Bachelor's project

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

Øyvor Vejlgaard Sørensen
Harald Aarseth
Kristian Fjelstad
Gloria Treider

# RSCCI: A User-Friendly Web Application for Evaluating Company Security Regulations and Cloud Security

Bachelor's project in IT-Operations and Information Security
Supervisor: Jia-Chun Lin

May 2021

NTNU
Norwegian University of
Science and Technology

Øyvor Vejlgaard Sørensen
Harald Aarseth
Kristian Fjelstad
Gloria Treider

# RSCCI: A User-Friendly Web Application for Evaluating Company Security Regulations and Cloud Security

Bachelor's project in IT-Operations and Information Security
Supervisor: Jia-Chun Lin
May 2021

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

NTNU
Kunnskap for en bedre verden

# Abstract

Axeptia Credit Intelligence AS is a small company in the financial industry. Like many companies nowadays, they use cloud services to host their infrastructure and software. Axeptia aims to help other companies get insight and control of their customer credit portfolio with details about credit risk, accounts receivables, and debt collection. Axeptia wants a service which is able to easily evaluate and control their security and regulatory compliance. This service should map generic regulatory requirements and generate understandable reports so that non-technical users are able to use the service without any effort. Therefore, in this project, we have developed a user-friendly web application called Regulatory Security Check for Cloud Infrastructure (RSCCI) for evaluating the security regulations of a company and the security of the cloud infrastructure used by the company. To be more specific, RSCCI offers a set of web-based questionnaires tailored to different roles of employees in a company. These are based on recognized standards from the cyber security community. RSCCI also provides a vulnerability scanner customized for cloud infrastructures. In this thesis, we introduce how we designed, developed, implemented, and evaluated RSCCI.

# Sammendrag

Axeptia Credit Intelligence AS er en liten bedrift innen finansbransjen. Som mange andre bedrifter benytter de skytjenester som sin løsning for infrastruktur og programvare. Axeptia sikter på å hjelpe andre bedrifter få innsikt og kontroll på sin kundeportefølje med detaljer om blant annet kredittrisiko, kundefordringer, og inkasso. Axeptia ønsker seg en tjeneste som lett gjør kontrollering av deres sikkerhet og samsvar med reguleringer. Denne tjenesten skal kunne kartlegge generiske regulatoriske krav og generere forståelige rapporter slik at brukere uten teknisk bakgrunn skal kunne bruke tjenesten enkelt. Vi har derfor utviklet en brukervennlig web-applikasjon kalt RSCCI i dette prosjektet for å evaluere sikkerhetskravene til et selskap og sikkerheten til sky-infrastrukturen selskapet bruker. Mer spesifikt tilbyr RSCCI et sett med webbaserte spørreskjemaer skreddersydd for ulike ansatt roller. Disse er basert på anerkjente standarder fra cybersikkerhets miljøet. RSCCI tilbyr også en sårbarhets-scanner tilpasset skybasert infrastruktur. I denne oppgaven introduserer vi hvordan vi designet, utviklet, implementerte og evaluerte RSCCI.

# Preface

The RSCCI bachelor project offered by Axeptia to Norges teknisk-naturvitenskapelige universitet/Norwegian University of Science and Technology (NTNU) has provided a unique opportunity for us to work with the security check of cloud infrastructure in various aspects. During this project we have become more familiar with cloud concepts and its requirements for security, worked with different international standards related to information security, made progress in scanning the vulnerabilities and subdomain enumeration, and utilized different programming languages to set up the whole RSCCI website in addition to developing and securing the needed infrastructure.

First and foremost, we would like to thank Per Nestor Warp, the Co-founder & CEO, and Håkon Viervoll, the CTO of Axeptia who were engaged in this journey with us and provided useful insight and support in each step of the project, in addition to participating in testing and giving feedback to us. The support Jia-Chun Lin (Kelly) has provided us both in writing the thesis and working with the project is inexpressible. Kelly spent an impressive amount of time and effort to help us with the rules and protocols of thesis, an area which there still are many things we can learn.

Last but not the least; we would like to thank all family and friends who read our thesis and gave us feedback on it, and also the people from Axeptia, who tested our RSCCI application, and enriched us with their advice and tips.

# Contents

# Figures

# Tables

# Code Listings

# Acronyms

| | |
|---|---|
| **API** | Application Programming Interface |
| **BMaaS** | Bare Metal as a Service |
| **BYOD** | Bring your own device |
| **CIS** | Cloud Infrastructure Scanner |
| **CSP** | Cloud Service Provider |
| **CVE** | Common Vulnerabilities and Exposures |
| **DoS** | Denial of Service |
| **DDoS** | Distributed Denial of Service |
| **DO** | Digital Ocean |
| **EER** | Extended Entity-Relationship |
| **ER** | Entity–Relationship |
| **GDPR** | General Data Protection Regulation |
| **GUID** | Globally Unique Identifier |
| **HTTPS** | Hypertext Transfer Protocol Secure |
| **IaaS** | Infrastructure as a Service |
| **IANA** | Internet Assigned Numbers Authority |
| **IBM** | International Business Machines Corporation |
| **IEC** | International Electrotechnical Commission |
| **ISO** | International Organization for Standardization |
| **JS** | JavaScript |
| **LEMP** | Linux Nginx MySQL PHP |
| **MITM** | man in the middle |
| **MVC** | Model-View-Controller |
| **NIST** | National Institute of Standards and Technology |
| **Nmap** | Network Mapper |
| **NSE** | NMAP Scripting Engine |
| **NTNU** | Norges teknisk-naturvitenskapelige universitet/Norwegian University of Science and Technology |
| **OS** | operating system |
| **OWASP** | Open Web Application Security Project |
| **PaaS** | Platform as a Service |
| **PDO** | PHP Data Objects |

| | |
|---|---|
| **PR** | Pull Request |
| **RSC** | Regulatory Security Check |
| **RSCCI** | Regulatory Security Check for Cloud Infrastructure |
| **SaaS** | Software as a Service |
| **SMTP** | Simple Mail Transfer Protocol |
| **TLS** | Transport Layer Security |
| **VM** | virtual machine |
| **VNC** | Virtual Network Computing |
| **XaaS** | Anything as a Service |
| **XSS** | Cross Site Scripting |

# Glossary

**AJAX** Asynchronous JavaScript and XML (AJAX) is a group of technologies used to create asynchronous requests between a client and a server. 29

**bcrypt** A password-hashing function. 48, 75

**CSRF** Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated. 46

**CSRF Token** A token generated on a web server, and sent to a client loading a web page. When the client sends data back to the web server, it includes this token so that the web server is sure that the request originated from the client. 46

**cURL** A computer software project providing a library and command-line tool for transferring data using various network protocols. 64

**Demilitarized zone** A DMZ is a physical or logical subnetwork containing an organisation's external facing services. A DMZ is used to protect an organization's internal local-area network from untrusted traffic. 62

**DevOps** A software development methodology that aims to automate and integrate the processes between development teams (Dev) and IT operation teams (Ops) so they can build, test, and release software faster and more reliably. 33–35

**DoS attack** A denial-of-service attack (DoS attack) is an attack where a malicious actor attempts to overload systems in order to prevent access from legitimate users. 14, 73, 79

**gamification** The use of game-design elements and game principles in non-game contexts. 85

**hypervisor** A software that creates and runs virtual machines. 7

**node** A client, server or peer. 24

**regular expression** Often referred to as regex, a regular expressions is a sequence of characters that specifies a search pattern. 46

**solution stack** The programming language(s) used to build and deploy a solution. 7, 27

**SQL injection** A code injection technique that might destroy your database. 46, 75

**Time-Based One-Time Password** A type of Two-Factor authentication. Generates a short one-time password, based on the current time. Apps like Google Authenticator can be used to generate this. 47

**Top Level Domain** A domain at the highest level in the hierarchical Domain Name System of the Internet after the root domain. In example.com, com is the Top Level Domain. 46

**Uncomplicated Firewall** A front end for the command-line firewall utility iptables. Makes firewall rules easier to manage. 64

**VNC** A graphical remote desktop system that allows remote control of another computer. 64

**VPC** A Virtual Private Cloud (VPC) network is a private network interface for collections of DigitalOcean resources. 64, 88

# Chapter 1

# Introduction

In this chapter, we give a description of project background, project goals, and problem area covered in the project. We also define the target audience of this thesis, outline the project scope and limitations, and introduce the project group.

## 1.1 Background

Our client is Axeptia who offers credit management software to their customers. They provide help for businesses to reduce outstanding invoices and reduce credit risk [1]. Axeptia that was established in 2017 connects a company's financial system to debt collection and credit information agencies with the service they offer. The Co-founder and general manager of Axeptia, Stian Bølskog Davidsen, tells in an interview with "DNB Nyheter" that their goal is to make the companies that use their service to gain, both operationally and strategically, increased insight and control over their own credit and debt collection portfolio.

As a startup, Axeptia, aims high and wants to digitalize the Norwegian credit and debt collection industry. They believe in their uniqueness in this concept in the Nordic region. Hiding the complexity for the customer, Axeptia has managed to connect to different data components which is a complicated task. They find great demand for the concept they offer. Updating information on customers with less manual work and no maintenance of integration Axeptia has reached their goal. This goal is to offer a tool which gives CFOs and credit managers the opportunity to make the best decisions using good insight and modern tools [2].

## 1.2 Project description

As Axeptia believes: "The vast majority of companies are dependent on changing their architecture to use more cloud-based infrastructure to remain profitable and competitive. It is often a myth that so-called on-premise infrastructure is more secure than cloud-based infrastructure. Today it is common that security requirements for companies that use cloud-based services are stricter than for those who use on-premise infrastructure." See Appendix A. The project was given from Axeptia to us is in this context.

In this project our client Axeptia has requested us to simplify the security task for companies that use cloud based services/infrastructure. This solution, which is implemented and called RSCCI in this project, should be developed in a way that it can provide an overview of how an organization[1] meets external requirements for the security of its architecture and infrastructure. RSCCI should also map relevant generic security requirements through obtaining information via a questionnaire, which could be like a type of gamification such as Kahoot, to illuminate how the organization itself views its security architecture and what requirements it should meet.

[1] In this thesis we have used phrases "organization" and "company" interchangeably.

Out from the organization's input, RSCCI must automatically generate a report that highlights which requirements completely or partially are not met, and provide advice, or link to advice, on how the organization can improve the compliance of the different requirements. RSCCI must be placed in a cloud-based hosting and be able to be integrated with a chosen platform. The platform can be hosted at any Cloud Service Provider (CSP) that the students have free access to.

RSCCI should recommend a solution, technical or through procedures, which ensures that only the organization responsible for completing the questionnaire and the administrators of RSCCI will have access to the results. RSCCI should also be able to support companies that have outsourced their ICT business, so that also they become aware of what requirements they still have to meet, and what requirements they have to set for their supplier. RSCCI must comply with the current guidelines for information security.

## 1.3 Project Goal

The project mainly consist of two parts: developing a regulatory security check application and creating a cloud infrastructure scanner. The regulatory security check application will be presented as a web-based questionnaire. The goal is to gather answers from an organization's participants about its security landscape and compare the answers with compliance regulations related to each problem area of the information security.

This will illuminate how an organization looks at its own security, regulatory compliance, and what security requirements it should fulfill. It is also desired, but not compulsory, to make gamification and short versions of the questionnaire. The second part of the project is to develop a cloud infrastructure scanner which is able scan an organization's cloud infrastructure for known security flaws and vulnerabilities. Note that the scanning is different from penetration testings, see Section 1.6.

Based on the questionnaires answered by a company and the scanning result produced by the cloud infrastructure scanner, the final part of this project should generate a report to illuminate what requirements are fulfilled completely, partial, or not at all. In addition the solution should give advice on how the organization may increase its security on different areas. This might be done by providing the participants guidelines through some web pages which gives more information, or any other solution which satisfies the same requirement.

## 1.4 Problem Area

The problem area for this bachelor thesis primarily focuses on the following topics related to cloud-based infrastructure security:

- **Security scanning for different XaaS:** A cloud-based infrastructure can come in many shapes and forms, since one can use the cloud in multiple ways. A cloud service provider may offer services in any of the following Anything as a Service (XaaS) platforms: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). We have researched how to scan the different types of XaaS platforms, and where research showed it is possible, implemented functionality to scan each of them.
- **Regulatory compliance:** All organizations have standards, laws, policies, or specifications that they need to follow regarding information security. Organizations using cloud are no exception, and this is therefore an important part of our thesis. We accomplished this task using International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC) and National Institute of Standards and Technology (NIST) standards.
- **Platform development:** The final product will offer companies a functional platform to obtain insight into their security. As it is important to develop this platform in a secure

manner we will pay enough attention and focus on a secure development of this application during the project.

- **Security in cloud-based infrastructure:** The project and thesis is based around security principles in cloud-based infrastructure and will try to follow the related security guidelines.

## 1.5 Target Audience

The main target audience of this thesis is our client Axeptia, as they wanted this project to be developed so they can use it as a security evaluation service. Other audiences of this thesis include anyone who is interested in what we have developed in this project, such as companies that wonder their security regulatory compliance or use cloud infrastructure services.

## 1.6 Scope and Limitations

As stated earlier, the main focus of this project is to develop a web-based application so that Axeptia could use it for security testing not only when it comes to compliance and management, but also scanning of known security flaws and vulnerabilities on an employed cloud infrastructure.

However, the scope and limitations are as important as the goals of this project as it gives us a chance to fulfil the task of the project in time. In the following section, we will discuss limitations in this project regarding scanning and generating a report to the user of the application. Lastly, we also discuss the effect of the Covid-19 pandemic on fulfilling this project.

### Scanning limitations

The scanner mainly scans after vulnerabilities using tools such as Nmap, Changeme, XSStrike, and Nikto. The steps which the scanner will perform includes among others identifying the target, port scanning, looking for remote access, determining software version, looking for CVEs based on the software version, and more. The purpose of these steps is to make organizations aware of potential vulnerabilities existing in their systems, see also Section 6.3.

What this active test, also referred to as a scan, is not going to do is a penetration testing. A penetration test will lead to downtime and harm to the systems [3]. The scanning process developed in RSCCI will only do the scanning and enumeration part of a full penetration test. This application scans anything that has an IP address regardless of the provider. However, providing IP addresses for the test is on the user to manage as this information is unique per organization.

When it comes to the scanning process itself, some organizations, including Axeptia, use solutions for "IP rate limiting", e.g. Imunify360, which blocks IP addresses if there are too many requests in a short period of time [4]. Organizations which use these types of solutions will not get a complete scan as they will block the scanner.

### The generated final report

The final report, generated by our application RSCCI, will not give a score compared with other organizations to say which organization has a better readiness and security compared with others. But the report will provide the participant information about the level of readiness in which the participant organization itself has admitted needing to have.

So there will not be an organization who scores highest compared to others, as there is no such "good" or "bad" secured, but it is up to each organization to determine how they will manage the risk. The report provides a list of controls that should be in place. This will be concluded based on the information provided by the participant. The risk management of the organization could be such as they decide to fix the control, accept the risk or choose to transfer it.

### Covid-19

Due to the ongoing Covid-19 pandemic, the abilities to have physical meetings and work sessions was limited and reduced to online sessions. However, we all had experience with working together online, and we tried to not let this situation affect our teamwork, productivity, or the final result.

## 1.7 Project Group

This group consist of four members who has different competencies and interest areas. Three of the group members study IT operations and information security and one study Engineering - Computer Science. However, for any project different competencies and understandings of various aspects of the project is needed. Therefore, this interest and knowledge variation has given us a better ability to overcome this project. The four student members of the team are as follows:

**Øyvor Vejlgaard Sørensen** is working on a bachelor in IT operations and information security at NTNU in Gjøvik, with a part time job as a junior developer for Axeptia Credit Intelligence AS at the side. She has also worked as a teaching assistant in programming for NTNU. Øyvor is interested in everything that has to do with code, technology, and information security.

**Kristian Fjelstad** is working on a bachelor in Engineering - Computer Science at NTNU in Gjøvik. He has also worked part time as a security engineer at a Security Operations Center (SOC) which now has become his full time job. He is interested in computers, programming, and server hosting.

**Harald Aarseth** is studying bachelor in IT operations and information security at NTNU in Gjøvik. He has interest in penetration testing and server hosting. He has practical IT experience from Elvebakken high school. He has also worked for NTNU's Norwegian Cyber Range (NCR) in penetration testing competitions.

**Gloria Treider** is a student of bachelor in IT operations and information security from NTNU in Gjøvik. She has worked as teaching assistant for many different subjects such as Data modeling and database systems, and Network programming. During her first year of bachelor study she worked voluntary for Norwegian Cyber Range (NCR). She already has a certificate of apprenticeship as a network administrator from Treider College, and has a variety of area of interest such as regulations for information security, networking, programming and cloud services. Gloria will start working for Sopra Steria as a network engineer in August 2021.

### Distribution of work

In the distribution of the team work main responsibilities are as follows:

- Technical development of the web application and its infrastructure: Øyvor and Kristian
- Technical development of the scanner: Harald
- Regulatory compliance questionnaire contents: Gloria

The distribution of other responsibilities in the group are as follows:

- Point of contact with Axeptia: Øyvor
- Contact person with NTNU and the supervisor: Gloria
- Group secretary: Gloria
- Group deputy secretary: Øyvor
- Inspector and responsible for Trello related workflow: Harald
- Logbook responsible: Harald
- Overleaf specialist: Kristian
- Confluence manager**:** Kristian

- Bitbucket repository and Jira responsible: Øyvor

For more details see also Appendix D. The tasks are divided as mentioned above, so there is a person responsible seeking for task to be done. But all group members have helped each other in different parts of tasks as that is how a group works.

## 1.8 Thesis Structure

- **Chapter 1 - Introduction**
  Description of the problem area, project description, project goals, scope and limitations, and the thesis structure. Introduction of the project group.
- **Chapter 2 - Background**
  Introduction to cloud computing, cloud security threats, security scanning, and regulatory compliance.
- **Chapter 3 - Requirements**
  Description of the functional and non-functional requirements for RSCCI, use cases, and misuse cases.
- **Chapter 4 - Design**
  Covers the design of RSCCI. Covers system architecture, architectural design, component communication, design driving technologies, and user friendliness.
- **Chapter 5 - Development Process**
  Covers the development process in this project.
- **Chapter 6 - Implementation**
  Description of how RSCCI and its desired functionalities are implemented.
- **Chapter 7 - Evaluation**
  Covers the testing and evaluation of RSCCI's compliance with the requirements.
- **Chapter 8 - Discussion**
  Discussion of the work completed during the project period.
- **Chapter 9 - Closing Remarks**
  Conclusion of the project and further work.

# Chapter 2

# Background

In this chapter we are going to introduce three different subjects, which are relevant to understand the contents of this thesis. Section 2.1 will talk about cloud computing, and introduce different therms and models. Sections 2.2 and 2.3 will cover the basics of threat actors and cyber vulnerabilities. Lastly, we will cover regulatory compliance, and talk about some of the standards we based our work on.

## 2.1 Cloud Computing

Cloud computing is a model for delivering on-demand availability of computer resources, like networks, servers, storage, applications, and services [5]. The resources are hosted by a CSP and used by cloud consumers. A CSP is a company that establishes public clouds, manages private clouds, offers on-demand computing services, or a combination of the three, while a cloud consumer is a customer who buys computer resources from a CSP.

According to NIST, there are five essential characteristics that define cloud computing: on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [5]. The collection of hardware and software that enables these characteristics of cloud computing is called cloud infrastructure. Each of these characteristics is defined by NIST as follows:

- **On-demand self-service**
  On-demand self-service means that cloud consumers can purchase storage, networks, and other computing resources when they require them, as well as return unneeded resources to their CSPs automatically without human interaction from their CSPs.
- **Broad network access**
  Broad network access means that the computing resources are basically available all the time, and can be accessed from anywhere by various kinds of devices such as mobile phones, laptops, workstations, and other devices. Cloud consumers can often access computing resources offered by a CSP via the web portal of the CSP.
- **Resource pooling**
  Resource pooling means that each type of computing resources provided by a CSP is organized and grouped into a logical pool to serve multiple cloud consumers at the same time. This is often accomplished using a multi-tenant model where different physical and virtual resources are assigned and reassigned automatically based on cloud consumer demand. A cloud consumer have no knowledge or control over the physical location of computing resources besides the high-level location information such as country, state, or data center where the resources are located.
- **Rapid elasticity**
  Rapid elasticity means that the computing resources can be scaled rapidly up or down (or out or in) to fit cloud consumers needs. The scaling possibilities often seem unlimited to cloud consumers, and can be granted anytime in any quantity as long as it gets paid for.

- **Measured service**
  Measured service means that CSPs automatically control and optimize resources usage by leveraging measurable values of computer resources. Exactly what value is used depends on the type of resource. Examples of measurable values include storage, processing, bandwidth, and number of active users.

In the rest of this section, we will introduce different service models, deployment models, and the shared security responsibility between a CSP and cloud consumers.

### 2.1.1 Service models

Cloud computing is offered in different service models, often referred to as different Anything as a Service (XaaS) [6]. The most common XaaS are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), but different CSPs list additional service models. For example, International Business Machines Corporation (IBM) lists Bare Metal as a Service (BMaaS) as a service model, while Microsoft Azure lists Serverless as another service model in addition to IaaS, PaaS, and SaaS [7, 8]. In this section, we will introduce the most common ones, i.e., IaaS, PaaS, and SaaS. For each of the service models, we will also describe what security responsibility falls to cloud consumers and CSP, as this is a shared responsibility.

#### Infrastructure as a Service (IaaS)

IaaS is a service model where a CSP delivers hardware in form of virtualized resources for processing, network, storage, and other fundamental computing resources to a cloud consumer. These resources are often hosted at a data center owned by a CSP. A cloud consumer can purchase this kind of services from a CSP to deploy and run software, including operating systems and applications. A typical example is that a cloud consumer rents one or more virtual machine (VM) from a CSP to deploy his/her application. The costs is often based on a pay-as-you-go model, and the service enables cloud consumers to scale their resource use based on their needs. For cloud consumers, IaaS removes the need to have unnecessary and expensive infrastructure "on hold" when the workload of their applications are low. In addition, it also eliminates the risk of not having sufficient resources when the workload is high [7].

IaaS is the service model where the cloud consumer has the largest security responsibility. When using IaaS, cloud consumers are responsible for their virtual networks, VM, operating system (OS), solution stack, applications, data, and interfaces such as application programming interfaces and graphical user interfaces, as well as their regulatory compliance. CSPs are not responsible for those, and have only security responsibility for processing and memory, data storage, network, hypervisors, physical hosts, and data centers when offering IaaS [9].

#### Platform as a Service (PaaS)

PaaS is a service model where a CSP delivers hardware, software, and infrastructure as a complete platform for cloud consumers to develop, run, and manage their applications without having to build and maintain the platform on-premise. This means that cloud consumers do not control or maintain the underlying infrastructure including network, servers, OS, or storage. Instead, CSP host the infrastructure at their data centers. PaaS therefore allows cloud consumers to focus on developing their applications without having to worry about maintaining or managing the underlying infrastructure and platform [10].

When cloud consumers are using PaaS, CSP take over some of their security responsibility. In addition to the responsibility when offering IaaS, CSP are also responsible for the virtual networks, VMs, and OS, while cloud consumers no longer has this responsibility [9].

#### Software as a Service (SaaS)

SaaS is a service model where a CSP delivers ready-made software to cloud consumers. The software is accessible by various compatible devices in form of a mobile application or via web browsers. As the software is provided over the Internet, a cloud consumer can be wherever. In SaaS, everything is controlled and managed by CSPs, releasing the responsibility of cloud consumers for maintaining the software and the underlying infrastructure and platform [11].

SaaS is the service model where cloud consumers have the least security responsibility, since CSPs control all the software, infrastructure, and hardware. When using SaaS, cloud consumers are only responsible for securing their data, interfaces, and regulatory compliance. The CSPs are responsible for the applications and solution stack, in addition to all previous mentioned responsibilities [9].

### 2.1.2 Deployment models

A deployment model is the way CSPs provision their cloud infrastructures and make it available for cloud consumers. There are four different deployment models: Public Cloud, Private Cloud, Community Cloud, and Hybrid Cloud [5, 12]. Each of them is separately introduced below.

#### Public Cloud

Public Cloud is a deployment model where a cloud infrastructure is owned, managed, and operated by a CSP, and is provisioned for public use. A Public Cloud is hosted at a CSP's data center, and is available via the Internet, hence making it available for whomever, individuals as well as businesses. CSPs are responsible for maintaining and managing all resources, whereas cloud consumers purchase needed resources based on the pay-per-usage model or subscription. Public clouds offer a flexible solution since cloud consumers do not pay for more than they need and they do not have to think about physical maintenance or other aspects of having physical servers [13].

#### Private Cloud

Private Cloud deployment model refers to a cloud infrastructure that is provisioned for exclusive use by one cloud consumer, which could be an organization or enterprise. A Private Cloud may be hosted on-premise or at a CSP's data center. The latter is often referred to as a *Virtual* Private Cloud. Private Cloud offers more control to cloud consumers, as they get the ability to customize the environment, manage it based on their specific needs, and keep their data private. This is the case for both on-premise Private Cloud and Virtual Private Cloud [14].

#### Community Cloud

Community Cloud is another deployment model where a cloud infrastructure is provisioned for exclusive use by a community of cloud consumers. In this case, a community could be a group of organizations that have shared concerns, such as mission, security requirements, policy, or compliance consideration. Similar to Private Cloud, a Community Cloud may be hosted on-premise or at a CSP's data center.

#### Hybrid Cloud

Hybrid Cloud refers to the combination of two or more of the other deployment models, for example the combination of a Private Cloud and a Public Cloud. A Hybrid Cloud may contain one or more infrastructures of each deployment model, and the infrastructures communicate via proprietary software. This deployment model gives cloud consumers great control over their private data, as well as providing great flexibility by allowing cloud consumers to move workloads between the cloud infrastructures based on cost or their need [15].

## 2.2 Security Scanning

In this section we will cover what a vulnerability is, and talk about why it is so important to have regulatory assess security in general. We will also talk about the consequences of a bad security practice.

### 2.2.1 Vulnerability

In computers, the word vulnerability is referred to as a weakness, that can be exploited to gain unauthorized access, or perform harm to a computer system [16].

A vulnerability can be explained as a hole in a wall, with which an attacker can expand using tools, to gain access to a building. Sometimes the wall may even be big enough for anyone to be able to walk into. Attackers gain access to systems using exploits, often made into command line tools [16]. Exploits are carefully crafted lines of code, which make the software behave different than what it is intended to do.

Vulnerabilities exists from programmers who writes code, which either is not tested enough, or does not know how to write code securely. New operative system updates or program language version updates can make legacy and outdated software even more vulnerable, by not following the latest standards, or still using for example a module which have found out is vulnerable to exploits.

As an organization, there are two main different types of vulnerabilities one needs to be on the lookout for. The ones that are caused internally (badly self-inflicted code), and the vulnerabilities in the software that an organization uses to do the rest of their normal tasks. Microsoft Word is an example of this.

### 2.2.2 The importance of vulnerability scanning

The consequences of an attack can harm an organization both financially, and with respect to reputation. Attacks in large scales can affect anywhere from thousands, to millions of user. This will result in big media publicity, which will leave a negative reputation against the organization. They may expect huge fines either from a class action lawsuit, or in form of a General Data Protection Regulation (GDPR) lawsuit. A customer would loose credibility to an organization, if they have been compromised. The customer could also feel frightened that its personal information has come in the wrong hands.

This is why computer security has become a huge market. Attackers and defenders each try to outperform each other, and companies spend billions defending from the risks they take when having everything online.

### 2.2.3 Legality of port scanning

Although being quite commonly to run a vulnerability scan, some people have received problems for port scanning some organizations. Running multiple scans directed towards the same network can pile up a lot of network traffic. For some small networks this can act as a denial of service attack, and may slow down the rest of the network traffic. Some people consider a port scan the first step towards breaking in, even though the intent was not to hurt the target. Within the year frame 1999 and 2006 a total of 3 people have been accused of attempted unauthorized access to computers, by running port scans on computers owned by nation states, and even a bank [17].

### 2.2.4 CVE

A Common Vulnerabilities and Exposures (CVE) is a identifier for vulnerabilities and exploits which are present in publicly released, in popular software and applications [18].

The CVE database is maintained by the MITRE Corporation, which is founded by Homeland Security in the USA. When a vulnerability is disclosed by the Mitre corporation, they give the vulnerability an ID, which first says the year the vulnerability was disclosed, followed by a unique

4 digit number. For example: CVE-2017–0143, which is the commonly exploited heart bleed exploit from 2017. A software might have multiple CVEs, even on the same software version. This will then be different weaknesses that the software contains.

Every CVE has different impact, and therefore there is a system for scoring the severity on a scale from 1-10. This is called a Common Vulnerability Scoring System (CVSS), and a table with the severity of the different scores can be seen in Table 2.1 [19].

**Table 2.1.** CVSS vulnerability severity ratings.

| Severity | Base Score |
|---|---|
| None | 0 |
| Low | 0.1-3.9 |
| *Medium* | 4.0-6.9 |
| High | 7.0-8.9 |
| Critical | 9.0-10.0 |

## 2.3 Security Threats

The exploits are performed by attackers, and can be categorized into threat actors. The attackers have different motives to preform the attacks [20].

- **Cyber Terrorists**
  Terrorist groups have in the last two decades used the internet to spread fear in larger cases than before. They cause damage to large companies, the authorities, and large sums of people. Their methods are often Distributed Denial of Service (DDoS) attack, and leaking the authorities' information, or the company's secrets.
- **Nation state agents**
  Nation state agents is probably the most skilled threat agent. They have almost unlimited funds, and they do not have to try to hide from their own authorities. Their motivation are mostly towards information gathering either at other nations, or towards inside threats. This information can affect politics, and investigations.
- **Organized Crime**
  Criminals often hack for financial gains. They target large companies, and gain access to users personal information, which they later sell in large bulks on black market. Other techniques they use are to spread ransomware, which demands the users to pay a sum, in order to receive their computer files back.
- **Hacktivists**
  Hacktivists use hacking to raise awareness about particular topics. They want to reveal government or company secrets, which are hidden from the public. Their methods are either by using insiders (like Edward Snowden), or hacking the systems themselves, in order to gain the information they want. Julian Assange, the founder of the organization and the website Wikileaks, is one of the most famous hacktivists [21]. The group Anonymous is also a well known group, which use hacking to promote their political opinions.
- **Insiders**
  Inside users with privileged access have the possibility of really harming its company. Reasons for the employees turning against them can range from financial motivation, to an disgruntled employee, who wants to hit back at the company. They can leak secret information, or add a secret backdoor into the network, which other attackers can use.
- **Script kiddies**
  With many public penetration tools being so easy and available to use, the increasing number of so called "script kiddies" have increased. The word is used towards individuals

with lacking hacking skills. They use attacking tools to see how far they can come, however often not making any progress. Their motivation is more towards the thrill itself of the attacking. They also brag about successful attacks to others.

- **Human errors**
  Human error is a non malicious threat, which is not intentional. Other threat actors rely on human coding error, in order to craft exploits. Most CVEs are because of human error. Websites and applications which are not tested enough, and leave small gaps which are exploitable to attackers.

## 2.4 Regulatory Compliance

The number of rules in form of specifications, laws, standards, or policies is increasing. Conforming to these rules is called compliance. There is a need to adapt the control of compliance to organizations as the number of the regulations are increasingly growing. This is why we talk about regulatory compliance to accomplish the need for operational transparency for organizations. These regulations are various as there are many standards which inform about how to comply with rules, such as Gramm–Leach–Bliley Act (GLBA), Federal Information Security Management Act of 2002 (FISMA), Control Objectives for Information and Related Technologies (COBIT), etc. [22].

Among these we have chosen some relevant standards to our project to accomplish this task. A short introduction to the main chosen standards in our project is as follows:

- NS-EN ISO/IEC 27017:2021 Information technology - Security techniques - Code of practice for information security controls for cloud services is based on ISO/IEC 27002 and provides guidelines for information security controls applicable to the provision and use of cloud services [23]. Also, additional controls with implementation guidance that specifically are related to cloud services, provide controls and implementation guidance for both cloud service providers and cloud service customers.

  Specifically, this questionnaire aims to support the implementation of information security controls for cloud service customers who implement the controls. ISO/IEC 27017 points to ISO/IEC 27002 as the baseline for the controls before it adds a few control checks to each point highlighted in 27002. Our cloud specific questionnaire is based on ISO/IEC 27017 standard.

- NS-EN ISO/IEC 27002:2017 Information technology - Security techniques - Code of practice for information security controls (ISO/IEC 27002:2013) provides guidelines for organizational information security standards and information security management practices [24]. This standard is designed for organizations to use as a reference for selecting controls within the process of implementing an Information Security Management System (ISMS) based on ISO/IEC 27001, or as a guidance document for organizations implementing commonly accepted information security controls.

  ISO/IEC 27002 guideline takes into consideration the organization's security risk environment. ISO/IEC 27017 Information technology Security techniques Code of practice for information security controls is based on ISO/IEC 27002 for cloud services. Thereby making ISO/IEC 27002 as the core part of this questionnaire. ISO/IEC 27002 is the main resource which we have utilized to achieve the objectives of this questionnaire. We utilized this standard in questionnaires for IT, HR, and Management employees.

- NIST Special Publication 800-209, Security Guidelines for Storage Infrastructure The main focus of NIST is to provide a comprehensive security recommendation that will address the threats. The recommendations span not only over security management areas that are common to an Information Technology (IT) infrastructure, but also those specific to storage infrastructure.

  This application RSCCI however, pays attention to chapter 4 of this document which focuses on Security Guidelines for Storage Deployment and includes the most related controls to cloud infrastructure security regulations [25]. The selected questions provide a software-based abstraction over all forms of background storage technologies. We have utilized this standard in case an organization would like to do a more in depth check of their cloud infrastructure security, or will choose this standard as the more preferred one.

The ISO/IEC International Standards such as ISO27001 has also been to a degree of helping this collection, but not as much as the three regulatory standards named above.

- ISO/IEC 27001:2013 Information technology - Security techniques - Information security management systems - Requirements, is for specifying the requirements for establishing, implementing, maintaining and continually improving an information security management system within the context of an organization. It is important that information security management system is part of and integrated with the organization's processes and overall management structure, and that information security is considered in the design of processes, information systems, and controls. This standard is a guidance document for organizations implementing commonly accepted information security controls [26]. A few questions for Management questionnaire are taken from this standard.

# Chapter 3

# Requirements

Axeptia has been very flexible about how the problem goal can be met, and only had a handful of requirements for the final solution of this project. We have therefore set some of our own functional and non-functional requirements in addition to theirs. This chapter will give a description of all the functional and non-functional requirements for the solution. The requirements are either *shall* requirements or *should* requirements, where shall means that the solution has to comply with the requirement and should means the solution is recommended to comply with the requirement.

## 3.1 Functional Requirements

Our functional requirements are primarily based on the provided task description and discussions with Axeptia regarding our solution's functionality. See Appendix A for the task description.

1. The solution shall map relevant generic regulatory requirements by the use of questionnaires to shed light on how a company views its security architecture and what requirements should be met.
2. The solution shall automatically generate a report to shed light on what security requirements are met, and give advice (or links to advice) about how a company can improve their security on specific areas, based on the answers from the questionnaires.
3. The solution shall be placed in a cloud environment and be able to be used by all companies with cloud infrastructure, regardless of CSP.
4. The solution shall make sure that only the company that answers the surveys or perform a scan is authorized to see the results.
5. Companies that outsource their IT business should be able to use the solution, mostly to make themselves aware of which requirements they should still fulfill and what requirements they should set for their supplier.
6. The solution shall be able to perform simple network scans against a users endpoints, to check if a users infrastructure is potentially vulnerable to known vulnerabilities.

## 3.2 Non-Functional Requirements

Our non-functional requirements are based on discussions with Axeptia regarding our solution's operation capabilities and constraints. We have grouped the non-functional requirements into the following categories. Note that we continue the numbering from the functional requirements.

7. **Security**
   Axeptia stated that the solution has to follow today's guidelines for cyber security. As this is very vague, we have specified it into the following:

   a. The solution shall support user authentication with salted, hashed passwords. Two-factor authentication shall be supported, but only enabled when a user desires to use two-factor authentication.
   b. The solution should support authorization by roles.
   c. The solution should support accountability for user actions, meaning that user actions should be traceable to the user who performed them [27].
   d. As the solution is web-based, it should be protected against the common web-vulnerabilities listed in OWASP Top 10, which is a list of the most common web vulnerabilities.
   e. The infrastructure used to host the solution shall use encryption when transmitting data over the network with use of approved protocols, such as HTTPS and SSH.
   f. The infrastructure used to host the solution shall be secured in accordance with best practices.

8. **Performance**
   Our web application should be able to load each web page fairly fast to provide a good user experience.
   a. The different pages in the application should load in about 1 second as this will provide a good user experience when navigating between the different pages. The questionnaire should load in about 0.1 seconds, as this gives users a feeling of instantaneous response [28].
   b. The web application shall achieve a Google PageSpeed score of over 90 [29].
9. **Reliability**
   a. The web application should have 99% availability within normal working hours to ensure users can use our web application.
   b. Backup of the web applications database should be done on a regular basis to minimize data loss in case of disaster.
10. **Scalability**
   The scanner part of the solution needs to perform its tasks linearly to minimize the risk of a scan being interpreted as a DoS attack, so it should be possible to run multiple scanner instances at the same time against multiple cloud infrastructures.

11. **Usability**
   The solution should be intuitive to use.
   a. Where needed, sufficient information about functionality shall be provided.
   b. Users who do not have background in information technology and cyber security should be able to understand and use the solution easily.

## 3.3 Use Cases

This section covers the different use cases for our web application, RSCCI. We have identified the following actors: User, Team admin, and System admin. A user is a person who uses the web application. The user can perform different actions based on the role they are assigned in RSCCI: Employee, HR, IT or Management. A Team administrator is a user who has registered and created a team, or is assigned the Management role. A System administrator is a user who owns and maintains RSCCI.

Figure 3.1 shows the use case diagram of the identified use cases. For each of these, Tables 3.1 to 3.9 will give a more detailed description.

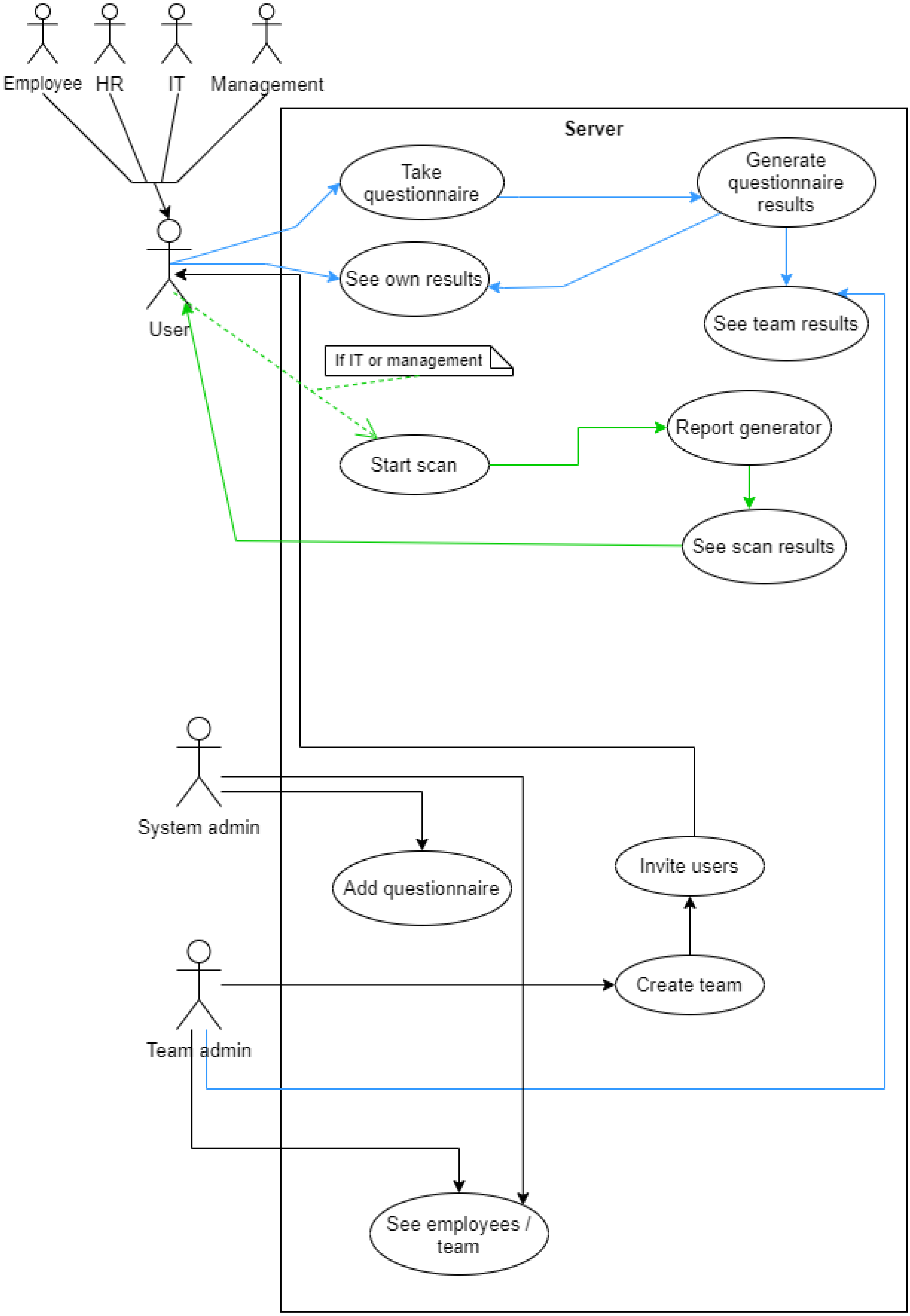


**Figure 3.1.** Use case diagram of our web application

**Table 3.1.** Use Case: Take Questionnaire

**Use Case:** Take Questionnaire
**Purpose:** Map generic regulatory requirements.
**Requirement(s):** 1, 11a, 11b
**Actor(s):** User
**Pre-condition(s):** A user is logged in and part of a team.
**Trigger(s):** The user chooses to start questionnaire.
**Description:**

1. The user chooses a questionnaire.
2. The user is presented with a question form the questionnaire.
3. The user answers the questions.
4. Point 2 and 3 are repeated until there are no more unanswered questions.

