None of the group members had prior experience and knowledge working with Overleaf. Our advisor recommended and challenged us to proceed with this tool for writing the thesis. We deemed it as a quite different from office as we normally use, which lead to some time being spent on getting comfortable with it. After making use of NTNU's template which contained a fairly good explanation on how to write this report, did we sense accomplishment towards this challenge.

12.1.4 Developing

During the development face we learned to utilise several services we had no prior experience with. It was challenging for us to figure out where to begin and how. ChatGPT as a searching tool combined with the discussion with our advisor played a central part to get specific answers to our question of where to begin. Consulting with our advisor in combination with ChatGPT was a decisive factor in the early stages of development as well throughout the project where we came across comprehensive decision on which path to take. When we had things up and running, we could start customizing it to the customer's needs.

12.2 Future Considerations

Although we have created a solid service that meets the expectation of our client, there are several improvements we see fit to enhance and optimize the result derived from our solution. The bullet points below describe a summary of what has been discussed in the discussion-chapter.

- Make Grafana scalable, enabling it to handle different load.
- Enhance Azure virtual Network
- Consider implement an application gateway with WAF to create autoscaling and protection against know web vulnerabilities and exploits.
- Implement a bastion in Azure, to improve the access management to VMs as well as security.
- Encryption during transit internally
- API Gateway
- Fine tune VM resources
- Azure AD Authentication Integration With GitHub
- Security Tests And Patching Before Putting It In Production

12.3 Conclusion

In conclusion, the bachelor group has committed effort into achieving their sat goals and requirements stated in the beginning of the report. The group has managed to fulfil their goal upon creating a sufficient and secure reporting system through a SaaS-Service for Tussa's Customers. The service updates and informs clients about critical device status utilizing API integration. Adhering best practice for guidelines and development as been spotlighted, attempting to withstand potential cyber threats. Constructing a risk assessments also provides enhanced focus on mitigating security risks, in addition to improving the overall security awareness.

Ultimately, by delivering updated reports from multiple sources do we greatly help Tussa in achieving efficient and reliable documentation and reporting to their clients. Furthermore, the group recognize improvements necessary to the product, to enhance scalability, reliability and security. Implementing these considerations can improve a long term success of delivering consistent information to Tussa's consumers with amplified security.

12.4 Final words

In retrospective of the project, we recognize a steep learning curve offering invaluable knowledge to be particularly rewarding. We feel we have gained remarkable and relevant experience in team work, development and security testing in a larger project. We deem the competence and experience acquired during this journey to be highly appropriate for further studies and occupations. As final words for this bachelor thesis do we truly hope this project can be useful for our client and as a whole do we wish to express our comprehensive appreciation towards all who have helped us accomplish this thesis.

Bibliography

[1] Standard. 'Ledelsessystemer for informasjonssikkerhet – iso/iec 27001.' (), [Online]. Available: https://www.standard.no/fagomrader/ikt/it-sikkerhet/isoiec-27001/. (accessed: 18.05.2023).

[2] IBM. 'What is an api?' (), [Online]. Available: https://www.ibm.com/topics/api. (accessed: 18.05.2023).

[3] P. A. Networks. 'What is the principle of least privilege?' (), [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-the-principle-of-least-privilege. (accessed: 05.05.2023).

[4] Harvard. 'Risk management audit services.' (), [Online]. Available: https://rmas.fad.harvard.edu/faq/what-does-information-systems-audit-entail. (accessed: 13.05.2023).

[5] B. Figureau. 'Logging best practices.' (), [Online]. Available: https://www.dataset.com/blog/the-10-commandments-of-logging/. (accessed: 13.05.2023).

[6] Docker. 'Use containers to build, share and run your applications.' (), [Online]. Available: https://www.docker.com/resources/what-container/. (accessed: 04.04.2023).

[7] Kubernetes. 'Kubernetes.' (), [Online]. Available: https://kubernetes.io/. (accessed: 09.04.2023).

[8] Docker. 'Swarm mode overview.' (), [Online]. Available: https://nodejs.org/en/about. (accessed: 08.05.2023).

[9] Z. Hira. 'Kubernetes.' (12th Jul. 2022), [Online]. Available: https://www.freecodecamp.org/news/kubernetes-vs-docker-swarm-what-is-the-difference/. (accessed: 09.04.2023).

[10] NTNU. 'Dcsg2003 - robuste og skalerbare tjenester.' (), [Online]. Available: https://www.ntnu.no/studier/emner/DCSG2003/2021#tab=omEmnet/. (accessed: 01.04.2023).

[11] D. Docs. 'Manage sensitive data with docker secrets.' (), [Online]. Available: https://docs.docker.com/engine/swarm/secrets/. (accessed: 04.04.2023).

[12] P. A. Netowrks. 'What is azure?' (), [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/. (accessed: 01.04.2023).

[13] Microsoft. 'About azure key vault.' (1st Mar. 2023), [Online]. Available: https://learn.microsoft.com/en-us/azure/key-vault/general/overview. (accessed: 03.04.2023).

[14] Microsoft. 'Azure container registry.' (), [Online]. Available: https://azure.microsoft.com/en-us/products/container-registry. (accessed: 05.04.2023).

[15] Microsoft. 'Quickstart: Register an application with the microsoft identity platform.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/develop/quickstart-register-app. (accessed: 05.04.2023).

[16] Microsoft. 'Application and service principal objects in azure active directory.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals. (accessed: 06.04.2023).

[17] Microsoft. 'Manage azure active directory groups and group membership.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/fundamentals/how-to-manage-groups. (accessed: 06.04.2023).

[18] C. Harris. 'Microservices vs. monolithic architecture.' (), [Online]. Available: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith. (accessed: 20.04.2023).

[19] raman$_2$57. 'Microservices vs. monolithic architecture.' (), [Online]. Available: https://www.geeksforgeeks.org/monolithic-vs-microservices-architecture. (accessed: 17.04.2023).

[20] RedHat. 'What is grafana?' (13th May 2022), [Online]. Available: https://www.redhat.com/en/topics/data-services/what-is-grafana. (accessed: 05.05.2023).

[21] Microsoft. 'Azure managed grafana.' (), [Online]. Available: https://azure.microsoft.com/en-us/products/managed-grafana. (accessed: 05.05.2023).

[22] Elastic. 'What is elasticsearch?' (), [Online]. Available: https://www.elastic.co/what-is/elasticsearch. (accessed: 04.05.2023).

[23] NodeJs. 'About node.js.' (), [Online]. Available: https://nodejs.org/en/about. (accessed: 08.05.2023).

[24] B. Semah. 'What exactly is node.js? explained for beginners.' (), [Online]. Available: https://www.freecodecamp.org/news/what-is-node-js/. (accessed: 16.05.2023).

[25] M. Contributers. 'What is javascript?' (), [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. (accessed: 16.05.2023).

[26] Uxpin. 'What is npm?' (), [Online]. Available: https://www.uxpin.com/studio/blog/what-is-npm/. (accessed: 16.05.2023).

[27] Amazon. 'What is the elk stack?' (), [Online]. Available: https://aws.amazon.com/what-is/elk-stack/. (accessed: 02.04.2023).

[28] S. Knight. 'Grafana vs kibana vs knowi: Battle royale 2022.' (12th Apr. 2022), [Online]. Available: https://www.knowi.com/blog/grafana-vs-kibana/. (accessed: 02.04.2023).

[29] Grafana. 'Visualize splunk easily with grafana.' (), [Online]. Available: https://grafana.com/solutions/splunk/visualize/. (accessed:04.04.2023.)

[30] Splunk. 'Managing indexers and clusters of indexers.' (), [Online]. Available: https://docs.splunk.com/Documentation/Splunk/9.0.4/Indexer/Howindexingworks/. (accessed:04.04.2023.)

[31] Splunk. 'Pricing.' (), [Online]. Available: https://www.splunk.com/en_us/products/pricing.html/. (accessed:04.04.2023.)

[32] KinneyGroup. 'Splunk licenses: What size do you need? here's how to estimate it.' (22nd Mar. 2023), [Online]. Available: https://kinneygroup.com/blog/splunk-license-estimations/. (accessed:04.04.2023.)

[33] P. Gorbachenko. 'What are functional and non-functional requirements and how to document these.' (), [Online]. Available: https://enkonix.com/blog/functional-requirements-vs-non-functional/. (accessed: 08.05.2023).

[34] M. Maynes. 'One simple action you can take to prevent 99.9 percent of attacks on your accounts.' (20th Aug. 2019), [Online]. Available: https://www.microsoft.com/en-us/security/blog/2019/08/20/one-simple-action-you-can-take-to-prevent-99-9-percent-of-account-attacks/. (accessed: 14.03.2023).

[35] Amazon. 'Microservices.' (), [Online]. Available: https://aws.amazon.com/microservices/. (accessed:22.05.2023.)

[36] R. KUĆ. 'Solr vs elasticsearch: Performance differences more. how to decide which one is best for you.' (), [Online]. Available: https://sematext.com/blog/solr-vs-elasticsearch-differences/. (accessed: 13.03.2023).

[37] apiconnection. 'How to understand your nginx server configuration.' (), [Online]. Available: https://apiconnections.co/blog/2022-05-31-nginx-super-user-config/. (accessed:22.05.2023.)

[38] Calomel. 'Nginx secure web server.' (), [Online]. Available: https://calomel.org/nginx.html. (accessed: 04.05.2023).

[39] Velosimo. 'What is siloed infrastructure?' (), [Online]. Available: https://www.velosimo.com/blog/what-is-siloed-infrastructure. (accessed: 17.04.2023).

[40] Microsoft. 'Linux virtual machines pricing.' (), [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/?ef_id=_k_Cj0KCQjw9deiBhC1ARIsAHLjR2Ai375Ok40SgEeCQbVCwX_9T9Jm-Kh0trYOq-giqHixISqGuIwMfWQaAoEyEALw_wcB_k_&OCID=AIDcmmbnk3rt9z_SEM__k_Cj0KCQjw9deiBhC1ARIsAHLjR2Ai375Ok40SgEeCQbVCwX_9T9Jm-Kh0trYOq-giqHixISqGuIwMfWQaAoEyEALw_wcB_k_&gclid=Cj0KCQjw9deiBhC1ARIsAHLjR2Ai375Ok4
9T9Jm-Kh0trYOq-giqHixISqGuIwMfWQaAoEyEALw_wcB#pricing. (accessed: 06.05.2023).

[41] Virtuozzo. 'Hardware node requirements.' (), [Online]. Available: https://www.knowledgehut.com/blog/web-development/install-nodejs-on-ubuntu#prerequisites. (accessed: 06.05.2023).

[42] J. Loisel. 'Elasticsearch: Optimization guide.' (), [Online]. Available: https://octoperf.com/blog/2018/09/21/optimizing-elasticsearch/#server-hardware. (accessed: 04.05.2023).

[43] G. Labs. 'Install grafana.' (), [Online]. Available: https://grafana.com/docs/grafana/latest/setup-grafana/installation/. (accessed: 04.05.2023).

[44] Trendmicro. 'Setting the reverse-proxy server of safesync for enterprise (ssfe) 2.1.' (), [Online]. Available: https://success.trendmicro.com/dcx/s/solution/1103620-setting-the-reverse-proxy-server-of-safesync-for-enterprise-ssfe-2-1?language=en_US&sfdcIFrameOrigin=null. (accessed: 04.05.2023).

[45] D. docs. 'Ucp system requirements.' (), [Online]. Available: https://docs.docker.com.zh.xy2401.com/v17.09/datacenter/ucp/2.2/guides/admin/install/system-requirements/. (accessed: 06.05.2023).

[46] ProductPlan. 'Rapid prototyping.' (), [Online]. Available: https://www.productplan.com/glossary/rapid-prototyping/. (accessed: 20.05.2023).

[47] Jira. 'Jira software.' (), [Online]. Available: https://www.atlassian.com/software/jira. (accessed:31.01.2023.)

[48] SimonMcCallum. 'Thesis-ntnu.' (), [Online]. Available: https://github.com/COPCSE-NTNU/thesis-NTNU. (accessed: 10.04.2023).

[49] Ø. Wahlstrøm. 'Bachelorprosjekt2023.' (21st May 2023), [Online]. Available: https://github.com/Oivindwa/Bachelor_Tussa. (accessed:21.05.2023.)

[50] github. 'Adding locally hosted code to github.' (23rd Dec. 2022), [Online]. Available: https://docs.github.com/en/migrations/importing-source-code/using-the-command-line-to-import-source-code/adding-locally-hosted-code-to-github. (accessed: 19.05.2023).

[51] npm. 'Azure identity client library for javascript.' (), [Online]. Available: https://www.npmjs.com/package/@azure/identity. (accessed: 16.05.2023).

[52] npm. 'Azure key vault secret client library for javascript.' (), [Online]. Available: https://www.npmjs.com/package/@azure/keyvault-secrets. (accessed: 16.05.2023).

[53] npm. 'Elasticsearch node.js client.' (), [Online]. Available: https://www.npmjs.com/package/@elastic/elasticsearch. (accessed: 16.05.2023).

[54] npm. 'Axios.' (), [Online]. Available: https://www.npmjs.com/package/axios. (accessed: 16.05.2023).

[55] npm. 'Node-cron.' (), [Online]. Available: https://www.npmjs.com/package/cron. (accessed: 16.05.2023).

[56] npm. 'Qs.' (), [Online]. Available: https://www.npmjs.com/package/qs. (accessed: 16.05.2023).

[57] E. D. Hardt. 'The oauth 2.0 authorization framework.' (), [Online]. Available: https://www.rfc-editor.org/rfc/rfc6749.html. (accessed: 20.03.2023).

[58] Microsoft. 'List devices.' (10th Mar. 2023), [Online]. Available: https://learn.microsoft.com/en-us/graph/api/device-list?view=graph-rest-1.0&tabs=http. (accessed: 18.05.2023).