**Sub-variations:**
3a. The user exits the questionnaire before all questions are answered.
**Post-conditions(s):** The user is guided back to the questionnaire overview page.

**Table 3.2.** Use Case: Start Scan

**Use Case:** Start Scan
**Purpose:** Scan a company's cloud infrastructure for vulnerabilities.
**Requirement(s):** 6, 7c, 11a, 11b
**Actor(s):** User
**Pre-condition(s):** A user is logged in, part of a team, and assigned the IT or Management role.
**Trigger(s):** The user goes to the scan page.
**Description:**

1. The user goes to the scan page.
2. The user agrees to the scan terms.
3. The user inputs the desired domain and IP addresses to scan.
4. The user starts the scan.

**Post-conditions(s):** The user is guided to the result page with a progress bar of the scan's progress.

**Table 3.3.** Use Case: See Own Results

**Use Case:** See Own Results
**Purpose:** Evaluate security.
**Requirement(s):** 2, 4, 11a, 11b
**Actor(s):** User
**Pre-condition(s):** A user is logged in, part of a team, and at least one questionnaire is completed.
**Trigger(s):** The user goes to the result page.
**Description:**

1. The user goes to the result page.
2. The user chooses which questionnaire to see the results for.
3. The system generates the results.
4. The results are presented to the user

**Post-conditions(s):** The user can see the results.

**Table 3.4.** Use Case: See Team Results

**Use Case:** See Team Results
**Purpose:** Evaluate the whole company's security.
**Requirement(s):** 2, 4, 11a, 11b
**Actor(s):** Team administrator
**Pre-condition(s):** A Team administrator is logged in and has created a team, and at least one team user has answered at least one questionnaire.
**Trigger(s):** The Team administrator goes to summary result page.
**Description:**

1. The Team administrator goes to the summary results page.
2. The Team administrator chooses which questionnaire to see the team results for.
3. The system generates the results.
4. The results are presented to the team admin.

**Post-conditions(s):** The Team administrator can see the results.

**Table 3.5.** Use Case: See Scan Results

**Use Case:** See Scan Results
**Purpose:** Evaluate the vulnerabilities found by the scanner.
**Requirement(s):** 2, 4, 11a, 11b
**Actor(s):** User
**Pre-condition(s):** A user is logged in, is part of a team, and assigned the IT or Management role. A scan has been initialized.
**Trigger(s):** The user goes to the scan results' page.
**Description:**

1. The user goes to the scan results' page.
2. The system generates the results for the user.
3. The results are presented to the user.

**Post-conditions(s):** The user can see the results.

**Table 3.6.** Use Case: Create Team

**Use Case:** Create Team
**Purpose:** Group together employees of the same company.
**Requirement(s):** 11a, 11b
**Actor(s):** Team administrator.
**Pre-condition(s):** A Team administrator has registered and is logged in.
**Trigger(s):** The Team administrator chooses to create a team.
**Description:**

1. The Team administrator chooses to create a team.
2. The Team administrator inputs the name of the team and saves.
3. The system creates the team and adds the team administrator as owner.

**Post-conditions(s):** The new team is created, and the team administrator is the owner.

**Table 3.7.** Use Case: Invite Users

**Use Case:** Invite Users
**Purpose:** Invite employees/co-workers to be a part of a team to get better team results.
**Requirement(s):** 11a, 11b
**Actor(s):** Team administrator.
**Pre-condition(s):** A Team administrator is logged in and has created a team.
**Trigger(s):** The Team administrator goes to the company page.
**Description:**

1. The Team administrator goes to the company page.
2. The Team administrator fills in name, email, and role for a new team member.
3. The Team administrator sends the invitation.
4. The system creates the new user with a random password.
5. Point 2, 3, and 4 is repeated for as many members as desired.

**Post-conditions(s):** The invited user receives an email with login credentials.

**Table 3.8.** Use Case: See Employees / Team

**Use Case:** See Employees / Team
**Purpose:** Get an overview of a team's members and the web application's users.
**Requirement(s):** 11a, 11b
**Actor(s):** Team administrator, System administrator
**Pre-condition(s):** A System administrator is logged in. A Team administrator is logged in and has created a team.
**Trigger(s):** The Team administrator goes to the company page. The System administrator goes to the "Admin" panel.
**Description:**

1. A user with the right permissions can see the employees and teams registered on the web application.

**Sub-variations:**
1a. The Team administrator can only see the users on their team, but can also remove them.
1b. The System administrator can not remove users.
**Post-conditions(s):**
1a. The Team administrator sees his/her team members.
1b. The System administrator either sees the registered users or the registered team.

**Table 3.9.** Use Case: Add Questionnaire

**Use Case:** Add Questionnaire
**Purpose:** Have questionnaires for the users to answer, make a System administrator's job easier.
**Actor(s):** System administrator
**Pre-condition(s):** A System administrator is logged in.
**Trigger(s):** The System administrator goes to the questionnaire page on the "Admin" panel.
**Description:**

1. The System administrator can add, update, and remove questions from questionnaires and change the order of the questions.

**Post-conditions(s):** The new question is added, or the old question is deleted or updated. The question order is changed.

## 3.4 Misuse Cases

This section covers misuse cases for our web application RSCCI. A misuse case helps us to identify potential security risks that might require mitigation(s). Figure 3.2 shows the misuse case diagram of the identified misuse cases. For each of these, Tables 3.10 to 3.13 will give a more detailed description.

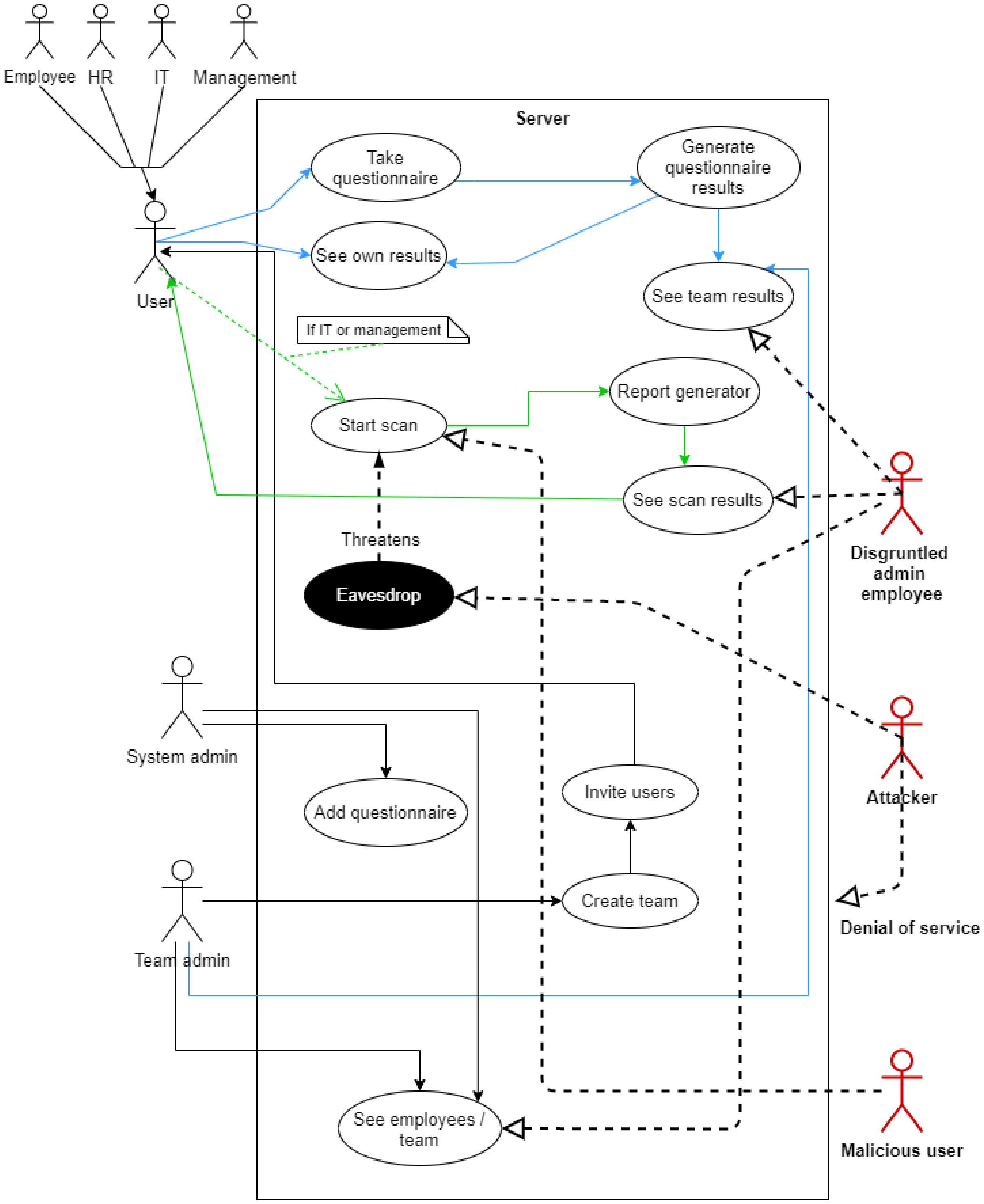


**Figure 3.2.** Misuse case diagram

**Table 3.10.** Misuse Case: Eavesdropping

**Misuse Case:** Eavesdropping
**Purpose:** Sabotage the results.
**Actor(s):** Attacker
**Pre-condition(s):** A user starts a scan
**Description:**

1. An attacker can potentially listen on what information users provide to the scanner, and in worst case act as a man in the middle (MITM), and tamper with the provided information to the scanner. This will result in the attacker being able to alter the IP addresses and domains. IP addresses and domains are what the users want the scanner to check for vulnerabilities.

**Countermeasure(s):** Using encrypted communication will ensure the information between a client and a host is unreadable for an attacker eavesdropping on the traffic.

**Table 3.11.** Misuse Case: Denial of Service

**Misuse Case:** Denial of Service (DoS)
**Purpose:** Taking down the web site.
**Actor(s):** Attacker
**Pre-condition(s):** The web site is running normally
**Description:**

1. An attacker can use one or many computers to overload the network, where the web site is placed on, with packets. This will cause the website to go down, and make it unusable for users. A Denial of Service (DoS) attack is possible to do against the vulnerability scanner machine. However, the IP of the scanner is not published anywhere, so a DoS on this machine is highly unlikely.

**Countermeasure(s):** Using an anti - DoS proxy in front of the server.

**Table 3.12.** Misuse Case: Company insider

**Misuse Case:** Company insider
**Purpose:** Revenge/sabotage on company.
**Actor(s):** Disgruntled ex-employee
**Pre-condition(s):** The disgruntled ex-employee has previously worked for the company, and is still has access as an administrator in RSCCI.
**Description:**

1. A formerly employed administrator can still have access to the "Admin" panel in RSCCI. His/her coworkers can have forgotten to remove this person's administrator access to the web site. This ex-employee can have access to lists of whoever is working with the companies which have used this web site, and also see the scan results. This could compromise the information registered in this application.

**Countermeasure(s):** Have a good routine to check and update the employees' access to the application, when their contract ends.

**Table 3.13.** Misuse Case: Vulnerability scanning abuse

**Misuse Case:** A user of the application scans addresses which are not their own
**Purpose:** Gaining information about other companies
**Actor(s):** Malicious user, or a competitor
**Pre-condition(s):** The malicious user has access to the system
**Description:**

1. There is nothing to stop a user from starting a vulnerability scan on IP addresses which belong to other companies. This results in the user gaining security information about any companies, which can be their competitors.

**Countermeasure(s):** The website could have a check, so a user only can scan domains or addresses of their own. The scan can also be logged to monitor and prevent such events from happening. So in case such events happen the malicious user can be traced and investigated.

# Chapter 4

# Design

In this chapter we will explain the design and the system architecture of RSCCI. This chapter also covers the architectural pattern, design driving technologies and languages chosen to realize the system architecture, communication, and user friendliness.

## 4.1 System Architecture

To be able to easily describe the system architecture, we have logically divided RSCCI into three main components: RSC, CIS and the web portal. RSC is the questionnaire part of our application, which handle the questionnaires, its questions and generating reports from the questionnaire answers. CIS is the security scanner, and handles the scanning of a company's endpoints and domain, and generating a report from the scan. The web portal is the website a user interacts with, and handles everything that falls outside the aspects of the more specialized RSC and CIS, like front end, authentication, and user management. This logical division is not representative of how our application works on a technical level, but describes how the interaction between components is designed. The technical aspects will be discussed in Chapter 6.

Figure 4.1 shows the high-level system architecture of the final product, and Figure 4.2 shows a flowchart of the application experience from a user perspective. A user logs into the web portal of RSCCI and chooses to either start RSC or CIS. If a user chooses to use RSC, RSC will fetch a set of questions that corresponds to the role of the user in his/her company from our database and presents them to a user. Whenever a user finishes answering a question, his/her answer will be immediately stored in the database for further analysis. After a user has answered all the questions, the RSC Report Generator will analyze all his/her answers and generate a report. This report will help the user evaluate how compliant the user's company are with different regulations.

On the other hand, if a user chooses to use CIS, he/she will be requested to provide the endpoints and domain of the cloud infrastructure used by his/her company that he/she wishes to check. Then CIS will automatically scan the endpoints for vulnerabilities, attempt to find sub-domains of the domain, and will filter and combine the results into a single file. Finally, the CIS Report Generator will produce a report based on the combined file. This report shows all the potential vulnerabilities found as well as a network map. The network map is a graphical representation of the endpoints and domain provided by the user in addition to the sub-domains and IP-addresses found by CIS.

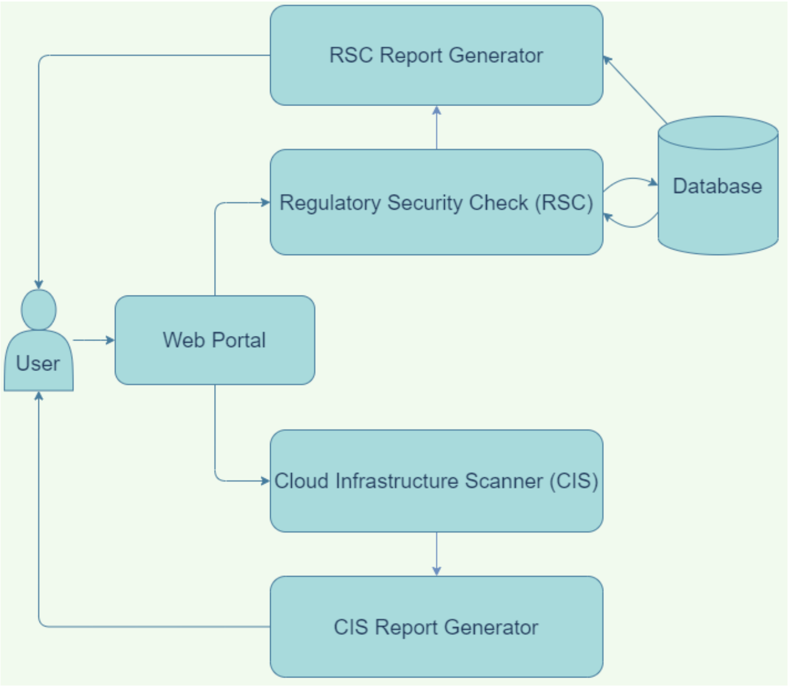

RSC Report Generator
Regulatory Security Check (RSC)
Database
User
Web Portal
Cloud Infrastructure Scanner (CIS)
CIS Report Generator

**Figure 4.1.** The high-level system architecture of RSCCI

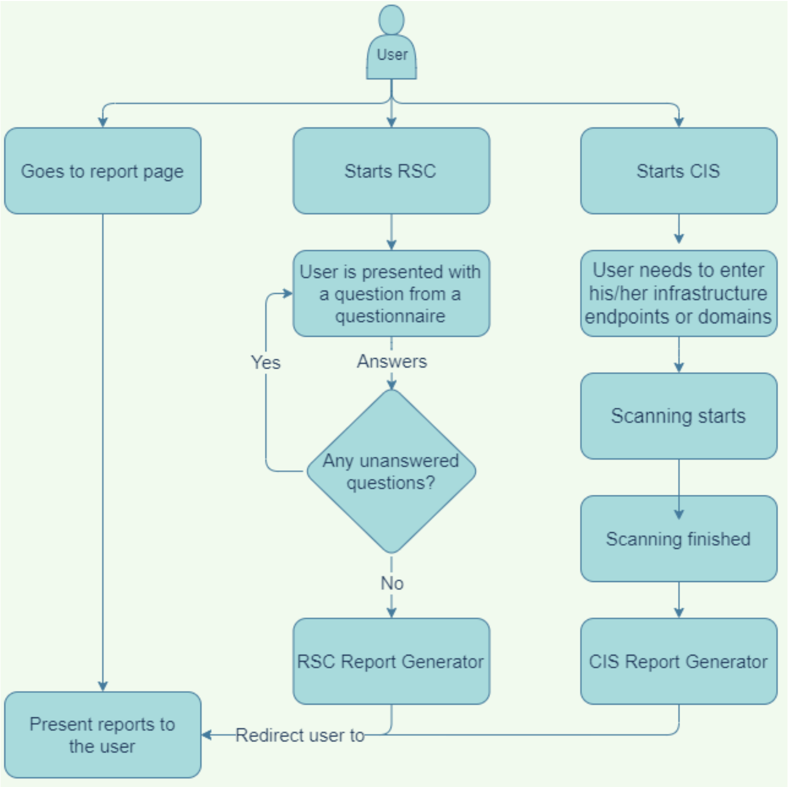

User
Goes to report page
Starts RSC
Starts CIS
User is presented with a question from a questionnaire
User needs to enter his/her infrastructure endpoints or domains
Yes
Answers
Scanning starts
Any unanswered questions?
Scanning finished
No
RSC Report Generator
CIS Report Generator
Present reports to the user
Redirect user to

**Figure 4.2.** Flowchart of web interface of RSCCI

There are multiple components and parts to both RSC and CIS. Figures 4.3 and 4.4 show low-level sequence charts of system functionality for these.

The sequence chart for RSC in Figure 4.3 shows the interaction and sequence for when a user answers a questionnaire. After starting RSC, it will get a question from the database which will be returned to the user. When a user submits his/her answer, RSC will store the answer in the database. This process will loop until there are no more unanswered questions. RSC will then generate a report and return it to the user.

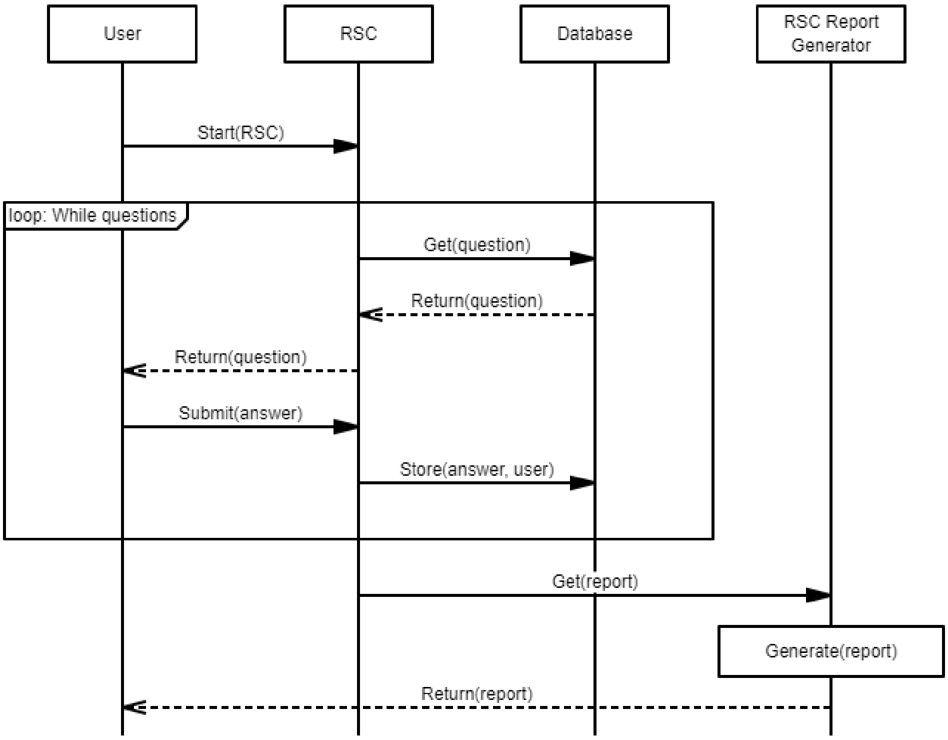


**Figure 4.3.** Sequence chart of RSC

The sequence chart for CIS in Figure 4.4 shows how a scan is performed. Whenever a user starts the scan, he/she needs to enter his/her desired endpoints. After the user has entered his/her information, CIS will start scanning the endpoints and analyze the returned data until there are no more endpoints to scan. CIS will then store the results in the database, and start the CIS report generator. This report generator will generate and return a report to the user.

## 4.2 Architectural Pattern

We decided early on to use the Model-View-Controller (MVC) design pattern for our web application, as it makes it simple to model and scale the software for future needs [30]. MVC is also one of the most used patterns in the software development industry, and many popular coding languages have existing MVC frameworks available. This includes C#, PHP, and Python [31].

MVC structures the code and architecture as its naming dictates; in models, views, and controllers. The model component manages system data and their associated operations. The view component is responsible for what the user sees, and it defines and manage how the data is presented to the user. The user does not interact directly with the views, but with the controllers. The controller component passes the user interactions to the views and models, and is responsible for returning the correct view with the correct data to the user.

Figure 4.5 shows a rough overview of our software architecture based on MVC. The dotted boxes represent a node, the fully drawn arrows represent communication between the different nodes, and the dotted arrows represent communication between the different software components.

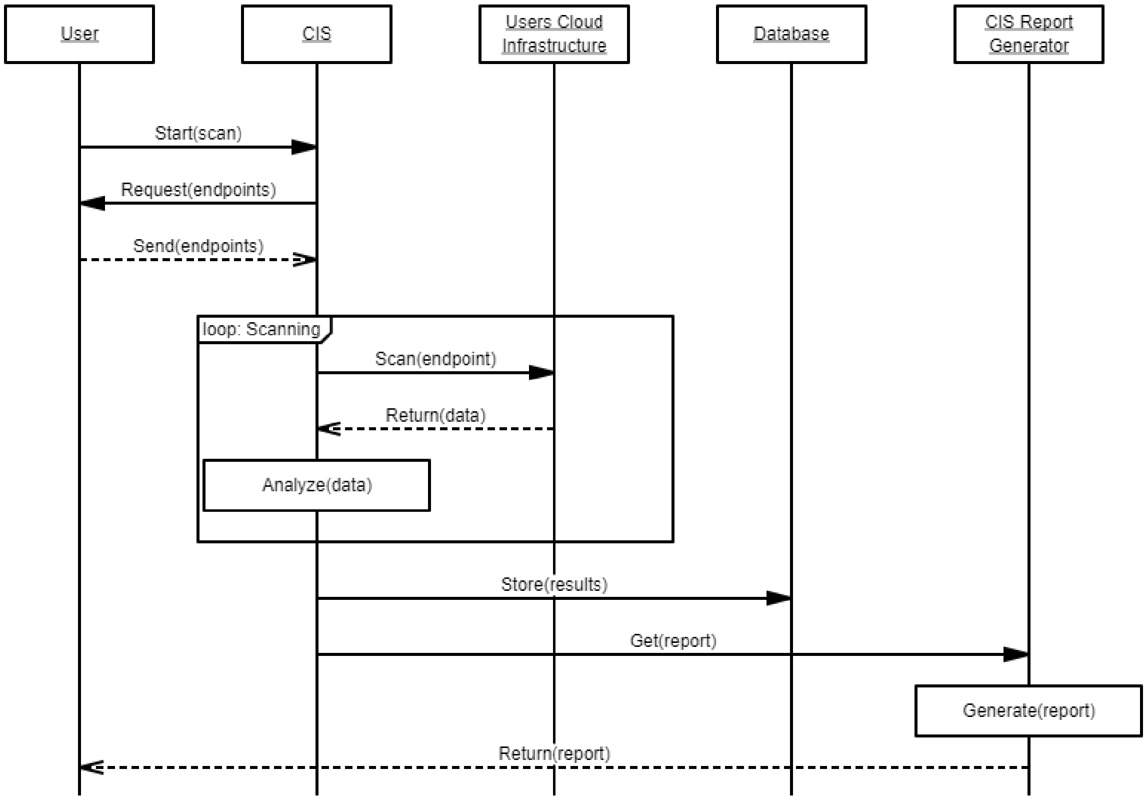


**Figure 4.4.** Sequence chart of CIS

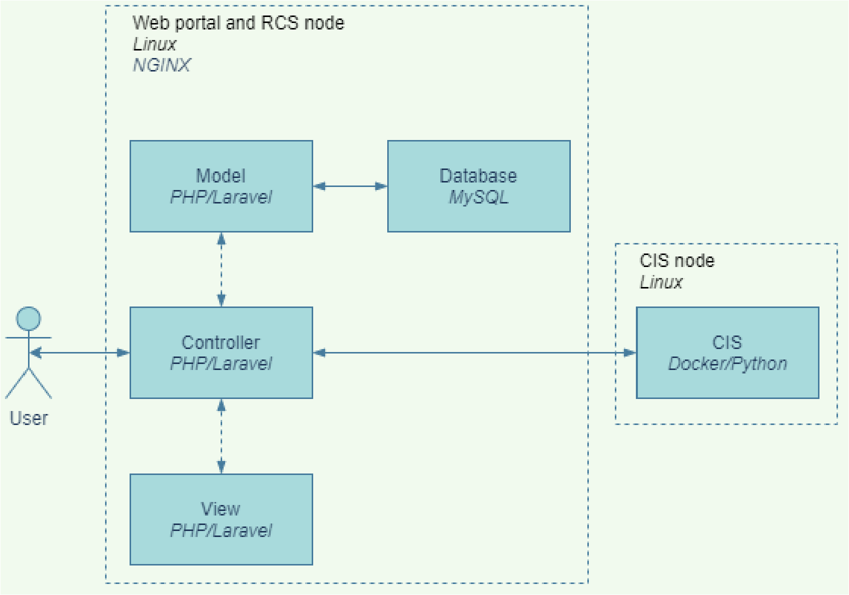
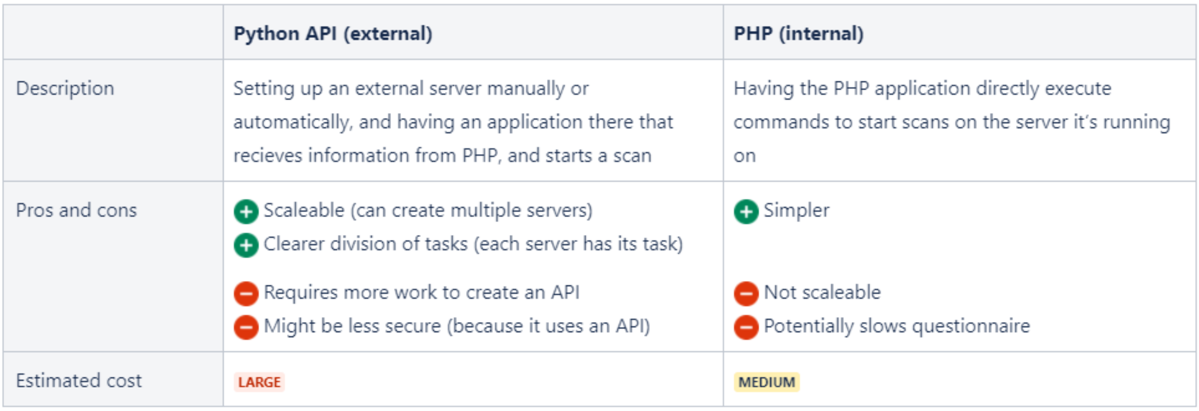


**Figure 4.5.** Our software architecture based on MVC

## 4.3 CIS communication

Different approaches for communication between CIS and the web portal were considered, and we ended up with two primary options. The first option was putting CIS on its own server and create an Application Programming Interface (API), which the web portal could communicate with. The other option was to have the web portal directly execute commands to start scan, and run code locally on the same server to start CIS and get the output. Figure 4.6 shows a table documenting our decision between the two options.

We ended up going with the first option and create an API, primarily due to the ability to scale up if needed, and make sure a scan does not affect the user experience for someone



| | Python API (external) | PHP (internal) |
|---|---|---|
| Description | Setting up an external server manually or automatically, and having an application there that recieves information from PHP, and starts a scan | Having the PHP application directly execute commands to start scans on the server it's running on |
| Pros and cons | ⊕ Scaleable (can create multiple servers)<br>⊕ Clearer division of tasks (each server has its task)<br>⊖ Requires more work to create an API<br>⊖ Might be less secure (because it uses an API) | ⊕ Simpler<br>⊖ Not scaleable<br>⊖ Potentially slows questionnaire |
| Estimated cost | LARGE | MEDIUM |

**Figure 4.6.** Decision between external and internal calling of scanner

using RSC at the same time. This option also requires an API endpoint for the web portal to receive the results from CIS. There are some security considerations for the API option, as API authentication details and user's endpoints are sent over the network. Therefore, this data should be sent encrypted, to ensure that no one can access it.

## 4.4 Design Driving Technologies and Languages

In this section we will talk about which design driving back end and front end technologies we use, and why we have chosen them.

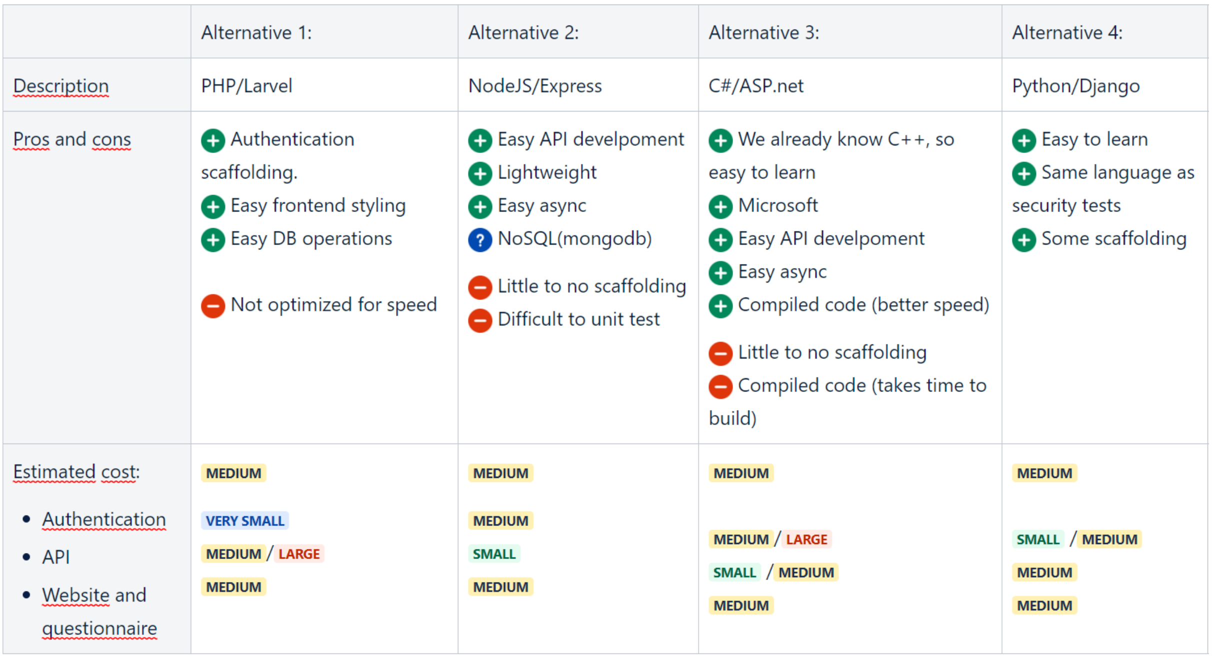
### 4.4.1 Back End

In any project, the selection of back end technologies will have a huge impact, as all the technologies has to work well together. Changing back end technologies after the implementation has started can be extremely time consuming, and a technology change in the middle of a project period might block developers from finishing their product. Therefore, we chose our back end technologies carefully.

When considering what language to use for development of the web application, we created a pro/con table as shown in Figure 4.7. We decided to go with "Alternative 1" in Figure 4.7, PHP with the Laravel framework, as PHP is a high-level language that is relatively easy to learn. In addition, the features that the Laravel framework provides, like scaffolding will help us save time in the development of the application. For the scanner part, the natural choice from the start was Python, as this is the most used scripting language in the penetration testing community, and Docker [32, 33]. In the following, we briefly introduce PHP, Laravel, LEMP stack, Python, and Docker.

- **PHP**
  PHP is a popular general-purpose language for creating dynamic web content. It is a fairly easy language to learn, and is often recommended as a beginner language for web development. Many websites and applications use PHP, most notably the content management system WordPress. According to W3Techs, WordPress is used by 40% of websites [34].
  By using PHP with the Laravel framework and some PHP packages, we have to write less code and can develop our application faster.

- **Laravel**
  Laravel is an open source PHP framework from 2011, and is intended to be used in development of applications following the MVC design pattern [35]. It provides scaffolding and built-in libraries, which simplifies the development process. Using Laravel's scaffolding, we get a complete user system with authentication, e-mail verification and team features from the start. The user system uses Laravel Fortify as its back end, which provides login,



| | Alternative 1: | Alternative 2: | Alternative 3: | Alternative 4: |
|---|---|---|---|---|
| Description | PHP/Larvel | NodeJS/Express | C#/ASP.net | Python/Django |
| Pros and cons | ⊕ Authentication scaffolding.<br>⊕ Easy frontend styling<br>⊕ Easy DB operations<br>⊖ Not optimized for speed | ⊕ Easy API develpoment<br>⊕ Lightweight<br>⊕ Easy async<br>? NoSQL(mongodb)<br>⊖ Little to no scaffolding<br>⊖ Difficult to unit test | ⊕ We already know C++, so easy to learn<br>⊕ Microsoft<br>⊕ Easy API develpoment<br>⊕ Easy async<br>⊕ Compiled code (better speed)<br>⊖ Little to no scaffolding<br>⊖ Compiled code (takes time to build) | ⊕ Easy to learn<br>⊕ Same language as security tests<br>⊕ Some scaffolding |
| Estimated cost:<br>• Authentication<br>• API<br>• Website and questionnaire | MEDIUM<br>VERY SMALL<br>MEDIUM / LARGE<br>MEDIUM | MEDIUM<br>MEDIUM<br>SMALL<br>MEDIUM | MEDIUM<br>MEDIUM / LARGE<br>SMALL / MEDIUM<br>MEDIUM | MEDIUM<br>SMALL / MEDIUM<br>MEDIUM<br>MEDIUM |

**Figure 4.7.** Back end languages considered

registration, password reset, and email verification [36].

- **LEMP stack**
  A Linux Nginx MySQL PHP (LEMP) stack is a popular solution stack used to serve web applications [37]. In a LEMP stack, Linux is used as the OS, NGINX as the web server, MySQL to store data, and PHP is used to process dynamic content. Below we introduce Linux, NGINX, and MySQL.

  - **Linux**
    Linux is a popular OS, and most servers nowadays runs on Linux [38]. There are multiple distributions of Linux available, but we chose to use Ubuntu for our LEMP stack, since it is relatively easy to use and learn and well documented. Specifically, we use Ubuntu 20.04, as this is the latest version of Ubuntu.

  - **NGINX**
    NGINX (pronounced Engine-X) is open source software, and is often used as a web server. However, it can also be used for reverse proxying, caching, load balancing, media streaming, and more. NGINX is one of the most used web service softwares, and beats Apache on several areas [39]. It also performs really well with Linux, and is easy to set up.

  - **MySQL**
    MySQL is the most popular open source SQL database management system. It structures collections of data, and uses the SQL language to query data. Instead of adding the data in one big storeroom, MySQL structures it into multiple tables. This is called a relational database. A relational database queries data fast, as it does not have to go through all the data to find what it is looking for [40].

  We chose to use a LEMP stack for our application since we have experience with this software, and because Linux and NGINX perform very well together.

- **Python**
  Python is another easy to learn, general-purpose language used to create everything from simple scripts, to complex machine learning and math applications [41]. Python is a great language to use for a security scanner, as the penetration testing community often use Python to create new tools and automate existing tools. In addition, Python has libraries for API and Docker communication, which we need in order to communicate between the web portal and CIS.

- **Docker**
  Docker is a set of products and technologies used to deliver software in packages called containers. These Docker containers are lightweight, standalone and executable packages of software that includes the code, tools, system libraries and settings needed to run an application [42]. Docker containers are similar to VMs, and provide nearly the same level of separation. However, Docker containers are a lot faster because they share the same Linux kernel as the host. We used Docker to package the different tools and code need for CIS into a single container. This means that we can easily spin up multiple CIS instances on demand and run multiple instances simultaneously.

### 4.4.2 Front End

This section covers the technologies used for the front end of our web application. We used a variety of different technologies and frameworks to create a dynamic and interactable front end. The team members that developed the web application are mostly used to back end developing, and creating a good front end is not always easy for back end developers. To make sure that we did not spend all our time trying different CSS properties, we needed to use some libraries. Here our choice was pretty easy, as we used the CSS and JavaScript (JS) frameworks that comes pre-installed in the Jetstream starter kit. We also used Laravel's built-in Blade templates and Laravel Livewire to create our front end.

- **CSS**
  A CSS framework is a library of pre-defined CSS styles that can simplify the styling of a web page, by giving users CSS classes they can use in the HTML, instead of manually editing a CSS stylesheet. The included CSS framwork is called Tailwind CSS, and is a relatively new framework that provide utility classes to use when designing a web page. It does not provide complete components like the popular CSS framework Bootstrap, but allows users to easily construct customized components by adding classes to HTML elements. There are also multiple sites[1] that provide pre-made components.

- **JavaScript**
  The JS frameworks we use is called Alpine.js, and is also a relatively new framework. On the frameworks Github page they say: "Think of it like Tailwind for JavaScript.", which summarises the framework pretty well. There is no need for a separate script section, or a separate file for components, everything is done in the HTML.

- **Blade Templates**
  Blade is the templating engine that is included with Laravel, and is the "view" part of the MVC architecture [43]. These Blade files combine and compile the required HTML, PHP, JS and CSS in to HTML which is sent to a user and rendered in the users browser. Since Blade is included with Laravel, it was a natural choice for our views.

[1] Many of the sites providing pre-made Tailwind components have been collected in the awesome-tailwindcss Github page.

- **Livewire**
  Livewire is another framework we use, and help us make dynamic content. Livewire is a connector between front end and back end, and makes it easy to create dynamic elements [44]. Livewire works by initially rendering a component within a page, like a blade template, and whenever a user interact with an element inside the component, Livewire makes an AJAX request to the back end server with data about the interaction. The server then re-renders the component and respond with some new HTML. Livewire then intelligently merges the old and the new HTML, based on what was changed.

## 4.5 User friendliness

What we have tried to acheive in RSCCI is to keep it intuitive, simple and easy to use. RSCCI is designed to be easy to understand how to use and rely on. High complexity is avoided and bugs are fixed so that the application functions properly. The development of this application from the beginning has aimed to meet the needs of our client Axeptia and satisfy their expectations. This makes user experience (UX) fullfiled and regarding user interface (UI) the easy to understand and well customized interface of RSCCI makes the target group well-pleased.

### Scanner

As shown in Figure 4.9 the progress bar developed shows the scanner is working as it should to make it easier for users to wait until it has finished the process of scanning. Figure 4.8 shows that the network map deployed in the application is based on the information entered by user. This map makes it more interesting for users to work with the application and find out about the existing vulnerabilities in their network. At the same time that it shows the result of the scan, it makes the whole process more exciting using colors and flags while presenting the domain, sub-domains and IP-addresses.

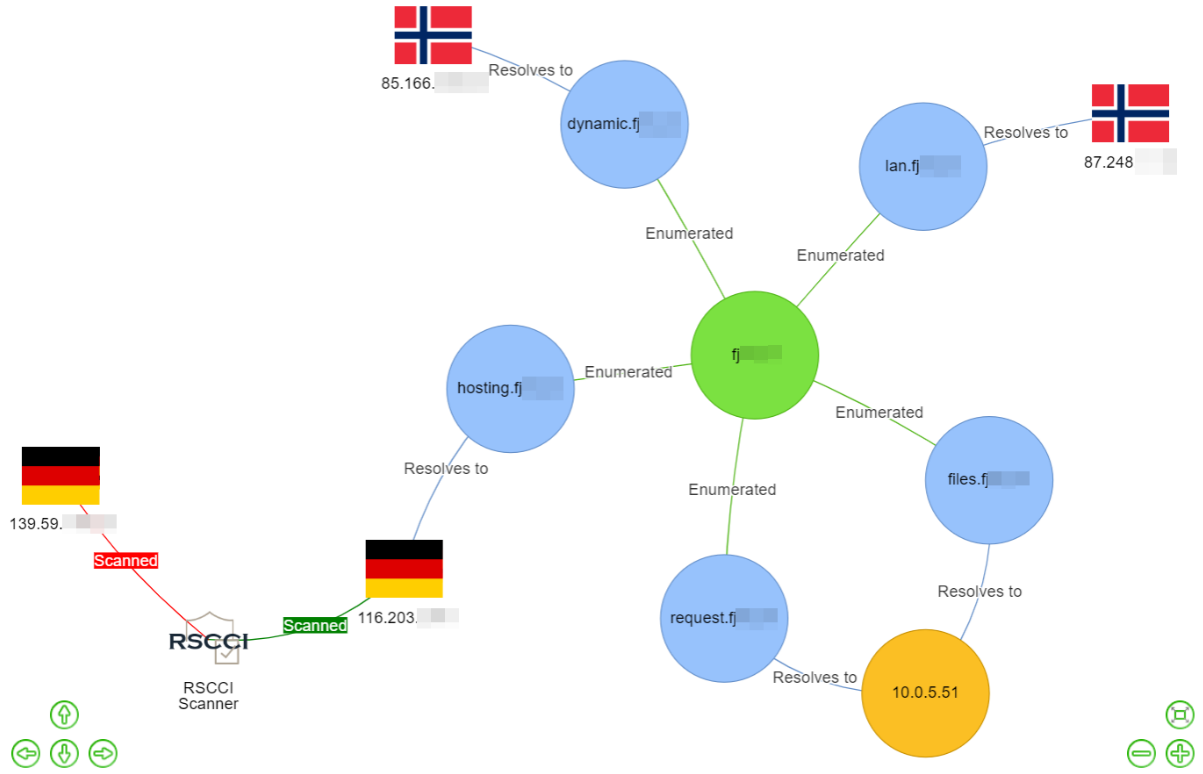


**Figure 4.8.** A network map based on user provided information.

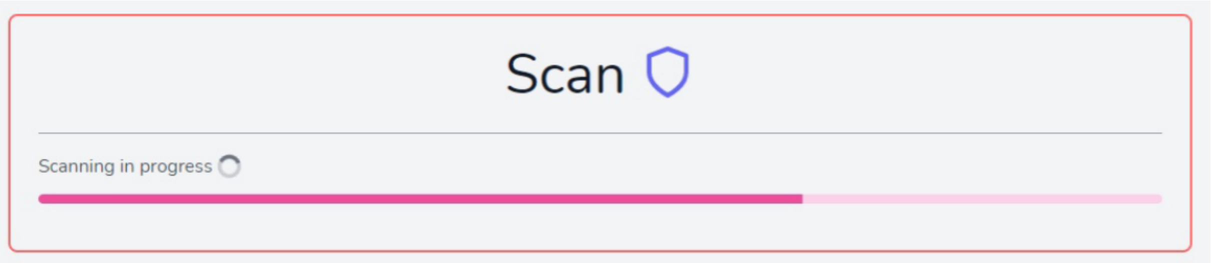
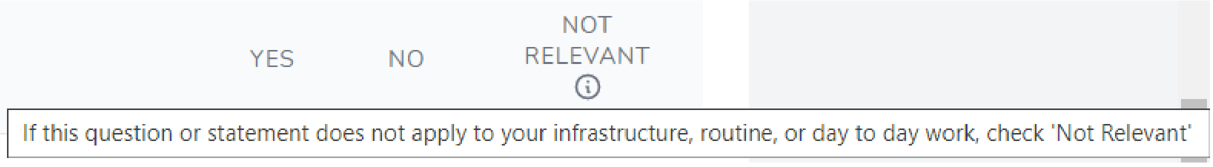


**Figure 4.9.** The progress bar shows how much of the scanning process is left to fulfil.

## Web application

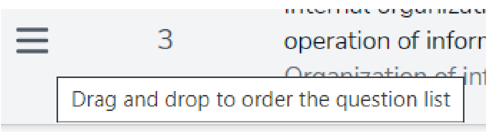
There are developed tooltips to give information to users both for administrators and participants regarding the functionality of each part. For an example of these please see Figure 4.10 and 4.11. In addition the web page is developed in a solid manner which offers a clean design to users. This makes it usable to a large extend for everyone as we have tried to use a universal design [45].

Furthermore, as shown in Figure 4.12, the progress bar is also a way to communicate with the participant about the remaining part of each scanning process or the questionnaire under completion. The same figure shows a button which the participant can click on to escape the current questionnaire and proceed in a time which is more desirable to fulfil it. In this manner the participant is not kept against their wish in one questionnaire but rather can enjoy the flexibility of this application and finish it when it suits them. Use of colors makes it easier for the user to see the state of each of questionnaire. In this context green color means that the test is done, yellow/orange means that the test is started but not finished yet, and purple means that the test is not started yet. See Figure 4.13.



If this question or statement does not apply to your infrastructure, routine, or day to day work, check 'Not Relevant'

**Figure 4.10.** This tooltip gives information about issues not relevant to the participant/company.



Drag and drop to order the question list

**Figure 4.11.** This tooltip tells the administrator how to interact with the structure of the questions.

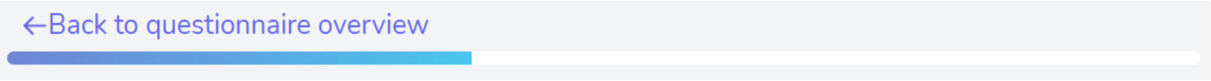
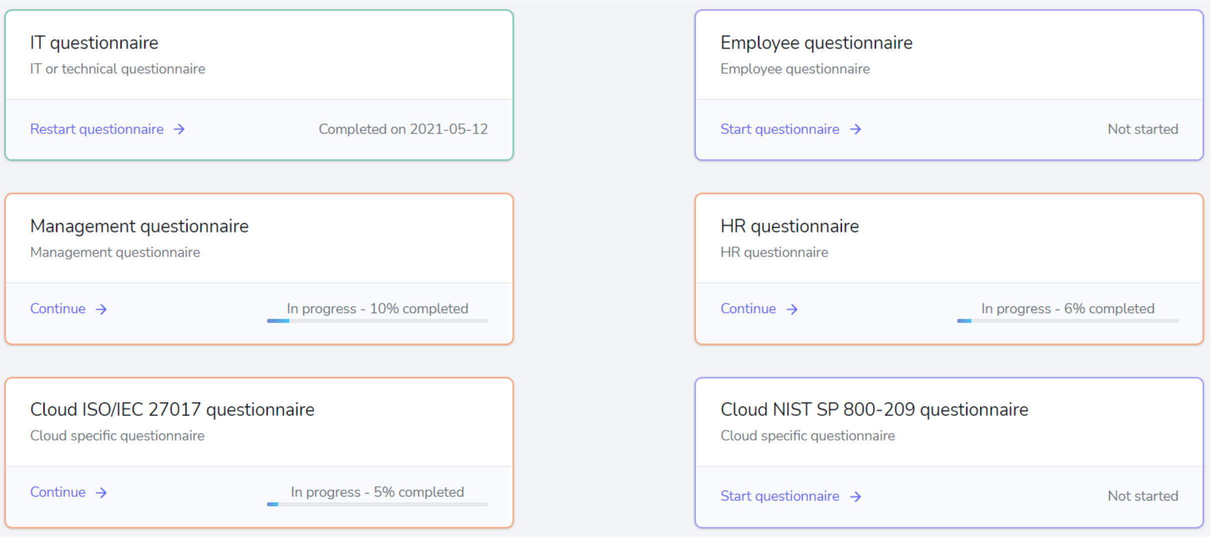


**Figure 4.12.** The progress bar shows the participant how much of the questionnaire is left to fulfil.

## Questionnaire

When it comes to the questionnaire to achieve user friendliness in this application we tried to keep it close to universal design standards [46, 47]. The format in which we have made the questionnaires is meant to make it easy for the participant to be able to choose if a specific control is relevant to their organization, if so is it in place or not. We have tried to keep same



IT questionnaire
IT or technical questionnaire
Restart questionnaire → Completed on 2021-05-12

Employee questionnaire
Employee questionnaire
Start questionnaire → Not started

Management questionnaire
Management questionnaire
Continue → In progress - 10% completed

HR questionnaire
HR questionnaire
Continue → In progress - 6% completed

Cloud ISO/IEC 27017 questionnaire
Cloud specific questionnaire
Continue → In progress - 5% completed

Cloud NIST SP 800-209 questionnaire
Cloud specific questionnaire
Start questionnaire → Not started

**Figure 4.13.** Use of colors in the questionnaire.

format all the way as when the participant gets used to it, they will complete the test easier. There is applied introduction to each questionnaire to explain the importance of the test and to convince the responders about why the answers should be as accurate as possible.

We have tried to briefly outline the purpose of the questionnaire. It has been a difficult task to keep the questionnaires as short as possible as we did not want to cut out parts of the ISO standards. Because each ISO standard is a whole and removing parts of it makes gap in security control checking. But we have tried to be as general as possible in the way questions are formulated. We have tried to outline the duration of each questionnaire by developing a progress bar for each one as mentioned above and shown in the Figure 4.12. In this way we make it easier for the participants to know how many questions are done, and how much is left.

We have tried to make the questionnaire as simple as possible, one question at a time so that a participant can answer with only "Yes" or "No". The questionnaire is more like a checklist that goes through some control checks and gives a result with some guidelines as a list that the participant can use to reach a better cybersecurity for their organization. Therefor, the question style suited best for this kind of choice was the closed question style [47]. On the other hand closed question style that we have chosen makes it take less time for the participants to fulfill the questionnaire, and at the same time have a good overview of the level of security in their organization.

Open question style was not suitable in this kind of questionnaire needed in this application. Despite this, questionnaires are developed to improve each organization's cloud information security and this is actually an opportunity for them. But we have provided an information page at the end of each test to thank the participants for their participation, as well as guide them regarding the further process to the report produced by the application. This questionnaire collects personally information such as name and e-mail address, but not personally sensitive information. To improve compliance with GDPR regarding these personal information we wish to do certain tasks in the future to provide a better security for the users' private data. Please see Section 9.3.

Also the test results are confidential and will be available only to the organization which attempts the cloud security check with this RSCCI application. Several different people from Axeptia, our client, have tested RSCCI application so that we get the opportunity to fix as much problems as possible in this application before we distribute it. For an overview of the test results please see Chapter 7. Whoever this progress is dependent on the time we have available during this bachelor thesis project. About the plan for future work on the application of RSCCI please visit Section 9.3.

# Chapter 5

# Development Process

In this chapter, we will cover the development process of the project. We will talk about the chosen development model, documentation, and routines.

## 5.1 Development model

All of our team members are familiar with working after either the Scrum model or the Kanban model. These two models are similar, but Kanban offers more flexibility. Therefore, we have chosen to use Kanban for this project. One of the main benefits of using Kanban is continuous flow [48]. Scrum usually has fixed sprint lengths where it is expected that a task is finished during one sprint [49]. This may cause complex tasks to be split into several sub tasks. Kanban has no sprints, and there is no need to split bigger tasks. This means that we spend less time on task defining, especially if we are unsure how big a task actually is and how much time a task will take.

The Kanban model also offers more continuous delivery, where the features are released when they are ready, without having to follow a time schedule or predefined release dates. In Scrum, on the other hand, the team usually releases the new features at the end of a sprint.

The Scrum model divides the team in multiple roles, where the most central are scrum product owner, scrum master, and scrum development team, whereas Kanban has no predefined roles. As we are a small team, we prefer not to use formal roles, but rather have a flat team hierarchy.

During Scrum sprints, the team should not make changes that are not agreed upon before the sprint started, but rather wait for the next sprint. When using Kanban, the team can do changes whenever needed. This has been useful during the project period.

A Kanban team's work resolves around a Kanban board to visualize and keep track of different stages and components of the project. As we were working remotely, we chose to use a digital Kanban board: the agile project management tool Jira [50]. During the project period, we divided the different development features and tasks into issues, and used Jira to track the issues as well as monitor the progress of our project. Any additional technical challenges or bugs we encountered were also added as issues.

## 5.2 Documentation

This section will cover how the different aspects of the project was documented. During the project period, we continually documented to make sure important decisions, figures, notes, and guides did not get lost or forgotten.

### Work Management

We used different tools in different parts of the project to keep track of which team member was working on what task. As stated in Section 5.1, we used Jira to keep track of the issues when developing the web application. For other tasks, like the the questionnaire related ones, Trello was used. When it came to keeping track of the final thesis, we chose to use an Excel document to make sure all chapters and sections were covered, if they were sent to review, and if they were finished.

### Report Writing

For the report writing, we used LATEXwith the editor Overleaf. This enabled us to work in real time with the same document, and easily compile the source code to see how the document would turn out as a PDF. As NTNU provides bachelor thesis templates for LATEX, this was a natural choice.

### Source Code

The source code for the web application and scanner were managed by the Git tool Bitbucket. Bitbucket provides source control, collaboration features, and enables DevOps methodology when developing code, making it perfect for our needs [51]. It was also possible to integrate Bitbucket with Jira to keep track of which issues had which branch, and automatically close an issue when the code in its corresponding branch was merged into master. We chose to split the code into two Bitbucket repositories: one for the web application and one for the scanner.

### Project Meeting Notes

There was taken notes from meetings within the group, with the supervisor, and with Axeptia. We had a dedicated secretary who took notes, and posted them either to our shared OneDrive or our Teams channel. In case the secretary could not make it to one of the meetings, we had appointed another team member to take over. This made sure we always had someone to take notes.

### Other Project Documentation

For other aspects of the project that needed to be documented, we used Confluence [52]. Confluence offers multiple templates for different documentation needs, e.g. decision making. It also offers integrations with multiple third-party tools, making it excellent for project documentation. We used Confluence for documenting decisions, the different figures needed for the design and development phase, and the user guide for setting up the final product. Some of these documents can be found in Appendix E.

## 5.3 Routines

The routines we planned and followed carefully in this teamwork were strongly impacted by the situation induced from the pandemic of Covid-19. Meetings mainly became online meetings and contacts went on Discord and Teams.

### With supervisor

We held regular weekly meetings with our supervisor Kelly on Teams. This channel was where we communicated with her, sharing our weekly reports and other documents with her. Each weekly report among others includes the following topics:

- What have you done in the previous week?
- Did you encounter any problems? How did you solve them?
- What do you want to discuss with the supervisor?
- What are you going to do in the following week?
- Are your group still on track?

As the report implies during these meetings we got guide and follow up by the supervisor. Each report was available to the supervisor one day before each meeting so that she could take a look at it before the meeting.

### With client

We held meetings every other week with Axeptia as our client on Teams. In the meetings we showed every change and heard their suggestions. In some of these meetings Per Nestor and Håkon tested and discussed the scanning functionality of RSCCI.

### Group Internal

We held internal group meetings every Tuesday using Discord. This platform was also used for internal chat and notifications. During these meetings we took a look at the tasks which was delegated to each group members, and divided new tasks to each of us. These meetings had the necessary functionality of both internal communication and control of the task progress.

We have grown fond of Atlassian software for agile teams, as they support our chosen development model well and can be configured to communicate with each other. The software we used are Jira Software, Trello, Bitbucket, and Confluence, see Figure 5.1. For the time track of each individual group members' work, we used Clockify since it was well suited to manage this task. Clockify was possible to integrate with Jira, which we took advantage of to help keeping track of how much time was spent on each specific issue.

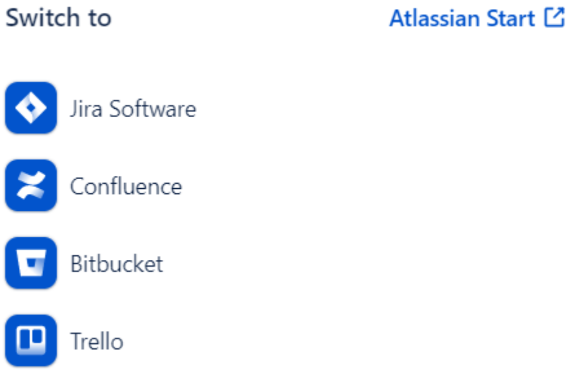


**Figure 5.1.** Atlassian tools used to accomplish internal group workflow.

## Development routines

As we were multiple people to develop the source code for RSCCI, we set up some development routines. The purpose of the routines were to reduce the risk of merge conflicts, have a DevOps approach to the development process, and make sure that the final code is of good quality. A graphical representation of the development workflow can be seen in Figure 5.2, and consists of the following steps/rules:

- Each contributor clones the project down to their local computer.
- Each contributor will work on their own branch, no code is ever to be pushed directly to Master. Exception: README.md.
- There should be created a new branch for each task/feature with a short, descriptive name. When the changes are merged to Master, the branch used should be closed/deleted.
- When a contributor is done with their task, a Pull Request (PR) is to be made with at least one other contributor as reviewer.
- When one reviewer has reviewed and approved the changes related to the task, the contributor is allowed to merge their changes into the Master.

- In case of pair programming, the PR does not need to be reviewed and approved by another reviewer.
- Each time there is a merge with the master branch, the other contributors has to pull down the changes and merge them into their own branch to prevent merge hell later.

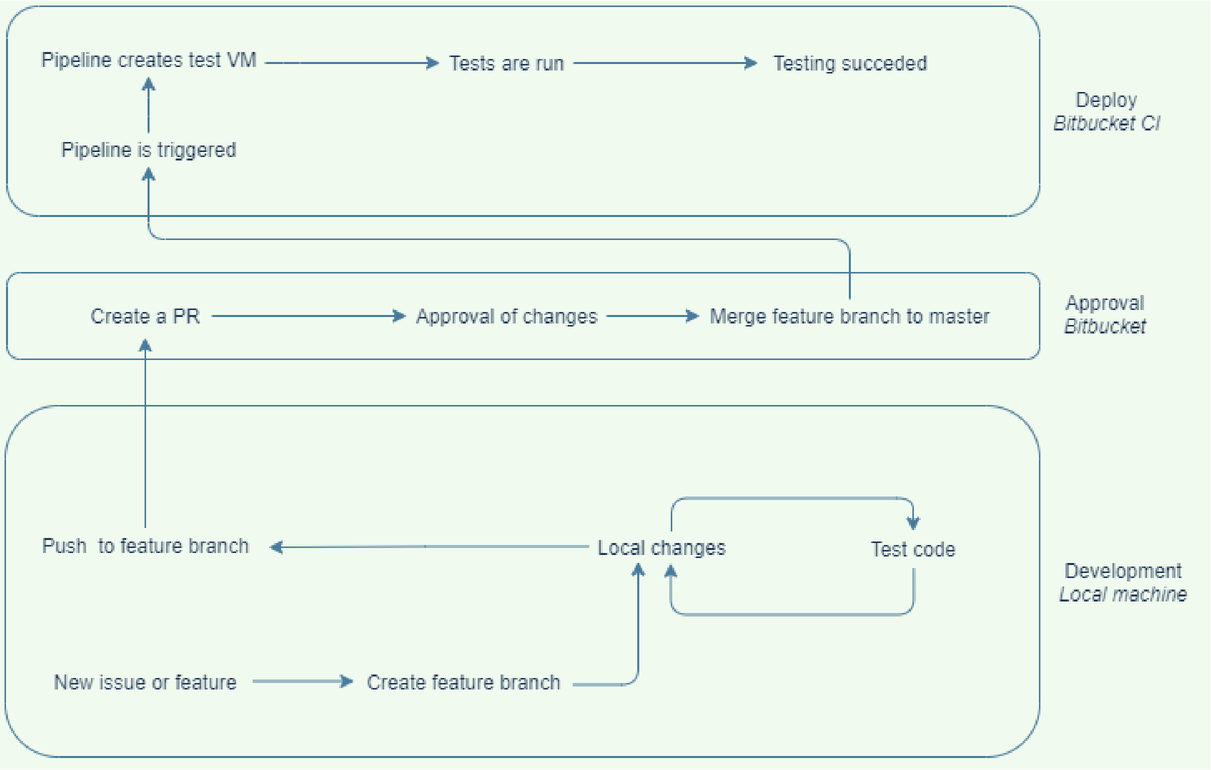


**Figure 5.2.** Graphical representation of the development workflow

These steps/rules are located in the Bitbucket repository's README.md. The full README.md file can be found as a PDF in Appendix F.

To take advantage of Bitbuckets DevOps functionality, we set up a simple CI/CD pipeline for the repository holding the code for the back end and front end. The pipeline code can be seen in Code listing 5.1. CI stands for *Continuous Integration*, and CD for either *Continuous Delivery* or *Continuous Deployment*, depending on pipeline functionality [53].

Continuous Integration means that the developers merge code changes into the master branch often. The merged changes are validated by automatic building and testing. In our pipeline, the validation happens every time a piece of code is pushed or merged to the master branch. The testing consists of running all implemented feature- and unit tests. If any of these tests fails, we get a notification. We also get a notification if anything fails. Continuous Integration helps with limiting integration challenges when all changes are pushed into the master branch at the same time for release.

Continuous Delivery means that the code tested during the Continuous Integration part is deployed to a testing and/or production environment after the building and testing is completed. Deployment to testing environment happens automatically, while deployment to production environment needs some kind of human interaction to be triggered. The deployment could be done at any time interval, for example every time the pipeline runs or once a week. Our pipeline was planned to support Continuous Delivery by having it deploy the new code to our development environment automatically on success. However, as we where hosting the development environment locally, this was not as easy as we initially thought, and was therefore not implemented. Read more about the development environment in Section 6.4.1.

Continuous Deployment is similar to Continuous Delivery, but takes it one step further: instead of needing human interaction for deploying the changes to production environment, the pipeline deploys it directly and automatically. Like Continuous Delivery, the deployment can happen at any time interval. Continuous Deployment was not considered for our project.

**Code listing 5.1.** Yaml code of our Bitbucket pipeline

```yaml
image: php:7.4-fpm

pipelines:
  branches:
    master:
      - step:
          name: Build and test
          caches:
            - composer
          script:
            - apt-get update && apt-get install -qy git curl libmcrypt-dev mariadb-client ghostscript
            - yes | pecl install mcrypt-1.0.3
            - docker-php-ext-install pdo_mysql bcmath pcntl
            - curl -sS "https://getcomposer.org/installer" | php -- --install-dir=./ --filename=composer
            - composer install
            - ln -f -s .env.pipelines .env #Creates link between .env.pipelines and .env
            - php artisan migrate #Creates DB
            - ./vendor/bin/phpunit #Runs unit- and feature-tests
          services:
            - mysql
            - redis
definitions:
  services:
    mysql:
      image: mysql:5.7
      environment:
        MYSQL_DATABASE: $PIPE_MYSQL_DATABASE #Has to be same as DB_DATABASE in .env.pipelines
        MYSQL_RANDOM_ROOT_PASSWORD: $PIPE_MYSQL_RANDOM_ROOT_PASSWORD
        MYSQL_USER: $PIPE_MYSQL_USER #Has to be same as DB_USERNAME in .env.pipelines
        MYSQL_PASSWORD: $PIPE_MYSQL_PASSWORD #Has to be same as DB_PASSWORD in .env.pipelines
    redis:
      image: redis
```

# Chapter 6

# Implementation

This chapter covers the code implementation, infrastructure configuration, and questionnaire implementation. The discussion of the code implementation is divided into three parts: Front End, Back End, and Scanner.

## 6.1 Front End

This section covers the front end implementation of the web portal, which is what a user sees and interacts with. We will provide code examples to show how the code is structured, and figures to illustrate what components we used to build up different parts of the web application.

### 6.1.1 Code structure and examples

As stated in Section 4.2, we use MVC as our architectural pattern and divide the code into models, views and controllers. Most of the views consist of multiple Blade templates and Livewire components to create the desired front end functionality and experience.

The most used Blade template is *layout.blade.php*. This Blade consists of our top and bottom navigation bars, as well as CSS and JS imports. All of our other Blade templates extends this page, meaning you will find the same navigation bars at all pages with different content in between. Code listing 6.1 shows how this is implemented on the Blade template *homepage.blade.php*. To place the content where desired in relation to the navigation bars, we use the code *@yield('content')* and *@yield('script')* in the Blade template *layout.blade.php*.

**Code listing 6.1.** homepage.blade.php

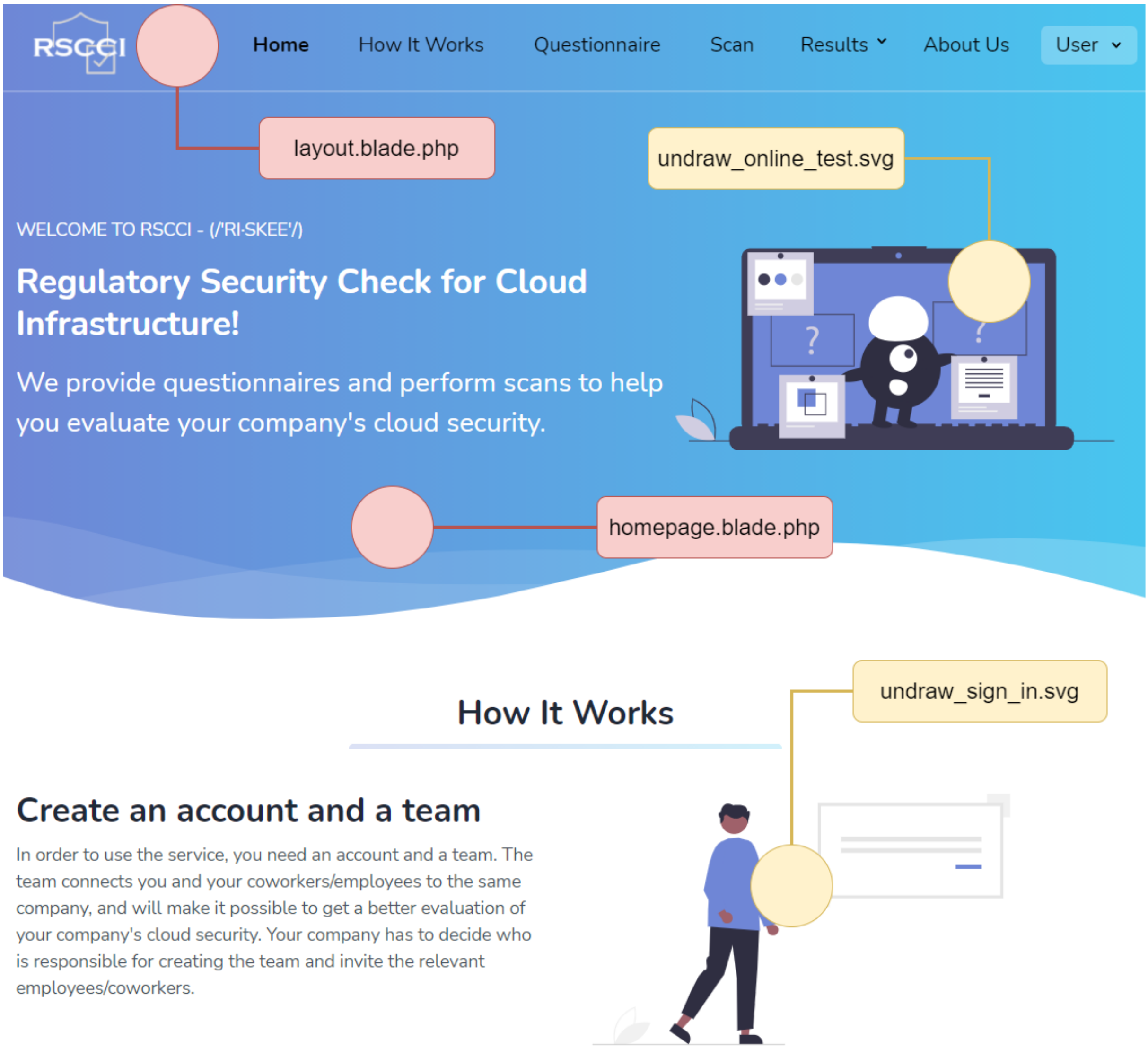
```
@extends('layout')

@section('content')
    [...]
@endsection
@section('script')
    <script>
        [...]
    </script>
@endsection
```

In the rest of this section we will provide examples of the code structure for the most important features in our application. For our code structure examples, we have given the different components their own color. The red components are Blade templates or Livewire components, the blue are routes, the green are functions, the yellow are figures, and the purple are JS components. It is possible to distinguish between the Blade templates and the Livewire components, as the Livewire components are prefixed with *livewire/* in the figures.

### Homepage

Figure 6.1 shows a selection of the components we use to build our homepage. The navigation bar is part of the layout.blade.php, and the main contents of the page is from the homepage.blade.php file. The page also consists of images loaded from the /img directory.



WELCOME TO RSCCI - (/'RI-SKEE'/)

Regulatory Security Check for Cloud Infrastructure!

We provide questionnaires and perform scans to help you evaluate your company's cloud security.

How It Works

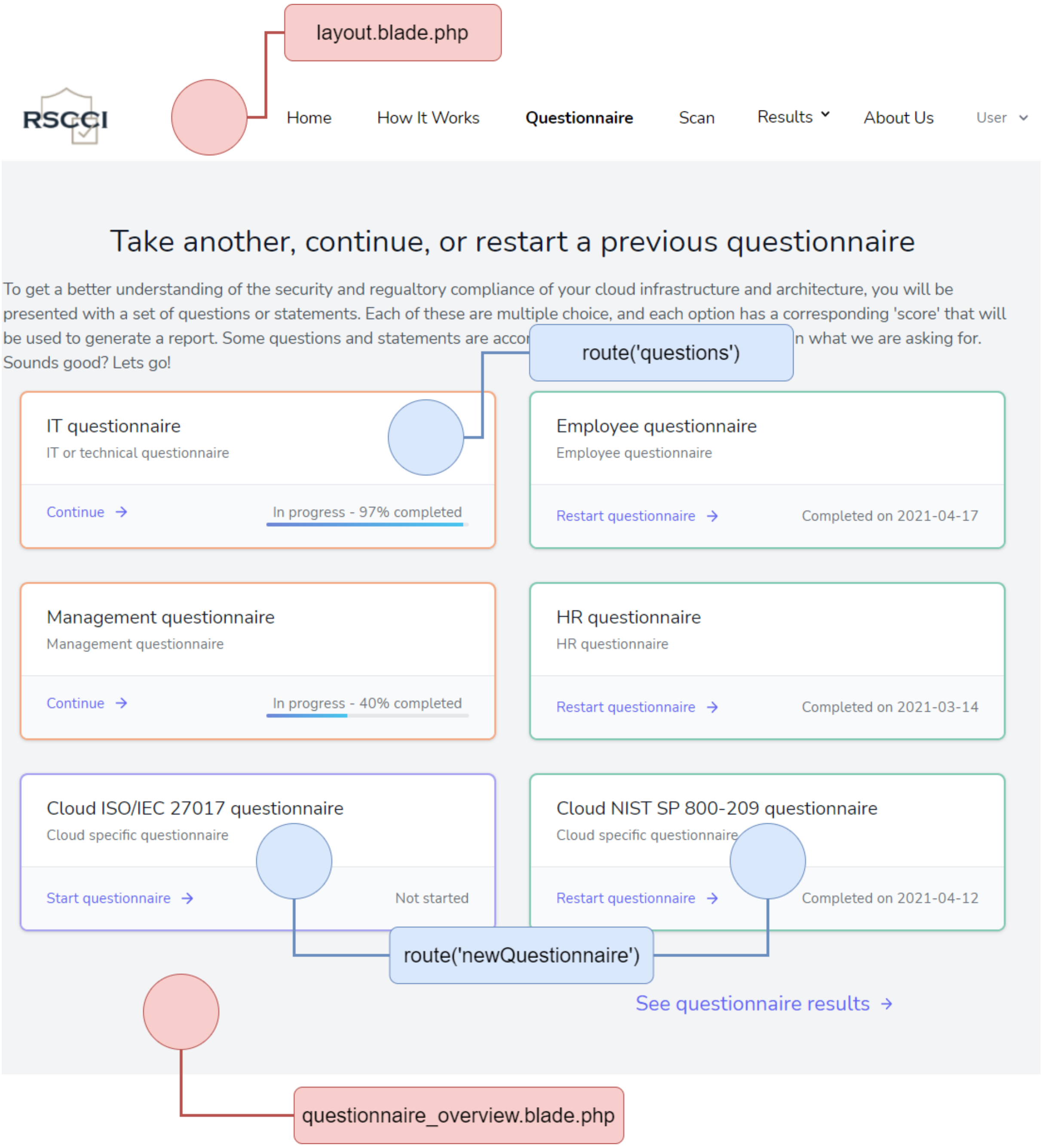
Create an account and a team

In order to use the service, you need an account and a team. The team connects you and your coworkers/employees to the same company, and will make it possible to get a better evaluation of your company's cloud security. Your company has to decide who is responsible for creating the team and invite the relevant employees/coworkers.

**Figure 6.1.** Components used to build Homepage

### Questionnaire overview page

Figure 6.2 shows what components are used to build up the page showing an overview of the available questionnaires. Like the homepage, the navigation bar is part of the *layout.blade.php* page, while the rest are from the Blade template *questionnaireoverview.blade.php*. There are two different routes on this page, based on whether a user should continue a questionnaire or start a new one. When this page gets rendered, the routes are translated into URLs, placed in a *href*, which when clicked redirects the user to either where he/she was on a started questionnaire, or to the start of the new one.



Take another, continue, or restart a previous questionnaire

To get a better understanding of the security and regualtory compliance of your cloud infrastructure and architecture, you will be presented with a set of questions or statements. Each of these are multiple choice, and each option has a corresponding 'score' that will be used to generate a report. Some questions and statements are accor n what we are asking for. Sounds good? Lets go!

IT questionnaire
IT or technical questionnaire
Continue → In progress - 97% completed

Employee questionnaire
Employee questionnaire
Restart questionnaire → Completed on 2021-04-17

Management questionnaire
Management questionnaire
Continue → In progress - 40% completed

HR questionnaire
HR questionnaire
Restart questionnaire → Completed on 2021-03-14

Cloud ISO/IEC 27017 questionnaire
Cloud specific questionnaire
Start questionnaire → Not started

Cloud NIST SP 800-209 questionnaire
Cloud specific questionnaire
Restart questionnaire → Completed on 2021-04-12

See questionnaire results →

**Figure 6.2.** Components used to build Questionnaire overview page

### Question page

Figure 6.3 shows the construction of the page presenting the questions to a user. This page consists of two Blade templates and one Livewire component. The Livewire component *showquestion.blade.php* is called from the Blade template *questionnaire.blade.php* as shown in Code listing 6.2. In the call, we also send the necessary variables *$survey* and *$questionnaire* to the Livewire component.

The Livewire component then renders the question form and progress bar, and uses Livewire functions to be able to load either the next or the previous question without reloading the entire page. The *nextQuestion()* function takes the user's answers and inserts them to the database before loading the next question. The *previousQuestion()* function does not save the user's input. It only loads the previous question and what the user answered on that question.

**Code listing 6.2.** questionnaire.blade.php

```
<div>
    <livewire:show-question :survey="$survey" :questionnaire="$questionnaire" />
</div>
```

Code listing 6.3 and 6.4 shows an example of how a simple "load next question, without reloading the page" function can be implemented using Livewire. The *$survey* and *$questionnaire* variables are sent from the *questionnaire.blade.php* template file, loading the Livewire component. Code listing 6.3 shows the code for the view that displays the intro, text, description, and category of a question.

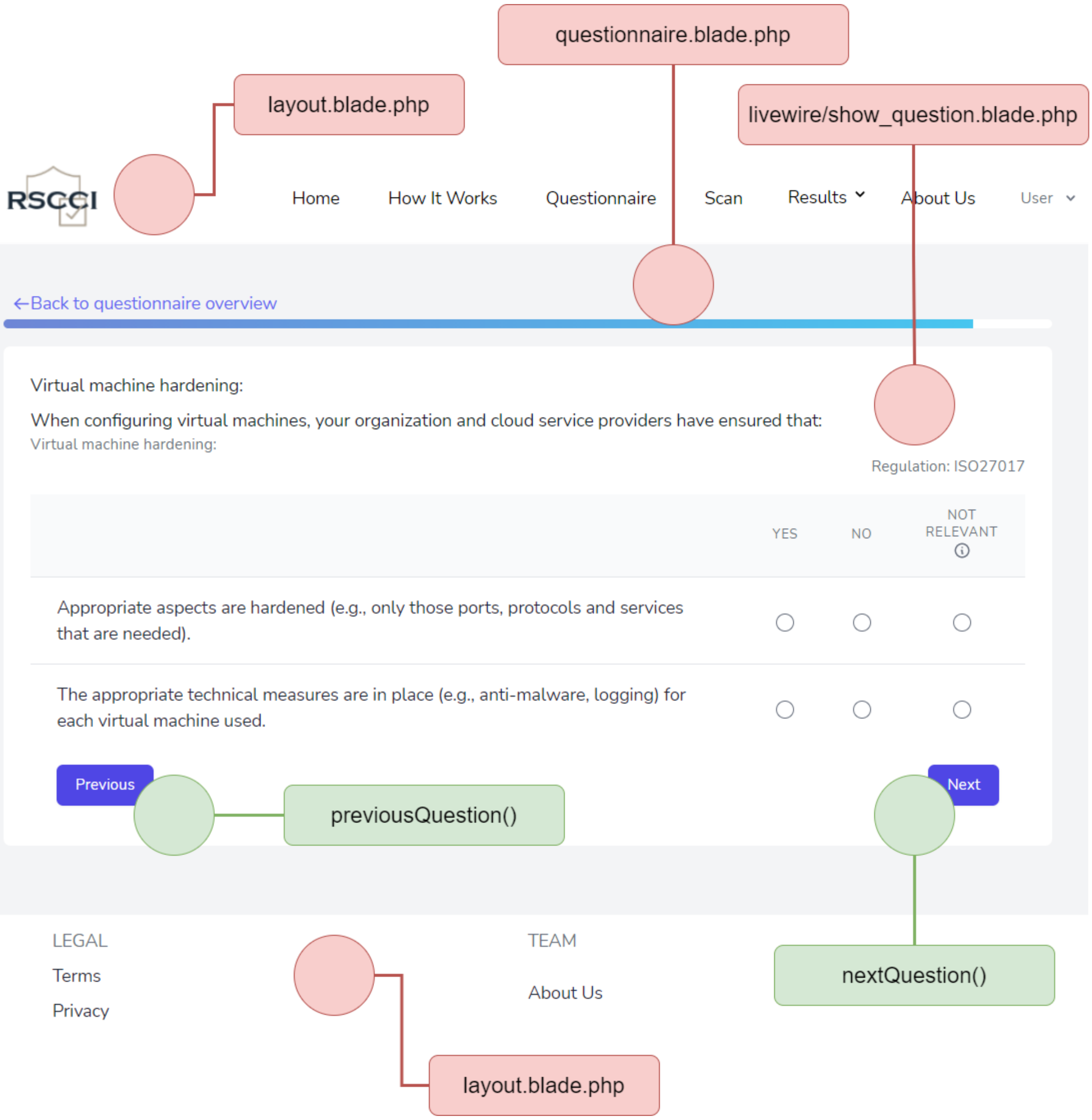
A question is represented by the variable *$question*. This variable is declared in the controller shown in Code listing 6.4 Line 1, and defined on Line 13 before being sent to the view on Line 20. The function on Line 2 is the function that gets triggered when a user clicks the *next question* button, and simply sets *$this→question* to the next question. Everything else is handled by Livewire, and the front end is updated automatically without ever reloading anything other than the elements that are changed.