[59] Docker. 'Getting started with swarm mode.' (), [Online]. Available: https://docs.docker.com/engine/swarm/swarm-tutorial/. (accessed: 19.05.2023).

[60] Docker. 'Install docker engine on ubuntu.' (), [Online]. Available: https://docs.docker.com/engine/install/ubuntu/. (accessed: 19.05.2023).

[61] Microsoft. 'Install the azure cli on linux.' (), [Online]. Available: https://learn.microsoft.com/en-us/cli/azure/install-azure-cli-linux?pivots=apt. (accessed: 19.05.2023).

[62] Microsoft. 'Quickstart: Create a key vault using the azure portal.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/key-vault/general/quick-create-portal. (accessed: 19.05.2023).

[63] Microsoft. 'Quickstart: Set and retrieve a secret from azure key vault using the azure portal.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/key-vault/secrets/quick-create-portal. (accessed: 19.05.2023).

[64] Microsoft. 'Key vault pricing.' (), [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/key-vault/. (accessed: 19.05.2023).

[65] Microsoft. 'Quickstart: Create an azure container registry using the azure portal.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/container-registry/container-registry-get-started-portal?tabs=azure-cli. (accessed: 19.05.2023).

[66] Microsoft. 'Container image storage in azure container registry.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/container-registry/container-registry-storage. (accessed: 19.05.2023).

[67] Microsoft. 'Create a role-assignable group in azure active directory.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/roles/groups-create-eligible?tabs=ms-powershell. (accessed: 19.05.2023).

[68] Microsoft. 'B2b collaboration overview.' (10th Mar. 2023), [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/external-identities/what-is-b2/. (accessed: 13.05.2023).

[69] Microsoft. 'Tutorial: Secure user sign-in events with azure ad multi-factor authentication.' (), [Online]. Available: https://learn.microsoft.com/en-us/azure/active-directory/authentication/tutorial-enable-azure-mfa. (accessed: 19.05.2023).

[70] M. Walkom. 'What are ports 9200 and 9300 used for?' (12th Apr. 2022), [Online]. Available: https://discuss.elastic.co/t/what-are-ports-9200-and-9300-used-for/238578. (accessed: 02.04.2023).

[71] G. Labs. 'Configure grafana.' (12th Apr. 2022), [Online]. Available: https://grafana.com/docs/grafana/latest/setup-grafana/configure-grafana/. (accessed: 02.04.2023).

[72] G. Labs. 'Configure azure ad oauth2 authentication.' (12th Apr. 2022), [Online]. Available: https://grafana.com/docs/grafana/latest/setup-grafana/configure-security/configure-authentication/azuread/. (accessed: 02.04.2023).

[73] D. Maksutenko. 'Loaning laptops to employees: Best practices and pitfalls to consider.' (), [Online]. Available: https://www.worktime.com/loaning-laptops-to-employees-best-practices-and-pitfalls-to-consider#A1. (accessed: 09.05.2023).

[74] R. Vanover. 'What is the 3-2-1 backup rule?' (), [Online]. Available: https://www.veeam.com/blog/321-backup-rule.html. (accessed: 10.04.2023).

[75] J. Duhamel. 'Api key best practices.' (), [Online]. Available: https://support.nmi.com/hc/en-gb/articles/360002813697-API-Key-Best-Practices. (accessed: 21.04.2023).

[76] M. Abaakouk. 'On api keys best practices.' (), [Online]. Available: https://blog.mergify.com/api-keys-best-practice/. (accessed: 21.04.2023).

[77] G. Wrenn. 'How to patch vulnerabilities and keep them sealed.' (), [Online]. Available: https://www.techtarget.com/searchsecurity/tip/How-to-patch-vulnerabilities-and-keep-them-sealed. (accessed: 10.04.2023).

[78] Certbot. 'User guide.' (), [Online]. Available: https://eff-certbot.readthedocs.io/en/stable/using.html#manual. (accessed: 15.04.2023).

[79] E. MÍNGUEZ. 'Container image scanning for azure pipelines with sysdig.' (), [Online]. Available: https://sysdig.com/blog/container-image-scanning-for-azure-pipelines-with-sysdig/. (accessed: 10.04.2023).

[80] C. Team. 'Horizontal vs. vertical scaling: How do they compare?' (), [Online]. Available: https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling. (accessed: 10.04.2023).

[81] NTNU. 'Iik3100 - etisk hacking og penetrasjonstesting.' (), [Online]. Available: https://www.ntnu.no/studier/emner/IIK3100#tab=omEmnet/. (accessed: 18.05.2023).

[82] Kali. 'What is kali linux?' (), [Online]. Available: https://www.kali.org/docs/introduction/what-is-kali-linux/. (accessed: 18.05.2023).

[83] Nmap. 'Nmap: Discover your network.' (), [Online]. Available: https://nmap.org/. (accessed: 18.05.2023).

[84] Nmap. 'Script ssh-hostkey.' (), [Online]. Available: https://nmap.org/nsedoc/scripts/ssh-hostkey.html/. (accessed: 18.05.2023).

[85] Kali. 'Dirb.' (), [Online]. Available: https://www.kali.org/tools/dirb/. (accessed: 18.05.2023).

[86] w3schools. 'Sql injection.' (), [Online]. Available: https://www.w3schools.com/sql/sql_injection.asp/. (accessed: 18.05.2023).

[87] Kali. 'Sqlmap.' (), [Online]. Available: https://www.kali.org/tools/sqlmap/. (accessed: 18.05.2023).

[88] Kali. 'Hydra.' (), [Online]. Available: https://www.kali.org/tools/hydra/. (accessed: 18.05.2023).

[89] Stackoverflow. 'What is the default username and password for grafana login page?' (), [Online]. Available: https://stackoverflow.com/questions/54039604/what-is-the-default-username-and-password-for-grafana-login-page/. (accessed: 18.05.2023).

[90] D. Wittman. '1000-most-common-passwords.' (), [Online]. Available: https://github.com/DavidWittman/wpxmlrpcbrute/blob/master/wordlists/1000-most-common-passwords.txt/. (accessed: 18.05.2023).

[91] Metasploit. 'Unix passwords.' (), [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/data/wordlists/unix_passwords.txt/. (accessed: 18.05.2023).

[92] Zaproxy. 'Owasp zap - getting started.' (), [Online]. Available: https://www.zaproxy.org/getting-started/. (accessed: 18.05.2023).

[93] Kali. 'Zaproxy.' (), [Online]. Available: https://www.kali.org/tools/zaproxy/. (accessed: 18.05.2023).

[94] O. Garrett. 'Trust no one: The perils of trusting user input.' (), [Online]. Available: `https://www.nginx.com/blog/trust-no-one-perils-of-trusting-user-input/`. (accessed: 19.05.2023).

[95] Snyk. 'What is snyk?' (), [Online]. Available: `https://snyk.io/product/`. (accessed: 19.05.2023).

[96] CVE-2023-0464. 'Openssl.' (22nd Mar. 2023), [Online]. Available: `https://www.cve.org/CVERecord?id=CVE-2023-0464`. (accessed: 19.05.2023).

[97] Snyk. 'Release of invalid pointer or reference.' (7th May 2021), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-AOM-1290331`. (accessed: 19.05.2023).

[98] Snyk. 'Use after free.' (3rd Jun. 2021), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-AOM-1298721/`. (accessed: 19.05.2023).

[99] Snyk. 'Buffer overflow.' (5th Jun. 2021), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-AOM-1300249/`. (accessed: 19.05.2023).

[100] Snyk. 'Out-of-bounds write.' (16th Mar. 2021), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-AOM-1085722/`. (accessed: 19.05.2023).

[101] Snyk. 'Cleartext transmission of sensitive information.' (15th Feb. 2023), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-CURL-3320493/`. (accessed: 19.05.2023).

[102] Snyk. 'Cleartext transmission of sensitive information.' (29th Aug. 2022), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-AOM-1085722/`. (accessed: 19.05.2023).

[103] Snyk. 'Cleartext transmission of sensitive information.' (23rd Dec. 2022), [Online]. Available: `https://security.snyk.io/vuln/SNYK-DEBIAN11-CURL-3179181/`. (accessed: 19.05.2023).

[104] passportjs. 'Passport.' (), [Online]. Available: `http://www.passportjs.org/`. (accessed: 08.03.2023).

[105] Microsoft. 'Microsoft identity platform id tokens.' (5th Feb. 2023), [Online]. Available: `https://learn.microsoft.com/en-us/azure/active-directory/develop/id-tokens`. (accessed: 08.03.2023).

[106] mdn web docs. 'Using http cookies.' (), [Online]. Available: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies#security`. (accessed: 09.03.2023).

[107] passportjs. 'Microsoft azure active directory passport.js plug-in.' (), [Online]. Available: `http://www.passportjs.org/packages/passport-azure-ad/`. (accessed: 08.03.2023).

[108] Nginx. ‘Restricting access by geographical location.’ (), [Online]. Available: https://docs.nginx.com/nginx/admin-guide/security-controls/controlling-access-by-geoip/. (accessed: 13.05.2023).

[109] G2. ‘F5 nginx pricing.’ (), [Online]. Available: https://www.g2.com/products/f5-nginx/pricing. (accessed: 13.05.2023).

[110] S. S. Team. ‘What is a certificate authority (ca)?’ (), [Online]. Available: https://www.ssl.com/faqs/what-is-a-certificate-authority/. (accessed: 13.05.2023).

[111] Microsoft. ‘What is azure bastion?’ (), [Online]. Available: https://learn.microsoft.com/en-us/azure/bastion/bastion-overview. (accessed: 13.05.2023).

[112] Microsoft. ‘Azure bastion pricing.’ (), [Online]. Available: https://azure.microsoft.com/en-us/pricing/details/azure-bastion/. (accessed: 13.05.2023).

[113] Redis. ‘Core capabilities.’ (), [Online]. Available: https://redis.io/. (accessed: 13.05.2023).

[114] Microsoft. ‘What is azure application gateway?’ (), [Online]. Available: https://learn.microsoft.com/en-us/azure/application-gateway/overview. (accessed: 13.05.2023).

[115] Microsoft. ‘What is azure application gateway v2?’ (), [Online]. Available: https://learn.microsoft.com/en-us/azure/application-gateway/overview-v2. (accessed: 13.05.2023).

[116] Microsoft. ‘Reduce service costs by using azure advisor.’ (), [Online]. Available: https://learn.microsoft.com/en-us/azure/advisor/advisor-cost-recommendations. (accessed: 19.05.2023).

Appendix A

Meeting minutes

Meeting minutes

2023-01-12

1 Participants

Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Decide the different group roles
- Discuss the work ahead

3 Notes on each matter

- Øivind is the group leader and communication manager, Magnus is quality control manager and Sindre is documentation responsible.
- The group have delegated the work towards the next meeting

4 Next meeting

Friday 13/01 11:00

Meeting minutes

2023-01-13

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- The bachelor group have some questions for Tussa:
  - Log sources
  - Lab environment
  - Does Tussa only use Microsoft azure?
  - What security criteria does Tussa demand
- General status report

3 Notes on each matter

- The task from Tussa is seen as a task with two parts. The first part is not technical, but rather a risk analysis and procedure.
- The log sources are coming. Benjamin is not sure what information they want to display to customers yet, but will come back with this information after talking with Vigleik.
- The Lab environment is coming soon. Most likely VM's in Tussas own datacenter.
- Tussa uses only Microsoft Azure

- Tussa will finish the issue setting, and then set more specific security critera. (ISM best practise and NSM)
- Both parties agree that a periodic portal that generates a daily report is the best solution.

4 Next meeting

Monday 16/01-23 13:00

Meeting minutes

2023-01-16

1 Participants

Muhammad Mudassar Yamin, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- The bachelor group have some questions for bachelor thesis:
  - Doing backups of our writing
  - Open source Threat Intelligence and sharing platforms
- General status report

3 Notes on each matter

- The group have decided together with the thesis to do weekly backup every Monday. The documentation responsible is responsible for this.
- Stix and Misp are open source tools we can use for cyber threats.
- The group have decided to use Jira as a planning framework through the bachelor.

4 Next meeting

Monday 23/01-23 09:30

Meeting minutes

2023-01-23

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Discuss the lab environment delivered by Tussa
- Examples of APIs the group will use
- General status report

3 Notes on each matter

- Benjamin will have a demo of the lab environment next meeting.
- Benjamin will get examples of APIs used, so the group will get baseline knowledge about the APIs used
- Vigleik will post documentation on PDF-export and organization chart

4 Next meeting

Monday 23/01-23 13:00

Meeting minutes

2023-01-23

1 Participants

Muhammad Mudassar Yamin, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- The bachelor group have some questions for bachelor thesis:
  - Defining the problem statement
  - Where to go from here
- General status report

3 Notes on each matter

- Muhammad have some suggestions on how to define the problem statement, but this is something the group needs to decide on together.
- Start working on the report. Begin with explaining the problem and how we are going to solve it according to plan.
- The group can also start playing around with fetching logs with APIs.
- Notes: IDA IPA cluster deployment, Wazuh SIEM, Cismon formating log.

4 Next meeting

Monday 30/01-23 09:30

Meeting minutes

2023-01-30

1 Participants

Muhammad Mudassar Yamin, Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Discuss the lab environment delivered by Tussa
- What logs does Tussa need to be pulled
- General status report

3 Notes on each matter

- The lab environment is nearly ready. The only thing missing is the external network traffic, and this will be in place over the next few days.
- The task is to pull logs from the APIs, which is in JSON format.
- The group presented Elastic as a possible way to pull logs. Tussa agrees this is a good way to go, with the availability to pull for a live update.