**Code listing 6.3.** Livewire view: show-question.blade.php

```
<div>
    @if ($question->question_intro)
    <legend>{{ $question->question_intro }}</legend>
    @endif
    <legend>{{ $question->question_text }}</legend>
    <p>{{ $question->description }}</p>
</div>
<div>
    <p>Regulation: {{ $category }}</p>
</div>
```

**Code listing 6.4.** Livewire controller: ShowQuestion.php

```
 1      public $question;
 2      public function nextQuestion()
 3      {
 4          $this->survey->current_question++;
 5          $this->survey->save();
 6          $this->question = $this->questionnaire->questions->where(
 7              'question_number',
 8              $this->survey->current_question
 9          )->first();
10      }
11      public function mount()
12      {
13          $this->question = $this->questionnaire->questions->where(
14              'question_number',
15              $this->survey->current_question
16          )->first();
17      }
18      public function render()
19      {
20          return view('livewire.show-question');
21      }
```



Virtual machine hardening:

When configuring virtual machines, your organization and cloud service providers have ensured that:

Virtual machine hardening:

Regulation: ISO27017

| | YES | NO | NOT RELEVANT |
|---|---|---|---|
| Appropriate aspects are hardened (e.g., only those ports, protocols and services that are needed). | ○ | ○ | ○ |
| The appropriate technical measures are in place (e.g., anti-malware, logging) for each virtual machine used. | ○ | ○ | ○ |

**Figure 6.3.** Components used to build Question page

### Scan results - network map

Figure 6.4 shows the map part of the scan result page. It uses a JS library called vis-network to visualize the connection between a user's endpoints, enumerated subdomains, and what IP addresses these domains resolve to. The green circle is the provided domain, the blue circles are the enumerated subdomains, and the yellow circle is a private IP address. If an IP address is not private, it will be accompanied by the flag of the country the IP address belongs to instead of a yellow circle. From the green circle there are green lines, which goes to enumerated subdomains. From the blue circles there are blue lines, which goes to the domain or IP that an enumerated subdomain resolves to.

CIS is shown as "RSCCI Scanner", and will show which IP addresses was provided by the user who initialized the scan, and whether or not any potential vulnerabilities were found. There are two different colors for connections from "RSCCI Scanner". One is green, which means that CIS did not find any potential vulnerabilities for a endpoint. The other is red, and means that the endpoint has at least one potential vulnerability.

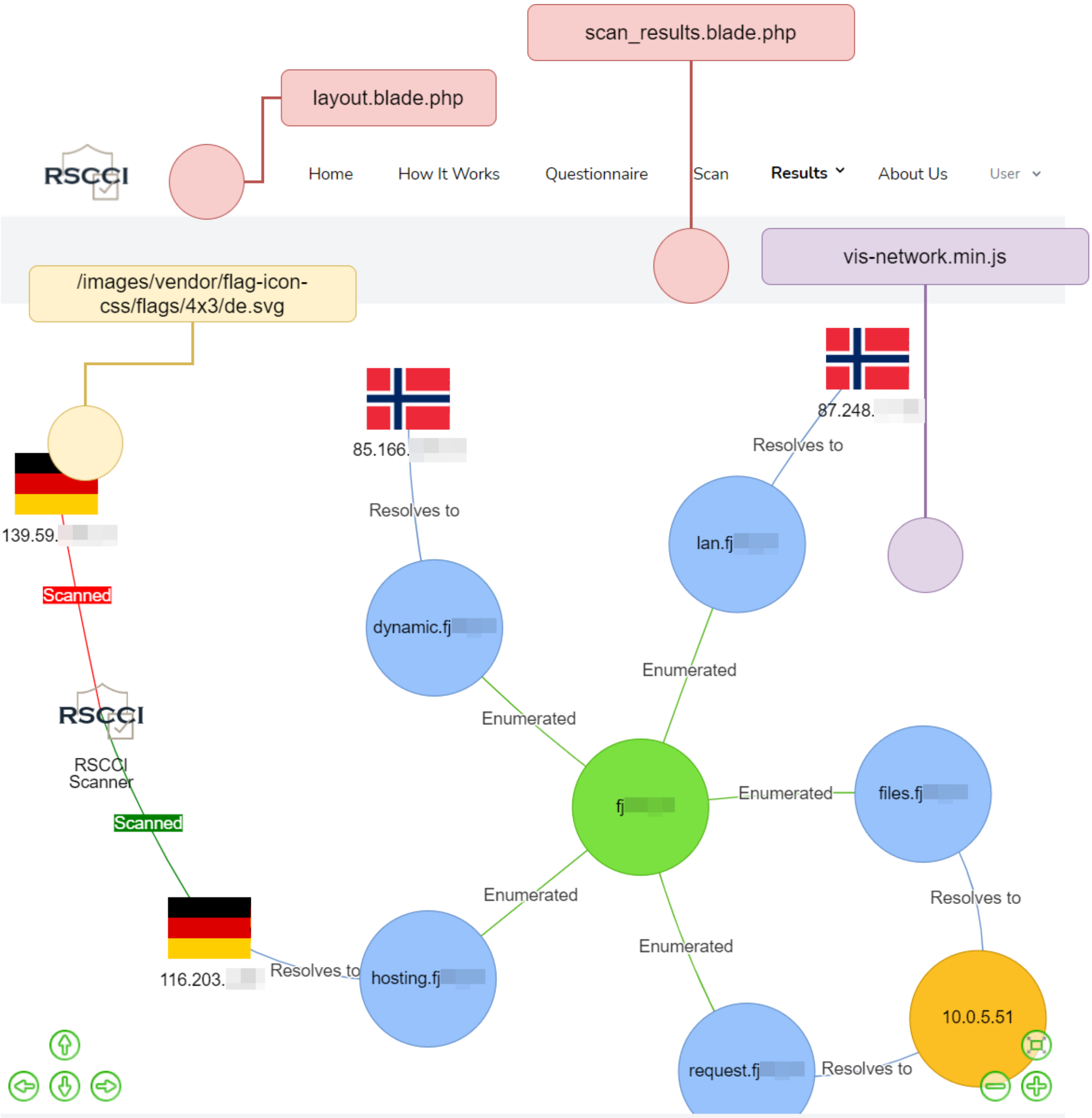
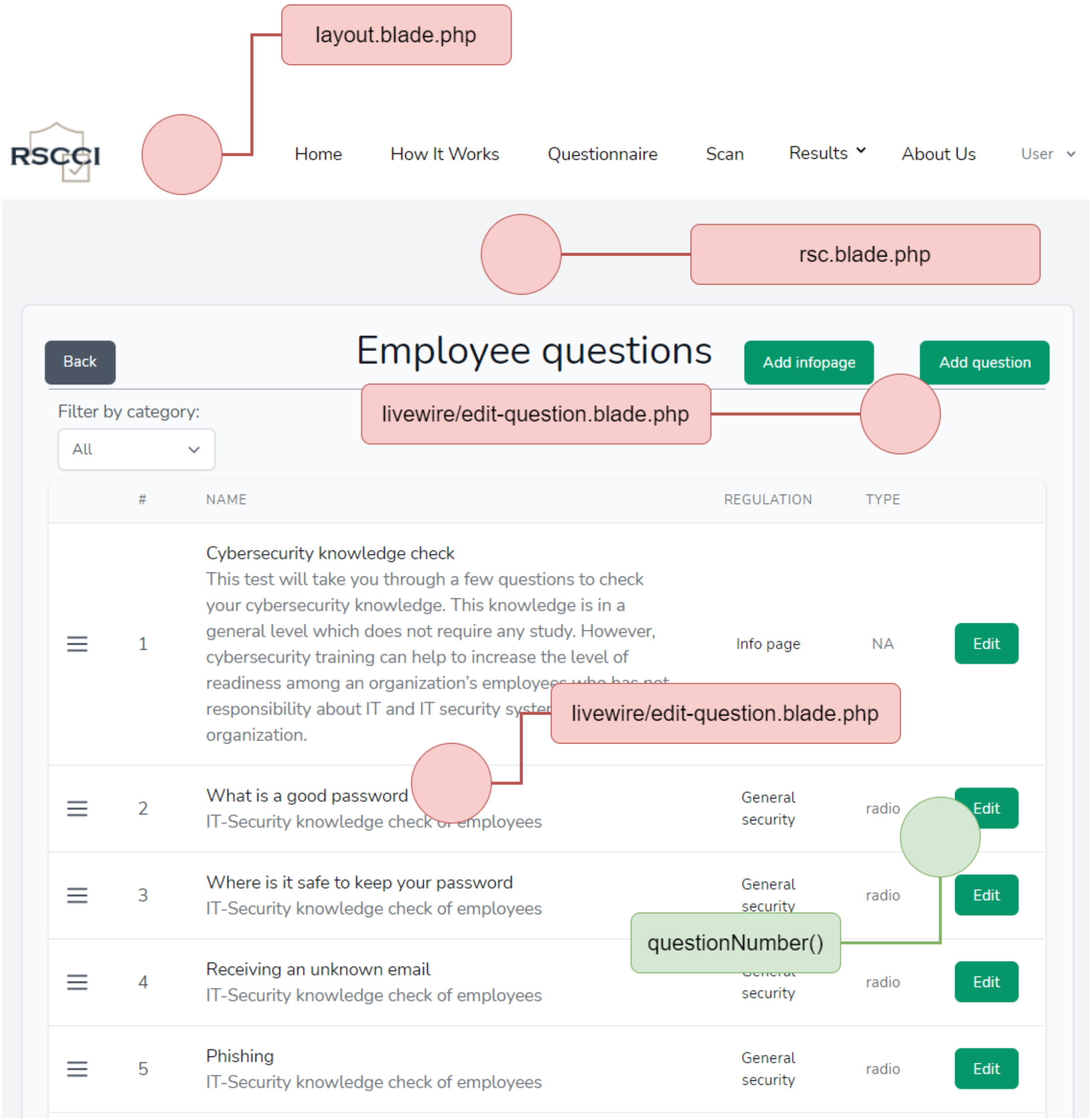


**Figure 6.4.** Components used to build Scan results - network map

### Edit questions page

Figure 6.5 shows the admin page for adding, editing, and deleting questions from a questionnaire, as well as changing the question order. This page uses a lot of JS in addition to two different Livewire components: *add-question.blade.php* and *edit-question.blade.php*. *add-question.blade.php* is a small component, and is used to add more answer input fields. *edit-question.blade.php* is a larger component, and contains the entire question list. It consists of functions to reorder the question list when a filter is selected, saving the order when the list has been reordered, and editing, and deleting questions.



Employee questions

| # | NAME | REGULATION | TYPE | |
|---|---|---|---|---|
| 1 | Cybersecurity knowledge check<br>This test will take you through a few questions to check your cybersecurity knowledge. This knowledge is in a general level which does not require any study. However, cybersecurity training can help to increase the level of readiness among an organization's employees who has not responsibility about IT and IT security syster organization. | Info page | NA | Edit |
| 2 | What is a good password<br>IT-Security knowledge check of employees | General security | radio | Edit |
| 3 | Where is it safe to keep your password<br>IT-Security knowledge check of employees | General security | radio | Edit |
| 4 | Receiving an unknown email<br>IT-Security knowledge check of employees | General security | radio | Edit |
| 5 | Phishing<br>IT-Security knowledge check of employees | General security | radio | Edit |

**Figure 6.5.** Components used to build Edit questions page

## 6.2 Back end

This section covers the back end implementation of the web portal and RSC, which is the underlying logic of our web application. We will talk about our database, the unit and feature tests we implemented, talk about implemented security measures, and provide code examples to show how the back end controller code is structured.

### 6.2.1 Database

Figure 6.6 shows the Extended Entity-Relationship (EER) diagram representation of our final database design. An EER diagram is an extension of the original Entity–Relationship (ER) model. The diagram uses Crow's foot notation to represent the relationships between the tables [54]. The light bulbs shows the columns that represents the primary keys, the red diamonds shows the columns that represents the foreign keys, and the blue diamonds shows normal data columns.

We used Laravel Jetstream for authentication and being able to divide users into teams. This is further described in Section 6.2.3. Laravel Jetstream comes with predefined tables, so not all tables in our database are created by us. However, we have altered these tables to work with our desired database design.

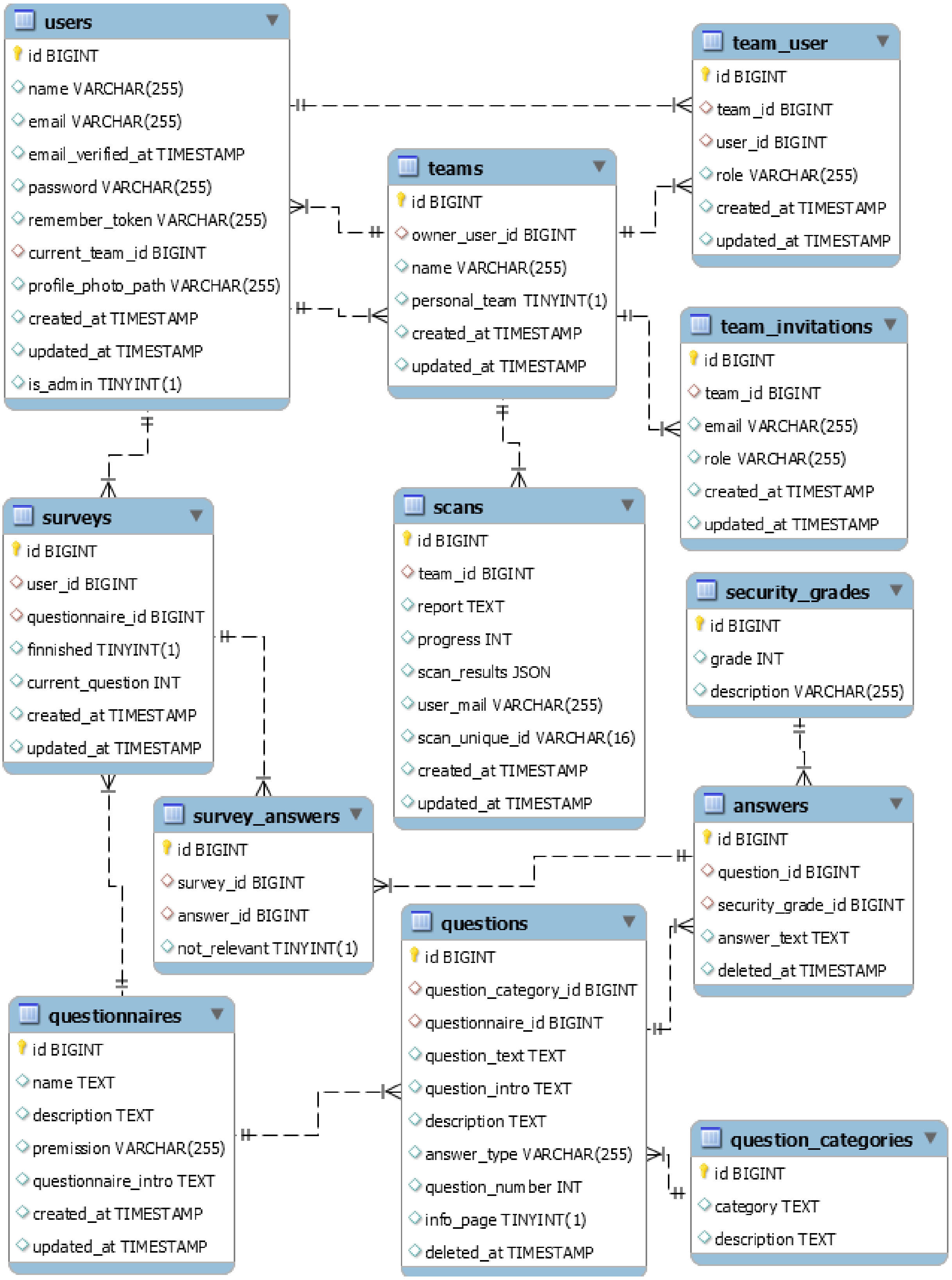


**Figure 6.6.** Extended Entity-Relationship diagram representation of our final database design

We interact with the database using Laravel Eloquent. This gives each table a model, which we use to interact with the tables. The benefit of using these models is that we can retrieve, insert, update and delete records without writing SQL code, but rather treating each model as an object with different properties. The database is a MySQL engine, as this is one of the four databases Laravel have first-party support for [55].

The tables are created using *migrations*. Migrations works like a version control for the database, and removes the need of having to make sure all developers have an updated database schema at all times. Code listing 6.5 shows how a database migration is implemented in the code. The up function creates the table *scans*, while the down function drops it.

**Code listing 6.5.** 2021_01_26_134615_create_scans_table.php

```
class CreateScansTable extends Migration
{
    public function up()
    {
        Schema::create('scans', function (Blueprint $table) {
            $table->id();
            $table->foreignId('team_id');
            $table->text('report');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('scans');
    }
}
```

If an existing table needed to be updated, a new migration file was created. The code in this new migration file would update table fields or add new columns to the existing table. This prevented the need to build up the database from scratch for each change, which could lead to data loss. Code listing 6.6 shows how we added the column *progress* to the existing *scans* table. The original migration is shown in Code listing 6.5. As the file names show, we added the progress column almost a month after the scan table was created.

**Code listing 6.6.** 2021_02_20_105918_add_progress_to_scans_table.php

```
class AddProgressToScansTable extends Migration
{
    public function up()
    {
        Schema::table('scans', function (Blueprint $table) {
            $table->integer('progress')->default(0);
        });
    }

    public function down()
    {
        Schema::table('scans', function (Blueprint $table) {
            $table->dropColumn('progress');
        });
    }
}
```

Like the code shown in Code listing 6.5, the up function is for creating and the down function is for dropping migrations. In theory, we would not need the down function for the column *progress*, as it will be dropped with the down function for the entire table. However, since migrations work as a version control, having a down function for the expansion enables us to roll back if we wanted, without having to drop the entire table and create it again.

### 6.2.2 Unit and feature testing

We wrote unit and feature tests for our web application, and our Bitbucket pipeline, described in Chapter 5, would automatically run these tests. This makes sure that new features and code updates does not break any existing functionality. Code listing 6.7 shows one of our feature tests. The feature test checks that a user can answer a questionnaire and that the answer is stored in the database.

**Code listing 6.7.** Feature test example from AnswerSurveyTest.php

```php
public function test_answer_questions()
{
    $this->actingAs($user = User::factory()->withPersonalTeam()->create());
    $questionnaire = Questionnaire::factory()->create();
    $question = Question::factory()->for($questionnaire)->hasAnswers(2)->create();
    Survey::factory()->for($user)->for($questionnaire)->create();
    $user->surveys->last()->answers()->attach($question->first()->answers->get(1));
    $this->assertCount(1, $user->surveys->last()->answers);
    $this->assertEquals(
        $user->surveys->last()->answers->first()->answer_text,
        $question->first()->answers->get(1)->answer_text
    );
}
```

### 6.2.3 Security

This section will describe the following selection of the total security features we have implemented or used in our application: input validation and sanitation, output sanitation, e-mail verification, authentication, authorization, accountability, and secure communication.

#### Input validation and sanitation

To validate the user input in our application we use some of Laravel and PHP’s built-in features, which provides most of the validation and sanitation. Laravel provides CSRF Token out of the box, which protect against CSRF attacks. We use Laravel query builder, which uses PHP’s PHP Data Objects (PDO) parameter binding, which protect against SQL injection [56]. This means that there is no need to clean or sanitize strings passed to the query builder. Because we do not clean the input, the data in our database can contain HTML and script tags. This is further described in Section 6.2.3 - Output sanitation.

Code listing 6.8 shows how we validate a user input from the scan page, where the user enters their domain, and infrastructure IP addresses. This validation first checks that the fields *domain* and *ipaddress* contain data in the form of an array, and the validations on Line 3 and 5 checks that the data contain valid domains and IP addresses. The IP address check on Line 5 is a built-in validator, while the *ValidDomain* validator on Line 3 is a validator we created ourselves to validate domains.

The code for the domain validator is shown in Code listing 6.9. To validate domains we use a regular expression that checks whether or not it is in the form of a valid domain. The domain "sub.domain.tld" would, for example, pass this validation. However, the validator could still be improved, as it does not check whether or not the given domain is a valid Top Level Domain. To do this, we could download the official Top Level Domain list provided by Internet Assigned Numbers Authority (IANA), and have the validator check if the domain is listed there.

**Code listing 6.8.** Validating values sent to the scanner

```php
1 $scanData = $request->validate([
2     ’domain’ => [’required’, ’array’],
3     ’domain.*’ => [new ValidDomain] ,
4     ’ipaddress’ => [’required’, ’array’],
5     ’ipaddress.*’ => [’ipv4’]
6 ]);
```

**Code listing 6.9.** ValidDomain validator

```php
if (!preg_match("/^\b((?=[a-z0-9-]{1,63}\.)(xn--)?[a-z0-9]+(-[a-z0-9]+)*\.)+[a-z]{2,63}\b$/", $value))
{
    return false;
}
return True;
```

### Output sanitation

When outputting data from the database, we use Blade " " echo statements, which are sanitized using PHP's *htmlspecialchars* [57]. This prevents HTML tags to be executed by the front end. Executing HTML tags would make the web application vulnerable for Cross Site Scripting (XSS) attacks. Code listing 6.10, Line 3, shows how the variables *$area→score* and *$area→possibleScore* are escaped.

**Code listing 6.10.** Example of escaping in questionnaire-results.blade.php

```
1 <p>
2     You checked yes for
3     {{ $area->score }} out of {{ $area->possibleScore }}
4     relevant controls in this area.
5 </p>
```

For some of the longer texts stored in the database, we wanted to keep the newlines (\n). However, HTML does not recognize \n. The workaround for this problem was to replace all the \n in the text from the database with the HTML <br> tag. However, we cannot escape the output if we want the HTML page to execute this tag, but not escaping the output is a possible XSS vulnerability. Our solution to this problem was to first strip the text of all HTML tags, insert the wanted <br> tags, and output the text without escaping it. Code listing 6.11 shows how we altered the text in the controller, while Code listing 6.12 shows how we output of the variables without escaping them.

**Code listing 6.11.** Example of stripping HTML tags and adding <br> in QuestionController.php

```
$question->description = strip_tags($question->description);
$question->question_intro = strip_tags($question->question_intro);
$question->question_text = strip_tags($question->question_text);

$question->description = nl2br($question->description, false);
$question->question_intro = nl2br($question->question_intro, false);
$question->question_text = nl2br($question->question_text, false);
```

**Code listing 6.12.** Example of non-escaping in show-question.blade.php

```
<div>
    {!! $question->question_intro !!}
    {!! $question->question_text !!}
    {!! $question->description !!}
</div>
```

### E-mail verification

Every user that registers to RSCCI has to verify their e-mail address, except for users invited by their Team admin. In addition to the e-mail verification, we have created a blacklist of domains that are not supported by the web application to minimize the probability of hackers abusing RSCCI to scan companies infrastructures.

### Authentication

By using Laravel Jetstream, we get a scaffolding for login, two-factor authentication, registration, password reset, and email verification. Jetstream scaffolds the front end views and uses Laravel Fortify as the back end of the user system. Combining this with Laravels existing features, we have the following built-in security features:

- Complete authentication system with login, register and forgot password pages.
- Email-verification, and two-factor authentication using Time-Based One-Time Password.

- Session guard which maintains state using session storage and cookies.
- Secure password storage by storing hashed passwords with salt using bcrypt.
- Login rate limiting after several attempts, unique to email and IP address.
- Built-in authentication facades to check whether a user trying to perform an action is authenticated or not.

Most of the views, like the result pages, are restricted to authenticated users. Therefore, almost all controllers verifies that the user is authenticated before returning the desired view. If not, the controller redirects the user to the view they originally came from. Code listing 6.13 shows example code for how the controllers authenticate a user with the built-in Laravel authentication facades.

**Code listing 6.13.** Example code for user authentication by the controllers

```
public function get_user_name() {
    if (Auth::check()) {
        $user = Auth::user();
        return $user->name;
    }else {
        return redirect()->back()->with(’status-alert’,  [’tittel’ => ’Error’,
        ’tekst’ => ’You␣need␣to␣log␣in’, ’type’ => ’red’]);
    }
}
```

### Authorization

Laravel Jetstream has an optional Team functionality, which we have enabled, and use to organize the different teams. It has built-in support for different roles, and individual permissions for those roles. Code listing 6.14 shows how we define a role in Jetstream with permissions, and Code listing 6.15 shows how we can check whether or not a user has a specific permission.

**Code listing 6.14.** Definition of a role in Jetstream

```
Jetstream::role(’hr’, ’Human␣resources␣(HR)’, [’survey:employee’, ’survey:hr’]);
```

**Code listing 6.15.** Permission check in Jetstream

```
if(!$user->hasTeamPermission($user->currentTeam, ’survey:hr’)){
    return redirect()->back()->with(’status-alert’,  [’tittel’ => ’Error’,
    ’tekst’ => ’You␣are␣not␣authorized␣to␣perform␣this␣action.’, ’type’ => ’red’]);
}
```

Some of the views, like the scan page, are restricted to specific user permissions. Therefore, almost all controllers check that the authenticated user is authorized to see the view before returning it. If not, the controller redirects the user to the view they originally came from.

The API endpoints used for scanner communication authorize requests by validating a secret key.

### Accountability

Since we have no method to check if the domains and IP addresses provided by users actually belongs to, or are used by them, it is important that we keep records about who initialized a scan on what domains and IP addresses. To achieve this, we store the user who initialized a scan and his/her team ID in the database along with the scan information. We are then able to go back in our database and prove who initialized a scan, and what infrastructure they scanned.

#### Secure communication

To ensure the browser traffic between our server and users cannot be eavesdropped on, we use Hypertext Transfer Protocol Secure (HTTPS). In HTTPS the traffic is encrypted using Transport Layer Security (TLS). We have configured Nginx to use Cloudflare's Origin CA certificate, and Cloudflare handles everything else to achieve end-to-end encryption. This is further described in Section 6.4.2.

### 6.2.4 Code structure and examples

This section covers the code structure and provide some code examples for the back end of the web application. The back end consists of the models and controllers, and is responsible for gathering, generating, and presenting data to users, as well as presenting the correct views. When we provide the code examples, we have chosen to remove the authentication check at the start of each controller function in order to save enhance clarity.

#### Presenting questions and saving user answers

The logic for presenting questions to a user and saving his/her answers is handled by two controllers: *QuestionController.php* and the Livewire controller *Livewire/ShowQuestion.php*. The function *show($id)*, shown in Code listing 6.16, is located in the controller QuestionController.php. This is the function that is triggered when a user wants to continue answering a questionnaire that he/she has started on earlier, but not finished.

The QuestionController.php's main task is checking whether or not the user is authorized to continue on the selected questionnaire. The function first checks if the user is part of a team, and then does multiple checks to determine whether or not he/she has started the selected questionnaire, but not completed it. If the code gets past each check, it returns *questionnaire.blade.php* along with the questionnaire object and a "answers" object that is used to save the user's answers for the selected questionnaire.

The questionnaire.blade.php view loads a Livewire component consisting of the Livewire/ ShowQuestion.php controller and the front end described in Section 6.1.1. The Livewire controller uses the questionnaire and answer objects from the QuestionController.php controller to find the next question for a user. The Livewire controller is also responsible for storing the user's answers whenever they go to the next question, and will return the user's next question after storing the answer. It also handles the functionality needed to go to previous questions by keeping track of how many questions back a user is and showing that question.

Originally we did not use Livewire to show questions and store user answers, so this functionality was previously handled by QuestionController.php. This meant that every time a user submitted their answers, the page would reload. The reload time was pretty fast, but it still used between 0,5-1 seconds to load as it had to load the entire page. Switching to a Livewire component instead means that the page is dynamically updated, so when a user goes to the next question, the Livewire view sends the answers to the Livewire controller.

The Livewire controller stores the answers and returns the next question to the view. The Livewire view will then take this next question and change out the question and answers on the page. Doing it this way cuts down the time it takes to show the next question to between 0,1 and 0,3 seconds. Although this is not a huge difference, the switch provides a better user experience, according a study done by the Nielsen Norman Group: "Even a few seconds' delay is enough to create an unpleasant user experience. Users are no longer in control, and they're consciously annoyed by having to wait for the computer" [28]. The switch to Livewire will therefore also help us meet the requirements described in Chapter 3.

**Code listing 6.16.** show($id) function in QuestionController.php

```
public function show($id) {
    $user = Auth::user();

    if ($user->allTeams()->isEmpty()) {
        return redirect(’company/show’)->with(’status-alert’,  [’tittel’ => ’Error’,
            ’tekst’ => ’You␣need␣to␣be␣part␣of␣a␣company␣to␣do␣a␣survey’, ’type’ => ’blue’]);
    }
    if ($user->surveys->isEmpty()) {
        $neverHadSurvey = $user->surveys->isEmpty();
        return redirect(’questionnaire/survey’);
    }
    $survey = $user->surveys->where(’questionnaire_id’ , $id)->last();
    if (!$survey || $survey->finnished) {
        return redirect(’questionnaire/survey’);
    }
    $questionnaire = $survey->questionnaire;
    if(!$user->hasTeamPermission($user->currentTeam, $questionnaire->permission)){
        return redirect()->back()->with(’status-alert’,  [’tittel’ => ’Error’,
            ’tekst’ => ’You␣are␣not␣authorized␣to␣perform␣this␣action.’, ’type’ => ’red’]);
    }

    $question = $questionnaire->questions->where(’question_number’,
    $survey->current_question)->first();

    if(!$question){
        return redirect(’questionnaire/done’);
    }
    [...]
    return view(’questionnaire’, compact(’questionnaire’, ’survey’));
    }
}
```

### Generating and presenting questionnaire results

Code listing 6.17 shows the code used for presenting questionnaire results to a user. All of a user’s questionnaires will be iterated through. If the user has not completed a certain questionnaire, an empty string will be added as the result for that questionnaire. If the user has completed a questionnaire, the results of this questionnaire will be generated.

**Code listing 6.17.** Code to display questionnaire results

```
public function showQuestionnaire(Request $request) {
    $allResults = collect();
    $categoryResults = ’’;
    $user = Auth::user();
    [...]
    foreach($user->questionnaires() as $questionnaire){
        $survey = $user->surveys->where(’questionnaire_id’, $questionnaire->id)->last();
        if($survey != null){
            if ($survey->finnished == 1) {
                $categoryResults = $this->generate($survey);
            } else {
                $categoryResults = ’’;
            }
        } else {
            $categoryResults = ’’;
        }
        $allResults->put($questionnaire->name, $categoryResults);
    }
    return view(’results/questionnaire-results’, compact(’allResults’));
}
```

The function that generates questionnaire results is *generate()*[1] function, which iterates over a user's answered questionnaire to find averages and results. The user's total possible score is calculated by counting all the relevant controls. The user's score is then calculated by checking how many of these relevant controls the user checked "Yes" for. The user's possible and total score, in addition to the relevant controls where the user checked "No" for will be added to a collection and returned to the view.

### Generating and presenting summary results

To generate the summary results, the code firsts verifies if the authenticated user has the correct permission as shown in Code listing 6.18. If so, it collects all the finished questionnaires for all the users in the authenticated user's team in an array called $allSurveys, and sends this array to the function *generate_summary()*.

**Code listing 6.18.** show_summary() function in ReportController.php

```
public function show_summary(){
    $user = Auth::user();
    if($user->hasTeamPermission($user->currentTeam, ’result:summary’)){
        $allUsers = \App\Models\User::where(’current_team_id’, $user->current_team_id)->get();
        $allSurveys = array();
        foreach($allUsers as $usr){
            if (!$usr->surveys->isEmpty()) {
                foreach($usr->surveys as $survey){
                    if($survey->finnished == 1){ array_push($allSurveys, $survey); }
                }
            }
        }
    }
    $results = $this->generate_summary($allSurveys);
    return view(’results/summary-results’, compact(’results’));
}
```

The *generate_summary()* function is about 100 lines, so the code in Code listing 6.19 only shows what is returned from this function. The code not shown in the code listing generates the questionnaire results. This is done by using the *generate()* function, for all of the questionnaires in the $allSurveys array. When this is finished, the results will be combined into a single result for each questionnaire before it is returned.

**Code listing 6.19.** generate_summary() function in ReportController.php

```
public function generate_summary(array $allSurveys){
    $summaryResults = array(
        ’IT’ => collect([]), ’Employee’ => collect([]),
        ’Management’ => collect([]), ’HR’ => collect([]),
        ’Cloud␣ISO/IEC␣27017’ => collect([]), ’Cloud␣NIST␣SP␣800-209’ => collect([]),
    );
    [...]
    foreach($allResults as $surveyName => $results){
        [...]
        $summaryResults[$surveyName]->put(’numberOfSurveys’, $numberOfSurveys);
        $summaryResults[$surveyName]->put(’averageScore’, $averageScore);
        $summaryResults[$surveyName]->put(’differentAnswers’, $answersSorted);
        $summaryResults[$surveyName]->put(’totalCurrentScore’, $totalCurrentScore);
        $summaryResults[$surveyName]->put(’totalTotalPoints’, $totalTotalPoints);
        $summaryResults[$surveyName]->put(’allAnswersInSurveyUnique’, $allAnswersInSurveyUniqueSorted);
        $summaryResults[$surveyName]->put(’totalNotRelevant’, $totalNotRelevant);
        $summaryResults[$surveyName]->put(’id’, $results[0]->id);
    }
    return $summaryResults;
}
```

[1] The generate function is not shown in any code listing, but can be found in the file *app/Http/Controllers/ReportController.php*, in the repository Bachelor-cloud-2021

### CIS communication - API

As mentioned in Chapter 4, CIS is the scanner, which is placed on its own server. We therefore need an API to communicate between CIS and our web portal. We have created two API endpoints in the web application to receive data from CIS: one for getting updates on scan progress and one for getting scan results. Code listing 6.20 shows the code for the two endpoints, where the function *update* on Line 1 is the endpoint for updating scan progress. The function *upload* on Line 15 is the endpoint for uploading scan results.

As mentioned in Section 6.2.3, the API endpoints authenticate the request by validating a secret key. The key is stored in the environment file *.env*. The code accesses this key by using the *env()* function, as shown on Line 4 in Code listing 6.20. If the secret is not the same as the key in the request, the functions return *401 Unauthorized*.

**Code listing 6.20.** The API endpoints in ScanResultsController.php

```
1  public function update(Request $request)
2  {
3      $key = $request->key;
4      $auth = env(’API_SECRET’);
5
6      if(strcmp($key, $auth)!== 0){
7          return ’401␣Unauthorized’;
8      }
9      $scan = Scan::where(’scan_unique_id’, $request->scanid)->firstOrFail();
10     $scan->progress = $request->progress;
11     $scan->save();
12     return ’Ok’;
13 }
14
15 public function upload(Request $request){
16     $key = $request->key;
17     $auth = env(’API_SECRET’);
18
19     if(strcmp($key, $auth)!== 0){
20         return ’401␣Unauthorized’;
21     }
22
23     $scan = Scan::where(’scan_unique_id’, $request->scanid)->firstOrFail();
24     $report = $request->report;
25     $userMail = $scan->user_mail;
26     $scan->scan_results = $report;
27
28     $scan->save();
29     $this->sendMail($userMail);
30     return ’Uploaded,␣mail␣sent␣to␣’.$userMail;
31 }
```

### Generating and presenting scan results

The scan results page shows a user his/her results of a scan performed against his/her company's cloud infrastructure. The user can see the results even though he/she was not the one who initialized the scan. Code listing 6.21 shows the code for displaying a scan to the user. The parse_scan function is the core function for presenting the scan results. This function gets the JSON scan data from the database, and iterates over the information, extracting and summarizing data.

The first data to be parsed is the vulnerability data, which is gathered from NMAP as described in Section 6.3.5. The data is iterated over, and if there are found any vulnerabilities, they are stored in a collection with the corresponding IP address. If any vulnerabilities have a CVSS score greater than 7.0, a warning will show in the scan results, since this is either a high or critical severity vulnerability[2]. The reason we show this warning is because it is important that whoever sees the results understands that it is a potentially severe vulnerability, and they might want to take action fairly quickly.

[2] Table 2.1 in Chapter 2 shows an overview of the different CVSS scores and their severity.

**Code listing 6.21.** Code to present scan results

```php
public function showScan(Request $request) {
    $user = Auth::user();
    $scan = ’’;
    $scanProgress = -1;
    $scanId = 0;

    if($user->hasTeamPermission($user->currentTeam, ’scan:run’)){
        $allTeamScans = \App\Models\Scan::where(’team_id’, $user->current_team_id)->get();
        $scanId = 0;
        if ($allTeamScans->isNotEmpty()) {
            $lastTeamScan = $allTeamScans->last();
            $scanProgress = $lastTeamScan->progress;
            $scanId = $lastTeamScan->id;
            if ($scanProgress > 99) {
                $scan = $this->parse_scan($lastTeamScan->scan_results);
            } else {
                $scan = ’In␣progress’;
            }
        }
    }

    return view(’results/scan-results’, compact(’scanProgress’, ’scanId’, ’scan’));
}
```

### Add/Edit questions to questionnaire

Code listing 6.22 shows how a question and its corresponding answers are added to the database. To make sure we get everything, we have chosen to use a transaction when creating the records. By using a transaction, we tell the database to treat all the create statements as "a whole", meaning that if the database fails at making one record or loses the connection before all the records are created, the already created records in the transaction will be rolled back [58].

**Code listing 6.22.** The function storeQuestion() in AdminController.php

```php
public function storeQuestion(Request $request){
    $questionnaire = Questionnaire::where(’id’, $request->role)->first();
    $question_number = $questionnaire->questions->count()+1;
    DB::transaction(function () use ($request, $question_number){
        $question = Question::create([
            ’question_text’ => $request->question,
            ’questionnaire_id’ => $request->role,
            ’question_category_id’ => $request->category,
            ’description’ => $request->description,
            ’question_intro’ => $request->question_intro,
            ’answer_type’ => $request->answer_type,
            ’question_number’ => $question_number,
        ]);

        foreach($request->answer as $option){
            if ($request->answer_type == ’yes-no’) {
                $grade = 1;
            } else {
                $grade = $option["grade"];
            }
            $question->answers()->create([
                ’security_grade_id’ => $grade,
                ’answer_text’ => $option["answer"],
            ]);
        }
    });
[...]
}
```

Code listing 6.23 shows the function that is called whenever an administrator updates a question. The code starts with defining the new question values based on the *$request* parameter, which holds the administrator's changes. It then checks if there are any changes to the answer values. This is done by comparing the existing answer ids in the database with the ids of the answers in $request. If they are the same, the code updates the existing answers. If not, the code will delete the answers from the database that are not in $request and update the ones that are. Finally, the code creates any new answers before saving the question.

**Code listing 6.23.** The function updateQuestion() in AdminController.php

```php
public function updateQuestion(Request $request){
    $question = Question::find($request->question_id);
    //Set new values for $question based on $request
    [...]
    $idExistingAnswers = $question->answers->pluck('id')->toArray();
    $idNewAnswers = array_column($request->answer, 'id');

    if($idNewAnswers == $idExistingAnswers){
        foreach($question->answers as $answer){
            //If there are no new answers, set new values based on $request->answer[$answer->id]
            [...]
            $answer->save();
        }
    } else {
        $removedAnswers = array_diff($idExistingAnswers, $idNewAnswers);
        $updatedAnswers = array_diff($idExistingAnswers, $removedAnswers);
        foreach ($removedAnswers as $removeAnswerId) {
            $answer = Answer::find($removeAnswerId);
            $answer->delete();
        }

        foreach($updatedAnswers as $answerId){
            $answer = Answer::find($answerId);
            //Set new values based on $answerId
            $answer->save();
        }
    }
    if ($request->new_answer) {
        foreach($request->new_answer as $option){
           //If new answers, create
           [...]
        }
    }
    $question->save();
[...]
}
```

### Import and export questions

To transfer the questionnaire contents between machines easily, we created functionality to import and export questions. The export function is shown in Code listing 6.24, Line 2. This function simply gets all the questions and answers, and exports them as JSON. We can then download that JSON file to import the questions later.

To import the questions we have a file upload section on the administrator page. When a file containing questions is uploaded, it is validated as shown in Code listing 6.24, Line 5, where the code validates that the file has the correct JSON format. Line 5 shows how the JSON file is decoded into a PHP variable. The code then goes through all the questions in the file, creates the questions and their corresponding answers, and stores everything in the database.

**Code listing 6.24.** Import and export questions

```php
// Export
$json = Question::withTrashed()->get()->load('answers')->toJson(JSON_PRETTY_PRINT);

// Import
$data = $request->validate(['questions' => 'required|file|mimetypes:application/json']);
$questions = json_decode(file_get_contents($data['questions']), true);
foreach ($questions as $question) {
    $currentQuestion = Question::create($question);
    foreach($question['answers'] as $answer){
        $currentQuestion->answers()->create($answer);
    }
}
```

### Generating and exporting PDF of results

There are multiple software and tools available to export data as a PDF. PHP has libraries to generate PDFs from text, but that require us to manually define tables and style the page in a PDF format. However, there are some PHP libraries like BrowserShot that creates PDFs from styled HTML. It works by using a headless browser called puppeteer to render the HTML and save a PDF of the page.