4 Next meeting

Monday 06/02-23 13:00

Meeting minutes

2023-02-06

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Discuss the lab environment delivered by Tussa
- General status report

3 Notes on each matter

- The lab environment is ready. The group have recived the information needed and will get the lab up and running soon.
- Benjamin will collect a JSON file with relevant information that the group will pull.
- The goup have decided to use elastic searched.

4 Next meeting

Monday 13/02-23 13:00

Meeting minutes

2023-02-13

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Docker on VM
- Risk analysis about private PC
- storing API credentials

3 Notes on each matter

- Benjamin will examen the possability to enable virtulazation in BIOS on the VM
- Will finalise the Risk analysis today
- Research about how to store API crednetials.

4 Next meeting

Monday 13/02-23 13:00

Meeting minutes

2023-02-20

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- The bachelor group will show some of their test data so far.
- General status report

3 Notes on each matter

- The data looks good, but one device can be in more than one endpoint.
- The only match of data on all the endpoints are ”hostname”

4 Next meeting

Monday 27/02-23 09:30

Meeting minutes

2023-02-27

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- The Bachelor group show what we have done so far regarding elastic. Made a docker cluster.
- The group have an error ”bad gateway” with using Elasticsearch in Grafana.
- General status report

3 Notes on each matter

- The plan from here is to get Kafka up and running. Elasticseach are going to send to Kafka and then to Grafana.
- The Bachelor group is on track with the project.

4 Next meeting

Monday 27/02-23 13:00

Meeting minutes

2023-03-06

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- The Bachelor group show the scripts they have made so far and how they work.
- The goup show their work regarding the risk analysis and have some questions regarding the template.
- General status report

3 Notes on each matter

- The scripts looks good so far, but the students have some errors running a crone job.
- The Bachelor group is on track with the project.

4 Next meeting

Monday 13/02-23 13:00

Meeting minutes

2023-03-16

1 Participants

Benjamin Yndestad Aam, Vigleik Hustadnes, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Short demo of the application so far
- Tussa's feedback on our work so far
- Discuss the next steps (Multi-tenant)
- The bachelor groups visit to Ørsta

3 Notes on each matter

- The bachelor group showed their work in grafana, with the tables and how it will look. Tussa had some feedback:
  - They want last sync from all three of the systems
  - They want one table where it is checked if all the syncs are on the same date, if not there is an error
  - Another table with "is active"
- We discussed if it is possible to have one API key for all of the customers. This is going to be part of the risk analysis
- The bachelor group does not need to give any thought to history and saving old data

- Discussed the different architectures. Didn't do a final conclusion.
- All in all, Tussa is happy with what we have done so far, but want us to also focus on the risk of the application, wand not just development

4 Next meeting

Monday 20/03-23 09:30

Meeting minutes

2023-03-20

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Discussion about the API keys
- General status report

3 Notes on each matter

- Secure Endpoint only gives the opportunity to change API keys to ”Read only” or ”Read and Write”
- Umbrella and Intune can make more changes, meaning they can share API keys across customers
- Tussa want one shared keyvault for every customer
- Every time the VM restarts, Grafana resets. Could possibly make a script that sets up Grafana again every time the VM restarts

4 Next meeting

Monday 27/03-23 09:30

Meeting minutes

2023-04-03

1 Participants

Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Status report on the solution
- Status report on the report
- Close the last sprint

3 Notes on each matter

- The status on the solution is that there are still some components that needs development. It is uploaded to Azure, but we still have hard coded API keys.
- We have set a deadline for the solution on 01/05. From that date we need to focus on the report.
- The status on the report is that we have started implementing things already written, but there are still a lot of things that needs to we written.
- We have set a deadline for the report on 15/05, from that date we want to be finished writing so we can control-check everything.

4 Next meeting

Tuesday 11/04-23 09:00

Meeting minutes

2023-04-11

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Show the work done through the easter break
- Ørsta visit
- General status report

3 Notes on each matter

- The change to Azure container registry was harder than expected, and will take some more time
- Keep focusing on the solution, but the group cant forget the report as well. Benjamin makes it clear that he understands that we have to focus on the report at some point.

4 Next meeting

Monday 17/04-23 09:30

Meeting minutes

2023-04-17

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Azure AD
- Ørsta visit
- General status report

3 Notes on each matter

- Azure AD authentication is harder than expected. But the group will find a solution together with the advisor.
- Ørsta visit plans are in order and the AirBnB is booked.
- Benjamin will ask someone in Tussa about the subdomain to be used.

4 Next meeting

Friday 21/04-23 10:00

Meeting minutes

2023-04-24

1 Participants

Benjamin Yndestad Aam, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- New subdomain
- General status report

3 Notes on each matter

- We need a new subdomain because the old one was deleted by a mistake. Benjamin will get a new one.
- The solution is almost finished, but need some fine tuning.
- Benjamin reminds us that we also need to focus on the report, and understands if not everything is finished in the solution.

4 Next meeting

Monday 01/05-23 09:30

Meeting minutes

2023-05-02

1 Participants

Muhammad Mudassar Yamin, Magnus Lekanger Voll, Øivind Wahlstrøm, Sindre Davidsen Schonhowd

2 Matters

- Questions from the bachelor group:
  - Assessment criteria
  - How to we refer to sources
  - What kind of testing should be do
  - Where in the thesis should we put the risk assesment
- General status report

3 Notes on each matter

- There is not any directly assessment criteria since a lot of the thesis are so different
- Risk analysis can be a seperate chapter
- Use Zap Proxy for testing + user feedback from Tussa
- The thesis should contain a description on how to deploy the solution

4 Next meeting

Monday 08/05-23 09:30

Appendix B

Project plan

Project plan

Magnus Voll, Øivind Wahlstrøm and Sindre Schonhowd

Spring 2023

Contents

1 Goals and framework

This section will explain the background for our project and what we will accomplish over the next 17 weeks, and we will explain the limitations to our project.

1.1 Background

Tussa has expressed that they have a need and a desire to build a platform to centralize security information for them to present to their customers. Tussa has stated that going to different applications to find the proper security information regarding their customers is cumbersome and inefficient way of working. Giving individual reports from each security platform/vendor to their customers has also been a thing that has been seen as a lack of optimizations of giving the full picture. Due to these reasons Tussa contacted NTNU to see if their Bachelor graduate students could develop a solution to this problem.

1.2 Project goal

1.2.1 Result goals

- The bachelor group should come up with an effective way to extract and report relevant information for the SaaS-suppliers.
- The reporting and storing of data need to be secure from possible cyber threats.
- The customers should be able to get a daily report with all their relevant information.
- The bachelor group will deliver a risk analysis of the new reporting system to make sure that it keeps up with the security standards for both Tussa and their customers.

1.2.2 Effect goals

- Our solution should greatly increase the efficiency and reliability on the documentation and reporting between Tussa and their SaaS-supplier.
- Our solution should be secure in a way that the customers can rely on that their information are stored in a secure manure and not in danger to cyber threats.
- The work we deliver should be modifiable at a later stage, either by Tussa or another group. Meaning that others outside our bachelor group should be able to expand, optimize and tweak our solutions at a later stage.

1.3 Frames

1.3.1 Time frames

- The project plan needs to be finished by 1th of February 2023
- The group have a given time frame to consider when completing the project. Our project began 11th of January and the deadline is 22th of May 2023. Both our solution and our report needs to be finished by the given deadline

1.3.2 Language frames

Even though this is a Norwegian study, our group have decided that we are going to be writing in English. The main reason is that a lot of technical terms are in English, and the report will be easier to understand when written in English. Another reason is that our supervisor is more fluent in English than Norwegian.

2 scope

This sections elaborate the scope of the project. it will give a feature explanation of the different aspect this task faces.

2.1 issue area (Problemområde)

Tussa IKT is a company under the Tussa group. The company delivers management IT, sky and security services for different companies in Norway. To manage this they have control their own data centers, a lot of SaaS- and communications suppliers together with their own fiber network. Tussa IKT is also ISO27001-certified MSP and ISP, which means that they have information security as a high priority.
The issue area lays between Tussa and their SaaS-suppliers. Tussa want the reporting and communications to run more smoothly. Up until now they collect their data from three different platforms and want all of the relevant data to be showed in the same interface.

2.2 issue limitations (Problemavgrensning)

As developers, will we develop a solution that makes it possible to extract specific information from various log sources from different vendors related to their customers and then store it safely in one place. It is Tussa IKT that decides what is to be extracted from the logs, and how they want to present it. We will listen to and rely on Tussa IKT's ideas and develop after that.

2.3 Task From the Client

The objective of the assignment is to research and improve the procedures for SaaS services integration using APIs, with a particular focus on risk assessment and securing the process around handling API credentials. Outline a solution that secures the data during transit, at rest, and define a best practice security strategy for managing access to scripts, code, API credentials and data in a multi-tenant environment. API credential vaults, test environments and SIEM are available.

Security requirements:

- Encryption
- Access control
- Least privilege access control principle
- Auditing and logging
- Compliance with the customer's security policies/ISMS
- Security testing

Proof of Concept:

- Build a prototype of the proposed solution that implements the key features and security requirements identified in the research and development phase.
- Test the prototype with a dataset from Microsoft Intune, Cisco Secure Endpoint and Cisco Umbrella to demonstrate how the solution could be used in a production environment
- Organize and compile data from the dataset into a structured format, such as a database or JSON file, for easy access and reference in future use.
- Evaluate and identify the result of the prototype and procedures, and suggest potential enhancements or limitations for future considerations

2.4 Problem statement

Based on the following needs from the customer, have we set our problem statement to be: *How can be develop an application which furfills the current demands of the customer?*

3 Project organising

3.1 Roles

- Leader - Øivind
- Documentation responsible - Sindre
- Quality control manager/supervisor/officer/responsible - Magnus
- Communication supervisor/officer/responsible - Øivind

3.2 Responsibilities

Everyone: The group members will have a common responsibility to do their best effort in order to achieve the project goals. They will also do their best to work consistently and to encourage the other members to do the same. Each member will also need to record and implement their workhouses in a common excel sheet. It is also important to do a correct documentation of which sources that has been utilized. Everyone will make decisions together as a team, and their voice will weigh equally, but if a disagreement cant be solved, the team leader will have the final say.

Leader: The leader responsibilities includes certainty that progress are accomplished during the project, as well as ensuring that the deadline will be met on the 22th of May. The leader will also be liable for de-escalate any conflict that may occur, and try to find the best diplomatic solution for every part. He will have the overall responsibility for the project.

Document responsible: The document responsible will be the one in charge to ensure a clear control of documents relevant to the project. This role will create meeting minutes for every meeting that's being held for the duration of this project. He will also be the one to maintain a clear structure on the groups main file sharing platform i.e. Overleaf.

Quality control manager: This role will ensure that what is being produced by the group meet the proper qualification and requirements expected of them. This includes finalising blocks of the project as well as do a finale check of every document before completion. This role will also secure a coherent structure throughout the report.

Communication Officer: The Communication officer will be responsible to achieve a satisfactory level of communication between all relevant parts during the project. This role will be the one to organize necessary meetings and will be the one to manage them. He will also relay information from his communication lines to the rest of the group members in a best possible manner.

3.3 Routines

The group thinks it is important to have routines both on meetings, workloads and internal problem solving. When we have set routines that makes the working more efficient and it makes sure that every team members knows whats expected of them. It also keeps the members on their toes and makes sure that we keep on schedule.

- Each member will fill out their part of the timetable at the end of each week.
- Every individual is to conduct a minimum of 30 hour workload per week.
- Being honest about dissatisfaction or problems is important to deal with and to solve as quickly as possible.
- We will conduct a meeting with our supervisor every Monday at 13:00. This meeting will have the opportunity to be both physical and digital, depending on what's most suitable. This is done to establish a high level of flexibility.
- The group will have an additional meeting every Friday at 12:00 to revise the week and to plan ahead for the upcoming Monday meetings.
- The group will also have a meeting with Tussa every Monday at 09:30 where the group can ask additional questions and keep everyone up to speed on how the project are going.
- Our workflow will consist of 25 minutes work sessions with a 5 minutes break. Between every fourth session there will be a longer break consisting of 15 minutes. Our lunch break will differ between 30-45 minutes.

3.4 Group rules

- Each member is obligated to be on time for meetings, and report absence, a minimum of 30 minutes before meetings with a respectively reason.
- Tasks will be completed on time, or it will be mention in advance, if delays are inevitable.
- The workload will be divided fairly to ensure that everyone contributes equally.
- We will have hundred percent focus when designated work hours are in play.
- All group members are equal and their voice is weigh the same, but the team leader will have the final say, if a disagreement occur.

4 Planing, follow-up and reporting

4.1 Main division of the project

The group has decided to use the scrum framework, for our method of working throughout the project. It is a well tested framework that suits our way of working in many ways, especially considering how agile it is[1]. Underneath will describe why we picked this framework and how it will benefit us.

The Scrum framework starts off by listing every task needed to be done throughout the project in something called the backlog. It is Never the less important to note that the overall backlog is editable throughout the project, We highlight this to emphasis our focus on flexibility and adaptability towards new and unknown challenges that may occur.

Scrum offers us to work in sprints where we have focus on specific tasks. We will utilize these sprints to deliver incremental products to the client for us to revise together. We find it important to be able to make changes based on feedback we get from our advisor and our client, which the sprints provide. This lets us prioritise the task, and focus on the task that are of most significance first. We also get to chose the different length of each sprint to make it as adaptable as possible, depending on the task.

The scrum framework provides a feature called sprint retrospective which is conducted after each sprint. This feature lets us improve our self for the next sprint trough discussing what went well, and what we need to improve. It also makes room to plan for the next sprint. Having this ability integrated into the framework was one of the main reasons for our decision to choose Scrum.

We also chose this framework based on that we can have tasks which are known, but don't know how to complete. Never the less, having them listed up so they are known will gives us a better overview of the overall workload we need to complete within the deadline.