To generate PDFs of questionnaire, scan and summary result pages we used a wrapper for Browsershot called laravel-browsershot. This enabled us to generate PDFs using Laravel views, as shown in Code listing 6.25, which is the code for generating a PDF of questionnaire results. The code starts with checking that a user actually has completed a questionnaire, before generating results using the generate() function. Line 9 shows how we use Browsershot's *loadView()* function to generate a PDF from a view. This view is called *pdf/questionnaire*, and is a modified version of the original questionnaire results view, where we removed unrelated things, like the navbar. We also run some additional functions on the loadView() function, where the most important one is *showBackground()*, which makes sure that the images and CSS display properly.

To have the PDF look nice and be useful, the code adds a header with the document title and date of completion. The title consists of the questionnaire name and the date the questionnaire was completed. The code also adds a footer with the page number. Finally, a PDF download object is returned with the title of the document.

**Code listing 6.25.** PDF generate of questionnaire results

```php
1  public function pdfQuestionnaire($id) {
2      $survey = $user->surveys->where('questionnaire_id' , $id)->where('finnished', 1)->last();
3      if (!$survey) {
4          return redirect()->back()->with('status-alert',
5          ['tittel' => 'Error', 'tekst' => 'Survey␣not␣found', 'type' => 'red']);
6      }
7      $results = $this->generate($survey);
8      $surveyName = $survey->questionnaire->name;
9      $pdf = PDF::loadView('pdf/questionnaire', compact('results', 'surveyName'))
10         ->showBrowserHeaderAndFooter()->showBackground()
11         ->format('A4')->margins(10, 10, 10, 10)->waitUntilNetworkIdle();
12     $pdf->headerHtml(view('pdf/header'));
13     $pdf->footerHtml(view('pdf/footer'));
14     return $pdf->download(
15         'RSCCI-'.$surveyName.'-Questionnaire-'.$results->date->format('d-m-Y').'.pdf');
16 }
```

### Sending mail to users

We have three scenarios where the application sends an email to users: when a user registers, when a user adds a team member to their team, and when a scan is done. The front end of the emails are generated in HTML, so all we have to do in the back end is to provide the correct variables and return the email. See Section 6.4 for information about the mail services used to send emails.

Code listing 6.26 shows an excerpt of the code that invites a new user to a team. The code sets the data we need for building the email before sending it to the user. The code shown in Code listing 6.27 is the controller that builds the invite mail to a user. This is the function that is triggered in Code listing 6.26 Line 11. Its only job is to return the email with the needed data. Returning the email is equivalent to sending the email.

**Code listing 6.26.** Putting together mail data in InviteTeamMember.php

```
 1  $maildata = [
 2      ’newUser’ => $newUser,
 3      ’user’ => $user->name,
 4      ’email’ => $email,
 5      ’role’ => $role,
 6      ’name’ => $name,
 7      ’passwordPlaintext’ => $passwordPlaintext,
 8      ’team’ => $team,
 9      ’questionnaires’ => $questionnaires,
10  ];
11  Mail::to($email)->send(new InviteToCompanyMail($maildata));
```

**Code listing 6.27.** Building mail in InviteToCompanyMail.php

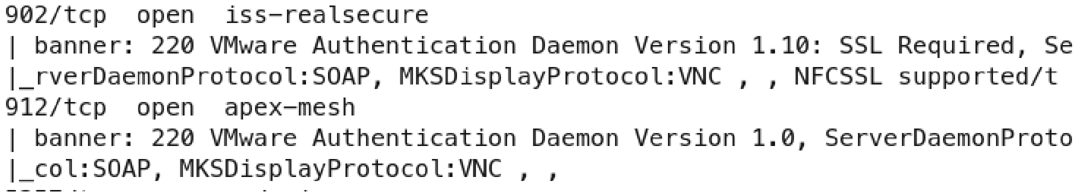
```
public function build()
{
    $address = ’noreply@rscci.cloud’;
    $subject = ’Welcome␣to␣RSCCI’;
    $name = ’The␣RSCCI␣team’;
    return $this->view(’mail.invite’)
                ->from($address, $name)
                ->subject($subject)
                ->with([
                    ’newUser’ => $this->data[’newUser’],
                    ’user’ => $this->data[’user’],
                    ’email’ => $this->data[’email’],
                    ’role’ => $this->data[’role’],
                    ’name’ => $this->data[’name’],
                    ’passwordPlaintext’ => $this->data[’passwordPlaintext’],
                    ’team’ => $this->data[’team’],
                    ’questionnaires’ => $this->data[’questionnaires’],
                ]);
}
```

## 6.3 Scanner

This section covers the scanner implementation of CIS. We will talk about how the scanner works, standards used, and what tools we used. We designed our scanner to be able to detect vulnerabilities, covering the top 10 highest threats ranked by Open Web Application Security Project (OWASP), which we introduce in Section 6.3.2.

1. **Target identification**
   The first step our scanner takes, is to do a network reconnaissance on the whole provided subnet. Our software discovers hosts on the network, and adds the IPs to the queue for the next step. If a single IP is provided, instead of a subnet, the software will then simply check if the host is reachable.
2. **Port scanning to find the running software**
   When we have established a list of IP addresses which are in our scope, we then check which ports are open on all of the machines. This may take some time, as there are 65535 (16 bits) ports which can be opened on each IP address. If we want to do a quick port scan, we can set the timeout for connection attempt to for example 0.1 seconds. This means that if the computer did not reply on that port within 0.1 second, the port scanner will move

on to check the next port. Sometimes this small amount of time is not enough. When the computer responds to a port connection request, it also sends back a small banner, which tells us what type of service is running on that port. We can see this in the Figure 6.7.



```
902/tcp  open  iss-realsecure
| banner: 220 VMware Authentication Daemon Version 1.10: SSL Required, Se
|_rverDaemonProtocol:SOAP, MKSDisplayProtocol:VNC , , NFCSSL supported/t
912/tcp  open  apex-mesh
| banner: 220 VMware Authentication Daemon Version 1.0, ServerDaemonProto
|_col:SOAP, MKSDisplayProtocol:VNC , ,
```

**Figure 6.7.** Nmap banner info

In the figure we can see a banner for the two open ports that were scanned. It contains information about the software, and its version that is running on the port. Sometimes the banner contains a large amount of information, such as long SSH keys, or raw website data. We will then run into a problem with using only a 0.1 second connection time limit. This is because the information from the banner takes longer than the specified time to retrieve the information. If we adjust the time to 1 second, the scanner will use 18 hours to complete for each machine. This is why we have chosen to use the highly popular tool Nmap, see Section 8.1.5. We describe Network Mapper (Nmap) more detailed in Section 6.3.5. Nmap allows us to scan machines much quicker. It has the ability to scan multiple hosts, and multiple ports in parallel.

3. **Look for remote access**
   The scanner now have a list of all the IP addresses, and the open ports. It can now start to analyze the findings. One easy way into controlling a machine, is to look at the different remote connection protocols. The most popular options are SSH, Telnet and Virtual Network Computing (VNC) on Linux, and RDP (Remote Desktop Protocol) on Windows. From looking at which ports are open, we can determine which of these services are running. If we see port 23 running, we know that this is Telnet, without needing to look at the banner information. These remote connections have all their own weaknesses, which an attacker most often will try to exploit.

4. **Scan for vulnerable software versions**
   Now when the scanner knows which software and versions of the software the machines contain, we can start to cross reference this information with databases of vulnerable software. The Nmap tool we use have this feature integrated. It goes on the internet, and checks if software versions have a weakness, which we earlier described as a CVE.

   With this, we are then able to notify users if they have vulnerable software running on their machines.

5. **Check configuration flaws**
   A lot of software ships with insecure default configuration options. Options such as having administrator panel accessible from everywhere, or authentication set to disabled by default, strongly increases the risk of a malicious event occurring. To check this, we use a tool called Nikto.

6. **Database design**
   If user inputs are not sanitised, a database can contain injection vulnerabilities. An attacker can send SQL queries, which is a database code, to reveal information hidden from users. The scanner runs a Nmap script called http-sql-injection, which scans web servers containing queries vulnerable to an SQL injection attack. It does this by scanning for URLs containing queries in order to find errors. These errors will be analysed, and some of them will most often be reported as vulnerable to an SQL injection attack.

7. **Cross-Site Scripting** Cross-site scripting is a type of injection attack to a website. Attacks like this can occur when a website allows user input, without validating, or encoding it. This allows an attacker to send malicious scripts using this input. These scripts can steal

information from other users on the website, display sensitive information, or even change the contents of the website for everyone. To check for injection vulnerabilities like this, we use a tool called XSStrike. The functionality of this tool is described in Section 6.3.5, but in a short explanation this tool generates custom payload for a specific target, and notifies if it finds a vulnerable input.

8. **Default credentials**
   Almost same as default configurations, credentials are often left untouched, and the manufacturers may have left a default username and password in the software. These could be "user, password", or "admin, admin". To mitigate this vulnerability, we implemented a tool in our scanner. This tool takes the findings from our Nmap scan, that is software and versions running, and tests it against a list of known credentials for the services.
9. **Filtering information**
   To provide users with only relevant security information about their systems, we filtered the output of the scanning tools using Python programming language.
10. **Send relevant information to website**
    The filtered information is then sent as JSON to the web portals API. The web portal will store this JSON information.

### 6.3.1 Subdomain enumeration

The subdomain enumeration function of our scanner is a scan which attempts to find a domain's valid subdomains. This is done by first querying multiple open source APIs that store DNS information using a program called Amass. We then used a list consisting of several thousand common subdomains in combination with a program called massdns to test other subdomains. The function goes through each item in the list, and tries to resolve the item as a subdomain. It queries several hundred open DNS servers at once to speed up the search. Because of the issues we had with false responses from DNS servers, all subdomain responses were looked up against a trusted DNS server. We talk more about this in Section 8.1.5.

### 6.3.2 Scanner standards

We followed the OWASP Top 10 standard, which represents the most critical security risks to web applications. OWASP is a non profit organization focused on web application security. Each year they rank what they see as the highest threads to software security. OWASP wants companies to use these recommendations, in order to minimize their risks online. The following is their 2020 list of top 10 highest threats [59].

1. **Injection**
   Injection can be done from an attacker by inserting their own code into a program input, or some sort of data submission. An example of this can be an input where a user is asked to type their name. However, instead of typing their name, the attacker constructs a malicious query. The malicious query is based on their knowledge of what technologies the application is running e.g. from SQL variants, to reveal information which the application is not supposed to reveal. In some cases it can also make the server execute command line code.
2. **Broken Authentication**
   Applications often let users use weak, or well known passwords (i.e "123456"), which allows an attacker to easily brute force a list of username and passwords on the application. Poorly designed systems, which does not rate limits the login attempts, make the attackers able to brute force this login much quicker. The authentication is also bad if it does not require a two factor authentication to gain access to the application.
3. **Sensitive Data Exposure**
   Sensitive Data Exposure is when badly encrypted data is captured by an attacker during transmission or storage of data. APIs which rely on insecure data transmission methods

are often the weak links that allow an attacker to gain access to this sensitive information.

4. **XML External Entities (XXE)**
   When using vulnerable web components, attackers are sometimes able to take advantage of components that use XML. If the website or platform allows uploading of XML files, an attacker can upload a malicious XML exploit file, which can execute hostile commands on the host machine.
5. **Broken Access Control**
   Broken access control is about not having strict enough rules for what a standard user of the platform should have access to. Attackers can abuse these rules, and be able to access data or functionalities which are designed and meant to be accessible to users with a higher user privileges.
6. **Security Misconfiguration**
   A common vulnerability today is actually default and misconfigurated software. For instance, when a system administrator does not change the default username and password, which is often quite easy to guess. Software or hardware products often ship with default username and credentials, which people do not bother to change. This makes it easy for an attacker to look up what the default credentials for that service are, and try them. When it comes to misconfigurated software, many administrators make it easy for them selves, and disable important security barriers, just to make it easier for themselves to access the platforms. Doing this, they drastically increases the likelihood of an attempted break-in.
7. **Cross-Site Scripting**
   XSS flaws make attackers able to execute scripts in the victim's browser. With this, they can hijack user sessions, deface websites, or redirect the user to malicious sites. This type of errors occur when a website takes user input without validating it properly.
8. **Insecure Deserialization**
   Insecure Deserialization is a vulnerability where deserialization flaws allow an attacker to remotely execute code in the system. Data from a user is serialized when it is stored on a server, which means combining multiple values into one line of information. We can take an example from a typical web shop, shown in Code listing 6.28.

   **Code listing 6.28.** Serialization

   ```
       {"productId": 14, "amount";"1", price:"29.90"}
   ```

   An attacker can exploit this by altering the serialized line before sending it to the server.

   If the server does not have a good validation, he can for example alter the price of the product from 29 dollars, to 1 dollar, and the server could accept the purchase, and only charge 1 dollar for the attacker.
9. **Using components with known vulnerabilities.**
   Using outdated or discontinued components such as libraries, frameworks, and old dependent code can contain known vulnerabilities. This will affect even new applications, which utilizes outdated code. Software updates often include security patches which prevent attackers from exploiting a platform.
10. **Insufficient Logging and monitoring**
    Failing to pick up events which have occurred towards a service, and not starting mitigation of the events soon enough. Within the time a company have discovered a security breach, it takes an average of 197 days before they discover that they have been attacked. During this time attackers have been inside their network gaining opportunity to do more harm [60].

### 6.3.3 Automating the vulnerability scanner

In order for our vulnerability scanner to operate without human interaction, we needed to automate a set of security scanning software, which all are mainly designed to be used manually. The only human interaction would be a user, which provides his/hers endpoints, and or domain addresses. The main portion of the vulnerability scanner lays in one python script. This file starts a few Linux terminal commands, and filters out the most relevant information, which is sent to the website RSCCI.

### 6.3.4 Communication with the web portal's API

Whenever a new scan starts or a scan has progressed, the scanner will send an update to the web portal's API. This update is sent using a POST request, and contains: an API key used to authenticate, the scan's unique ID, and the scan's progress. When the scan is completed, the scanner will send the results of the scan in a JSON report. This report contains filtered raw scan data and some statistics from the scan.

### 6.3.5 Scanning tools

We utilized some powerful tools to help us with scan of the target machines. We originally wrote the code in Python which we concluded was way too slow, and the help of already made tools speeded the scanning process up a lot. The following are the tools we used in this project.

#### Nmap

Nmap, a network scanner, which can gather a lot of information about a machine, or a subnet, by looking at the open ports the machines have. The Nmap tool uses banner grabbing to find out what software, and which version of the software the machines are running. We can use this information to find out if the machine is running outdated and/or exploitable software.

One powerful feature of Nmap is the NMAP Scripting Engine (NSE). NSE enables users to write scripts using the Lua programming language. These scripts allow us to use already made scripts, to save time trying to automate every vulnerability scan. For instance, we use a script called vulners, which looks up if the software on a machine has a corresponding CVE. One other script we used to detect SQL injection vulnerabilities on a target, is the NSE script "http-sql-injection".

Code listing 6.29 shows how we used Python to run Linux commands on the server. Most of the Nmap arguments are hard coded, besides from the IP address ("ipAddr"), and output logfile ("nmap_vulners"). There are a couple of lines of codes, shown in Code listing 6.29, which do not output anything to the server's terminal. The code sends the output to the specified file. This just makes things more clean.

**Code listing 6.29.** Nmap

```
subprocess.call(["nmap", "-sV", "--script=vulners", "-v", "-oX", nmap_vulners, ipAddr],
                                          stdout=subprocess.DEVNULL,
                                          stderr=subprocess.STDOUT)
```

#### Changeme

Changeme is a default credential scanner. We have configured it to use the network scanning done by Nmap, for detecting the services on the network. Then Changeme will try default credentials, which in this case are usernames and passwords, to the known services/software detected by Nmap. This works by using the preexisting Nmap scan result's file, which is an XML file. Changeme will then from the Nmap file, identify which software the machines contain, and use its built-in word lists to see if there is a match. This method is called a dictionary attack, because it tries a set of known working match for the username and password.

A dictionary attack is a type of brute force attack, which we originally did not want to use on the scanned systems. However, we think that it is important to at least check a few of the default credentials, which the software often has as standard configuration. These credential combinations are something an attacker often tries for gaining access. To run the program, we simply call it from our main.py Python file. See Code listing 6.30. The arguments provide the source from the Nmap file, and the destination for where the output for the results should be.

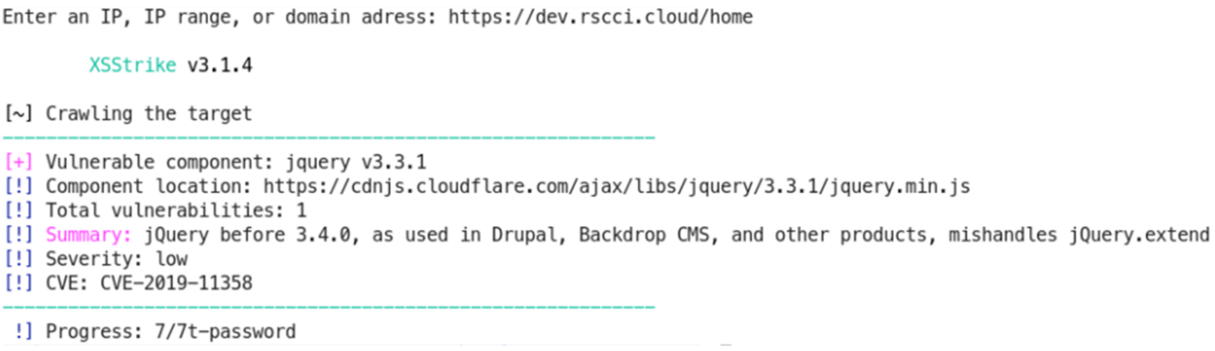
**Code listing 6.30.** Changeme

```python
def default_cred():
    subprocess.call(["python3", "changeme.py", nmap_vulners, "-oa", default_cred_file],
                    stdout=subprocess.DEVNULL,
                    stderr=subprocess.STDOUT)
```

### XSStrike

This tool detects websites vulnerable to XSS injection attack. The program is written in Python, and is a Cross Site Scripting detection suite [61]. The software generates payload for s specific website, and checks if the website is vulnerable to any form of XSS attack.

The tool will also find weaknesses in dependencies correlated to the web components. We tested this tool on our development web page earlier in March, and found a such vulnerability on our own site. See Figure 6.8.



```
Enter an IP, IP range, or domain adress: https://dev.rscci.cloud/home

        XSStrike v3.1.4

[~] Crawling the target
------------------------------------------------------------
[+] Vulnerable component: jquery v3.3.1
[!] Component location: https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js
[!] Total vulnerabilities: 1
[!] Summary: jQuery before 3.4.0, as used in Drupal, Backdrop CMS, and other products, mishandles jQuery.extend
[!] Severity: low
[!] CVE: CVE-2019-11358
------------------------------------------------------------
 !] Progress: 7/7t-password
```

**Figure 6.8.** Vulnerability on our own development website.

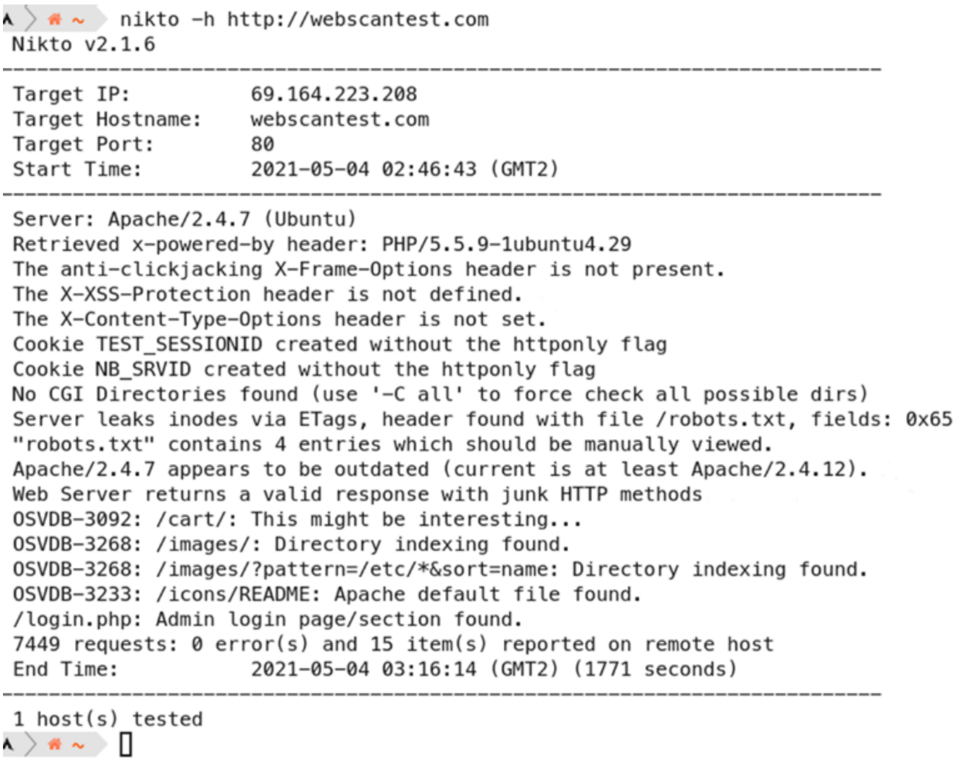
We have automated running the process of this tool as well. See Code listing 6.31. When we call on the tool, the parameters such as the IP address and file output are specified as variables, which are defined earlier in our Python script.

**Code listing 6.31.** Xsstrike

```python
xss_strike = ("cd␣"+xssDir+"␣&&␣python3␣xsstrike.py␣-u␣"+ipAddr+"
--crawl␣-l␣10␣--file-log-level␣INFO␣--log-file␣"+logfile)
```

### Nikto

Nikto is a web application reconnaissance tool, which is useful for scanning after server configurations. It can detect present default configurations on machines. These credentials should be change to mitigate vulnerabilities as mentioned before. To run the tool, we parse our Nmap scan results, which lets Nikto know which addresses it should scan. Figure 6.9 shows Nikto's findings from an open test web page.



```
nikto -h http://webscantest.com
Nikto v2.1.6
---------------------------------------------------------------------------
 Target IP:          69.164.223.208
 Target Hostname:    webscantest.com
 Target Port:        80
 Start Time:         2021-05-04 02:46:43 (GMT2)
---------------------------------------------------------------------------
 Server: Apache/2.4.7 (Ubuntu)
 Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.29
 The anti-clickjacking X-Frame-Options header is not present.
 The X-XSS-Protection header is not defined.
 The X-Content-Type-Options header is not set.
 Cookie TEST_SESSIONID created without the httponly flag
 Cookie NB_SRVID created without the httponly flag
 No CGI Directories found (use '-C all' to force check all possible dirs)
 Server leaks inodes via ETags, header found with file /robots.txt, fields: 0x65
 "robots.txt" contains 4 entries which should be manually viewed.
 Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12).
 Web Server returns a valid response with junk HTTP methods
 OSVDB-3092: /cart/: This might be interesting...
 OSVDB-3268: /images/: Directory indexing found.
 OSVDB-3268: /images/?pattern=/etc/*&sort=name: Directory indexing found.
 OSVDB-3233: /icons/README: Apache default file found.
 /login.php: Admin login page/section found.
 7449 requests: 0 error(s) and 15 item(s) reported on remote host
 End Time:           2021-05-04 03:16:14 (GMT2) (1771 seconds)
---------------------------------------------------------------------------
 1 host(s) tested
```

**Figure 6.9.** Here we can see Nikto's findings from an open test web page. It has found a default Apache login page, and given some other tips towards finding vulnerabilities.

## 6.4 Infrastructure

This section covers the technologies used for the infrastructure hosting the scanner and the web application, as well as how these are configured.

### 6.4.1 Development environment - Local hosting

The following section covers our development environment, which we used to test and develop RSCCI. We will talk about the infrastructure, security, and the e-mail service we used.

#### Infrastructure

For our development environment we use a VM on a server in one of the group member's house, to avoid unnecessary expenses, and potentially only use free cloud credits for the production environment. The development environment does not have any requirements regarding uptime or speed, so hosting at home is not an issue. The VM is running Ubuntu 20.04, and has been allocated 2 CPU cores and 2 GB RAM, which is plenty for the development environment. In the development environment we host both RSC and CIS on the same server.

#### Security

The server is not in a dedicated Demilitarized zone VLAN, so to secure the network we have added some firewall rules. The hypervisor is configured to block all traffic between the VM and the local network, except for remote management traffic initiated from the local network. Cloudflare is used for HTTPS, caching of content and anti-DDoS. The VM's firewall is configured to only allow traffic from Cloudflare IP addresses using an open source script. This script downloads a list of Cloudflares IP addresses, and adds these as allowed in the firewall so that only traffic going through Cloudflare can access the server.

### E-mail

Before having a mail service up and running, we used Mailtrap for developing and testing the mail functionality. Instead of sending mail out to the receiver address, the application sends it via Simple Mail Transfer Protocol (SMTP) to Mailtrap. Mailtrap then collects the mail in an inbox on their web page for the development team to see [62].

Laravel has built-in support for Mailtrap, and it was therefore a natural choice to use this solution for testing and developing [63]. To get it up and running, we only had to add the variables provided by Mailtrap in our environment file as shown in Code listing 6.32.

**Code listing 6.32.** .env when using Mailtrap

```
MAIL_MAILER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=<username>
MAIL_PASSWORD=<password>
MAIL_ENCRYPTION=tls
MAIL_FROM_ADDRESS="noreply@rscci.com"
MAIL_FROM_NAME="RSCCI"
```

Mailtrap has a *Free forever* plan where we get 500 free emails per month, making it perfect for our development and testing needs [64].

## 6.4.2 Production environment - Cloud hosting

This section covers our production environment, which is the stable, public environment we use to host RSCCI. This is the environment Axpetia got access to when testing RSCCI. We will talk about the infrastructure, security, and what e-mail service we used in this environment.

### Service provider and infrastructure

We wanted to use a known, commercial CSP to host our production environment. GitHub Backpack offers credits for five; Digital Ocean (DO), Microsoft Azure (Azure), Heroku, Education Host, and Netwise. Out of these five we already had knowledge and experience with DO and Azure, so these were the cloud service providers we considered.

For easy migration from our development environment to our production environment, we chose to put RSC and CIS on two separate VMs. Therefore, only IaaS was needed from the service provider. Both Azure and DO offers IaaS, but from personal experience we were more comfortable using DO, as the setup and configuration of VMs is pretty simple compared to Azure. We therefore chose to host our environment at DO. DO uses the term *Droplets* for their VMs, and we will therefore use the same term when talking about our VMs in the DO environment.

Our environment at DO is an IaaS platform consisting of two Droplets. Table 6.1 shows their specifications.

**Table 6.1.** Specification for our DO Droplets

| | **RSC Droplet** | **CIS Droplet** |
|---|---|---|
| OS | Ubuntu 20.04 | Ubuntu 20.04 |
| Number of vCPUs | 2 | 1 |
| Memory (RAM) | 2GB | 1GB |
| Storage (SSD) | 60GB | 25GB |
| Transfer (bandwidth) | 3TB | 1TB |

### Security

The environment at DO is configured to be secure. We have implemented multiple best practice security measures for the Droplets, with the main focus on securing them from unauthorized access.

We have configured SSH to only use key-based authorization for connections, and disabled login from the user "root". Key-based authorization makes it close to impossible to brute-force login, and the root user login is disabled to make sure that if an attacker somehow get access to our keys, they will not have root access on the Droplets.

We have set up and configured firewall rules on the Droplets using Uncomplicated Firewall. The inbound firewall rules allow traffic on port 22, 80 and 443, as we only expect traffic on those ports. All other ports and services are closed. The firewall is also configured as described in Section 6.4.1, only allowing HTTPS traffic from Cloudflare IP addresses. In addition, we have opened for communication between the RSC Droplet and CIS Droplet on port 80 via a private VLAN, or VPC, provided by DO.

DO provides a VNC web console to the Droplets, where root login is allowed. Since the web console is only accessible from one of the team members DO account, we have not disabled root login here. We have a strong password generated by and stored in the password vault Bitwarden, in case we lose our authentication keys or the sudo password. The Bitwarden vault is hosted locally by one of the team members, and is not accessible to the public.

To avoid unnecessary down time and data loss in case of an unexpected event, either natural disaster, cyber attack, or human error, we have enabled backups in DO to ensure disaster recovery. Once a week, DO automatically takes a disk image of the desired Droplets. We have chosen to only take backup of the RSC Droplet, as this holds the database with user data. CIS does not store any scan data, so we do not need backups of that Droplet.

The Droplets are both connected to the Internet, but the internal API communications between them happens over over a DO VPC network. The VPC network is internal and separated from the public, other VPC networks, and DO customers, which makes the connection more secure than if the communication was over the Internet, according to DO [65].

### E-mail

Since the application needs to send emails to a user, we needed a way to send emails to actual email accounts instead of trapping them with Mailtrap. For this purpose, we chose to use Mailgun.

Unlike Mailtrap, Mailgun uses an API connection to send emails. This API can be reached using cURL or existing libraries for Python, Ruby, Java, C#, and PHP [66]. Like Mailtrap, Laravel has built-in support for Mailgun, so using this functionality was a natural choice [63].

Even though the Mailgun integration uses an API connection for sending emails, the migration process from Mailtrap was not a lot of work. Both Mailgun and Mailtrap integrations are configured as environment variables, and the code to send email is the same the same no matter which integration we use. Because of this, we did not have to worry about the possibility of the new integration breaking existing functionality, as we only had to change some environment variables. Code listings 6.33 shows the variables used for the Mailgun integration.

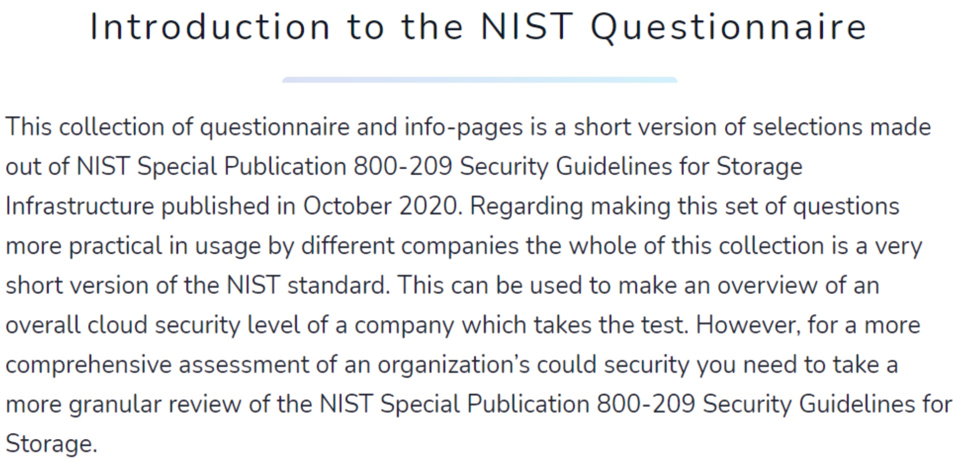
**Code listing 6.33.** .env when using Mailgun

```
MAIL_MAILER=mailgun
MAILGUN_DOMAIN=rscci.cloud
MAILGUN_SECRET=<secret>
MAILGUN_ENDPOINT=api.eu.mailgun.net
```

No other solution was considered for the mail service, since the Mailtrap integration and Mailgun integration are so similar and both are supported by Laravel by default. In addition, Mailgun offers "20,000 free emails and 100 free email validations each month for up to 12 months" via the GitHub Student Developer Pack (GitHub Backpack), which should be enough for our application.

## 6.5 The provided questions and the setup of the questionnaires

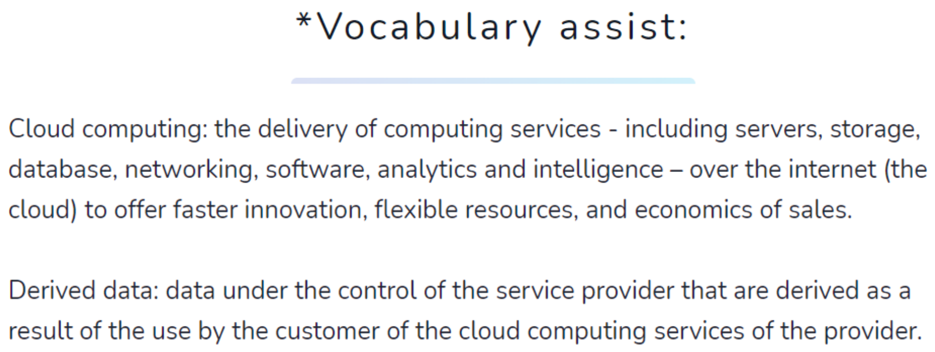
The questionnaire starts with a page informing about which resources are used as shown in the Figure 6.10. In some cases where necessary there is provided a vocabulary assistance in the start of the questionnaire as shown in the Figure 6.11. Then the participant will start the survey by clicking on the "Next" button. The participant can navigate to the previous page using the "Previous" button as shown in the Figure 6.12.



Introduction to the NIST Questionnaire

This collection of questionnaire and info-pages is a short version of selections made out of NIST Special Publication 800-209 Security Guidelines for Storage Infrastructure published in October 2020. Regarding making this set of questions more practical in usage by different companies the whole of this collection is a very short version of the NIST standard. This can be used to make an overview of an overall cloud security level of a company which takes the test. However, for a more comprehensive assessment of an organization's could security you need to take a more granular review of the NIST Special Publication 800-209 Security Guidelines for Storage.

**Figure 6.10.** This page informs about the resource(s) used in this questionnaire.



*Vocabulary assist:

Cloud computing: the delivery of computing services - including servers, storage, database, networking, software, analytics and intelligence – over the internet (the cloud) to offer faster innovation, flexible resources, and economics of sales.

Derived data: data under the control of the service provider that are derived as a result of the use by the customer of the cloud computing services of the provider.

**Figure 6.11.** The vocabulary assistant at the start of the questionnaire.

**Figure 6.12.** The navigation buttons at the bottom of the questionnaire.

### 6.5.1 Resources used in the questionnaires

To set up these questionnaires we have tried to use known trusted international standards to make this collection more useful. In addition, we have used our own knowledge and experience to make some of the general questions. The main resources we have leveraged for this meaning include:

- NS-EN ISO/IEC 27017:2021 Information technology - Security techniques - Code of practice for information security controls based on ISO/IEC 27002 for cloud services.
- NS-EN ISO/IEC 27002:2017 Information technology - Security techniques - Code of practice for information security controls (ISO/IEC 27002:2013).
- NIST Special Publication 800-209, Security Guidelines for Storage Infrastructure.

The ISO/IEC International Standards such as ISO/IEC 27001:2013 Information technology - Security techniques - Information security management systems - Requirements, has also been to a degree of helping this collection. For more information provided about these standards please see Section 2.4.

### General security questions

The questions under the topic of General Security test the participants' common knowledge about cybersecurity. The level of cybersecurity knowledge and readiness of an organization's employees in identifying cyber threats can make an organization's defense line against cyber-attacks, which are widespread and undetected, even more robust. These questions are more general and have not a specific resource other than our own knowledge and experience of information security.

## 6.5.2 About the questionnaires

There is information provided at the beginning of each questionnaire to inform the participants about the resources used in each questionnaire. This is important with regard to making the participants aware of the importance of the questions provided in the questionnaire. The following information, is also provided at the beginning of each questionnaire. The way questions are formulated is however a bit special as using a trick we have managed to make sentences somehow special. It means that same sentence used as a question will fit in the result as an advice. So we do not need two databases one for questions, one for advice to participants.

### IT questionnaire

This questionnaire is based on ISO/IEC 27002 international standard in regard to Information Technology (IT) personnel responsibilities about selecting controls. These controls are within the process of implementing an Information Security Management System, implementing commonly accepted information security controls, and developing their own information security management guidelines [24]. For an organization to perform a security check of their cloud infrastructure, the IT personnel should first fulfil this questionnaire, and then do the one called "Cloud ISO/IEC 27017 questionnaire". This is the base for the cloud specific questionnaire, and should be taken as serious and as important as "Cloud ISO/IEC 27017 questionnaire".

### Management questionnaire

This questionnaire is based on a combination of different resources to cover and test a variety of the understanding of security concepts and readiness of the management personnel of an organization. Hence, this questionnaire is composed of ISO/IEC 27002, ISO/IEC 27017, ISO/IEC 27001, and a few general security questions. The goal of this questionnaire aimed for management of the organization is to involve and aware them with IT security.

### HR questionnaire

This collection of questionnaire is made for Human Resource (HR) employees about IT security concepts as HR personnel of an organization have control on other employees' preparation and training programs regarding IT security knowledge and readiness. The goal of this study prior to employment is to ensure that employees and contractors understand their responsibilities and are suitable for the roles for which they are considered [24]. This questionnaire is based on ISO/IEC 27002, ISO/IEC 27017 international standards.

### Cloud ISO/IEC 27017 questionnaire

This questionnaire is made could specific security check and is based on only ISO/IEC 27017 standard. The base of this standard is ISO/IEC 27002 which is used in IT, HR, and management questionnaires. So for example the IT personnel should first perform a control test on their own questionnaire and then do this cloud specific questionnaire. This task is explained in the information page at the beginning of the IT questionnaire as well.

The management personnel can also use this questionnaire but it is more well suited for IT personnel who are directly responsible and have sufficient knowledge about the organization's cloud infrastructure set up. This questionnaire is the main and most related questionnaire in RSCCI collection. However, we could not exclude ISO/IEC 27002 because as mentioned before, that is the base for this.

### Cloud NIST SP 800-209 questionnaire

This questionnaire is only based on NIST standard and no combination of other standards are used in it. As we prepared a research to find the related standards to cloud infrastructure baselines and security, we found some interesting, related and useful parts in NIST. Also, we thought some organizations may see NIST more interesting than ISO/IEC standards. Therefore, we saw it necessary to make a questionnaire separated with the one named "Cloud ISO/IEC 27017 questionnaire".

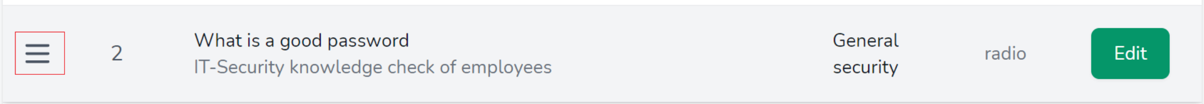
This collection of questionnaire is a short version of selections made out of NIST Special Publication 800-209 Security Guidelines for Storage Infrastructure published in October 2020. Regarding making this set of questions more practical in usage by different organizations the whole of this collection is a very short version of the NIST standard. This can be used to make an overview of an overall cloud security level of an organization which takes the test. However, for a more comprehensive assessment of an organization's could security organizations need to take a more granular review of the NIST Special Publication 800-209 Security Guidelines for Storage.

### Employee questionnaire

This questionnaire is made for all normal employees who does not have any specific responsibility about IT components or jobs in the organization. This is because nowadays Bring your own device (BYOD) is everywhere in form of personal laptops, mobile phones, iPads etc. A normal employee could be target of a phone call which expose the person for a social engineering, or opening an email which contains ransomware. It is of a high priority to train normal employees of the organization regarding readiness for different critic situations. This questionnaire is based on general security questions.

## 6.5.3 The order of questions in the questionnaire

It is possible for the person with the administrator privileges to change the order of questions to another desired order by moving questions up and down in Admin view. However we do not recommend this for other than the Employees' questionnaire, as the order of the questions follows the order of the controls in the ISO standards. Figure 6.13 shows where the administrator can handle and move a question and change the order. Speaking about what administrator can do, same figure shows the "Edit" button which makes it possible for the administrator to edit each question and its sub-questions as desired.



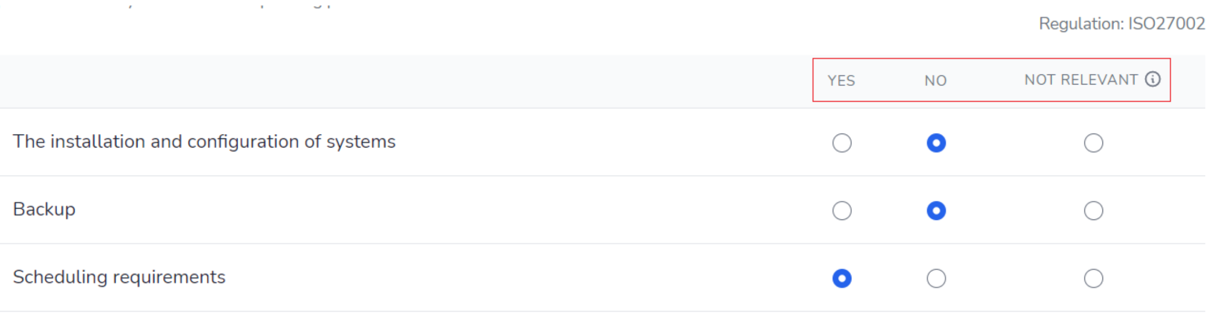
☰ 2 What is a good password
IT-Security knowledge check of employees
General security radio

**Figure 6.13.** The handle for moving each question to change their place in the question order, marked with red rectangle.