We also chose this structure of working because of the floating roles it offers. We are not limited by a fix role only one person can accomplish, but one person can do multiple tasks which makes us cross-functional. This means that if one person is absent, the work dose not stop, because another group member will just fill his role for the duration of his absent.

The scrum framework also offers a scrum board which will give the overview to always see our progress and to see what's needs to be done. It's a simple setup of every task that we have and what state it is. The different states are as follows: To Do, In Progress, To Verify, and Done.

Finally we will have a Scrum master ensuring that progress are being made, that the group stay on time to deliver according to the the schedule. In our case, will the group leader be the Scrum master.

4.2 Plan for status meetings and decisions

4.2.1 Status meeting

We have made the decision to implement a status meeting every Friday 12:00 to do a recap of the week, and prepare for our Monday meetings with our client and supervisor. Here we will discuss what has gone well, and what we improve to the following week.

4.2.2 Decision basis

The biggest decisions we will comprehend will take place after near the end of each iteration, to prepare for the next one. We will also make decisions a long the way as we work to impose a dynamic work method, and to be adaptable toward changes. We have planed to be available for daily conversation in case a decision is needed.

5 Organise quality assurance

5.1 Documentation and Standards

The vast majority of our documentation will be stored and maintained in Overleaf, where our advisor and our client will have access. The documentation here will consist of the Project plan, Tasks to do, Report, Meeting minutes... Furthermore will a weekly copy of our most important documents, i.e. the Project plan and the bachelor thesis to be stored in OneDrive as a safety measure to prevent data loss. Additionally will the group store documents of less significant value in the a different sub-folder in OneDrive. Here among other things you will find the Timetable and email drafts there.

All code written, will be maintain to meet the criteria of professional standards. Suitable comments will also be added to the code where necessary. we will use GitHub for a secure and common platform to store our code, where every member has access to. We chose GitHub because it's an easy to use platform, that everyone is familiar with, from previous projects.

We will be using Jira to manage our sprint schedule, as well as our to do list. We looked at different kinds of platform to manage our planing schedule, but we concluded that Jira was the best fit for us since it has a built in scrum framework, and of Magnus and Øivind's previous experience with it. it also provides a road-map which visualize the time frame for each sprint in a calendar. Jira is a great tool to keep track of what need to be done, what is currently being worked on and what as been completed.

5.2 Plan for inspection and testing

Our advisor will inspect our work prior to our Monday meeting and will gives us feedback on our progress. In addition will an inspection with Tussa take place every Monday later in the day. Although it should be mentioned that these meetings can be decreased to only one meeting every second week, depending on what we perceive to be necessary. As mentioned earlier, offers the scrum framework an inspection on what has been done after every sprint. Here it's discussed the things that went well, and what needs to me improved for the next sprint. Furthermore, will we revise our work from the current week in our internal Friday meetings as well as looking at the Gantt-timetable to see if we are on schedule to meet our goals.

We will conduct the testing of our application in a designated could environment provided by Tussa. This will be a secure environment where all features to do a proper testing will be available for us. Tussa will be responsible to set this up and maintain it so we can utilize it.

5.3 Risk analysis

The table showcases different risks related to the project, as well as the probability of the event occurs with consequences related to it. The probability is divided in to unlikely, likely, and very likely. Consequences are divided in to critical, moderate, minor, and negligible. In the table below have we identified the following risk events:

Number	Risk	Probabilty	Consequence	Action required
1	We do not manage to finish the project within the deadline. There are plenty of reason why something can be delayed, such as technical issues, miscalculation of time, miscommunication,loss of backup, etc.	Unlikely	Critcal	Yes
2	Big changes in the requirement from the customer regarding the task during the period.	Likely	Minor	Yes
3	One of the participants of our group gets sick over a longer period, or has to quit the project.	Unlikely	Critical	Yes
4	Loss of report or work related to the development of the product.	Unlikely	Critical	Yes
5	Other companies develop similar solutions	Very likely	Negligible	No
6	Tussa IKT decides to cancel the project	Unlikely	Critical	No
7	Problems with the communication between the product we have developed and the information it is supposed to fetch.	Likely	Moderate	Yes
8	Some of the demands to the functionality might not be possible to implement due to technical limitations	Unlikely	Moderate	yes

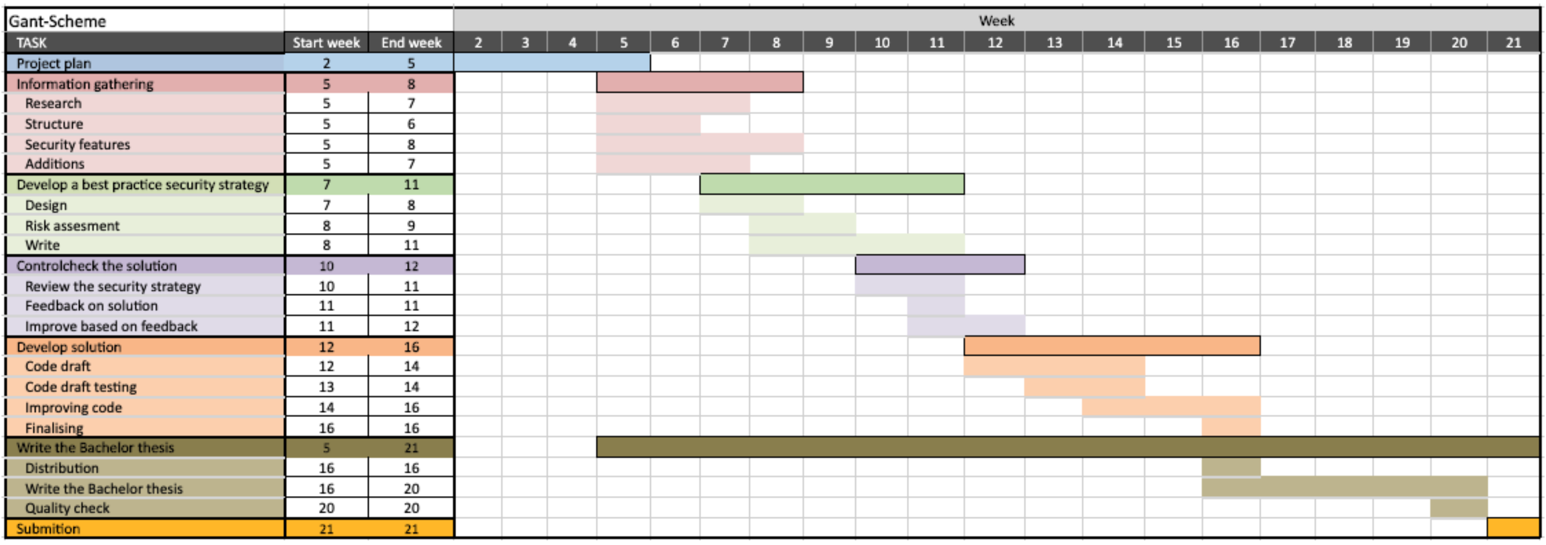
5.4 Action Required to handle the risk

Based on the risk analysis in 5.3, have we chosen several risk events where it is discussed several measures to implement to reduce the negative consequences if the risk event occurs. This are the measures which reduces the probability of negative consequences if an event occurs:

Number	Action required
1	in the initial phase of the project, we create a GANTT chart where we allocate planned time use. As this is dynamic work, some changes might occur, but if we detect that the product we are going to deliver gets delayed do we have to inform Tussa. A viable solution might be to change the functionality. To reduce the probability of delay, do we have to make sure that we are on schedule with the project. If we experience that something is delayed, is this something that require action immediate. A good plan for how to work to stay on schedule is required according to our project plan.
2	Communication between the group and Tussa IKT of which functionality are required for the product are important for both parts so we share the same understanding. During the project, is it important that we meet with Tussa IKT often, so we tell how we are doing and status. Sharing the same knowledge and meeting often will make it easier for Tussa IKT to share their requirements and wishes, and easier for us to implement the changes.
3	To reduce the consequence if someone gets sick, or something unexpected happends which means they will be away from the project for a longer time, or has to quit the project do we meet often to discuss what we have done. We will teach each other what is required for progress. This will minimize the risk of progress if some of the participants either has to quit or stays away for a longer period
4	To prevent the risk of loss of any work, do we save the report in ShareLatex and locally in our OneDrive provided by NTNU. We have also set up a GitLab-repository where we put the source code for the product as well as saving this locally as well.
7	It is important to find out early how the systems log their information, and later identify how the technology regarding fetching information works. With the understanding can we start testing to fetch the required information. As we are going to fetch log-information from multiple vendors might some of the templates/reports change in the future, so we have to create something that is easy to develop further for future changes.
8	We have to make sure that the demand from Tussa IKT is something realistic within the technical knowledge from us and limitation of the technical and security aspect of it. Communication and meeting often is key for realistic development.

6 Plan for implementation

6.1 Gantt-scheme / Project activities



Gant-Scheme			Week																			
TASK	Start week	End week	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Project plan	2	5	■	■	■	■																
Information gathering	5	8				■	■	■	■													
Research	5	7				■	■	■														
Structure	5	6				■	■															
Security features	5	8				■	■	■	■													
Additions	5	7				■	■	■														
Develop a best practice security strategy	7	11						■	■	■	■	■										
Design	7	8						■	■													
Risk assesment	8	9							■	■												
Write	8	11							■	■	■	■										
Controlcheck the solution	10	12									■	■	■									
Review the security strategy	10	11									■	■										
Feedback on solution	11	11										■										
Improve based on feedback	11	12										■	■									
Develop solution	12	16											■	■	■	■	■					
Code draft	12	14											■	■	■							
Code draft testing	13	14												■	■							
Improving code	14	16													■	■	■					
Finalising	16	16															■					
Write the Bachelor thesis	5	21				■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Distribution	16	16															■					
Write the Bachelor thesis	16	20															■	■	■	■	■	
Quality check	20	20																			■	
Submition	21	21																				■

Appendix C

Time tracking

Uke 2	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre			3	4	3			**10**
Magnus			3	2				**5**
Øivind			5	4,5	4,5			**14**

Navn	Totale timer
Sindre	588
Magnus	591
Øivind	656,5

Totalt samlet	1835,5

Uke 3	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	5	8	5	4	5		4	**31**
Magnus	6	4	7	5	7	3		**32**
Øivind	6	1	5	7	4	7	1	**31**

Uke 4	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	5	3	7	5	3		**29**
Magnus	5	6	4	6	2	6		**29**
Øivind	6	7	4	5	7		3	**32**

Uke 5	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	4	4	8	3	8	4	1	**32**
Magnus	7	5	8	1	6	5	1	**33**
Øivind	5	8	8	2	6	4	1	**34**

Uke 6	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	6	7	5	5	3		**32**
Magnus	6	8	5	5	6	2		**32**
Øivind	8	8	8	8		1		**33**

Uke 7	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	8	5	6	7	4	1	2	**33**
Magnus	6	8	5	5	4	3	1	**32**
Øivind	8	8	2	6	6	1	1	**32**

Uke 8	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	7	6	6	4	3		**32**
Magnus	5	8	4	5	4	5	1	**32**
Øivind	7	7	4	5	6		3	**32**

Uke 9	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	8	8	4	6	6	2		**34**
Magnus	6	6	7	6	5	3		**33**
Øivind	12	4	6	8	3	4		**37**

Uke 10	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	5	5	7	5	4	6	1	**33**
Magnus	4	5	7	7	4	5		**32**
Øivind	6	6	7	6	4	1	9	**39**

Uke 11	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	8	6	9	5	5	2		**35**
Magnus	7	5	5	8	7	2		**34**
Øivind	11	10	10	4	7			**42**

Uke 12	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	6	8	5	5	3		**33**
Magnus	8	8		6	7	5	1	**35**
Øivind	5		12	13			9,5	**39,5**

Uke 13	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	6	7	6	7	3		**35**
Magnus	6	5	8	8	4	3		**34**
Øivind	13	11		4	7	3		**38**

Uke 14	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	7	7	8	4	1	6		**33**
Magnus	8	8	7	5	1	5		**34**
Øivind	4	7	8	8	1	6		**34**

Uke 15	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	5	4	6	3	1	6		**25**
Magnus	5	4	6	5	1	4		**25**
Øivind	8	8	8	6	1	5		**36**

Uke 16	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	4	4	8	9	5	2		**32**
Magnus	8	7	7	6	4	3		**35**
Øivind	6	7	4	7	4	2		**30**

Uke 17	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	7	7	6	5	2	1		**28**
Magnus	7	7	8	4	2			**28**
Øivind	7	7	6	6	2	4		**32**

Uke 18	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	7	6	8	8			**35**
Magnus	5	6	6	7	3	5		**32**
Øivind	4	6	8	7	5	5		**35**

Uke 19	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	5	6	5	4	4	1	3	**28**
Magnus	5	7	5	5	7	1	3	**33**
Øivind	8	7	8	8	3	1	3	**38**

Uke 20	Mandag	Tirsdag	Onsdag	Torsdag	Fredag	Lørdag	Søndag	Totalt
Sindre	6	7	7	6	5	4	3	**38**
Magnus	6	7	8	8	7	3	2	**41**
Øivind	9	9	9	7	5	6	3	**48**

Appendix D

Project assignment

TUSSA

Bachelor thesis for Digital Infrastructure and cyber security

Working title: Framework for secure data collection and through integration with various APIs

Client

Tussa IKT is a company in the Tussa group. We deliver managed ICT, cloud services, and security services mostly to companies based in Norway. Services are based on various SaaS and communication providers in addition to our own data centers and our own regional fiber network.

We are an ISO27001 certified MSP and ISP with head offices in Ørsta and office locations in Ålesund, Åndalsnes and Stryn.

Tussa IKT AS, Langemyra 6, 6160 Hovdebygda
Contact:
Kjetil Sætre, tel 90012640, kjetil.satre@tussa.no
Vigleik Hustadnes, tel 48118820, vigleik.hustadnes@tussa.no

Technical contact:
Benjamin Yndestad Aam, tel 97525789, benjamin.yndestad.aam@tussa.no

Assignment

The objective of the assignment is to research and improve the procedures for SaaS services integration using APIs, with a particular focus on risk assessment and securing the process around handling API credentials. Outline a solution that secures the data during transit, at rest, and define a best practice security strategy for managing access to scripts, code, API credentials and data in a multi-tenant environment. API credential vaults, test environments and SIEM are available.

Security requirements

- Encryption
- Access control
- Least privilege access control principle
- Auditing and logging
- Compliance with the customer's security policies/ISMS
- Security testing

Proof of Concept:

- Build a prototype of the proposed solution that implements the key features and security requirements identified in the research and development phase.
- Test the prototype with a dataset from Microsoft Intune, Cisco Secure Endpoint and Cisco Umbrella to demonstrate how the solution could be used in a production environment.
- Organize and compile data from the dataset into a structured format, such as a database or JSON file, for easy access and reference in future use.
- Evaluate and identify the result of the prototype and procedures, and suggest potential enhancements or limitations for future considerations.

Appendix E

Agreements

Electronic signature

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Yamin, Muhammad Mudassar	12.01.2023 11.57.27
(Identity verified with BankID (NO))	**Date of birth**
bankID	1992-03-24
	Signature method
	BankID (NO)

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Voll, Magnus Lekanger	14.01.2023 19.06.23
(Name entered manually by the signer) This document was signed with a SMS one-time password, sent to +4795526004	**Signature method**
SMS	SMS OTP

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Schonhowd , Sindre Davidsen	12.01.2023 12.27.33
(Name entered manually by the signer) This document was signed with a SMS one-time password, sent to +4748252086	**Signature method**
SMS	SMS OTP

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Wahlstrøm, Øivind	13.01.2023 10.44.36
(Identity verified with BankID (NO))	**Date of birth**
bankID	1999-10-04
	Signature method
	BankID (NO)

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Kalstad, Nils	20.01.2023 08.30.09
(Identity verified with BankID (NO))	**Date of birth**
bankID	1978-07-16
	Signature method
	BankID (NO)

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Driveklepp, Ivar	12.01.2023 10.37.42
	Date of birth

(Identity verified with BankID (NO))

1968-06-28
Signature method
BankID (NO)

NTNU

Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt for informasjonssikkerheit og kommunikasjonsteknologi
Veileder ved NTNU: Muhammad Mudassar Yamin muhammad.m.yamin@ntnu.no 96999968
Ekstern virksomhet: Tussa IKT AS Ekstern virksomhet sin kontaktperson, e-post og tlf.: Vigleik Hustadnes vigleik.hustadnes@tussa.no 48118820 Kjetil Sætre kjetil.satre@tussa.no 90012640 **Teknisk kontaktperson:** Benjamin Yndestad Aam benjamin.yndestad.aam@tussa.no 97525789
Student: Magnus Lekanger Voll magnuslv@stud.ntnu.no 955 26 004 Fødselsdato: 31.03.1997
Student: Sindre Davidsen Schonhowd sindreds@stud.ntnu.no 482 52 086 Fødselsdato: 19.05.1999
Student: Øivind Wahlstrøm oivindwa@stud.ntnu.no 948 09 046 Fødselsdato: 04.10.1999

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	X
Prosjektoppgave	
Annen oppgave	

Startdato: 5.1.2022
Sluttdato: 22.5.2022

Oppgavens arbeidstittel er: *Rammeverk for sikker datainnsamling og rapportering i ein sosioteknisk kontekst gjennom integrasjon mot ulike API*

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Ekstern virksomhet skal dekke følgende utgifter til utførelse av oppgaven: Reise og opphald etter behov og avtale

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[1]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten

[1] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

X	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

X	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer – digital signering

Instituttleder: Nils Kalstad nils.kalstad@ntnu.no tlf 45492425
Veileder ved NTNU: Muhammad Mudassar Yamin tlf 96999968
Ekstern virksomhet: Ivar Driveklepp ivar.driveklepp@tussa.no
Studenter: Magnus Voll, Sindre Schonhowd og Øivind Wahlstrøm

Electronic signature

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Yamin, Muhammad Mudassar	12.01.2023 11.58.19
(Identity verified with BankID (NO))	**Date of birth**
	1992-03-24
	Signature method
	BankID (NO)

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Lekanger Voll, Magnus	12.01.2023 14.24.15
(Name entered manually by the signer) This document was signed with a SMS one-time password, sent to +4795526004	**Signature method**
	SMS OTP

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Schonhowd, Sindre Davidsen	12.01.2023 12.36.06
(Name entered manually by the signer) This document was signed with a SMS one-time password, sent to +4748252086	**Signature method**
	SMS OTP

Signed by	**Date and time** (UTC+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna
Wahlstrøm, Øivind	13.01.2023 10.48.31
(Identity verified with BankID (NO))	**Date of birth**
	1999-10-04
	Signature method
	BankID (NO)

TUSSA

Brukaransvar for informasjonsverdiar eigd eller forvalta av Tussa IKT

Mottakarar:	Muhammad Mudassar Yamin Magnus Lekanger Voll Sindre Davidsen Schonhowd Øivind Wahlstrøm		Organisasjon: NTNU Gjøvik
Oppdragsgjevar:	Tussa IKT	Kontaktperson:	Vigleik Hustadnes

Mottakar får tilgang til:

Passord, API-nøklar, rutinar, prosessar og annan info i kategoriane Begrensa og Konfidensielt.

Føremål:

Arbeid med bacheloroppgåve som skal publiserast offentleg.

Varigheit/tidsavgrensing for oppdraget:

Januar-mai 2023

OBS Teieplikta gjeld også etter at oppdraget er utført

Ved å ta i bruk tilgangen/informasjonen aksepterer du følgjande ansvar:
- ✓ Informasjonen kan berre vidareformidlast etter avtale med Oppdragsgjevar
- ✓ Du skal sikre informasjonen mot tilgang for uvedkomande ved hjelp av gode rutinar, kontroll over alle kopiar av informasjonen og tilstrekkelege tekniske sikringstiltak som t.d. kryptert overføring og lagring
- ✓ At alle som får tilgang til informasjonen skal vere informerte om sitt ansvar og teieplikt etter denne avtalen og eventuell databehandlaravtale for personopplysningar, samt relevante lover
- ✓ Å slette alle kopiar av informasjonen når avtalen eller føremålet opphøyrer

Ved pålogging til system eller tenester gjeld også følgjande:
- ✓ Tildelt brukarkonto er personleg og skal ikkje formidlast til andre
- ✓ Om kollegaer eller tredjepart treng tilgang, skal dette godkjennast av oppdragsgjevar og deretter tildelast av Tussa IKT
- ✓ Du skal ikkje søke informasjon som ikkje er nødvendig for det konkrete oppdraget som skal utførast
- ✓ Du skal rapportere alt utført arbeid til oppdragsgjevar
- ✓ Endeutstyret skal vere oppdatert og tilstrekkeleg sikra med tiltak mot vondsinna programvare (malware)
- ✓ Når du ikkje lenger har bruk for tilgang, skal du informere oppdragsgjevar om dette, slik at tilgangen kan fjernast

Ved systemutvikling/integrasjon:
- ✓ Utviklarane skal sikre systema mot ”CWE Top 25 Most Dangerous Software Weaknesses” (https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html)

Appendix F

Security testing results

ZAP Scanning Report Of Our Web-application

A security scan of the login-portal.

Site: https://securityportal.tikt.no

Generated on Thu, 18 May 2023 13:35:15

Summary of Alerts

Risk Level	Number of Alerts
High	1
Medium	6
Low	3
Informational	3

Alerts

Name	Risk Level	Number of Instances
Cloud Metadata Potentially Exposed	High	1
CSP: Wildcard Directive	Medium	4
CSP: script-src unsafe-eval	Medium	4
CSP: script-src unsafe-inline	Medium	4
CSP: style-src unsafe-inline	Medium	4
Hidden File Found	Medium	4
Multiple X-Frame-Options Header Entries	Medium	2
Server Leaks Version Information via "Server" HTTP Response Header Field	Low	16
Strict-Transport-Security Header Not Set	Low	14
Timestamp Disclosure - Unix	Low	6
Information Disclosure - Suspicious Comments	Informational	24
Re-examine Cache-control Directives	Informational	4
User Agent Fuzzer	Informational	72

Alert Detail

High	Cloud Metadata Potentially Exposed
Description	The Cloud Metadata Attack attempts to abuse a misconfigured NGINX server in order to access the instance metadata maintained by cloud service providers such as AWS, GCP and Azure. All of these providers provide metadata via an internal unroutable IP address '169.254.169.254' - this can be exposed by incorrectly configured NGINX servers and accessed by using this IP address in the Host header field.

URL	https://securityportal.tikt.no/latest/meta-data/
Method	GET
Attack	169.254.169.254
Evidence	
Instances	1
Solution	Do not trust any user data in NGINX configs. In this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker.
Reference	https://www.nginx.com/blog/trust-no-one-perils-of-trusting-user-input/
CWE Id	
WASC Id	
Plugin Id	90034

Medium	CSP: Wildcard Directive
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
Instances	4
Solution	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference	http://www.w3.org/TR/CSP2/ http://www.w3.org/TR/CSP/ http://caniuse.com/#search=content+security+policy http://content-security-policy.com/ https://github.com/shapesecurity/salvation https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources

CWE Id	693
WASC Id	15
Plugin Id	10055

Medium	**CSP: script-src unsafe-eval**
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
Instances	4
Solution	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference	http://www.w3.org/TR/CSP2/ http://www.w3.org/TR/CSP/ http://caniuse.com/#search=content+security+policy http://content-security-policy.com/ https://github.com/shapesecurity/salvation https://developers.google.com/web/fundamentals/security /csp#policy_applies_to_a_wide_variety_of_resources
CWE Id	693
WASC Id	15
Plugin Id	10055

Medium	**CSP: script-src unsafe-inline**
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website

	owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
Instances	4
Solution	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference	http://www.w3.org/TR/CSP2/ http://www.w3.org/TR/CSP/ http://caniuse.com/#search=content+security+policy http://content-security-policy.com/ https://github.com/shapesecurity/salvation https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources
CWE Id	693
WASC Id	15
Plugin Id	10055

Medium	**CSP: style-src unsafe-inline**
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks. Including (but not limited to) Cross Site Scripting (XSS), and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page — covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;

URL	https://securityportal.tikt.no/
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	
Evidence	default-src 'self'; script-src 'self' 'unsafe-inline' 'unsafe-eval'; style-src 'self' 'unsafe-inline'; img-src 'self' data:;
Instances	4
Solution	Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.
Reference	http://www.w3.org/TR/CSP2/ http://www.w3.org/TR/CSP/ http://caniuse.com/#search=content+security+policy http://content-security-policy.com/ https://github.com/shapesecurity/salvation https://developers.google.com/web/fundamentals/security/csp#policy_applies_to_a_wide_variety_of_resources
CWE Id	693
WASC Id	15
Plugin Id	10055

Medium	**Hidden File Found**
Description	A sensitive file was identified as accessible or available. This may leak administrative, configuration, or credential information which can be leveraged by a malicious individual to further attack the system or conduct social engineering efforts.
URL	https://securityportal.tikt.no/._darcs
Method	GET
Attack	
Evidence	HTTP/1.1 302 Found
URL	https://securityportal.tikt.no/.bzr
Method	GET
Attack	
Evidence	HTTP/1.1 302 Found
URL	https://securityportal.tikt.no/.hg
Method	GET
Attack	
Evidence	HTTP/1.1 302 Found
URL	https://securityportal.tikt.no/BitKeeper
Method	GET

Attack	
Evidence	HTTP/1.1 302 Found
Instances	4
Solution	Consider whether or not the component is actually required in production, if it isn't then disable it. If it is then ensure access to it requires appropriate authentication and authorization, or limit exposure to internal systems or specific source IPs, etc.
Reference	https://blog.hboeck.de/archives/892-Introducing-Snallygaster-a-Tool-to-Scan-for-Secrets-on-Web-Servers.html
CWE Id	538
WASC Id	13
Plugin Id	40035

Medium	**Multiple X-Frame-Options Header Entries**
Description	X-Frame-Options (XFO) headers were found, a response with multiple XFO header entries may not be predictably treated by all user-agents.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	
Instances	2
Solution	Ensure only a single X-Frame-Options header is present in the response.
Reference	https://tools.ietf.org/html/rfc7034
CWE Id	1021
WASC Id	15
Plugin Id	10020

Low	**Server Leaks Version Information via "Server" HTTP Response Header Field**
Description	The web/application server is leaking version information via the "Server" HTTP response header. Access to such information may facilitate attackers identifying other vulnerabilities your web/application server is subject to.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/login
Method	GET
Attack	

Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/6291.cb5ce8837ec621d8b6a1.js
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/8683.9259ad853ca27103e2cc.js
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/app.c52aa5cf53b2693cadba.js
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/grafana.dark.922c73a268c5f56fe5fe.css
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/build/runtime.2434a49086cee1380c4e.js
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/img/apple-touch-icon.png
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/img/browserconfig.xml
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/public/img/fav32.png
Method	GET
Attack	
Evidence	nginx/1.24.0

URL	https://securityportal.tikt.no/public/img/grafana_mask_icon.svg
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/robots.txt
Method	GET
Attack	
Evidence	nginx/1.24.0
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	
Evidence	nginx/1.24.0
Instances	16
Solution	Ensure that your web server, application server, load balancer, etc. is configured to suppress the "Server" header or provide generic details.
Reference	http://httpd.apache.org/docs/current/mod/core.html#servertokens http://msdn.microsoft.com/en-us/library/ff648552.aspx#ht_urlscan_007 http://blogs.msdn.com/b/varunm/archive/2013/04/23/remove-unwanted-http-response-headers.aspx http://www.troyhunt.com/2012/02/shhh-dont-let-your-response-headers.html
CWE Id	200
WASC Id	13
Plugin Id	10036