### 6.5.4 The way the questionnaires works

When a questionnaire starts either the question is of type of "Yes/No/Not relevant", radio button, or multiple choice. Regarding the ones of "Yes/No/Not relevant" type, each question has three solutions to choose between, as shown in Figure 6.14.

- "**Yes**" which means that the control is relevant to the participant's organization in addition to having it in place.
- "**No**" means that the control is relevant to the participant's organization, but it is not practiced. Which means that they need to set it up in their organization.
- "**Not Relevant**" which means that this control is not relevant to the participant's organization and the final result will not take it into account.



Regulation: ISO27002

| | YES | NO | NOT RELEVANT ⓘ |
|---|---|---|---|
| The installation and configuration of systems | ○ | ● | ○ |
| Backup | ○ | ● | ○ |
| Scheduling requirements | ● | ○ | ○ |

**Figure 6.14.** The participant is offered three choices to answer among, marked with a red rectangle.

From the start of each questionnaire until it ends there will be a progress bar which shows how much of the current questionnaire is fulfilled and how much is left. The progress bar is an effort to make the test more user friendly, see Section 4.5. As the questionnaires are comprehensive and will demand time and attention of the participant. Due to this matter we have made it is possible for each participant to do a part of a questionnaire, take a break, and then continue. The break could be needed both because the participant might need to check the control in their system to provide a more correct and updated answer to the test, or could be needed because of other reasons. This could be accomplished using the "Back to questionnaire overview" button at the above of the progress bar. See also Figure 6.15.

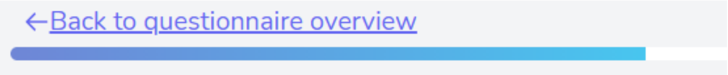


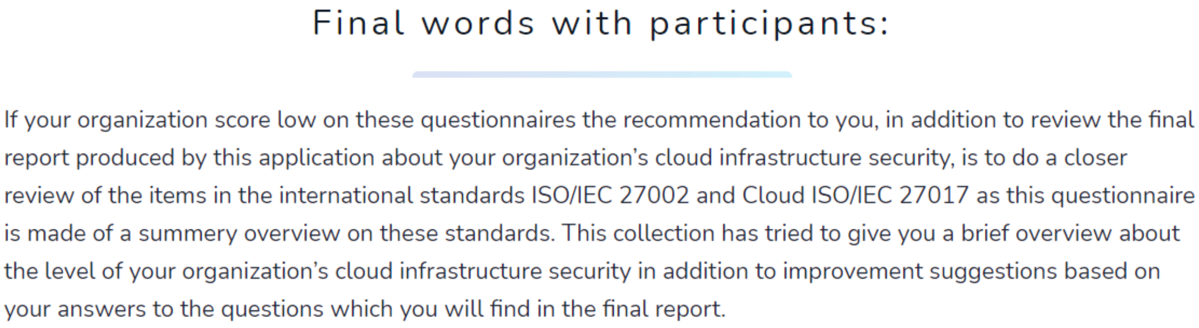
**Figure 6.15.** To take a break for any reason the participant can click on this button, escape the questionnaire at the time while getting back at any desired time in the future.

Before ending the questionnaire there will be a final word with the participant, especially IT personnel. This will come with some recommendations about how to interpret and use the result of the questionnaire completed as show in the Figure 6.16.

When all questions are answered the questionnaire will end with a "Thank you!" page as shown in Figure 6.17. This will inform the participant at the same time about the way forward which is to review the report and set up the controls which should be in place. These are the controls that at current time are not there, as the participant themselves has confirmed during the fulfilling of the questionnaire.

### 6.5.5 The result of the questionnaire

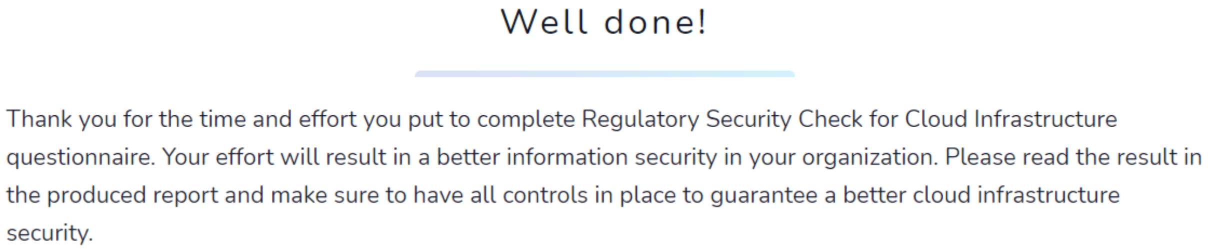
The results are made available from the drop-down list in the top menu both as individual report per questionnaire and as summary report as shown in Figure 6.18. The summary report is useful for instance for managers to give an overall overview of their organization's cloud information security level.



Final words with participants:

If your organization score low on these questionnaires the recommendation to you, in addition to review the final report produced by this application about your organization's cloud infrastructure security, is to do a closer review of the items in the international standards ISO/IEC 27002 and Cloud ISO/IEC 27017 as this questionnaire is made of a summery overview on these standards. This collection has tried to give you a brief overview about the level of your organization's cloud infrastructure security in addition to improvement suggestions based on your answers to the questions which you will find in the final report.

**Figure 6.16.** A final word with the participant from the IT department.



Well done!

Thank you for the time and effort you put to complete Regulatory Security Check for Cloud Infrastructure questionnaire. Your effort will result in a better information security in your organization. Please read the result in the produced report and make sure to have all controls in place to guarantee a better cloud infrastructure security.

**Figure 6.17.** "Thank you!" page at the end of each questionnaire.

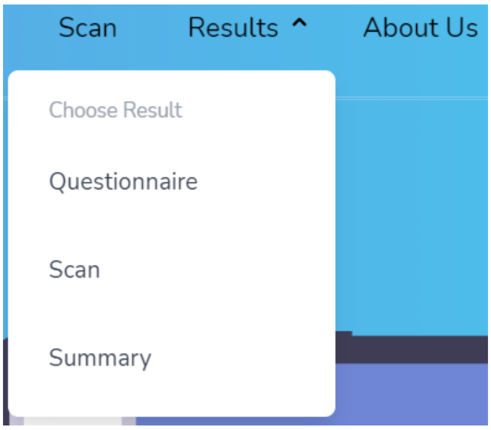
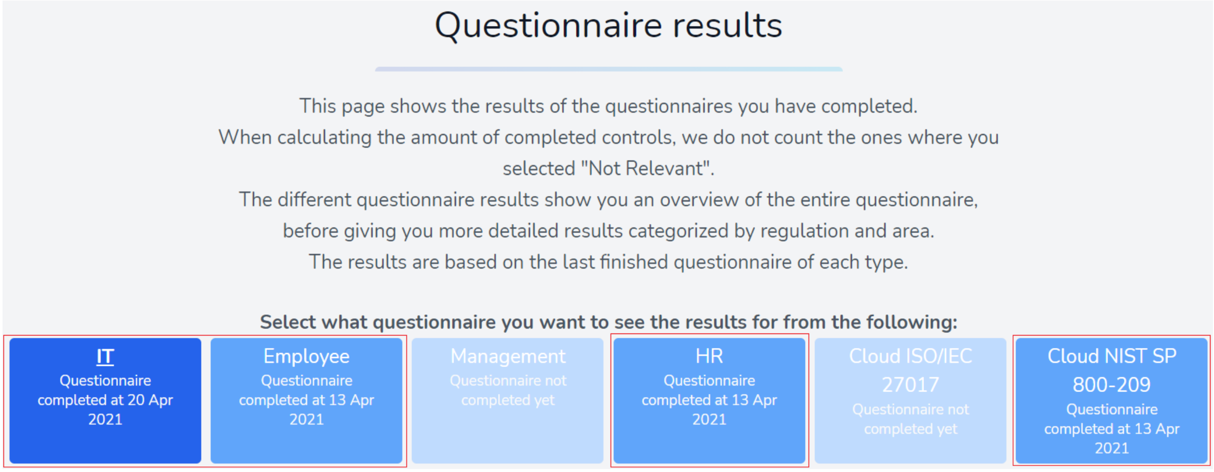


**Figure 6.18.** The drop-down menu makes the results of scan and questionnaires available. The result of the questionnaires are accessible both per individual questionnaire and as a summary for all questionnaires taken per department.

At the result page user can choose to see the desired questionnaire result by clicking on the related button. Only the buttons which the participant has fulfilled the related questionnaires are activated as shown in the Figure 6.19 with dark blue color. For instance, this person has fulfilled four tests, which are available to him/her related to the job/position in the organization, and have access to each individual questionnaire result as marked with red rectangles in the Figure 6.19. The reason we have chosen IT, Management, and HR as individual tests is that the ISO/IEC standards, mainly ISO/IEC 27002, centrally addresses these three groups.

Adding two sets of cloud questionnaires is to have this security check more complete once from the Cloud ISO/IEC 27017 perspective and once from Cloud NIST SP 800-209 perspective. The questionnaire for normal employee is provided because almost all employees in the organization have a relation to information technology in form of having an email or working with software etc. which makes it necessary for them to contribute with cybersecurity requirements and have a basic understanding of both concepts and the way to handle basic events.

The final result could be downloaded and is made available in the form of a .pdf file by clicking on the "Download PDF" button as shown in the Figure 6.20. If the user has clicked on the "Summary" from the drop-down list in the menu, shown in the Figure 6.19, the report is



Questionnaire results

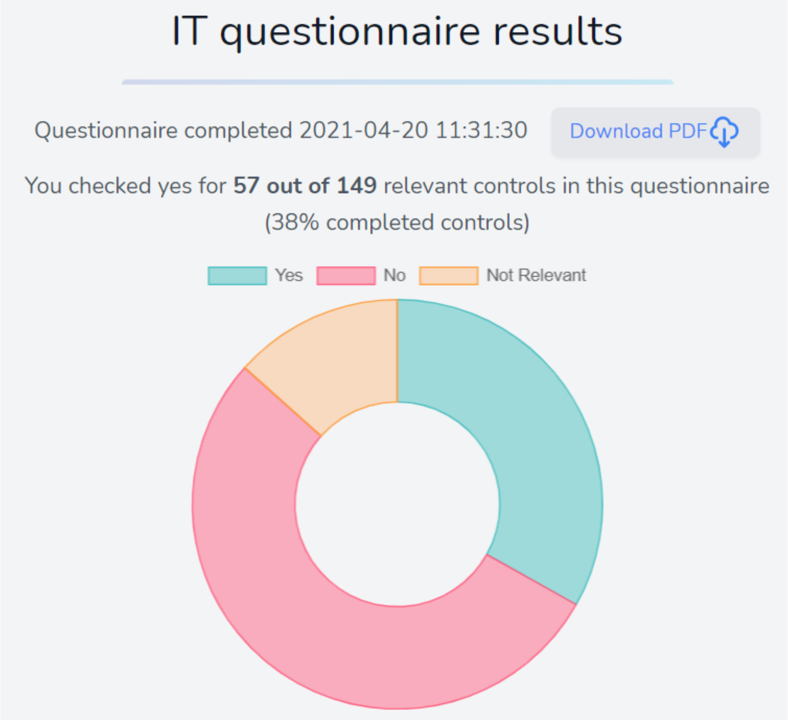
This page shows the results of the questionnaires you have completed.
When calculating the amount of completed controls, we do not count the ones where you selected "Not Relevant".
The different questionnaire results show you an overview of the entire questionnaire, before giving you more detailed results categorized by regulation and area.
The results are based on the last finished questionnaire of each type.

**Select what questionnaire you want to see the results for from the following:**

IT
Questionnaire completed at 20 Apr 2021
Employee
Questionnaire completed at 13 Apr 2021
Management
Questionnaire not completed yet
HR
Questionnaire completed at 13 Apr 2021
Cloud ISO/IEC 27017
Questionnaire not completed yet
Cloud NIST SP 800-209
Questionnaire completed at 13 Apr 2021

**Figure 6.19.** Individual questionnaire results made available to the participant.

available as a summary of all questionnaires taken. The PDF version of the result is identical to the one offered in the application. The report has a score of what percentage of the standard rules, related to the cloud service customer, are fulfilled in a total overview, and what percentage should be working on as they are not provided yet in their organization. The report then provides a list of controls that need to be practiced by the participant's organization as the cloud service customer to improve their level of cloud information security.



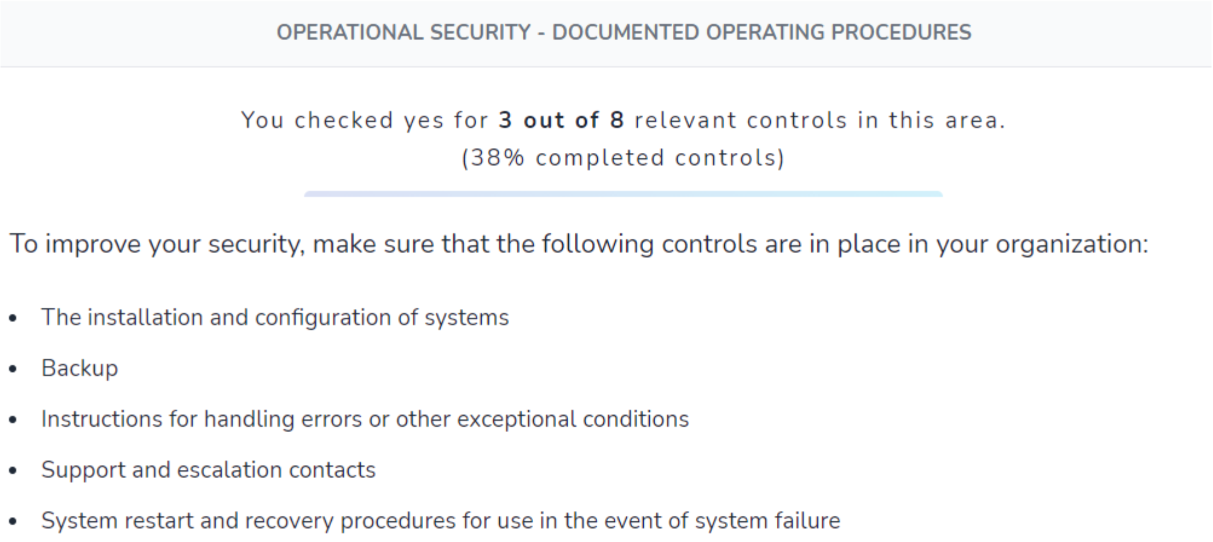
IT questionnaire results

Questionnaire completed 2021-04-20 11:31:30

You checked yes for **57 out of 149** relevant controls in this questionnaire (38% completed controls)

**Figure 6.20.** Figure shows a total result of a fulfilled questionnaire with an estimate of 38% of the controls completed. The result page also provides the date and time by which the questionnaire was fulfilled. In addition, the button to download the result as a PDF file is shown in the figure.

Furthermore, the result is observable as a list of recommendations with an estimate of level of information security controls in place for exactly this control. Figure 6.21 shows a small part of the reported results related to the questionnaire completed.



OPERATIONAL SECURITY - DOCUMENTED OPERATING PROCEDURES

You checked yes for **3 out of 8** relevant controls in this area.
(38% completed controls)

To improve your security, make sure that the following controls are in place in your organization:

- The installation and configuration of systems
- Backup
- Instructions for handling errors or other exceptional conditions
- Support and escalation contacts
- System restart and recovery procedures for use in the event of system failure

**Figure 6.21.** Figure shows a small part of the results from the report with an estimate of completed security controls.

# Chapter 7

# Evaluation

This chapter covers the evaluation of RSCCI. We explain our evaluation methods and evaluate how RSCCI complies with the functional and non-functional requirements specified in Chapter 3. We will also evaluate RSCCI based on the feedback from Axeptia.

## 7.1 Evaluation method

To evaluate RSCCI, we have tested the web application ourselves, had Axeptia do user testing, and sent out a survey about RSCCI for Axeptia's employees[1] to answer. When we tested the web application ourselves, we relied on the uses cases specified in Chapter 3. This means that we performed the actions as the use cases describes and checked if the system responded according to the related requirements and post-conditions. We also did research and other tests where necessary. The user testing consists of having Axeptia testing the functionalities, like CIS, and giving general feedback based on their experience. They also answered a survey created by us to help evaluate RSCCI. The survey with answers can be found in Appendix G. Tables 7.1 and 7.2 show what evaluation methods are used for the different requirements.

**Table 7.1.** Evaluation method for the functional requirements

| **Functional Requirement \Evaluation method** | **Own testing** | **User testing** |
|---|---|---|
| **Requirement 1:** The solution shall map relevant generic regulatory requirements by the use of questionnaires to shed light on how a company views its security architecture and what requirements should be met. | X | X |
| **Requirement 2:** The solution shall automatically generate a report to shed light on what security requirements are met, and give advice (or links to advice) about how a company can improve their security on specific areas, based on the answers from the questionnaires. | X | X |
| **Requirement 3:** The solution shall be placed in a cloud environment and be able to be used by all companies with cloud infrastructure, regardless of CSP. | X | X |
| **Requirement 4:** The solution shall make sure that only the company that answers the surveys or perform a scan is authorized to see the results. | X | |
| **Requirement 5:** Companies that outsource their IT business should be able to use the solution. | | |
| **Requirement 6:** The solution shall be able to perform simple network scans against a users endpoints, to check if a users infrastructure is potentially vulnerable to known vulnerabilities. | X | X |

[1] Excluding our team member who works at Axeptia.

**Table 7.2.** Evaluation method for the non-functional requirements

| **Non-Functional Requirement \Evaluation method** | **Own testing** | **User testing** |
|---|---|---|
| **Requirement 7a:** The solution shall support user authentication with salted, hashed passwords. Two-factor authentication shall be supported, but only enabled when a user desires to use two-factor authentication. | X | |
| **Requirement 7b:** The solution should support authorization by roles. | X | X |
| **Requirement 7c:** The solution should support accountability for user actions. | X | |
| **Requirement 7d:** As the solution is web-based, it should be protected against the common web-vulnerabilities listed in OWASP Top 10. | X | |
| **Requirement 7e:** The infrastructure used to host the solution shall use encryption when transmitting data over the network with use of approved protocols, such as HTTPS and SSH. | X | |
| **Requirement 7f:** The infrastructure used to host the solution shall be secured in accordance with best practices. | X | |
| **Requirement 8a:** The different pages in the application should load in about 1 second. The questionnaire should load in about 0.1 seconds. | X | |
| **Requirement 8b:** The web application shall achieve a Google PageSpeed score of over 90. | X | |
| **Requirement 9a:** The web application should have 99% availability within normal working hours to ensure users can use our web application. | X | X |
| **Requirement 9b:** Backup of the web applications database should be done on a regular basis to minimize data loss in case of disaster. | X | |
| **Requirement 10:** The scanner part of the solution needs to perform its tasks linearly to minimize the risk of the scan being interpreted as a DoS attack, so it should be possible to run multiple scanner instances at the same time against multiple cloud infrastructures. | X | |
| **Requirement 11a:** Where needed, sufficient information about functionality shall be provided. | | X |
| **Requirement 11b:** Users who do not have background in information technology and cyber security should be able to understand and use the solution easily. | | X |

## 7.2 Evaluation - Functional Requirements

In this section, we evaluate how RSCCI complies with the functional requirements specified in Chapter 3.

### Requirement 1

As we have used known, recognized standards from the cyber security community to generate the questionnaires, we know the questionnaires cover relevant requirements. The testing done by Axeptia shows that this is in fact true. We therefore conclude requirement 1 as passed.

### Requirement 2

The survey results show that the questionnaire results are understandable and useful. In addition, the employees who answered say they got a better insight of Axeptia's cloud security after using RSCCI. One of Axeptia's employees once said: "I thought I knew more about security before taking these questionnaires". We can therefore conclude that requirement 2 is passed.

### Requirements 3

When it comes to CIS, Axeptia tested it against Microsoft Azure, while we have tested it against DO, Hetzner, and our local environment. These tests have been successful, meaning the scanner can be used against multiple CSPs. For RSC, we have not created any CSP specific questions. Requirement 3 is therefore passed.

### Requirement 4

For the evaluation of requirement 4, we have performed tests to ensure that a user cannot see other companies' or team members' results. In our production environment, we have the companies Axeptia and RSCCI. A user from RSCCI cannot see Axeptia's results, or other team members' results. We have also anonymized each team members results in when showing a team's summary results. The only way a company can access another company's results is by compromising our database or the other company's RSCCI accounts. Requirement 4 is therefore passed.

### Requirement 5

Due to the time limit, we have not been able to get any companies that outsource their IT business to test our solution. We therefore do not know if requirement 5 is passed or not. This will be discussed further in Chapter 8.

### Requirement 6

Axeptia had some more security measures enabled on their Azure environment, meaning the scanner were not able to find any vulnerabilities. However, Axeptia were happy about the result and the functionality of the scanner. When it comes to the questionnaires, they are generated to support all types of cloud infrastructures, regardless of CSP. Therefore, requirement 6 is passed.

### Summary

Table 7.3 shows a summary of the evaluation of RSCCI's compliance with the functional requirements. The web application works as desired, and Axeptia are happy with the result.

**Table 7.3.** Summary evaluation of RSCCI's compliance with the functional requirements

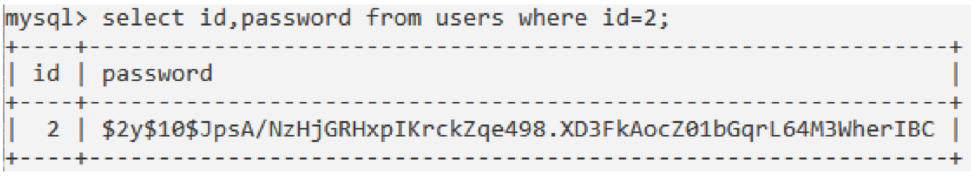
| **Functional Requirement** | **Passed/Failed** |
|---|---|
| **Requirement 1:** The solution shall map relevant generic regulatory requirements by the use of questionnaires to shed light on how a company views its security architecture and what requirements should be met. | Passed |
| **Requirement 2:** The solution shall automatically generate a report to shed light on what security requirements are met, and give advice (or links to advice) about how a company can improve their security on specific areas, based on the answers from the questionnaires. | Passed |
| **Requirement 3:** The solution shall be placed in a cloud environment and be able to be used by all companies with cloud infrastructure, regardless of CSP. | Passed |
| **Requirement 4:** The solution shall make sure that only the company that answers the surveys or performs a scan is authorized to see their own results. | Passed |
| **Requirement 5:** Companies that outsource their IT business should be able to use the solution. | Not tested |
| **Requirement 6:** The solution shall be able to perform simple network scans against a user's endpoints, to check if a user's infrastructure is potentially vulnerable to known vulnerabilities. | Passed |

## 7.3 Evaluation - Non-Functional Requirements

In this section, we evaluate how RSCCI complies with the non-functional requirements specified in Chapter 3.

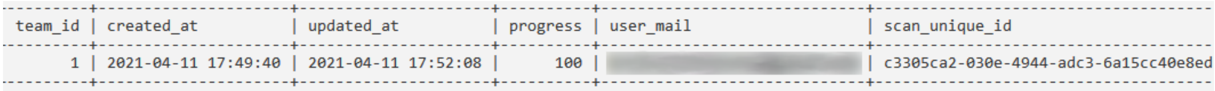
### Requirements 7a-c

The evaluation of requirements 7a-c is based on the testing and verifying of the database by us. As described in Section 6.2.3 we have secure user authentication, and two-factor authentication. As Figure 7.1 shows, we salt and hash the passwords stored in the database using bcrypt. Most of the actions on the web applications are restricted to specific roles, meaning a user has to be authorized to perform actions. See Section 6.2.3. For the scanning action, we save the user email and scan information to the database to support accountability, as described in Section 6.2.3 and shown in Figure 7.2. What the figure excludes is the results field where the IP addresses are stored. This is excluded because then the figure will be unreadable. The conclusion is that requirements 7a-c is passed.



```
mysql> select id,password from users where id=2;
+----+--------------------------------------------------------------+
| id | password                                                     |
+----+--------------------------------------------------------------+
|  2 | $2y$10$JpsA/NzHjGRHxpIKrckZqe498.XD3FkAocZ01bGqrL64M3WherIBC |
+----+--------------------------------------------------------------+
```

**Figure 7.1.** Hashed password from the database



```
---------+---------------------+---------------------+----------+---------------------------+------------------------------------
 team_id | created_at          | updated_at          | progress | user_mail                 | scan_unique_id
---------+---------------------+---------------------+----------+---------------------------+------------------------------------
       1 | 2021-04-11 17:49:40 | 2021-04-11 17:52:08 |      100 | [redacted]                | c3305ca2-030e-4944-adc3-6a15cc40e8ed
---------+---------------------+---------------------+----------+---------------------------+------------------------------------
```

**Figure 7.2.** Scan table from the database

### Requirement 7d

To evaluate requirement 7d, we checked our web application for the vulnerabilities mentioned in OWASP Top 10, either by research or testing. The following list describes how RSCCI complies with OWASP Top 10 and how we came to the conclusion:

1. **Injection**
   As mentioned in Section 6.2.3, we use Laravel query builder for our SQL statements. The query builder uses PDO binding, which protects the web application from SQL injection attacks [56]. We did a sample test to check if this is correct, by attempting a SQL injection attack on the web applications login page. This was not successful. We therefor conclude that the web application is not vulnerable for SQL injection.

2. **Broken Authentication**
   By default, Laravel Fortify and Laravel Jetstream protect from lots of broken authentication vulnerabilities. As mentioned in Section 4.4.1 Laravel Fortify provides safe logic for login, registration, password reset, and email verification, while Laravel Jetstream have rate limiting applied for login attempts [36, 67]. From research, we have not found any known vulnerabilities for both Laravel Fortify and Laravel Jetstream.

3. **Sensitive Data Exposure**
   The website does not store sensitive personal information, only a user's first name, last name, and email address. However, we do store sensitive information when it comes to the scan results, since this is information that potentially could be abused by a malicious attacker. The scan results are not exposed on the web application, and only users with

the right permission from the company that did the scan has access to it. For password security, please see evaluation of Requirements 7a-c.

4. **XML External Entities**
   Our web application does not accept any XML documents, so it is not vulnerable.

5. **Broken Access Control**
   To evaluate this vulnerability, we tried to act as an malicious user and do actions we are not authorized to do. We use strict authorization as discussed in Section 6.2.3 We first tried to change the URL and access a page as an unauthenticated user. This resulted in error. We then did the same as an authenticated user without scan permissions. This also resulted in error.

6. **Security Misconfiguration**
   For all the frameworks scaffolding templates, and programs we have used, we checked the default configuration settings. If a setting needed a password change or the default where not secure enough, we changed it to something more secure. For the infrastructure, see Section 6.4.2 for the security configuration.

7. **Cross-Site Scripting**
   To check for possible XSS vulnerabilities, we tried to perform XSS on all the input fields in our web application. For the input fields we have created, this was not possible, as we have implemented input validation and sanitation. Section 6.2.3 describes how this is implemented. As also mentioned in Section 6.2.3, we manually strip tags and add <br> for the questions in the questionnaires. Therefore, we tried to add a simple XSS statement to one of the questions and see if the script tag was stripped or executed. As expected, the tags where stripped.

8. **Insecure Deserialization**
   There are two scenarios where we need to deserialize JSON format: when importing the questions and when parsing the scan results. Both scenarios use JSON, which is a safe interchange format, with the safe function json_decode() [68].

9. **Using Components with Known Vulnerabilities**
   During our research, we have not found the components used to develop the web application and set up the infrastructure to have any known vulnerabilities.

10. **Insufficient Logging and Monitoring**
    By default, NGINX logs all requests to our Droplets. We also get a lot of change logs an monitoring metrics from DO about our Droplets. When it comes to the web application, we save user email with every scan, as mentioned in the evaluation of Requirements 7a-c.

Based on the individual evaluations of OWASP Top 10, we conclude that requirement 7d is passed.

### Requirements 7e-f

Section 6.4.2 describes how we secured the infrastructure hosting RSCCI. If we have secured it according to best practice is not easy to answer, as best practice is not a set definition. However, since we are using DO as our CSP, we have chosen to check our implemented measures against their recommended security measures. As far as we can see, we have followed all of the relevant measures DO recommend. We also use HTTPS, as mentioned in Section 6.2.3. This means that requirements 7e-f are passed.

### Requirements 8a-b

Figure 7.3 shows the Google PageSpeed score for RSCCI. This shows that the web application got a score of 96. For the load time of the different pages on the web application, we check manually by loading the different pages and taking notes of the load time. On average, the different pages loaded in about 0.5 seconds. The pages with the most calculations, like the questionnaire result page and the summary result page, loads in about 1 second. Therefore requirement 8a is mostly passed, while requirement 8b is passed.

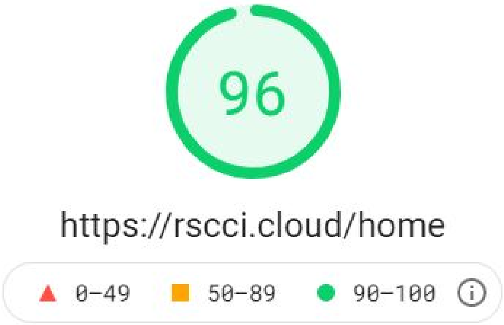


**Figure 7.3.** Google PageSpeed score for https://rscci.cloud/home

### Requirements 9a-b

Testing the web application's availability is not an easy task, and we can therefor not give a definite answer to whether or not RSCCI is available 99% of the time during normal working hours. However, DO has a 99.99% uptime guarantee for their Droplets and block storage, meaning the infrastructure should be available almost all the time [69]. As we have not implemented Continuous Deployment, as mentioned in Chapter 5, we can plan our software maintenance and deployment to make sure the web application is available during normal working hours. As mentioned in Section 6.4.2, DO takes a disk image once a week of the RSC Droplet to make sure we have a backup of the database. Based on this information, we can conclude that requirements 9a-b are passed.

### Requirement 10

To evaluate requirement 10, we started a scan against two cloud infrastructures at the same time. This was a successful test, and both of the cloud infrastructures were scanned at the same time. This is not surprising, as CIS is inside a Docker container and it is easy to start multiple Docker instances at the same time. We have also implemented support for multiple CIS instances in the API endpoint at the web application. Although we have not stress tested multiple instances, the only thing that should come in the way for it to work is the capacity of the DO Droplet the instances run on. We therefore evaluate requirement 10 as passed.

However, due to the fact that Axeptia is the only company that tested our web application, this feature is currently disabled. The reason for this is discussed further in Section 8.1.5.

### Requirements 11a-b

The evaluation of requirements 11a-b is based on feedback from Axeptia. With only 12 employees, we can see that there is a good spread of how understandable Axeptia found the questionnaire results. As shown in Figure 7.4, the answers range from 3 to 10 with an average of about 8. The answer 3 stands out from the other answers, but since the data only consists of 12 answers it is hard to say if it is an anomaly. When it comes to how understandable Axeptia found the scan results, the answers are more similar. The answers range from 6 to 10 with an average of about 9, as shown in Figure 7.5. Figure 7.6 shows how Axeptia answered when it comes to how understandable the summary results are. The answers range from 8 to 10 with an average of about 9.

The results were, on average with one decimal, 8.6 out of 10 understandable. We believe this to be very good, as it is almost top score. We therefore conclude requirements 11a-b as

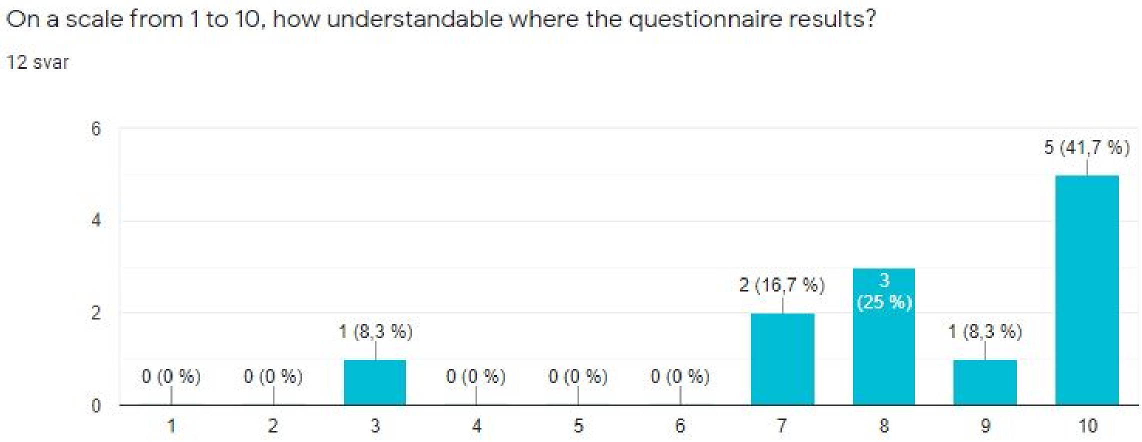


**Figure 7.4.** User evaluation of the questionnaire results

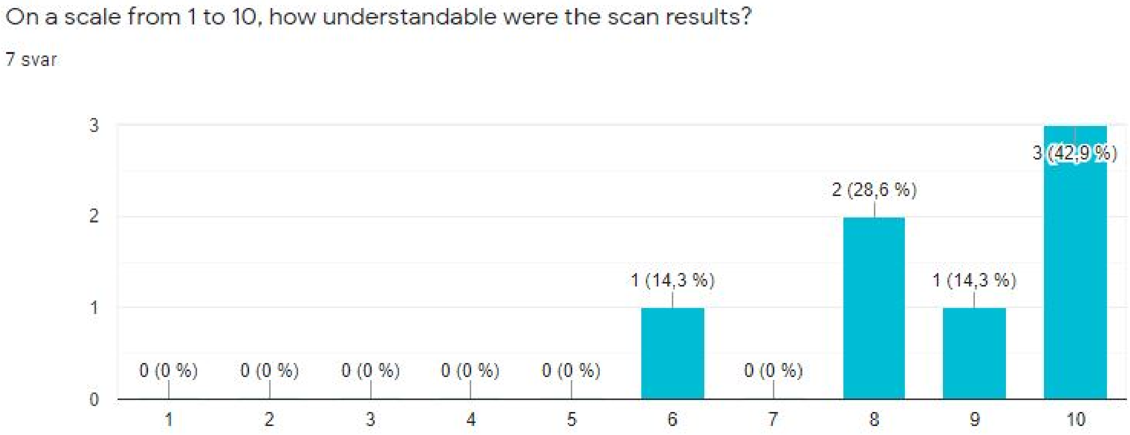


**Figure 7.5.** User evaluation of the scan results

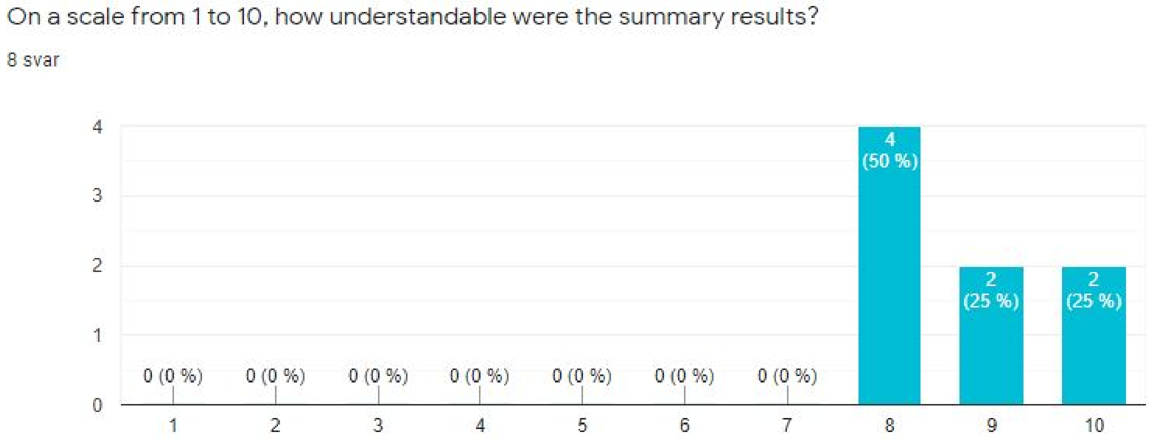


**Figure 7.6.** User evaluation of the summary results

passed. However, as there was someone who answered lower than the others and we have too few participants to know if this is a anomaly. This means that we could still do better to make sure that the information about functionality is sufficient and make RSCCI more user-friendly for users who do not have background in information technology and cyber security.

### Summary

Table 7.4 show a summary of the evaluation of RSCCI's compliance with the non-functional requirements.

**Table 7.4.** Summary evaluation of RSCCI's compliance with the non-functional requirements

| **Non-Functional Requirement** | **Passed/Failed** |
|---|---|
| **Requirement 7a:** The solution shall support user authentication with salted, hashed passwords. Two-factor authentication shall be supported, but only enabled when a user desires to use two-factor authentication. | Passed |
| **Requirement 7b:** The solution should support authorization by roles. | Passed |
| **Requirement 7c:** The solution should support accountability for user actions. | Passed |
| **Requirement 7d:** As the solution is web-based, it should be protected against the common web-vulnerabilities listed in OWASP Top 10. | Passed |
| **Requirement 7e:** The infrastructure used to host the solution shall use encryption when transmitting data over the network with use of approved protocols, such as HTTPS and SSH. | Passed |
| **Requirement 7f:** The infrastructure used to host the solution shall be secured in accordance with best practices. | Passed |
| **Requirement 8a:** The different pages in the application should load in about 1 second. The questionnaire should load in about 0.1 seconds. | Mostly passed |
| **Requirement 8b:** The web application shall achieve a Google PageSpeed score of over 90. | Passed |
| **Requirement 9a:** The web application should have 99% availability within normal working hours to ensure users can use our web application. | Most likely passed |
| **Requirement 9b:** Backup of the web applications database should be done on a regular basis to minimize data loss in case of disaster. | Passed |
| **Requirement 10:** The scanner part of the solution needs to perform its tasks linearly to minimize the risk of the scan being interpreted as a DoS attack, so it should be possible to run multiple scanner instances at the same time against multiple cloud infrastructures. | Passed |
| **Requirement 11a:** Where needed, sufficient information about functionality shall be provided. | Passed |
| **Requirement 11b:** Users who do not have background in information technology and cyber security should be able to understand and use the solution easily. | Passed |

## 7.4 Evaluation - Application

This section covers the feedback we got from the user survey that is not directly connected to a functional or non-functional requirement.

### Design

Figure 7.7 shows how the Axeptia employees answered about if they liked the design of RSCCI or not. Almost everyone liked the design, except for one. We look at the results as normal, as liking something is a personal preferences. As mentioned previously, the team members who developed RSCCI are mostly used to back end development, so this is a good result.

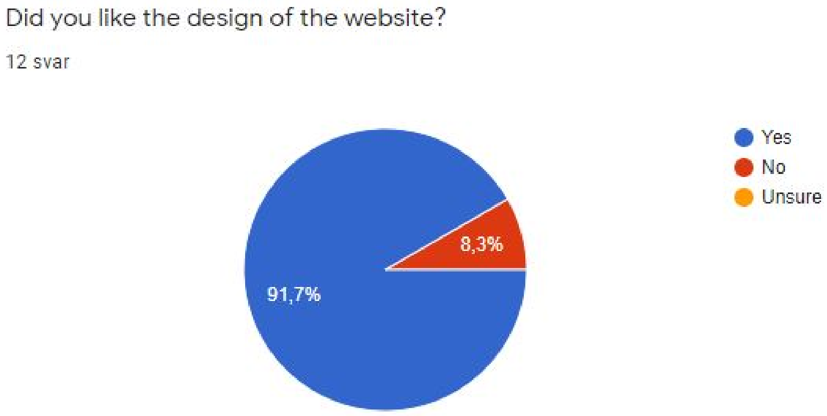


**Figure 7.7.** Answers about the design of RSCCI

## Usability

Figure 7.8 shows how useful the questionnaire results were for Axeptia. For most of the employees, the questionnaire results were very useful. As for how understandable the results were, one person answered 3, as shown in Figure 7.4. An individual check of the answers tells us that this is the same person who answered 3 for how understandable the results were, so there is most likely a connection between these answers.

Figure 7.9 shows how Axeptia feels about the usability of the scan results. Here we can see that most of the employees who were able to see the results think they are useful, while one answered with a lower score. However, there is a reason for this. Figure 7.14 shows the optional feedback Axeptia could give us. Here, the person who answered 4 says: "The score is low since I have not used it [the scanner] as much, and therefore I am not sure about the results. I hope I can try it more in the future, it seems like a really good tool and I am convinced it has high value!".

Figure 7.10 shows how Axeptia answered about the usability of the summary results. The answers show that the summary results are useful.

We have tried to create results that not only are understandable, but that can also be used to improve cloud security and regulatory compliance. The feedback from Axeptia shows that the results are in fact useful.