Low	**Strict-Transport-Security Header Not Set**
Description	HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTPS connections (i.e. HTTP layered over TLS/SSL). HSTS is an IETF standards track protocol and is specified in RFC 6797.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/build/6291.cb5ce8837ec621d8b6a1.js
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	

URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/build/8683.9259ad853ca27103e2cc.js
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/build/app.c52aa5cf53b2693cadba.js
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/build/grafana.dark.922c73a268c5f56fe5fe.css
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/build/runtime.2434a49086cee1380c4e.js
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/img/apple-touch-icon.png
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/img/browserconfig.xml
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/img/fav32.png
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/public/img/grafana_mask_icon.svg
Method	GET
Attack	
Evidence	
URL	https://securityportal.tikt.no/robots.txt
Method	GET
Attack	
Evidence	
Instances	14

Solution	Ensure that your web server, application server, load balancer, etc. is configured to enforce Strict-Transport-Security.
Reference	https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html https://owasp.org/www-community/Security_Headers http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security http://caniuse.com/stricttransportsecurity http://tools.ietf.org/html/rfc6797
CWE Id	319
WASC Id	15
Plugin Id	10035

Low	**Timestamp Disclosure - Unix**
Description	A timestamp was disclosed by the application/web server - Unix
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	1682353621
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	1682353621
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	1431655765
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	1494410783
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	1494410983
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	1540483477
Instances	6
Solution	Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.
Reference	http://projects.webappsec.org/w/page/13246936/Information%20Leakage
CWE Id	200
WASC Id	13
Plugin Id	10096

Informational	Information Disclosure - Suspicious Comments
Description	The response appears to contain suspicious comments which may help an attacker. Note: Matches made within script blocks or files are against the entire content not only comments.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	TODO
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	TODO
URL	https://securityportal.tikt.no/public/build/6291.cb5ce8837ec621d8b6a1.js
Method	GET
Attack	
Evidence	Query
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	admin
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	administrator
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	bug
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	Db
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	from
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	query
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET

Attack	
Evidence	select
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	TODO
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	User
URL	https://securityportal.tikt.no/public/build/6749.32f81d7fa137df51b539.js
Method	GET
Attack	
Evidence	where
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	bug
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	db
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	from
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	QUERY
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	select
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	user
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	

Evidence	username
URL	https://securityportal.tikt.no/public/build/7490.cf2da2a42f577bdb1843.js
Method	GET
Attack	
Evidence	WHERE
URL	https://securityportal.tikt.no/public/build/8683.9259ad853ca27103e2cc.js
Method	GET
Attack	
Evidence	bug
URL	https://securityportal.tikt.no/public/build/8683.9259ad853ca27103e2cc.js
Method	GET
Attack	
Evidence	from
URL	https://securityportal.tikt.no/public/build/runtime.2434a49086cee1380c4e.js
Method	GET
Attack	
Evidence	query
Instances	24
Solution	Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.
Reference	
CWE Id	200
WASC Id	13
Plugin Id	10027

Informational	**Re-examine Cache-control Directives**
Description	The cache-control header has not been set properly or is missing, allowing the browser and proxies to cache content. For static assets like css, js, or image files this might be intended, however, the resources should be reviewed to ensure that no sensitive content will be cached.
URL	https://securityportal.tikt.no
Method	GET
Attack	
Evidence	no-store
URL	https://securityportal.tikt.no/login
Method	GET
Attack	
Evidence	no-store
URL	https://securityportal.tikt.no/public/img/browserconfig.xml
Method	GET
Attack	
Evidence	public, max-age=3600
URL	https://securityportal.tikt.no/robots.txt
Method	GET

Attack	
Evidence	public, max-age=3600
Instances	4
Solution	For secure content, ensure the cache-control HTTP header is set with "no-cache, no-store, must-revalidate". If an asset should be cached consider setting the directives "public, max-age, immutable".
Reference	https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html#web-content-caching https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Cache-Control https://grayduck.mn/2021/09/13/cache-control-recommendations/
CWE Id	525
WASC Id	13
Plugin Id	10015

Informational	**User Agent Fuzzer**
Description	Check for differences in response based on fuzzed User Agent (eg. mobile sites, access as a Search Engine Crawler). Compares the response statuscode and the hashcode of the response body with the original response.
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3739.0 Safari/537.36 Edg/75.0.109.0
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Evidence	
URL	https://securityportal.tikt.no

Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/91.0
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (iPhone; CPU iPhone OS 8_0_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12A366 Safari/600.1.4
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Evidence	
URL	https://securityportal.tikt.no
Method	GET
Attack	msnbot/1.1 (+http://search.msn.com/msnbot.htm)
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko
Evidence	
URL	https://securityportal.tikt.no/

Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3739.0 Safari/537.36 Edg/75.0.109.0
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/91.0
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (iPhone; CPU iPhone OS 8_0_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12A366 Safari/600.1.4
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Evidence	
URL	https://securityportal.tikt.no/
Method	GET
Attack	msnbot/1.1 (+http://search.msn.com/msnbot.htm)
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Evidence	

URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3739.0 Safari/537.36 Edg/75.0.109.0
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/91.0
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (iPhone; CPU iPhone OS 8_0_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12A366 Safari/600.1.4
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Evidence	
URL	https://securityportal.tikt.no/public
Method	GET
Attack	msnbot/1.1 (+http://search.msn.com/msnbot.htm)

Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3739.0 Safari/537.36 Edg/75.0.109.0
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/91.0
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
	Mozilla/5.0 (iPhone; CPU iPhone OS 8_0_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML,

Attack	like Gecko) Version/8.0 Mobile/12A366 Safari/600.1.4
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Evidence	
URL	https://securityportal.tikt.no/public/build
Method	GET
Attack	msnbot/1.1 (+http://search.msn.com/msnbot.htm)
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3739.0 Safari/537.36 Edg/75.0.109.0
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/91.0
Evidence	
URL	https://securityportal.tikt.no/public/img

Method	GET
Attack	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (iPhone; CPU iPhone OS 8_0_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12A366 Safari/600.1.4
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Evidence	
URL	https://securityportal.tikt.no/public/img
Method	GET
Attack	msnbot/1.1 (+http://search.msn.com/msnbot.htm)
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0)
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Trident/7.0; rv:11.0) like Gecko
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/75.0.3739.0 Safari/537.36 Edg/75.0.109.0
Evidence	

URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/91.0
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html)
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp)
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (iPhone; CPU iPhone OS 8_0_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0 Mobile/12A366 Safari/600.1.4
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	Mozilla/5.0 (iPhone; U; CPU iPhone OS 3_0 like Mac OS X; en-us) AppleWebKit/528.18 (KHTML, like Gecko) Version/4.0 Mobile/7A341 Safari/528.16
Evidence	
URL	https://securityportal.tikt.no/sitemap.xml
Method	GET
Attack	msnbot/1.1 (+http://search.msn.com/msnbot.htm)
Evidence	
Instances	72
Solution	
Reference	https://owasp.org/wstg
CWE Id	
WASC Id	
Plugin Id	10104

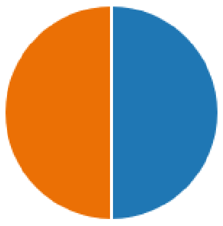
Appendix G

User test results

1. How satisfied are you with the final outcome of the project?

Flere detaljer

●	Very satisfied	1
●	Somewhat satisfied	1
●	Neither satisfied nor dissatisfied	0
●	Somewhat dissatisfied	0
●	Very dissatisfied	0



2. Do you have any commets you want to add?

Flere detaljer

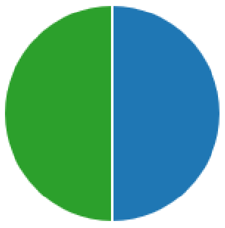
1
Svar

Siste svar
"Looking for some specific recommendations on risk mitigation when using ...

3. Will it be usable in the your production environment?

Flere detaljer

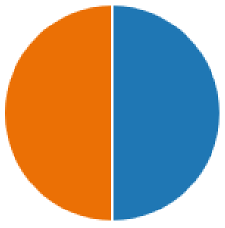
●	Yes	1
●	No	0
●	Maybe	1



4. Do you think the solution needs user training to use when it is in production?

Flere detaljer

●	Yes	1
●	No	1



5. What do you think will be the main challenge with the solution when its in production?

Flere detaljer

2
Svar

Siste svar
"Don't know of any big challenges, of course we must do some adaprion work"
"Don't see any direct challenges"

6. Do you have any feedback or suggestions for what could be improved?

Flere detaljer

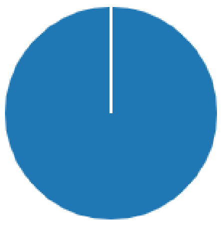
2
Svar

Siste svar
"No"
"No"

7. Did the project meet your initial goals and requirements?

Flere detaljer

●	Yes	2
●	No	0



8. Do you have any commets you want to add?

Flere detaljer

1
Svar

Siste svar
"It seems to me that the students are skilled, well organized and hard-worki...

Appendix H

Risk assessment for using local computers

Risk-assessment of private PC in work environment **Første vurdering dato:** 15.02.2023 **Oppdatert dato:**

ID	NR	**Risikobeskrivelse** Stikkord om: (1) innledende hendelse(r) (2) informasjonssikkerhetsbruddet (3) de uønskede konsekvensene som kan oppstå	**Begrunnelse for konsekvensvurdering**	**Initiell Konsekvens**	**Initiell sannsynlighet**	**Begrunnelse for vurdering av tilhørende sannsynlighet**
R1	1	1. increased risk of data breach, viruses, malware and other cyber-attacks 2. Company owned computers usually have a higher level of security features implemented than a privet PC. 3. infection of other workpace PCs and/or create a backdoor into the company. Steal API credentials	Exfiltrate, Alter and view data without detection from Tussas workspace. Spread more malicious code into their internal network. steal API credentials to access information.	Stor	Svært liten	It is not public information that we collaborate, which makes it hard to find working connection between bachelor writers and Tussa. We are cybersecurity students and have security culture integrated in our daily life. Only connection to Tussa workspace is through the VM which is seperated onto an isolted VLAN. API crendetials are only used to generate test data.
R2	2	1. Poor password management and encryption. 2. Unwanted persons gain access to API credentials. 3. Gain access to the data they can retrive.	Only test data	Svært liten	Svært liten	API crendetials are only used to generate test. General cybersecurity knowledge of how to store API credentials.
R3	3	1. No surveillance or managed control of a private PC. 2. Harder to discover security breach as well as when a breach has occurred. 3. Advisories go undetected	Stealing API credentials, or data without detection. Can't control the PC and isolate it in case of breach.	Stor	Liten	API crendetials are only used to generate test. General cybersecurity knowledge of how to store API credentials. Private PC does not have access to any importantinformation related to Tussa.
R4	4	1. Access control 2. Tussa has little to no control of whom may have access to the private PC 3.An unauthorasied person can gain access to restricted information.	Someone steals the PC or someone may know the password to the PC. Multiple users share same files on PC.	Liten	Svært liten	We see the most likely scenario is that someone steals the PC, But we consider that as very unlikely.
R5	5	1. Compliance Violations 2. Personal computers might be excluded from rules and guidelines that apply to company-owned computers 3. raising the possibility of privacy, security, and other compliance rules being broken.	worst case: breach GDPR rules.	Svært liten	Liten	We have recived severel docuements related guidelines different topics. These guidelines should not be broken, but might happend due to an accident.

Initielt risikonivå	Handtering - spesifiser på arkfane Tiltak	Forventa konsekvens etter tiltak	Forventa sannsynlighet etter tiltak	Planlagt risikonivå	Risikoeier	Gjeldende konsekvens	Gjeldende sannsynlighet	Gjeldende restrisiko
Svært liten	1, 2, 4, 5, and 6	Liten	Svært liten	Svært liten		Stor	Svært liten	Svært liten
Svært liten	1, 2, 4, 5, and 6	Svært liten	Svært liten	Svært liten		Svært liten	Svært liten	Svært liten
Liten	1, 2, 4, 5, and 6	Svært liten	Svært liten	Svært liten		Stor	Liten	Liten
Svært liten	1, 3, and 5	Svært liten	Svært liten	Svært liten		Liten	Svært liten	Svært liten
Svært liten	4	Liten	Svært liten	Svært liten		Svært liten	Stor	Svært liten

Initiell risiko

Sannsynlighet	Svært liten	Liten	Stor	Svært stor
Svært stor	🟨	🟧	🟥	🟥
Stor	🟩	🟨	🟧	🟥
Liten	🟩 R5	🟩	🟨 R3	🟧
Svært liten	🟩 R2	🟩 R4	🟩 R1	🟨

Konsekvens

Forventa/planlagt risiko

Sannsynlighet	1	2	3	4
	Svært liten	Liten	Stor	Svært stor
Svært stor	🟨	🟧	🟥	🟥
Stor	🟩	🟨	🟧	🟥
Liten	🟩	🟩	🟨	🟧
Svært liten	🟩 R2,R3,R4	🟩 R1,R5	🟩	🟨

Konsekvens

Gjeldande oppdatert risiko

Sannsynlighet	1	2	3	4
	Svært liten	Liten	Stor	Svært stor
Svært stor	🟨	🟧	🟥	🟥
Stor	🟩	🟨	🟧	🟥
Liten	🟩 R5	🟩	🟨 R3	🟧
Svært liten	🟩 R2	🟩 R4	🟩 R1	🟨

Konsekvens

Tiltak ID eller change nr	**Beskrivelse**	**Status**
1	Password manager	Forslag
2	store API credentials in a secure place with encryption	Forslag
3	Encrypted hardisk	Forslag
4	Enrolled into SOC overwatch (intune, umbrella or amp)	Forslag
5	MFA to logg onto private PC	Forslag
6	secure network (VPN or Proxy)	Forslag

Appendix I

Tussa's internal guidelines

Systemutvikling - integrasjon - script- DevOps

Ansvar for oppdatering : Vigleik Hustadnes

Kapittel i ISO27002:

A.14.1.1 Behovsanalyse og beskrivelse av krav til informasjonssikkerhet

A.14.2.7 Utkontraktert utvikling

Overordna krav

Retningslinjer for sikkerheit i digitalisering

Retningslinjer for identitets- og tilgangskontroll

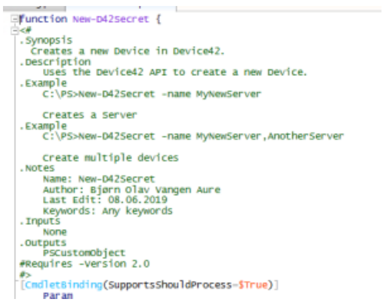
DevOps

Script/kode som er en sentral del av en driftkritisk prosess skal være dokumentert/beskrevet i dokumentasjonen for den gjeldende prosessen. Legg gjerne inn en kopi av scriptet/koden der du dokumenterer dette, forklar også hvordan dette blir kjørt. (eks, manuelt, scheduled task o.l.)