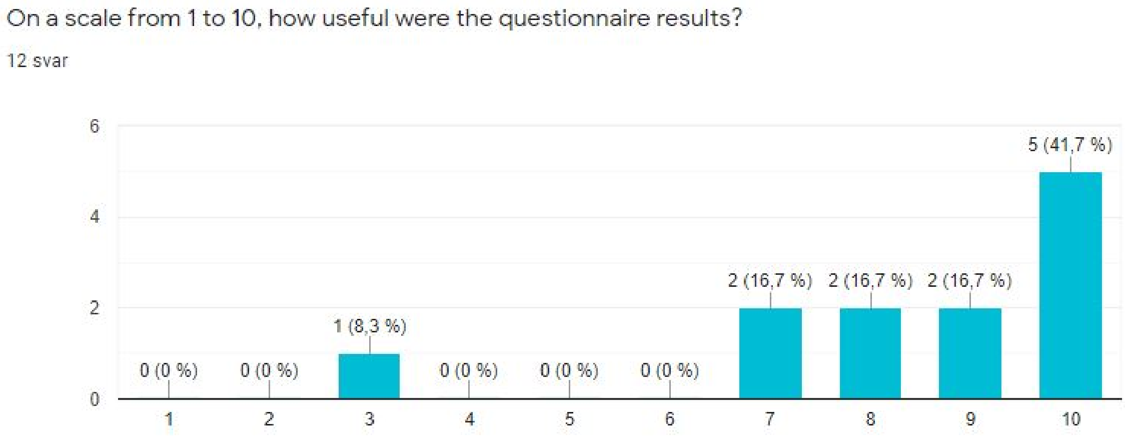
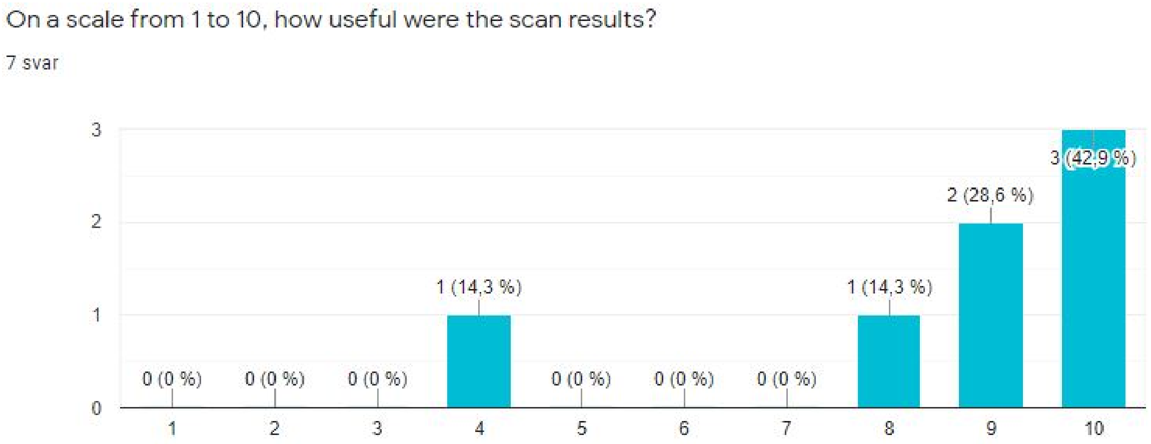
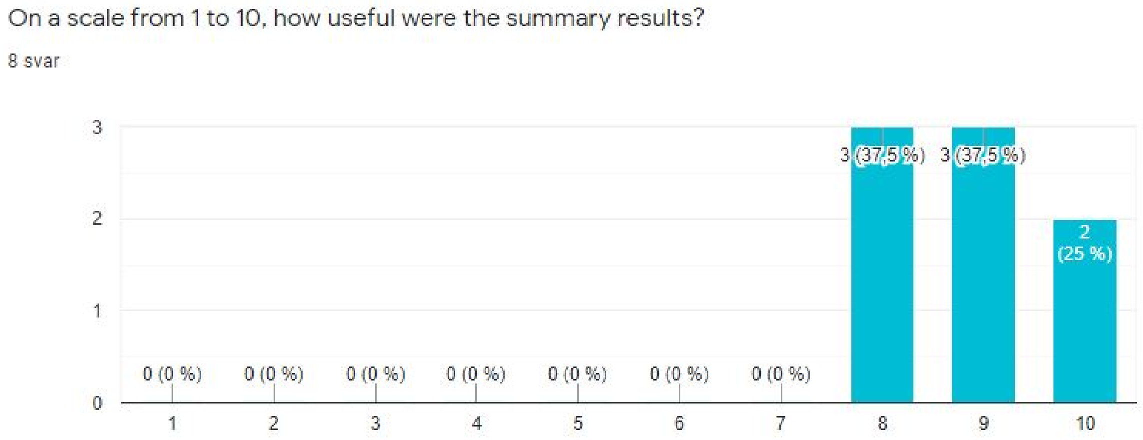


**Figure 7.8.** Answers about the questionnaire results usability


On a scale from 1 to 10, how useful were the scan results?

7 svar

**Figure 7.9.** Answers about the scan results usability


On a scale from 1 to 10, how useful were the summary results?

8 svar

**Figure 7.10.** Answers about the summary results usability

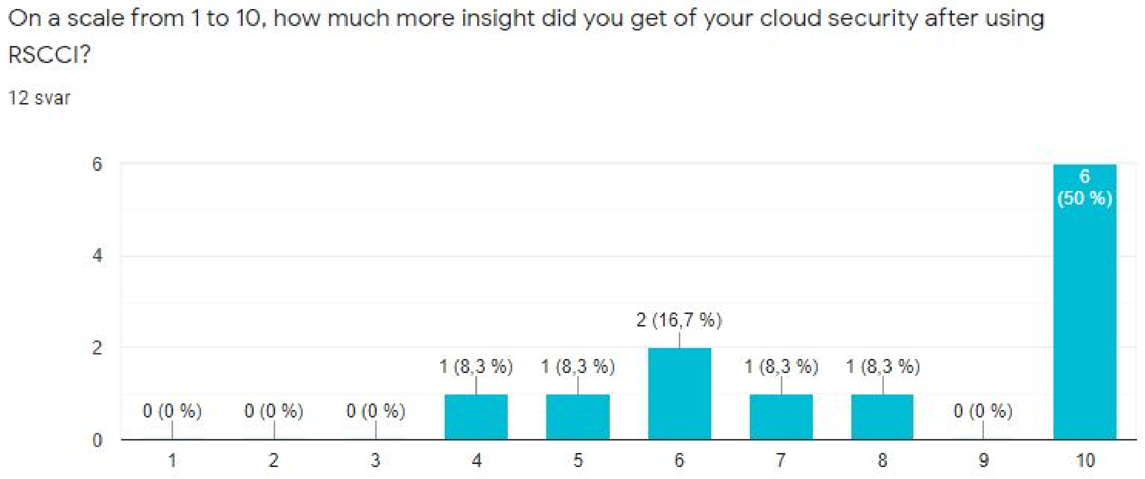
### Insight

Figure 7.11 shows what Axeptia's employees answered in regards to how much more insight they gained of their cloud security after using RSCCI. The answers are evenly spread from 4 to 10. As the results varies a little, we looked into the individual answers and compared them with what questionnaire the employee took. The results show that the employees who only took the Employee questionnaire did not get as much insight into their cloud security as the employees who took the other questionnaires. This is not surprising, as the functionality that generates the Employee questionnaire results where implemented late in the implementation period, and therefore only shows how many correct and wrong answers the user has. This is a problem we are aware of, and something we would have liked to improve. More about this in Section 9.3.

As for the usability, the person who answered 3 on how understandable the questionnaire results were, also reported a lower score for how much more insight into his/her cloud security he/she got after using RSCCI. There is most likely a connection between these answers as well.

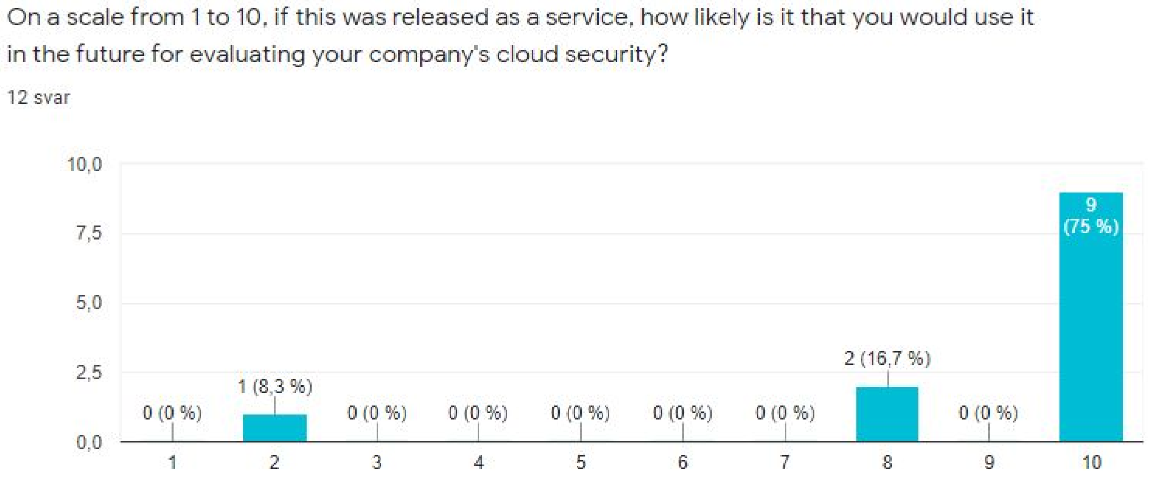
### Further use

Figure 7.12 shows how likely the employees of Axeptia are to use RSCCI as a cloud security evaluation tool if it was released as a service. The answers are very uplifting, as 9 of 10 gives us the highest score. Again, the the person who answered 3 on how understandable the questionnaire results were also reported a lower score, which most likely has a connection. We also get the same type of answers when it comes to if Axeptia would recommend RSCCI for other companies, shown in Figure 7.13. This indicates that Axeptia is happy with the overall result of the project.


On a scale from 1 to 10, how much more insight did you get of your cloud security after using RSCCI?

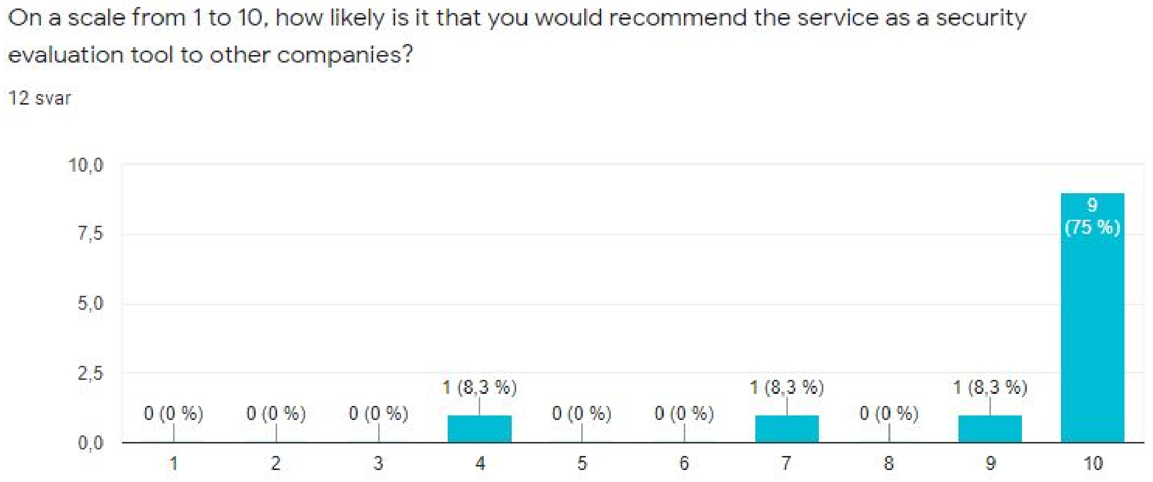
12 svar

**Figure 7.11.** Answers about insight after using RSCCI


On a scale from 1 to 10, if this was released as a service, how likely is it that you would use it in the future for evaluating your company's cloud security?

12 svar

**Figure 7.12.** Answers about further use of RSCCI


On a scale from 1 to 10, how likely is it that you would recommend the service as a security evaluation tool to other companies?

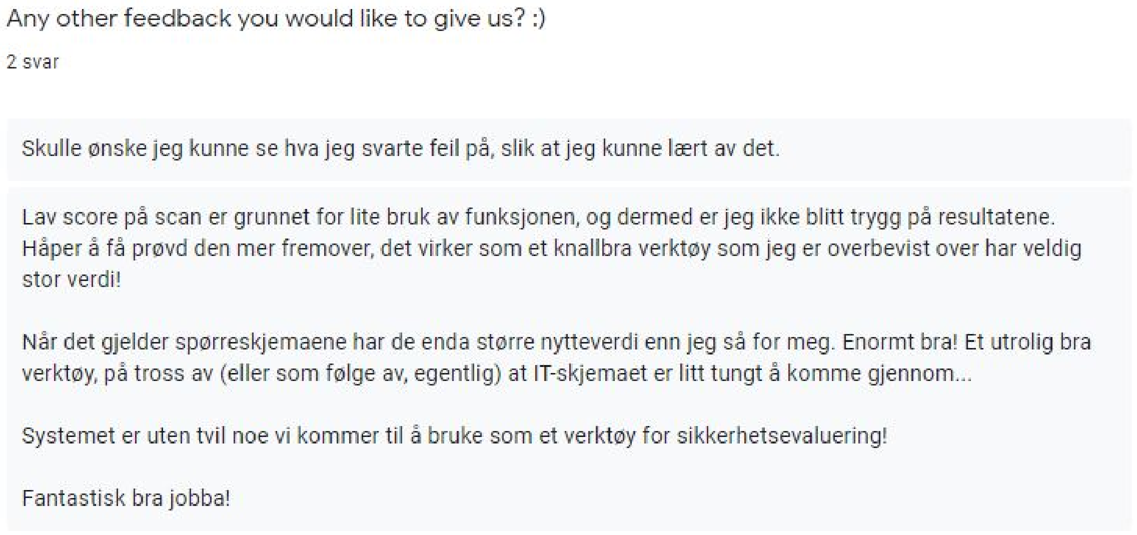
12 svar

**Figure 7.13.** Answers about recommending RSCCI to other companies

## Other

The last question we had on the user survey was an open question, in case anyone had any other feedback that could be useful to us. The two answers to this are shown in Figure 7.14. The first answer were from someone who took the Employee questionnaire, and he/she would like to be able to see what he/she answered incorrectly on. As mentioned earlier, the functionality that generates the Employee results was implemented with some shortcomings, so this is not surprising. The second answer says that RSCCI has a bigger utility value than he/she thought it would have. He/she also said that RSCCI is an incredibly good tool because the IT questionnaire takes a long time to complete.

As only two of Axeptia's employees answered this question, it is difficult to conclude what the overall feedback means about RSCCI. However, as the general rating on the other questions are high, it is possible that the lack of other feedback indicates that no one encountered any major problems while testing our web application.



Any other feedback you would like to give us? :)

2 svar

Skulle ønske jeg kunne se hva jeg svarte feil på, slik at jeg kunne lært av det.

Lav score på scan er grunnet for lite bruk av funksjonen, og dermed er jeg ikke blitt trygg på resultatene. Håper å få prøvd den mer fremover, det virker som et knallbra verktøy som jeg er overbevist over har veldig stor verdi!

Når det gjelder spørreskjemaene har de enda større nytteverdi enn jeg så for meg. Enormt bra! Et utrolig bra verktøy, på tross av (eller som følge av, egentlig) at IT-skjemaet er litt tungt å komme gjennom...

Systemet er uten tvil noe vi kommer til å bruke som et verktøy for sikkerhetsevaluering!

Fantastisk bra jobba!

**Figure 7.14.** Other feedback from Axeptia

### Summary

Axeptia seems really happy with RSCCI. Some of the answers varies more than others, but it is hard do conclude if the variations are anomalies or not since only 12 people tested RSCCI. On average, Axeptia's employees find the web application understandable and useful. They would also use RSCCI themselves, as well as recommend it to other companies, if it was released as a service. The feedback indicates that Axeptia is satisfied with the overall result of the project.

# Chapter 8

# Discussion

In this chapter we will discuss our approach to different parts and problems we encountered during the project period. We will also discuss some of the things that changed from our original project plan.

## 8.1 Approach

As stated in Chapter 3, Axeptia was very flexible on how the problem goal could be met. We therefore considered different approaches: an application that could be deployed directly on companies servers, an API that companies could integrate with, create the whole solution as Infrastructure as Code/Docker container that could be deployed in any environment, or a web application hosted by us. It has hopefully been clear that we chose to create a web application hosted by us.

The reason we chose this is because this is the easiest and most user friendly approach. Since we have a web application, potential users only have to read the information available on the web page instead of e.g. a README.md file in our Bitbucket repository. In addition, the ones using the service only need to register and does not have to think about the setup and maintenance of RSCCI. Doing it this way also makes the scanning easier to implement, as the scanner is placed outside a company network.

The decision to make a web application has impacted how the rest of the project was conducted. In the rest of this section we will discuss problems we encountered, user testing, design decisions, choices around the scanner and the report generation, as well as do a short risk assessment.

### 8.1.1 Problems

- **Existing bugs in used libraries**
  There was a bug in the JS library we used, Alpine.JS. This bug was caused by the JS library incorrectly handling element duplication. When changing the order of the questions on the administrator page, the moved elements would create a hidden duplicate. This meant that after reordering, when opening, closing or editing the questions, there were multiple elements with the same ID, which created multiple issues. There was no way to reorder the elements without duplication, or instructing the JS library to ignore the old duplicate versions. Our solution to this was forcing a reload whenever someone changed the order.

- **Jetstream**
  The Jetstream package comes with a lot of logic for the login page, registration page, mails, etc. This have come in handy for getting the web application up and running pretty quickly, but it also came with a lot of of unexpected problems. Most of the existing logic was either hidden or over-abstracted, spread over multiple interfaces, contracts, and actions. This was especially problematic when we wanted to modify the team invitation email. We used

multiple hours just to understand how all the different files worked, and ended up having to duplicate a lot of the functionality to be able to design our own email.

- **PDF generation**
  We spent almost 17 hours working on PDF generation, as we had some issues with generating a good looking and readable PDF report [70]. We tried multiple different libraries for PDF generation, but we almost always had issues related to keeping CSS styles and table structure in the PDF. The problems seemed to be that some of the libraries we used were too old to support the CSS styles we used. We eventually managed to do it with laravel-browsershot and the "showBackground" argument.

### 8.1.2 User Testing

We planned to have multiple companies test our web application, not only Axeptia. However, due to the time limit, we were not able to do so. As mentioned in Chapter 7, we did not get to test one of the functional requirements because of this. This is unfortunate, as we would have gathered useful feedback for the application. On the other hand, since we got the entire staff of Axeptia to test the solution, we do believe that we got enough feedback to do a thorough evaluation of the requirements.

The results from the survey Axeptia's employees answered has given us helpful feedback for the evaluation of RSCCI and thoughts about how the web application could be improved in the future. It has also been uplifting to read the feedback, as it is mostly positive and shows that what we have created measures up to Axeptia's expectations, and that RSCCI is something they would like to use as a cloud security evaluation tool in addition to recommend it to other companies.

### 8.1.3 Gamification

When it comes to gamification of the questionnaires, as also discussed with Axeptia and they pointed to it, it requires more expertise and experience to build such a platform. Neither of the group members with the developing responsibility had experience with design and implementation of game elements. As gamification is not directly relevant to our courses of studies and the project period had a strict time limitation, we decided not to spend time on learning the technologies needed to add game elements to the questionnaires.

So both Axeptia and us were satisfied that the project was done the way it is now as the questionnaires are granular and provide a more in depth cloud infrastructure security check than a gamification. However, the questionnaire for normal Employees is the closest we got to a gamification instance. All together, Axeptia and us agreed that the way to make the gamification models was through this granular models and to get there we just needed more time.

### 8.1.4 Design Decisions

Front end design has not been a big part of our project, as it is not directly relevant to our courses of studies. However, we have tried our best to keep the web application clean and user friendly by choosing bright colors, contrasts between background and text, and fun, relevant images.

### 8.1.5 Scanner

When performing scans we only allow scanning of one domain, because of problems with DNS servers. When doing a domain enumeration we continuously query multiple DNS servers, and sometimes a DNS server would respond with an IP for all domains queried, even though the domains were not valid. We do not know if this is because some of the DNS servers were malicious, or if it was some kind of anti DDoS feature.

We also implemented a queue for the scanner, which only allows one scan to be run at the same time. This is mostly because of the issues with DNS servers, and a lack of stress testing multiple scans at once. With the amount of users who currently use the application, only allowing one scan at a time is not an issue.

At the start of the project we planned to make our own port scanner. We discussed this with our supervisor, and she recommended to use already developed tools. Therefore, we changed our port scanner to the already existing tool Nmap. Before we changed the tool we had already used some time developing our own port scanner. Changing to Nmap saved time and allowed us to use our time more effectively.

### 8.1.6 Report Generation

Reports for users are generated every time a user loads the result page, adding a delay between 0.5 and 1 second. These reports could have been generated and stored after every questionnaire was completed, but we decided that storing the generated results added too much complexity, compared to the time saved. This was also the case for generating a PDF version of results, which took even longer to generate. It took about 3-4 seconds for a PDF to start downloading. To solve this, we could have created a Laravel queue to generate PDFs in the background. However, we did not have time to properly set up and test a queue, and properly store the PDFs.

### 8.1.7 Regulatory Concerns

To be able to handle users, we collect their first names, last names, and e-mail addresses. This is defined as personal data, and since we operate in Europe we have to follow GDPR. None of the data we store are defined as sensitive personal information. As the project time is limited, and this is more a proof of concept than a real service, we have not prioritized to document and follow GDPR to the letter. However, we do feel that we have sufficiently mitigated the known risks that might lead to a data breach.

A dynamic IP address is, in come cases, classified as personal data. RSCCI stores IP addresses, but due to lack of time, we have not configured NGINX to give us the real IP addresses from the Cloudflare reverse proxy. The IP addresses we store are therefore Cloudflares IP addresses, and are not connected to a logged in user.

### 8.1.8 Time Usage

Since we tracked our time using Clockify, we were able to see how much time we spent in total on the project, and how much time we spent on each project area. In total the group spent approximately 1355 hours working on the project that was divided into different areas. Figure 8.1 shows how our focus changed throughout the project period, from Planning the project in the beginning using the orange color, to doing research, working with development and compliance in the middle, and finally a complete focus on the report writing at the end using the green color.

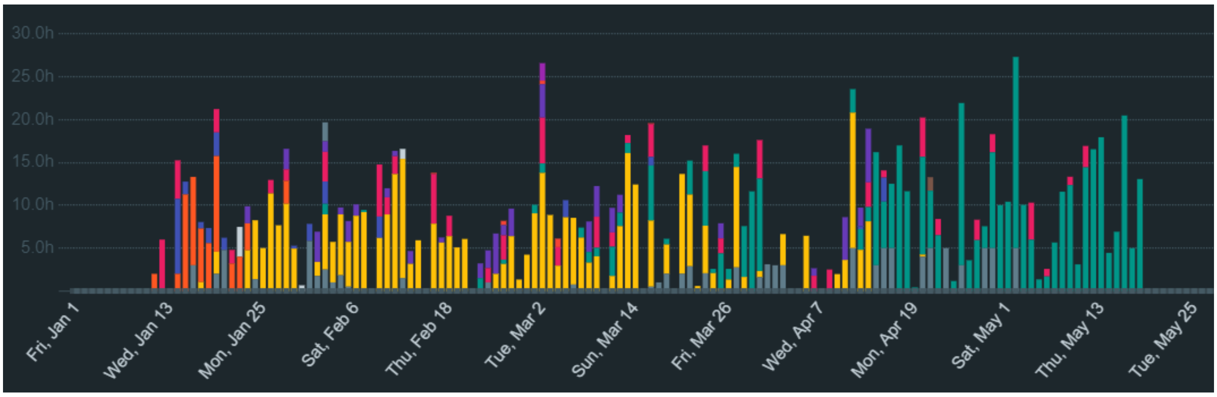


**Figure 8.1.** Figure shows the time tracking of the group work on the project until the current time of writing this thesis.

The project which started in January and ended at the end of the May, had a peak of over 300 hours of total group members' work in March. While total hours working on project in January was about 228 hours, in February about 268 hours, in April 317 hours, and about 245 hours in May. Among these the heaviest task have been "development and programming" with a peak of 480 hours which includes both developing the code for scanner and programming the application. This task has been active during January, February, March and April.

The second heaviest task in this project was writing the thesis report itself with a total hours of about 375 hours. This task has been active during March, April, and May with a peak of about 150 hours work in April. The two next biggest topics are "Regular meetings", which includes both meeting with supervisor, client and internal group meetings, with about 136 hours, and "Research and learning" with a total of about 143 hours.

In the fourth place is the task of "Planning the project" with about 82 hours and "Compliance" with about 62 hours. The task of "Planning the project" got accomplished during January, while for example "Regular meetings", "Compliance", and "Research and learning" has been going on all along. There has also been some miscellaneous tasks that together will not increase above 10 hours such as designing the logo of the web site etc.

Tracking the time during the project using Clockify has been a motivating factor for the group members. All group members have been contributing actively during the project to different tasks. However, as mentioned also in Section 1.7, each group member has been delegated specific responsibilities due to the tasks in the group so that each task has one who seeks for it getting done.

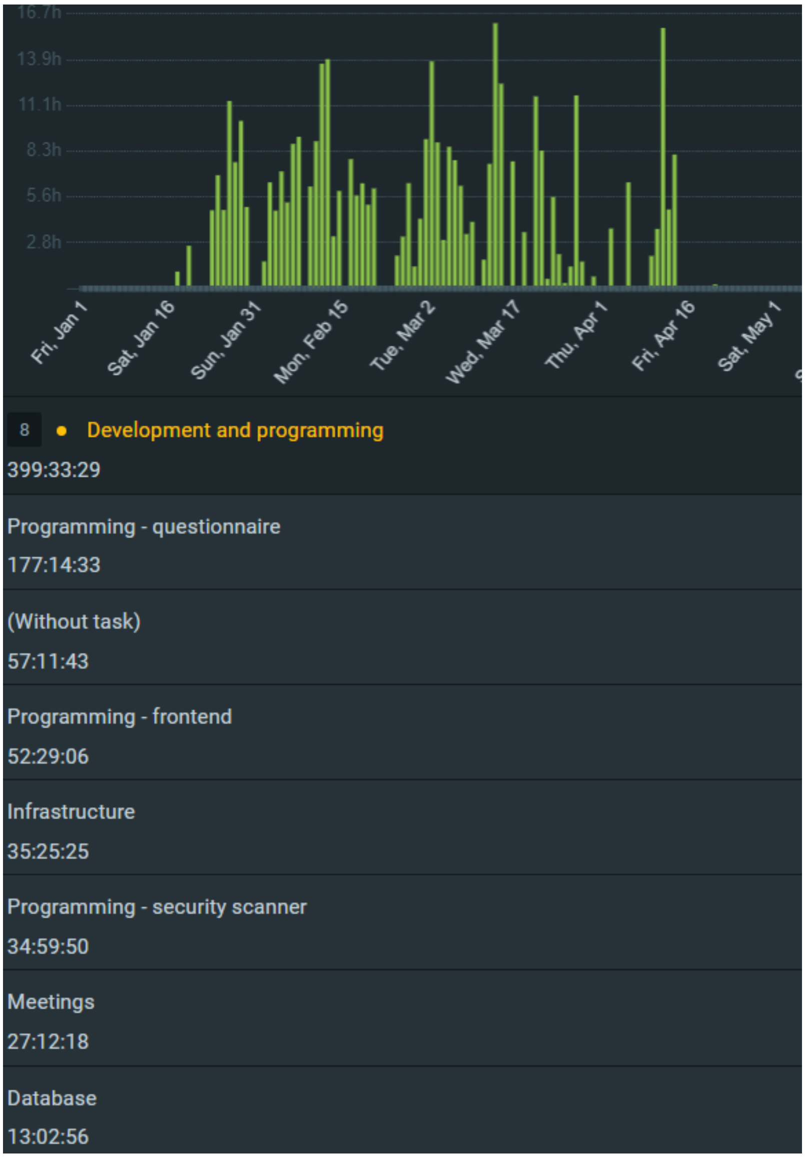
Since we sometimes simultaneously worked on different parts of the project, or a task belonged to multiple categories, the time spent on each category may not be completely accurate. For example, most of our time spent testing were logged as "Development and programming", and not as "Testing". When it comes to the "Development and programming" part, we divided our categories into different tasks as shown in Figure 8.2.

The "Programming - questionnaire" has been by far the most worked on task, and was mainly used for PHP/Laravel which includes back end of the web portal, RSC and CIS report generator. We believe that we have spent our time well and on the correct tasks. None of the team members with the developing responsibility had a lot of experience with front end development, so naturally this took some time to learn and get familiar with. We therefore spent a little more time on the front end than desired, keeping us from improving back end functionality. However, we are happy with both the front end and back end results.

## 8.2 Plan Changes

In this section we discuss changes incurred to the project due to the initial work plan. When it comes to project description, see Section 1.1, in the initial plan we had an understanding that there would be one questionnaire covering the whole areas of cloud infrastructure security. But studying ISO/IEC more closely we found out that this could be accomplished making separate questionnaire for each category of employees. This could be done with regard to the employees' responsibilities, participating in the questionnaires as for instance questions related to IT personnel could not be answered by HR personnel, and vice versa. We also made some research to find out if there is other standards regarding security controls of cloud infrastructure that we could utilize. NIST was one of the suggestions we got from the professors at the NTNU who consulted us.

We also tried to make shorter versions of the questionnaires which was not part of the plan from the start. But this also was one of the tasks that we, despite having a desire to do, would have to add to the "Future work list", see Section 9.3. Speaking about "Future work list", most of these works that are listed was initially not part of the plan we made at first. But through continuous work on the project we came to different and better ideas that could and was needed to make the it more efficient. Unfortunately, the time for this thesis work is limited and we only can make a list of them to have in mind in case we or Axeptia wish to expand and build more



● Development and programming

399:33:29

Programming - questionnaire

177:14:33

(Without task)

57:11:43

Programming - frontend

52:29:06

Infrastructure

35:25:25

Programming - security scanner

34:59:50

Meetings

27:12:18

Database

13:02:56

**Figure 8.2.** Time spent on different development tasks

on this project.

At the end of the development period we rewrote a lot of the code logic and functionality, so almost all of the original tests started failing. For most of these tests, fixing them meant rewriting the whole testing logic as well. Due to the complexity and time limit, we decided to drop rewriting the tests and rather focus on getting out the remaining functionality. As a result, we tested the web page behavior more thoroughly before releasing new features. Since we had no automatic tests to rely on.

Encryption between the scanner API and the web portal was not implemented. This is because enabling encryption on the API would require a reverse proxy, which we did not have time to properly set up and test. To minimize the risk of eavesdropping and man-in-the-middle attack, we used a VPC Network provided by DO for API communication between our two servers. Because the VPC Network is not reachable from the Internet or any other customers, we believe it to be sufficient to protect the API communication.

Regarding the vulnerability scanner, we originally planned to do a cloud specific vulnerability scan. This turned out to be rather challenging, considering the terms of service each of the cloud hosting providers have. To do a cloud specific vulnerability scan, we would need to have different tests for each of the most popular cloud providers such as Amazone AWS, Microsoft Azure, or Google Cloud. From this base, we changed the scanner approach, and made it to handle all machines the same, and do the same testing on them.

# Chapter 9

# Closing Remarks

This project has learnt us many new things both when it comes to technical aspects and when it comes to team work. We got to know and learned a lot from each other during the project. The project it self however had a huge capacity to expand and improving the quality of it in different aspects. As Per Nestor, the Co-founder & CEO at Axeptia, once in our weekly meeting said to us: "This project could expand to one more year as it had both the capacity and need to deepen and improve!". Here, we count briefly some of these aspects as well as some about what we have learnt during the project, the conclusion, and the works which will be postponed to the future.

## 9.1 Learning outcome

This section will cover what we have learned during the project period. It covers what we have generally learned about project work and the special topics needed to conduct the project. It covers what we have learned about project work and thesis writing, as well as what we learned when it comes to the more technical and security related aspects of the project.

### General

Not everyone in the project group had experience with creating a product from scratch. Being able to do so during the project period has been frustrating, time consuming, and demanding, but also exciting and fun. Working with the project has also given us a valuable experience we will get use of in the future.

### Regulatory Compliance

During the period of the project, we got the opportunity to work more in depth with different regulatory standards, such as ISO/IEC 27002, ISO/IEC 27017, and NIST 800-209, than we have done previously during the bachelor program. Using a variety of international standards have given us a deeper insight of the most common areas of security concerns.

### Cloud Computing and Cloud Security

As this project is about security in cloud infrastructure, we naturally learned a lot about cloud computing and cloud security. This knowledge will most likely be useful to us in the future, as more businesses choose to use cloud computing over on-premise infrastructure.

### Software Development

Software development has not been a big part of our bachelor studies, but the team members with the development responsibility had experience with it from leisure time and work life. However, we had little experience with the chosen language, PHP, and the different frameworks chosen for front end and back end. See Section 4.4 for an overview of the front end and back end technologies used during this project. These technologies was something we got to learn a lot about during the project period, both when it comes to front end, back end, and API endpoints.

### Infrastructure

Although we have experience with infrastructure from leisure time, work life, and our studies, we have learned more about it during the project period. For example, Mailgun and Mailtrap was unfamiliar to us, as well as the way the database was implemented with migrations. In addition, we learned how CI/CD can be implemented in Bitbucket and how to set up a Laravel production environment on a Linux server.

### Vulnerability Scanning

Writing a vulnerability scanner, which starts automatically scanning the provided addresses from a user, requires a lot of automation. All tasks which a user normally does in the Linux terminal for starting scanning tools, and filtering out the right result must be automated using a Python script. By doing this, we learned about script automation, and how to write code for filtering out the relevant information. We also learned to write a port scanner from the scratch even though this got replaced by an already built tool see Section 8.1.5.

### Problem Solving

During the project period, we have learned how to prioritize and break down problems, and ask for help when needed. Creating a product from scratch has not been a hassle-free process. However, problems that have occurred have been solved quickly either with the help of other team members, our supervisor Kelly, or our client Axeptia.

### Teamwork and Communication

When we worked with the project, we divided the project group into two teams: a technical team and a compliance/management/regulation team. The technical team was again divided into a web application team and a security scanner team. All the teams and sub-teams worked with different aspects of the final application that later was sewn together to one final product. Dividing the project group this way meant we needed to have a good communication both within and between the teams, especially since the ongoing COVID-19 pandemic has digitized the form of our collaboration. This has taught us a lot about teamwork and communication, as well as giving us a taste of the work life that is waiting for us when we finish our studies.

### Thesis Writing

We have spent a lot of time writing the bachelor thesis. This includes trying to format the document with LaTeX, find good sources to cite and keep track of them, create figures, tables and code listings, as well as the actually writing part. As many groups before us, we underestimated how comprehensive and time consuming this part of the project actually is. However, this has not stopped us from trying to produce the best possible thesis. With good help from Kelly's reviews and a thesis writing course, we have learned a lot about academic writing.

## 9.2 Conclusion

As stated earlier, the main goal for this project was to develop a regulatory security check customized for companies using the cloud. We met this goal by developing the user-friendly web application RSCCI. With one main component of RSCCI, RSC, the web application is able to map generic regulatory requirement, give companies useful insight into their regulatory compliance, and help them evaluate their cloud security. CIS gives a useful overview of how vulnerable an organization's infrastructure potentially is, as well as help them to discover subdomains which they might have overlooked.

During our research in cloud infrastructure, we learned that even though we thought a cloud service provider like Azure would require a completely different set of tests and scanner, most of it was the same. Whether an application is hosted in someones garage, on-premise or in a public data center, it still is hosted on a server, and the application has the same regulatory requirements. Our scanner reflects this, as scanning a VM in Azure is the same as scanning a VM from any other provider. We can therefore say that using cloud infrastructure to host an application is more secure than other options, as cloud service providers are usually large organizations that spend vast amount of money on securing their cloud services. IaaS will be at least as secure as on-premise, while PaaS and SaaS will be more secure than hosting the services on-premise.

We have tested the requirements to ensure RSCCI works as intended. The functionality of the entire web application has been tested by both us and Axeptia to ensure that everything works as desired. Axeptia's feedback indicates the information from the web application is useful and understandable. This applies to the questionnaires, scans, and results. In addition, they had very little issues using our web application, and were generally happy with the result and cooperation.

## 9.3 Further Work

As also mentioned in Section 8.2 during the project we faced many topics that could make this project more effective and at the same time attractive for our now and future clients. Due to the constraint of time we had to keep our efforts inside the specified scope of this project to be able to finish this thesis in time. Even so, we have made a list of the work we wish to do to make this project more functional and productive. This list is as following:

**Regarding the questionnaire:**

- The project description talks about a high expectation and importance of cloud security checks with standards and at the same time it comes with the point that a gamification version of the application is desired but not compulsorily, see Section 1.1. We accomplished a more granular version of questionnaire, however we did not reach to make the short version of it to have also the gamification in place. The both cannot be accomplished in one version, as a gamification will not be sufficient for a regulatory security check, and the two can not be done in one place.

  In future what we would like to do is dividing each questionnaire in sub-parts, in groups of 15-20 questions per group, as it will be easier for the participants to fulfil each part and start another one when suitable for them. Due to the time we did not reach to do this, as this part required a complete task in programming of this project which falls out of the scope and capacity of time available for this thesis. However, we have tried to make a temporary solution to this problem as the participant can go out of each questionnaire and come back to it at any time so each questionnaire will not require fulfilling it at once. Also the user friendly front-end of RSCCI is a close step to make the gamification version of the application.

  As mentioned before, another idea was to make short gaming versions of the questionnaires so that participants could have fun with it while also learning something. This was also one of the suggestions that our client Axeptia had in the project description as it mentions: "the solution should also map relevant generic safety requirements through obtaining information via a questionnaire, which could be like a type of gamification such as Kahoot, to illuminate how the company itself views its security architecture and what requirements it should meet", see Section 1.1. This task is left to the future works, due to the time constraints.

- Developing guide links such as "Read More" for each recommendation in result page which navigates to the best practice for each control recommended by the standard. This helps to guide about how the organization may increase its security on different areas. The aim of this is to provide the user more in depth information about how to improve the specific problem area of security. This task was also mentioned in project description by Axeptia which we did not reach to do due to the time limitations, see Section 1.1.

- At the time we have developed questionnaire for normal employees which shows how is their understanding of IT security and its concepts. However, the constraint of time did not let us finish this task as we was going to make a summary report of all employees of an organization, who took the test, for the management of the organization. This would have shown a summary of all organization's employees' understanding and readiness due to IT security concepts and principles.

  In addition to this, we would like to improve the report generator for the Employee questionnaire. Due to lack of time, the results are insufficient and only shows how many correct and wrong answers a user got without any more information about which answers were wrong. It is important for us to implement this as it also is a part of the feedback we got from Axeptia.
- Making the questionnaire, especially the one for normal employees, "mobile friendly" so that they can take the test on their mobile as well as their iPad or PC's. As this application is supposed to work almost same as Kahoot when it comes to be interesting and offering possibility for competition.
- Another work to do is to correct all small mistakes in the questionnaire as we did not reach to do during this time. These are such as some places in the questionnaire using the word "survey" instead of "questionnaire" as it was not clear for us from the beginning that we will choose to call this for questionnaire. Also making each questionnaire, specially the one for normal employees, more effective. The questions under the topic of general security need to be processed and remade to get more suited in the way we wish them to be.
- Refurbishment of the questions making them shorter, more effective, and more general while at the same time hit more precise security control checking.

**Regarding functionalities:**

- Implementing Continuous Delivery functionality for the pipeline to automatically deploy code to our development environment.
- Implementing possibility to join an organization by their Globally Unique Identifier (GUID) instead of being invited. The idea is to create each organization with a GUID, so that a user could join the organization by putting this GUID in instead of having to be invited by someone else.
- When verifying an email-domain for a user that is making a registration, check that the users email is not disposable in addition to checking with the blacklisted email-domains.
- Create a demo site so people can see how the questionnaire and results works without creating a user.
- Developing processes which periodically delete users' private data saved in the database of RSCCI to be in compliance with GDPR rules. These processes will ensure that the user's private data will not be stored in the application more than necessary. To be in full compliance with GDPR and to protect Personally identifiable information (PII), if any is registered in this application, we will develop processes to have the consent of the users [71].

**Regarding the scanner:**

- Implementing more scanning tools, to further improve the detection rate of the vulnerability scanner. We have tested multiple other tools, and tried to automate these. Some tools are really only made to be used manually, and filtering the interesting output of the results from these tools are very time consuming, which means we did not have enough time to implement them.