Det skal som standard ikke lagres passord eller API nøkler i klartekst, dette gjelder også for script. I noen tilfeller er det selvsagt ikke mulig å komme videre uten å lagre passord i klartekst, det kan være f.eks et programmeringsspråk som ikke støtter kryptering av passord, i disse tilfellene skal DevOps teamet diskutere og avgjøre sammen med sikkerhetsavdelingen hvordan vi løser dette på en best mulig måte som vi kan godta som "sikker nok".

Det skal alltid brukes kommentarer når vi skriver kode, beskriv så godt du kan hva som blir gjort og hvorfor. Legg også gjerne inn informasjon om dato, funksjon og forfatter i toppen av scriptet ditt slik at den neste som skal se på dette får mest mulig info.

Eksempel på kommentar for en PowerShell funksjon



```
function New-D42Secret {
<#
.Synopsis
  Creates a new Device in Device42.
.Description
    Uses the Device42 API to create a new Device.
.Example
    C:\PS>New-D42Secret -name MyNewServer

    Creates a Server
.Example
    C:\PS>New-D42Secret -name MyNewServer,AnotherServer

    Create multiple devices
.Notes
    Name: New-D42Secret
    Author: Bjørn Olav Vangen Aure
    Last Edit: 08.06.2019
    Keywords: Any keywords
.Inputs
    None
.Outputs
    PSCustomObject
#Requires -Version 2.0
#>
[CmdletBinding(SupportsShouldProcess=$True)]
    Param
```

PowerShell

For PowerShell **skal** alle passord og nøkler lagres som en SecureString slik at denne kan importeres og benyttes videre for autentisering på en sikker måte.

Det skal også benyttes nyeste versjon av PowerShell der det er mulig, PowerShell versjon 2.0 skal avinstalleres.

PowerShell - SecretManagement

PowerShell signering av script - Code sign

Tradisjonell systemutvikling

Desse krava gjeld ved systemutvikling uavhengig av om det blir gjort av interne eller innleigde ressursar.

Sjå også Rutiner for bruk av leverandørar og partnarar for Tussa IKT om avtale med leverandøren

Utarbeiding av kravspesifikasjon

1. Beskriv funksjonalitetskrav
2. Vurder konsekvensar ved potensielle uønska hendingar, bruk t.d. desse spørsmåla
   a. Kva kategoriar og mengder av informasjon og personopplysningar skal handterast i systemet
   b. Konsekvensar om uvedkomande får tilgang til informasjonen/systemet
   c. Kva om brukarane ikkje får tilgang til systemet, eller om det blir ustabilt
   d. Kva om informasjonen ikkje vert oppdatert eller synkronisert med master-data, om gamle data blir liggande i årevis, eller om det er store feil i informasjonen
3. Angrepsflate

    a. Eksponering for angrep frå uvedkomande/uautoriserte brukarar
    b. Kva skade kan autentiserte brukarar i systemet gjere.
4. Beskriv sikkerheitskrav og vurder minimum dette
    a. Kva kan og skal gjerast for å redusere eller unngå risikoane under pkt 2 og 3
    b. Autentisering: Om SSO og MFA ikkje kan brukast, skal gode grunnar for dette og evt kompenserande tiltak dokumenterast
    c. Autorisering: Trengst det ulike roller og tilgangsstyring i systemet (admin og ulike brukarroller) ?
    d. Integrasjon/kommunikasjon med andre system skal sikrast tilfredsstillande, inkludert passord/nøklar og muligheit for misbruk av desse

Systemarkitektur og design

Skal dokumenterast og godkjennast opp mot kravspesifikasjonen før implementeringsarbeidet startar.

Følgjande skal brukast i denne fasen, der det er relevant:

- http://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html
- Secure Application Model framework ved integrasjon mot O365

Akseptansekriterier og behov for testing

Skal dokumenterast før implementeringsarbeidet startar, kan utvidast i implementasjonsfasen

Implementering, testing og dokumentasjon

Før systemet går i produksjon skal følgjande dokumenterast

Beskrivelse av risikoar og implementerte sikringstiltak inkludert

- sikkerheitskrava i kravspesifikasjonen
- Mitigering av http://cwe.mitre.org/top25/archive/2019/2019_cwe_top25.html i den grad dei er relevante
- Testresultat og samsvar med kravspesifikasjon og akseptansekrav
- Plan for systemvedlikehald, support og sikkerheitspatching

TUSSA

Dato: 20220822

Retningslinjer for identitets- og tilgangsstyring Tussa IKT

Innhald

1. Dokumentforvalting

Målgruppe: Alle leiarar i Tussa IKT, systemeigarar og systemansvarlege
Innhald: Krav til styring av brukaridentitet og tilgang til IT-system og informasjonsverdiar i Tussa IKT
Konfidensialitet: Begrensa, jfr «Retningslinjer for lagring og deling av informasjon»
Godkjent: 22.08.2022 Ivar Driveklepp

1.1 Endringslogg

Dato	**Endringar**
20220822	Header: Lagt til dato. Kap 1: Nytt avsnitt med endringslogg.

2. Føremål med retningslinjene

Kontroll på identiteten til brukarar samt korrekt tildeling av brukarane sine tilgangsrettigheiter skal sikre autorisert tilgang til Tussa IKT sine informasjonsverdiar. Dette dokumentet regulerer både fysisk og logisk tilgang til alle typar informasjonsverdiar i alle deler av organisasjonen. Tilgang for leverandørar og partnerar er også omfatta av desse retningslinjene. Følgjande hovudprinsipp skal leggast til grunn:

- Det skal til ei kvar tid vere korrekt identifisert person som er registrert og gitt tilgang til Tussa IKT sine informasjonsverdiar.
- Det skal sikrast god styring av tilgang til informasjonsverdiar med korrekte rettigheiter basert på behov.
- All bruk av Tussa IKT sine informasjonsverdiar skal kunne førast tilbake til eit individ
- Tilgang som ikkje kan tildelast i samsvar med krava i dette dokumentet skal berre brukast unntaksvis, og skal alltid dokumenterast og godkjennast av nærmaste leiar

3. Krav til identitetsstyring

3.1 Handtering av brukaridentitetar

Identiteten til brukarane skal kontrollerast før dei får tilgang til Tussa IKT sine informasjonsverdiar.

Det skal dokumenterast ein prosess for registrering og fjerning av interne og eksterne brukarar. Tilgang for innleigde og midlertidige brukarar skal vere tidsavgrensa. Prosessen skal inkludere rutinar for tildeling og utlevering av autentiserings-informasjon (som passord m.m.) til brukarane. Prosessen skal også inkludere administrative eller tekniske tiltak for å unngå misbruk av roller. Dette kan vere fordeling av ansvar når det er hensiktsmessig, eller andre kompenserande tiltak.

Autoritativt system for brukarar skal definerast og brukast som grunnlag for identitets-styring.

Alle som får utlevert påloggingsinformasjon og/eller adgangskort, skal først signere teieplikterklæring og arbeidsavtale eller tilsvarande. Dei skal også oppfylle krava i datadisiplinerklæringa, som skal inkludere krav til handtering av personleg påloggingsinformasjon.

Alle tilsette som skal ha tilgang til Tussa IKT sine informasjonsverdiar skal gjennomføre obligatorisk opplæring i informasjonssikkerheit. Nærmaste leiar er ansvarleg for at slik opplæring blir gitt.

Systemeigar for det enkelte system skal sørge for sikker styring av påloggingsinformasjon til brukarar, jfr rollebeskrivelsen i «Retningslinjer for styring av informasjonssikkerheit». Det skal dokumenterast kva tiltak applikasjonen/systemet brukar for å ivareta sikker pålogging.

Ved etablering av nye system, skal støtte for single-sign-on vere eit kriterium som blir vektlagt.

4. Krav til tilgangsstyring

4.1 Autorisering

Tilgang til Tussa IKT sine informasjonsverdiar skal definerast på grunnlag av den enkelte si rolle og tilhøyring til Tussa IKT. Den enkelte leiar skal sikre rett tildeling av tilgangsrettigheiter for tilsette som han/ho har ansvar for.

Dei som godkjenner tilgang, skal ta omsyn til relevante lover, kundekrav, klassifisering og andre sikkerheitskrav som gjeld for den enkelte applikasjonen/systemet, og for informasjonsverdiar som er tilgjengelege igjennom applikasjonen/systemet.

Prinsippet for behovsstyrt tilgang gjeld både fysisk adgangskontroll, tilgang til applikasjonar og informasjon, og nettverkstilgang.

Systemeigar skal definere rutine for bestilling, godkjenning og tildeling av tilgang i sine system. Når ikkje anna er avtalt og dokumentert, skal nærmaste leiar godkjenne tildeling av tilgang for sine tilsette.

Brukarrettigheiter for leverandørar/forretningspartnarar skal tildelast i samsvar med den gjeldande avtalen. Eigaren av avtalen skal sikre rett spesifisering av tilgangsrettigheiter i avtalen.

Bestemmelsar for medlemskap i tilgangsgrupper skal dokumenterast.

4.2 Autentisering

Val av autentiseringsmekanismer skal tilpassast sikringsnivået som gjeld for informasjonsverdien som skal beskyttast, og i samsvar med «Retningslinjer for lagring og deling av informasjon». For eksempel er passord i dei fleste tilfelle utilstrekkeleg som autentiseringsmekanisme for informasjon som er tilgjengeleg frå internett.

Datadisiplinerklæringa skal inkludere retningslinjer for passordkvalitet.

5. Tilgang til nettverk og nettverkstenester

Brukarar skal berre få tilgang til dei deler av nettverk og nettverkstenester som dei har bruk for og er autorisert til å bruke. Dette gjeld både LAN, WLAN og VPN-tilgang. Sjå kapittel 4.1 om autorisering.

5.1 Styring, sikring og kontroll av nettverk

Ansvar for dei ulike nettverks-tenestene som Tussa IKT brukar, og tilhøyrande sikringsmekanismar, skal plasserast. Som minimum skal tenestene oppfylle følgjande krav:

- Rutinar for drift og overvaking skal dokumenterast.
- Kabla og trådlaust klientnettverk og VPN-tilgang skal sikrast med autentisering av brukarar eller utstyr. Når intern informasjon blir overført over internett eller trådlaust nett skal den krypterast med robuste algoritmar.
- Nettverks-segment med muligheit for administrasjon av infrastruktur skal vere skilt frå nettverk for vanleg brukartilgang, og skal sikrast med strengare tiltak.
- Segmentering av nettverk skal brukast for å hindre innsyn på tvers av ulike kundar, når det er relevant, og i den grad det er nødvendig for å oppfylle avtalefesta krav frå kundar.
- Nettverkskommunikasjon mellom interne nett og eksterne partnerar skal filtrerast i brannmur
- Logging i samsvar med kap. 9.1

6. Privilegert tilgang

Brukarar med privilegerte rettigheiter omfattar brukarar med rot- eller administratorrettigheiter («administratorar») og brukarar med spesielle rettigheiter i applikasjonar, system og tenester («superbrukarar»). Kontroll over denne typen brukarar er særleg viktig. Systemeigar skal dokumentere og følgje opp ei rutine for provisjonering, bruk og kontroll i det enkelte system eller grupper av system. Følgjande krav skal gjelde rutinane:

- Tildeling av privilegerte rettigheiter skal avgrensast til dei som treng det, jfr kap 4.1 Autorisering
- «Servicekontoar» skal ikkje brukast for fysiske personar
- Felles brukarnamn skal unngåast så langt det er råd. Ved unntak frå denne regelen skal systemeigar sørge for rutinemessig skifte av passord og sikring av dette.
- Bruk av privilegerte rettigheiter skal alltid kunne sporast tilbake til eit individ.
- Logging og rapportering i samsvar med kap. 9

Ved dagleg arbeid som ikkje krev utvida rettigheiter, skal det brukast ein standard brukarkonto. Pålogging med privilegerte brukarkontoar skal berre gjerast når det er nødvendig.

7. Tilgangsstyring i system og applikasjonar

Kapittel 4.1 om autorisering gjeld også for tilgang til applikasjonar og system, og differensiering av tilgang for brukarane i applikasjonen. Det er systemeigar som har ansvaret for å oppfylle krava i dette dokumentet og i «Retningslinjer for styring av informasjonssikkerheit» med vedlegg, for sitt system. Systemansvarleg sitt ansvar inkluderer å velge sikringstiltak som påloggingsprosedyrer, passordstyring, kontroll med privilegerte hjelpeprogram og sikring av API for applikasjonen.

8. Fysisk tilgangsstyring

Informasjonsverdiar og fasilitetar skal sikrast med fysiske tiltak. Fysiske sikkerheitssoner skal definerast, og sonene skal sikrast med relevante sikkerheitstiltak som inkluderer behovsbasert tilgangskontroll, overvaking og logging. Ansvaret for soneinndeling og for sikkerheitstiltaka skal plasserast i samsvar med «Retningslinjer for styring av informasjonssikkerheit» med vedlegg.

9. Oppfølging og revisjon

Det skal etablerast tiltak for å oppdage uautorisert tilgang eller forsøk på tilgang til Tussa IKT sine informasjonsverdiar. Dersom det vert oppdaga at det har skjedd ein uautorisert tilgang, skal dette handterast som eit avvik.

9.1 Krav til logging

Aktivitet relatert til vanlege brukarkontoar og kontoar med privilegerte rettigheiter skal loggast og oppbevarast i samsvar med gjeldande lover. Dette gjeld brukaraktivitet som er relevant for sikkerheit, og endring av påloggingsinformasjon. Det er systemeigar sitt ansvar å vurdere kva som må loggast og korleis loggane skal handterast, basert på risiko.

Viktige sikkerheitsloggar inkludert bruk av privilegerte kontoar skal lagrast og vernast mot uautorisert tilgang i samsvar med prinsippet om fordeling av ansvar.

Historiske opplysningar om brukaridentitetar, tilgangsrettigheiter og autentisering skal oppbevarast i samsvar med krav frå kundar, styresmakter og konsern.

Metodar for systematisk gjennomgang og analyse av sikkerheitsloggar skal implementerast, for å avdekke uønskt aktivitet.

9.2 Revisjon

Når det gjeld fjerning/endring og avvikling av tilgangsrettigheiter skal følgjande krav gjelde:

- Systemeigar er ansvarleg for at rutine for fjerning, endring og avvikling av tilgang for den aktuelle applikasjon eller system vert dokumentert og følgt.
- Når det er føremålstenleg, skal tilgangsgrupper som gir tilgang til Tussa IKT sine informasjonsverdiar ha ein ansvarleg eigar som periodisk reviderer kven som skal vere medlem i gruppa.
- Tildelte tilgangsrettigheiter skal gjennomgåast minst årleg, og gjennomgangen skal loggførast. Privilegerte tilgangsrettigheiter skal gjennomgåast minst kvart halvår. Tilgangen skal fjernast når brukarane ikkje lenger har bruk for den i arbeidet.

Appendix J

Stage one of the infrastructure

Stage one of the infrastructure

Working throughout the project, the group has concluded to call the following achieved architecture for stage one. Stage one represents an elementary solution to the task provided by the client, Tussa. It fulfils the basic requirements, and the main task which is to collect unique information about devices from different vendors and display it as a collective information on a single platform.

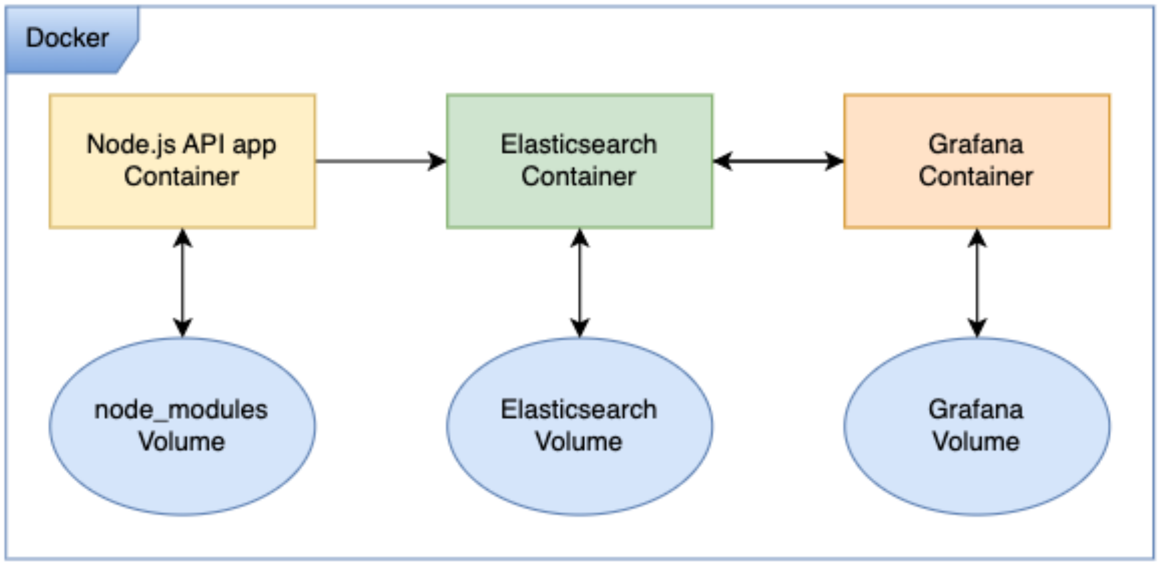


Fig x.x, The picture displays stage on of the infrastructure

The infrastructure consists of three main parts, all running in a single environment, i.e. on a single virtual machine. Each part runs as a container with a volume attached to it for persistence purposes, which will be described further below.

Containers

containerizing our applications present numerous benefits, which is why we have determined to utilize it. Containers are standalone executable packages which consist of everything needed to run an application, like libraries, configuration files, binary code, and executables. Containers do not persist of an operating system image, unlike machine or server virtualization. This makes them lightweight and appropriate for transport.

Applications running as a container has the abilities to easily be deployed onto multiple different environments, without the concern of configuration issues and dependencies. This trait can prove useful for Tussa in the future, as they don't have to think about which system they need to utilize for further development, if they decide to do so later. This technology is also suitable for us as the group work with two different operating systems. This makes building and deploying the application across the group versatile.

Containerizing application additionally offers to isolate each of the applications to its own environment while running on the same machine. This enhances the ability to manage each of the applications individually, as well as ensuring that they will not interfere with each other in an unwanted manner[1].

Here is a short description of the three containers, see figure x.x above. The first container marked in yellow represents the Node.js API app. This container holds the code that utilize APIs to fetch the required security data about devices from each vendor. It also sorts this information into a single data structure which is sent to the Elasticsearch container. The Elasticsearch container coloured in green is responsible for storing data it got from the Node.js API app in an index it in a JSON format. The last container marked in orange is running Grafana. Grafana is used to fetch the data stored in the Elasticsearch container and display it to the user in a dashboard. Separating the responsibilities of data collection, storage, and visualization, into tree individual entities make the system more flexible and modular, that can be managed independently.

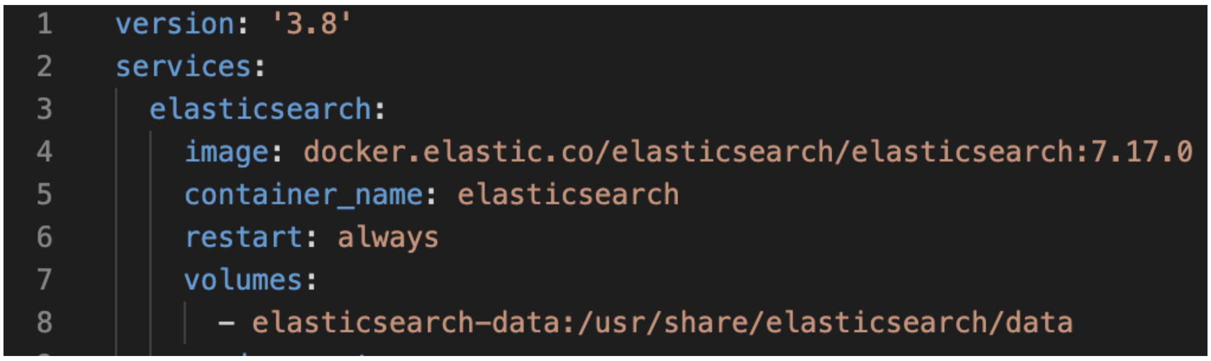
[1] NetApp. Required: (09.03.2023). What are containers? https://www.netapp.com/devops-solutions/what-are-containers/

Volume

This section will explain why we have attached a volume to each of the containers, and what it is used for. It is important to highlight that connection a volume to the containers will make each of the containers stateful rather than stateless. Stateful and stateless are different approaches which offers individual capabilities and features.

Statefulness means that the container will store everything that is done on it, which means storing the state of the container. A deeper understanding on how this will profit our solution is described in the next paragraph. Statefulness also amplifies the speed of the information flow. A stateless container will do the opposite, and not store any information on it. A stateless approach offers the benefits of enhanced horizontal scaling capabilities. For the program to be able to store information and be stateless, the implementation of a different persistent service is needed e.g. a database[1]. This approach will be discussed later in the thesis for future development.

A stateful container will prevent data loss from occurring in case a container goes down unintentionally. Attaching a volume to the containers will store the state of the container before it goes down and reload it when it is running again. The container will be unavailable for the duration it is down but will fully function as before once it is back up again due to the volume. In other words, a volume is a way to persistent data for containers, and to share it between them. They are independent form containers which gives them the ability to save data even if the container goes down. This means that the data is stored outside of the container. Although we highlight the importance of adding the field "restart: always" in the docker compose file as shown beneath on line 6. See fig x.x. Reason for this is to enable the containers to restart again automatically if it goes down. Without this field, you would have to restart them manually for it to be restored to the old state. Line 8 also shows the path of the volume.



```
version: '3.8'
services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.17.0
    container_name: elasticsearch
    restart: always
    volumes:
      - elasticsearch-data:/usr/share/elasticsearch/data
```

Fig x.x, The picture displays our docker-compose file.

The Node.js API app does not store any information, but it utilizes a container to store its node modules which is crucial for it to function.
The Elasticsearch container only receive data from the Node.js API container once a day, due to the implementation of the cron job in the Node.js API app. Therefore, if the container does not have a volume attached to it and it goes down it will have no date to forward to Grafana before the Node.js API container sends new data to it. The potential outcome could be that Grafana has no date to display for almost an entire day.
Grafana container uses a volume to store its User interface (UI) design, which is configured with Graphical User Interface (GUI) after the container is deployed. Meaning that the dashboards configured to display data in a certain way will not disappear if the container goes down.

The disadvantage utilizing a docker compose file is that it is impossible to roll out image updates without down time. If you have a new version of an image you want to implement, you need to first bring the containers down, then change out the image with the new one, and then start the containers again. The reason for this is that docker compose was never designed for the concept of always uptime. A way of fixing this problem is to use a mulit-server orchestration tool like docker swarm and Kubernetes. These solutions will be compared and presented later on[2]

[1] BasuMallick. 20.09.2022. Stateful vs. Stateless: Understanding the key differences. https://www.spiceworks.com/tech/cloud/articles/stateful-vs-stateless/
[2] Bert Fisher. 27.02.2019. Docker compose or swarm for a single server. Docker Compose or Swarm For A Single Server

Appendix K

Iterations

Iterations retrospective

Iteration one

a. What went well?
   - Started early with planning and working on the application.
   - Establishing common understanding of the task ahead with all parties. i.e., Advisory, Client (Tussa), and the group.

b. What didn't go well?
   - Misunderstanding between the parties concerning the goal of the assignment.
   - Confusion regarding where to start and get an overview of what needs to be done in such a huge assignment.

c. What can be improved?
   - Better communication between the Client and Advisor. The bachelor group should have set up a meeting earlier with all the parties, so everyone understood the project better.
   - More frequent meetings in the beginning to get and overview of what needs to be done.

Iteration two

a. What went well?
   - Planning the infrastructure to get an overview of how the application will be built.
   - Communication with the parties has gone well, with a clear communication line.

b. What didn't go well?
   - Communication within the bachelor group could have been better for everyone to understand how the application works.
   - Dividing of work assignment and its workload. Some tasks are unequally distributed.
   - All the team members could be better to take initiative.

c. What can be improved?
   - Communication within the group. Update each other frequently of what they are working on to keep everyone in the loop.
   - Use Jira more actively, even for smaller tasks that might be added at a later stage.

Iteration three

a. What went well?
    - The internal communication worked better, and the Jira-solution worked well to always keep everyone up to speed.
    - Good communication with the client with regular weekly meetings.
    - The work on the solution went well, and the team have done a lot of developing on the solution.

b. What didn't go well?
    - For a period of time we lacked some enhanced communication with our advisor as he canceled three weeks with meeting in a row due to travel. This led to impaired communication as it lacked verbal exchange, which is often easier when asking more complex questions.
    - The bachelor group could be better on digging into answers from the advisor.

c. What can be improved?
    - Our willingness to ask multiple questions to get the right answers from our advisor. Meaning that we don't just take his first answer as a final answer.
    - The whole group needs to keep a better focus when working on the project. Be dedicated to the task when working on it.

Iteration four

a. What went well?
    - The problems regarding the communication with the advisor have been a lot better.
    - The team has done a lot of development, and the application is really starting to take shape.

a. What didn't go well?
    - The team need to focus more on the thesis. We have set a deadline for the development of the application (01/05). From that point we need all the focus on the thesis.

a. What can be improved?
    - The communication within the group is somewhat lacking. We need to get better at asking each other for help if we are stuck on a problem.

Iteration five

a. What went well?
    - Every member worked hard the last couple of weeks.
    - Kept to the deadlines we sat for our self, regarding development.
    - Optimized our time usage.

a. What didn't go well?
    - The members always find something that could be better and are never truly satisfied with the work.

a. What can be improved?
    - Everyone could be more satisfied with their work, and not always focus on what could be better.

NTNU

Norwegian University of
Science and Technology