- Limiting the users to only be able to scan their own domains. The way the website is set up now is that an organization can easily use the scanner to scan other organizations' machines. However, it is impossible for the scanner to know who owns a random IP address. The only way we could have restricted the users, is to only let them scan web domains with the same organization name in the domain.
- Our original plan was to do more specific scans on cloud specific vulnerabilities. Though, after some testings, we discovered that this was impossible to implement within the time frame we had as it required different testings for each cloud platform. Some of these cloud specific testings also crosses some of the CSPs rules. What we wish to do about this in the future is to make more specific scans on cloud specific vulnerabilities.
- Before scanning an IP, checking if it is not a Private IP, as it is not possible to scan a company's internal network from the outside with a Private IP.
- When listing vulnerabilities, try to also give recommendations for how a user can mitigate each vulnerability.

**Regarding the report:**

- In retrospect, we would have liked to do a more complete risk assessment of RSCCI platform. Due to the lack of time we did not reach to add this part completely to our report. We think this might be good and will give a better weight to our report.

# Bibliography

[1] Axeptia, *Get paid, faster.* https://www.axeptia.com/, (Downloaded 29/04/2021).

[2] E. D. Holm, *Get paid, faster.* https://www.dnb.no/dnbnyheter/no/grunder/vil-fornye-kreditt-og-inkassobransjen, (Downloaded 29/04/2021), Jun. 2018.

[3] K. Johnson, *What are the risks of a penetration test?* https://secureideas.com/knowledge/what-are-the-risks-of-a-penetration-test, (Downloaded 11/05/2021), Aug. 2019.

[4] W. contributors, *Rate limiting*, https://en.wikipedia.org/w/index.php?title=Rate_limiting&oldid=999977463, (Downloaded 24/04/2021), Jan. 2021.

[5] T. G. Peter Mell, *The nist definition of cloud computing*, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf, (Downloaded 12/04/2021), Sep. 2011.

[6] S. J. Bigelow, *Xaas (anything as a service)*, https://searchcloudcomputing.techtarget.com/definition/XaaS-anything-as-a-service, (Downloaded 25/04/2021), Nov. 2017.

[7] IBM, *Iaas (infrastructure-as-a-service)*, https://www.ibm.com/cloud/learn/iaas, (Downloaded 08/03/2021), Jul. 2019.

[8] Microsoft, *What is cloud computing?* https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/#cloud-computing-models, (Downloaded 23/02/2021).

[9] C. S. I. G. P. S. S. Council, *Pci data security standard (pci dss)*, https://www.pcisecuritystandards.org/pdfs/PCI_DSS_v2_Cloud_Guidelines.pdf, (Downloaded 18/03/2021), Feb. 2013.

[10] IBM, *Paas (platform-as-a-service)*, https://www.ibm.com/cloud/learn/paas, (Downloaded 08/03/2021), Oct. 2019.

[11] Cloudflare, *What is saas? | saas definition*, https://www.cloudflare.com/learning/cloud/what-is-saas/, (Downloaded 09/03/2021).

[12] Citrix, *Citrix homepage*, https://www.citrix.com/, (Downloaded 17/04/2021).

[13] Citrix, *What is a public cloud?* https://www.citrix.com/no-no/glossary/what-is-public-cloud.html, (Downloaded 01/03/2021).

[14] Citrix, *What is a private cloud?* https://www.citrix.com/no-no/glossary/what-is-private-cloud.html, (Downloaded 01/03/2021).

[15] Citrix, *What is a hybrid cloud?* https://www.citrix.com/no-no/glossary/what-is-hybrid-cloud.html, (Downloaded 01/03/2021).

[16] W. contributors, *Vulnerability (computing)*, https://en.wikipedia.org/wiki/Vulnerability_(computing), (Downloaded 15/05/2021), May 2021.

[17] Wikipedia contributors, *Port scanner*, https://en.wikipedia.org/w/index.php?title=Port_scanner&oldid=1020597951, (Downloaded 15/05/2021), May 2021.

[18] redhat.com, *What is a cve?* https://www.redhat.com/en/topics/security/what-is-cve, (Downloaded 15/05/2021).

[19] National Institute of Standards and Technology(NIST), *Vulnerability metrics*, https://nvd.nist.gov/vuln-metrics/cvss, (Downloaded 15/05/2021), Dec. 2017.

[20] Bernard Brode, *7 cyber threat actors to watch for in 2021*, https://www.securityinfowatch.com/cybersecurity/article/21207268/7-cyber-threat-actors-to-watch-for-in-2021, (Downloaded 15/05/2021), Jan. 2021.

[21] Wikipedia contributors, *Hacktivism*, https://en.wikipedia.org/wiki/Hacktivism#Notable_hacktivist_peoples/groups, (Downloaded 16/05/2021), May 2021.

[22] W. contributors, *Regulatory compliance*, https://en.wikipedia.org/w/index.php?title=Regulatory_compliance&oldid=1020486750, (Downloaded 30/04/2021), Apr. 2021.

[23] I. O. for Standardization(ISO), *Information technology security techniques code of practice for information security controls based on iso/iec 27002 for cloud services*, https://www.standard.no/nettbutikk/sokeresultater/?search=ISO+27017&subscr=1, (Downloaded 23/03/2021), Jan. 2021.

[24] I. E. C. International Organization for Standardization, *Information technology security techniques code of practice for information security controls*, https://www.standard.no/nettbutikk/sokeresultater/?search=ISO+27002, (Downloaded 23/03/2021), 2017.

[25] D. P. Ramaswamy Chandramouli, *Security guidelines for storage infrastructure*, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-209.pdf, (Downloaded 23/03/2021), Oct. 2020.

[26] I. E. C. International Organization for Standardization, *Information technology - security techniques - information security management systems - requirements*, https://www.standard.no/nettbutikk/sokeresultater/?search=27001&subscr=1, (Downloaded 25/03/2021), 2013.

[27] NIST, *Accountability*, https://csrc.nist.gov/glossary/term/accountability, (Downloaded 01/05/2021).

[28] J. Nielsen, *Website response times*, https://www.nngroup.com/articles/website-response-times/, (Downloaded 18/03/2021), Jun. 2010.

[29] G. developers, *Lighthouse performance scoring*, https://web.dev/performance-scoring/#color-coding, (Downloaded 25/04/2021), Feb. 2021.

[30] I. Sommerville, *Software Engineering*, 10th ed. Pearson Education Limited, 2016, ch. 6 Architectural design, pp. 176–177.

[31] Wikipedia, *Model–view–controller*, https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, (Downloaded 11/04/2021).

[32] A. Kumar, *The 10 best programming languages for hacking*, https://www.fosslinux.com/40111/the-10-best-programming-languages-for-hacking.htm, (Downloaded 07/05/2021), May 2020.

[33] M. Hasan, *The 15 best programming languages for hacking (ethical hacking)*, https://www.ubuntupit.com/best-programming-languages-for-hacking/, (Downloaded 07/05/2021), Apr. 2021.

[34] M. Gelbmann, *40% of the web uses wordpress*, https://w3techs.com/blog/entry/40_percent_of_the_web_uses_wordpress, (Downloaded 21/04/2021), Feb. 2021.

[35] Laravel, *Meet laravel*, https://laravel.com/docs/8.x#meet-laravel, (Downloaded 01/05/2021), Apr. 2021.

[36] Laravel, *Laravel fortify*, https://laravel.com/docs/8.x/fortify, (Downloaded 05/05/2021).

[37] DigitalOcean, *How to install linux, nginx, mysql, php (lemp stack) on ubuntu 18.04*, https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-ubuntu-18-04, (Downloaded 01/05/2021), Mar. 2018.

[38] Linux, *What is linux?* https://www.linux.com/what-is-linux/, (Downloaded 01/05/2021).

[39] Nginx, *What is nginx?* https://www.nginx.com/resources/glossary/nginx/, (Downloaded 01/05/2021).

[40] Oracle, *1.2.1 what is mysql?* https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, (Downloaded 01/05/2021).

[41] P. S. Foundation, *About python*, https://www.python.org/about/, (Downloaded 01/05/2021).

[42] Docker, *What is a container*, https://www.docker.com/resources/what-container, (Downloaded 01/05/2021).

[43] Laravel, *Blade templates*, https://laravel.com/docs/8.x/blade, (Downloaded 01/05/2021), Apr. 2021.

[44] C. Porzio, *Livewire*, https://laravel-livewire.com/, (Downloaded 22/02/2021), Feb. 2021.

[45] W. contributors, *Universal design*, https://en.wikipedia.org/w/index.php?title=Universal_design&oldid=1017733026, (Downloaded 15/04/2021), Apr. 2021.

[46] V. Rogers, *12 questionnaire design tips for successful surveys*, https://www.snapsurveys.com/blog/12-questionnaire-design-tips-creating-successful-surveys/, (Downloaded 16/03/2021).

[47] S. Survey, *Questionnaire design*, https://www.smartsurvey.co.uk/questionnaire-design, (Downloaded 16/03/2021).

[48] Atlassian, *What is kanban?* https://www.atlassian.com/agile/kanban, (Downloaded 12/04/2021).

[49] Atlassian, *What is scrum?* https://www.atlassian.com/agile/scrum, (Downloaded 12/04/2021).

[50] Atlassian, *The #1 software development tool used by agile teams*, https://www.atlassian.com/software/jira, (Downloaded 13/04/2021).

[51] Atlassian, *Built for professional teams*, https://bitbucket.org/product/, (Downloaded 01/05/2021).

[52] Atlassian, *Accomplish more together*, https://www.atlassian.com/software/confluence, (Downloaded 01/05/2021).

[53] S. Pittet, *Continuous integration vs. continuous delivery vs. continuous deployment*, https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment, (Downloaded 20/04/2021).

[54] Wikipedia, *Entity–relationship model*, https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model, (Downloaded 11/04/2021).

[55] Laravel, *Database: Getting started*, https://laravel.com/docs/8.x/database, (Downloaded 21/04/2021).

[56] Laravel, *Database: Query builder*, https://laravel.com/docs/8.x/queries, (Downloaded 09/05/2021).

[57] Laravel, *Blade templates*, https://laravel.com/docs/8.x/blade, (Downloaded 08/05/2021).

[58] W3Resources, *Mysql transaction*, https://www.w3resource.com/mysql/mysql-transaction.php, (Downloaded 08/05/2021), Feb. 2020.

[59] OWASP Foundation contributors, *Owasp top ten*, https://owasp.org/www-project-top-ten/, (Downloaded 16/05/2021), Apr. 2021.

[60] Michael Vizard, *Survey finds breach discovery takes an average 197 days*, https://securityboulevard.com/2018/07/survey-finds-breach-discovery-takes-an-average-197-days/, (Downloaded 16/05/2021), Jun. 2018.

[61] Somdev Sangwan, *Xsstrike*, https://github.com/s0md3v/XSStrike, (Downloaded 16/05/2021), Dec. 2019.

[62] Mailtrap, *Email sandbox service*, https://mailtrap.io/, (Downloaded 03/05/2021).

[63] Laravel, *Mail*, https://laravel.com/docs/5.1/mail, (Downloaded 03/05/2021).

[64] Mailtrap, *Choose a plan that suits your needs*, https://mailtrap.io/pricing/, (Downloaded 09/05/2021).

[65] DigitalOcean, *Vpc*, https://docs.digitalocean.com/products/networking/vpc/, (Downloaded 09/05/2021), May 2021.

[66] Mailgun, *The email service for developers*, https://www.mailgun.com/, (Downloaded 03/05/2021).

[67] Laravel, *Authentication*, https://laravel.com/docs/8.x/authentication#login-throttling, (Downloaded 04/05/2021).

[68] OWASP, *Deserialization cheat sheet*, https://cheatsheetseries.owasp.org/cheatsheets/Deserialization_Cheat_Sheet.html, (Downloaded 07/05/2021).

[69] DigitalOcean, *Droplet policies*, https://docs.digitalocean.com/products/droplets/resources/policies/, (Downloaded 09/05/2021), Mar. 2021.

[70] Clockify, *Rscci clockify summary report*, https://clockify.me/shared/6097bb6132b3397d3aa13602, (Downloaded 09/05/2021).

[71] Datatilsynet, *Personalmappe - hva lagres, hvor lenge og hvorfor?* https://www.datatilsynet.no/personvern-pa-ulike-omrader/personvern-pa-arbeidsplassen/personalmappe/, (Downloaded 16/05/2021), Jul. 2019.

# Appendix

## A Task Description

## Oppdragsgiver Axeptia

# Regulatorisk sikkerhetssjekk for skybasert infrastruktur

De aller fleste selskaper er avhengig av å legge om sin arkitektur til å bruke mer skybasert infrastruktur for å fortsette å være lønnsomme og kunne være konkurransedyktige. Det er ofte en myte at såkalt on-premise infrastruktur er sikrere enn skybasert infrastruktur. I dag er det vanlig at det stilles strengere krav til sikkerhet hos selskaper som bruker skybaserte tjenester, enn for de som bruker on-premise infrastruktur.

## Oppgave

For å forenkle sikkerhetsoppgaven for selskaper som benytter seg av skybaserte tjenester/infrastruktur, skal det utvikles en semi-automatisert regulatorisk sikkerhetssjekk som kan gi en oversikt over hvordan selskapet  oppfyller eksterne krav til sikkerheten i sin arkitektur og infrastruktur. Løsningen må kartlegge relevante generiske sikkerhetskrav gjennom innhenting av informasjon via et spørreskjema (gjerne med bruk av gamification, feks Kahoot!) som belyser hvordan selskapet selv ser på sin sikkerhetsarkitektur og hva den burde oppfylle av krav. Ut fra selskapets innmelding (gjerne flere fra ledelsen i selskapet/ressurser som ikke daglig jobber med sikkerhet) må løsningen automatisert generere en rapport som belyser hvilke krav som helt eller delvis ikke er oppfylt, og gi råd (eller lenke til råd) for hvordan selskapet kan bedre sikkerheten for det enkelte punkt.

Løsningen må plasseres i en skybasert hosting og kunne integreres med plattformen som velges for XXX (både AWS og Azure tilbyr gratis hosting til studenter).

Løsningen må anbefale en løsning, teknisk eller gjennom prosedyrer, som sikrer at kun den virksomheten som står for utfyllingen av spørreskjema og ansvarlig for tjenesten (XXX) får tilgang til resultatene.

Løsningen bør også kunne støtte opp under selskaper som har outsourcet sin IKT-virksomhet, slik at også de blir klar over hvilke krav som de fortsatt må oppfylle, og hvilke krav de må stille overfor sin leverandør.

Løsning må selvsagt overholde dagens retningslinjer for informasjonssikkerhet.

Kontaktpersoner

Per Nestor Warp, per.nestor.warp@axeptia.com

Håkon Viervoll, hakon.viervoll@axeptia.com

# B Project Agreement

NTNU

Vår dato Vår referanse

**Norges teknisk-naturvitenskapelige universitet**

# Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

**Axeptia Credit Intelligence AS**________________________________________________

________________________________________________ (oppdragsgiver), og

**Øyvor Vejlgaard Sørensen, Kristian Nicolai Hjelmtveit Fjelstad, Harald Aarseth og Gloria Treider**__________

________________________________________________

________________________________________________ (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra __**11.01.2021**__ til__**20.05.2021**__.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle beståtte bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): **Jia-Chun Lin**

Oppdragsgivers kontaktperson (navn): **Håkon Viervoll**

Student(er) (signatur): Gloria Treider dato 12.01.2021

Harald Aarseth dato 12.01.2021

Øyvor V. Sørensen dato 12.01.2021

Kristian Fjelstad dato 12.01.2021

Oppdragsgiver (signatur): dato 14.01.2021

*Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.*
*Godkjennes digitalt av instituttleder/faggruppeleder.*

*Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.*
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): ______________________ dato ________

## C Project Plan

# Regulator Security Check for Cloud Infrastructure Bachelor Thesis Project Plan

Øyvor Vejlgaard Sørensen - 505059 - BITSEC
Harald Aarseth - 505089 - BITSEC
Kristian Fjelstad - 509256 - BIDAT
Gloria Treider - 505124 - BITSEC

January 2021

## Contents

# 1 Project information

Title: Regulator Security Check for Cloud Infrastructure
Projectnr: 11
Participants: Øyvor Vejlgaard Sørensen (BITSEC), Harald Aarseth (BITSEC), Kristian Fjelstad (BIDAT), Gloria Treider (BITSEC)
Project owner: Axeptia
Supervisor: Jia-Chun Lin

# 2 Goals and limits

## 2.1 Background

More and more companies today migrate to the cloud to become more cost-efficient and be more competitive in their industry. A normal myth among consumers and customers is that on-premise infrastructure is more secure than the cloud, as the companies then has more control over their servers. Companies using cloud-based infrastructure also has higher requirements for security than similar companies hosting on-premise.

Axeptia Credit Intelligence AS, from here just Axeptia, is a relatively small company with technical and commercial domain experience from the credit information, debt collection, and the software industry. They offer credit management software trying to help businesses to reduce outstanding invoices and reduce credit risk. With their solution, you get full information about accounts receivables, credit info, and debt collection.[1]

As mentioned above, more and more companies migrate to the cloud, and Axeptia is no exception. With a growing customer base, there is a need to know how they are doing in regards to security and regulatory compliance grows as well. Therefore, to simplify their work and other companies work to have an overview of their security status on a regular basis, Axeptia wants a semi-automated regulatory security solution customized for companies using the cloud.

## 2.2 Project goals

The project is in theory twofold, although the two parts will be integrated into one final product.

The first part, and goal, of the project is to create regulatory security check application presented as a web-based questionnaire. It will gather answers from a company about its security landscape and compares the answers with compliance regulations related to information security.

The second goal is to develop a cloud infrastructure scanner that will scan the company's cloud infrastructure for known security flaws and vulnerabilities.

Lastly, as a part of both goals, it is to be generated reports for both the regulatory security check and the cloud scanner to show an overview of the security of the company with links to information about how to improve its security.

# 3 Scope

## 3.1 Subject area

The subject area for the bachelor thesis will primarily focus on different topics related to cloud-based infrastructure security. We will, among others, cover the following subjects:

- Security in cloud-based infrastructure
  - The project and thesis will be based around security principles in cloud-based infrastructure
- Regulatory compliance
  - All companies have standards, laws, policies, or specification that they need to or should follow regarding information security. Companies using cloud are no exception, and this is therefore an important part of the thesis.
- Security scanning for different XaaS
  - A cloud-based infrastructure can come in many shapes and forms, since you can use the cloud in multiple ways. A cloud service provider may offer services in the following XaaS: SaaS, PaaS, and IaaS. We will research how to scan the different types of XaaS, and, if research shows it possible, implement functionality to scan each of them.

- Platform development
  - Our final product will be a functional platform for businesses to get an insight about their security.
  - It is important that our platform is developed in a secure manner. We will therefor have high focus on secure development during the project.

## 3.2 Task description

The project consists of two tasks; developing a regulatory security check application and creating a cloud infrastructure scanner. The solution to the first task will use a web-based questionnaire to gather information about how a company looks at its own security, regulatory compliance, and what security requirements it should fulfill. From the questionnaire, supported by the cloud infrastructure scanner, the solution should generate reports showing what requirements are fulfilled completely, partial, or not at all, in addition to giving advice (or links to advice) about how the company may increase its security on different areas.

Companies that has their IT business and cloud managed by another company should be able to use the regulatory security check and the cloud infrastructure scanner, mostly to make themselves aware of which requirements they should still fulfill and what requirements they should set for their IT/cloud suppliers.

## 3.3 Limitations

### 3.3.1 Service provider

There are a lot of cloud service providers to choose from, and generating a security scanner for all of them may be too much for a bachelor thesis with a duration of about four months. In cooperation with Axeptia, we have chosen to prioritize the Microsoft Azure cloud computing and infrastructure platform, as this is what they use for their cloud infrastructure. This said, we will still conduct research and implementation of a cloud security scanner for other service providers, but the Azure implementation will be prioritized before others if time is limited.

On-premise infrastructure security check will not be covered in the thesis.

Since the companies using cloud infrastructure is not the owner of the physical computers and servers they use, we also have to limit the scope of the active scan, as for example Microsoft has strict, defined rules of engagement when it comes to penetration testing. [2]

### 3.3.2 Penetration test

The active test, also referred to as a scan, should not be considered a penetration test, as it will only do the scanning and enumeration part of a full penetration test. The gathering of IP-addresses and endpoints for this part of the test will be the users responsibility.

### 3.3.3 The generated user report

The final report generated by the solution should not give a score that could wrongfully give an impression about the security in the company. This means that the report should not say ”Company X scored 10/100 points”, but rather give an indication on areas where the answers and scan shows they have good control and which areas where they have less insight about their implemented security features.

### 3.3.4 Covid-19

Due to the ongoing covid-19 pandemic, the abilities to have meetings and work sessions physical is limited, but not impossible. However, we have all experience with working in groups using digital media, and believe that it will not affect the teamwork, productivity, or the final report.

# 4 Project organization

## 4.1 Roles and responsibilities

For the group roles, and responsibilities, see appendices A.

## 4.2 Policies and routines

### 4.2.1 Workflow

The group uses Discord for messages and voice call meetings. Discord is convenient, and everyone in the group are familiarized on how to use it. Within privacy concerns, Discord does collect a lot of data about its users and messages, however the convenience of the platform, and no strict guidelines from Axeptia makes us continue using it.

For time tracking, we use Clockify. This allows us to see a estimate of the amount of hours we put into the project. It is possible to connect Clockify to Jira, so that we can track how much time each task, and issue takes.

We will use Overleaf with Latex to write our report. This allows us all to write at the same document, at the same time, without having merge issues. To share files, we use a shared OneDrive folder. In order to communicate with our supervisor Kelly, we use Microsoft Teams to meet, we will also use Teams to have scheduled meetings with Axeptia. Trello is used for having a Kanban board tracking our progress within tasks. All of our code will be hosted on Bitbucket, to allow us to have git version control. We have chosen Bitbucket since this is what we have been using in other NTNU projects. Confluence is used to store documentation for ourselves, and notes/plans done for development/programming. Jira is used for tracking and issues and development progress.

**Coding languages** To develop our intended vulnerability scanner platform, and questionnaire servers, we will use some different coding languages, which not all in the group will be as familiarized to as others. For the pentest side, Python will be mostly used, since this is arguably the easiest way to make large scripting files for Linux. We will probably also use some bash scripts, and containerize the code inside a Linux machine using Docker.

**Git workflow** Figure 1 describes our git workflow

- All commits shall have a professional message in English that briefly explains what has been done with the modified file(s). Example: "Added logging of database queries".
- A contributer is not allowed to merge his or her changes to master without another reviewers approval.
- If two or more contributers has worked together on a change (pair programming), another contributer does not need to review the changes.
- All changes has to be relevant to the project goal, and should have a task/issue/trello-card connected to it.
- When possible, one or more unit-tests should be created when adding new functionality to the solution.
- The group members shall remember to push up their code often, so that the others can see what have been done.

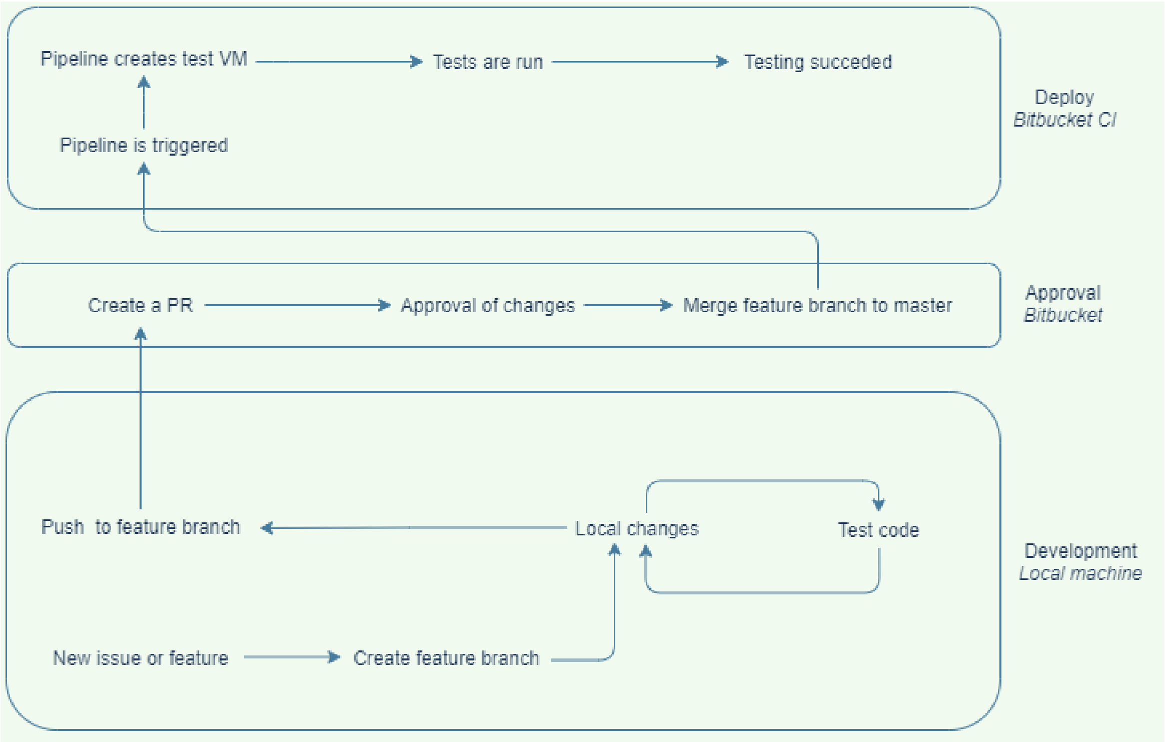


Figure 1: Git workflow

# 5 Planning and reporting

## 5.1 Main project sections

### 5.1.1 Development model

The development model we have chosen to use for this project is kanban[3], as it seems to fit our needs best. Kanban allows us to split the different stages and components of the project into Jira issues. We will then use Jira to track the issues, and monitor the progress and eventual problems of our project.

### 5.1.2 Method and Approach

Figure 2 shows the desired system architecture for the final product.

Explanation for technical side of how a security check will work using figure 2 as a reference:

- The user logs into the web-page and chooses to either start the cloud infrastructure scanner or the regulatory security check.
- The regulatory security check gets the questions from the database, presents them to the user and sends the answers back to the database.
- The cloud infrastructure scanner asks for the desired endpoints and domains to check, scans them, and analyzes the results.
- The report generators creates the corresponding reports, based on the regulatory security check and scan results, so that a user with low knowledge of information security is able to understand it. It will most likely be two independent reports, one for scan and one for questionnaire.
- The reports will be presented to the user on the web page, with a possibility to download a PDF version of the files.

Figure 3 shows how the user is supposed to experience the web interface.

Figures 4 and 5 shows low-level sequence charts of system functionality for the two components.

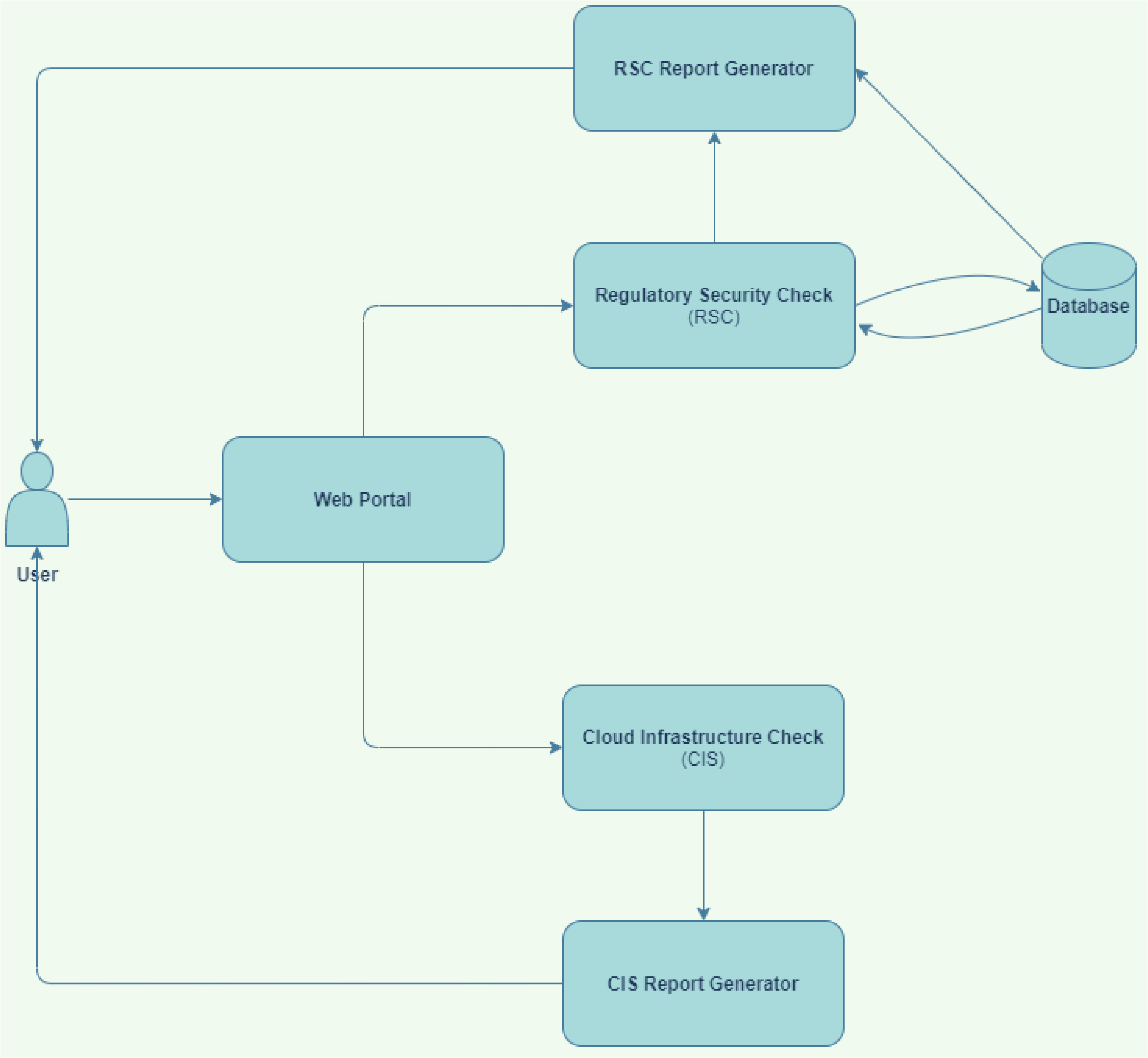

RSC Report Generator
Regulatory Security Check (RSC)
Database
User
Web Portal
Cloud Infrastructure Check (CIS)
CIS Report Generator

Figure 2: Proposed system architecture

## 5.2 Planning of project status meetings

The group will gather online and meet every Tuesday at 1000-1100. In this meeting we will agree on other meeting hours and the report we produce for the supervisors information. We will also take a look on the tasks that should have been done until the meeting time and assign new tasks to group members.
After this meeting we will send an informative report to the supervisor about our progress and next step plan. This report will let the supervisor to help us better in the coming meeting per week as there will be a meeting every Thursday with the supervisor at 1200-1230. In this meeting we inform the supervisor about the tasks that are done during the past week and the tasks that will be done during the next week.
The supervisor will question and guide us as she (Kelly) find it necessary. She will also make sure that we follow the progress plan we have set up from the start of the semester.
As we are going to produce the final report during the project progresses there will be meetings to work on the report, and meetings for development of the application and the project it self.
The meeting with Axeptia will take place every other week.

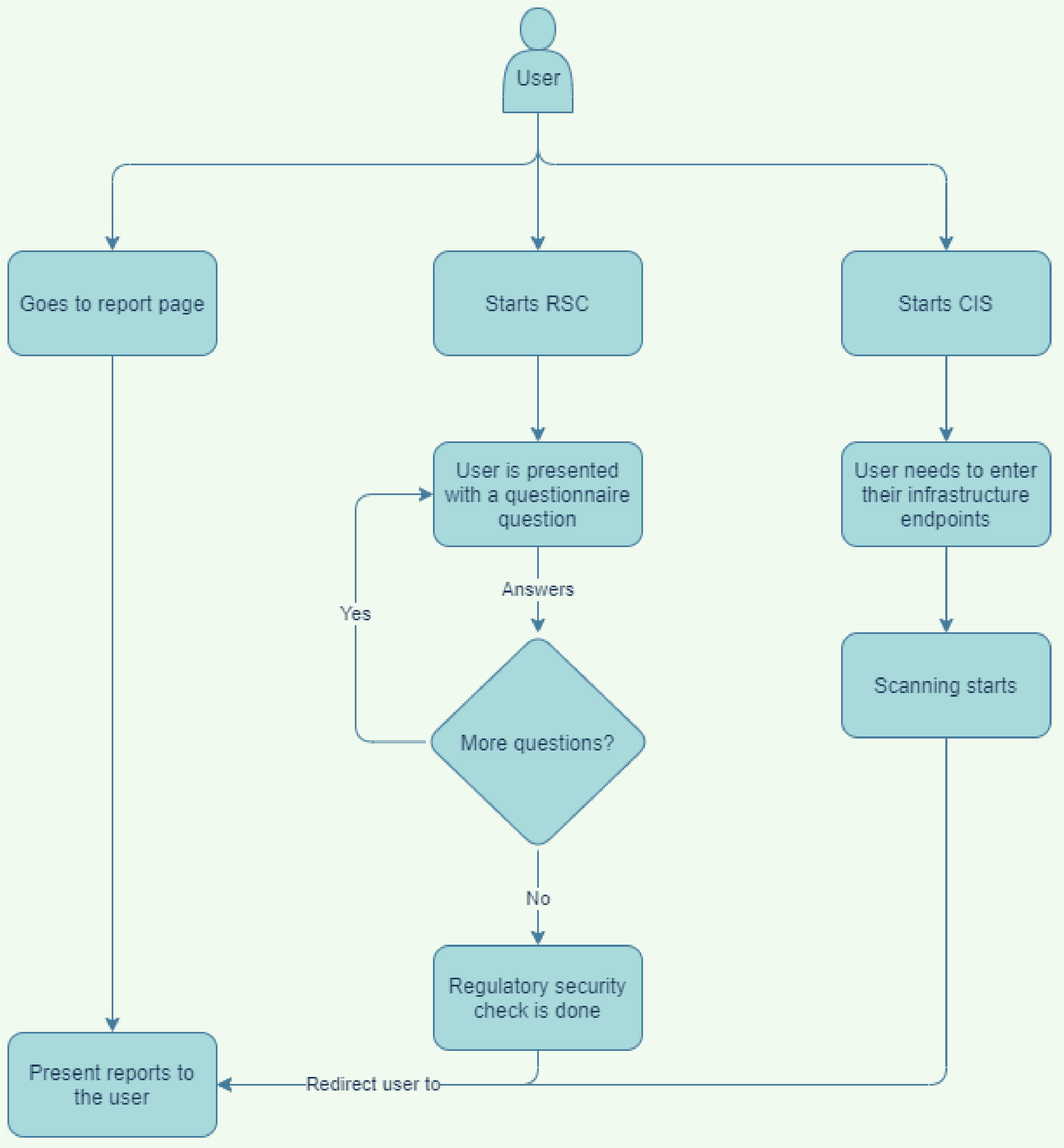

User
Goes to report page
Starts RSC
Starts CIS
User is presented with a questionnaire question
User needs to enter their infrastructure endpoints
Answers
Yes
More questions?
Scanning starts
No
Regulatory security check is done
Present reports to the user
Redirect user to

Figure 3: Flowchart of web interface

# 6 Quality assurance

## 6.1 Documentation, standards, and source code

Sources found during research for the thesis will be thoroughly documented and cited. Where necessary, sources will be added as appendices in the final report.

All meetings during the project period will be documented by the designated secretary. This includes meetings with the group, supervisor, and client. If the secretary is not able to attend a meeting, the appointed backup will do the documentation.

The source code for the final product will be stored on BitBucket. Its documentation will be stored in Confluence.

In the case of using external libraries and code, there will be a thoroughly research about licensing and copyright to make sure we can legally use it for the desired purpose.

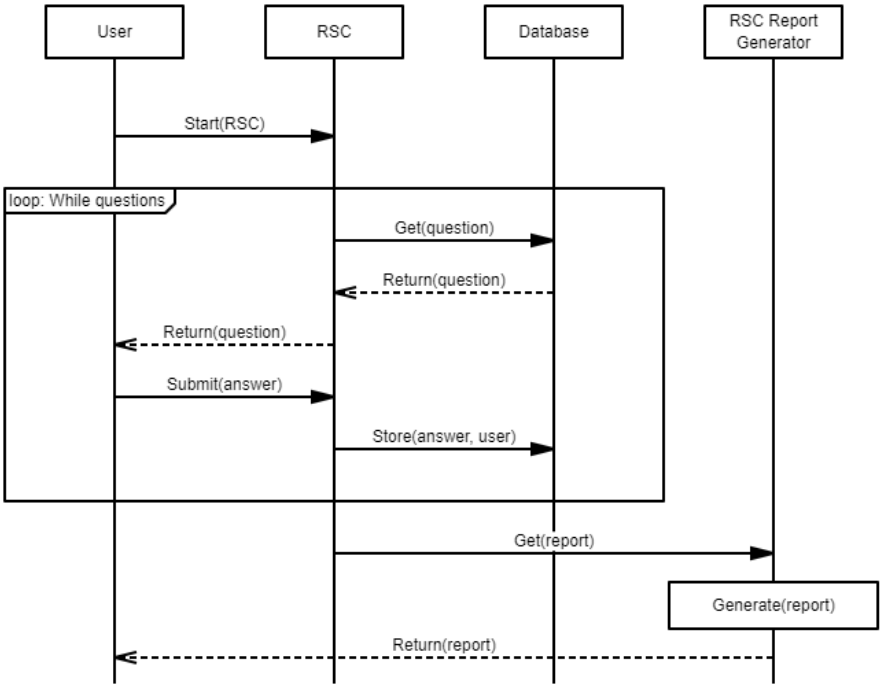


Figure 4: Sequence chart of the RCS

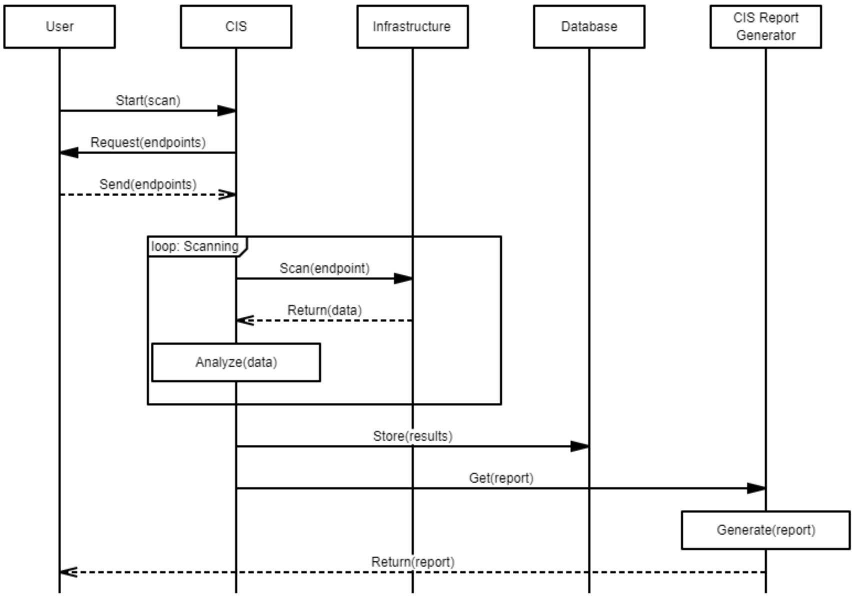


Figure 5: Sequence chart of the CIS

## 6.2 Configuration management

- The bachelor project report will be written in Overleaf.
- Code will be stored in Bitbucket and managed using Git to provide version control.
- Programming and coding tasks will be managed using Jira issues.
- Other tasks will be managed using Trello.
- Internal programming and coding documentation will be stored in Confluence.
- File storage, minutes of meeting and other administrative documents such as group rules will be stored on OneDrive.

## 6.3 Risk analysis

**Type:** Project
**Risk:** A group member falls ill for a significant time period
**Consequence:** Medium
**probability:** Usually Low, but Medium due to covid-19
**Action:** Stated by group rules, a group members has to notify the others if they cannot do their assigned tasks in time. The group will then redistribute the workload amongst themselves.

**Type:** Project
**Risk:** Loss of project report, code, or other data
**Consequence:** High
**probability:** Low
**Action:** All of our tools are cloud-based, so the risk of data loss due to hardware failure is very low, but there is a risk of data loss due to user error or bugs, so OverLeaf project data will be downloaded regularly. Git data will be cloned locally, so this data will be stored at multiple places.

Overleaf, BitBucket, and most of the other applications use version control, which minimizes the consequence of user error.

**Type:** Project
**Risk:** Underestimated the work that needs to be done to complete the project.
**Consequence:** Medium
**probability:** Medium
**Action:** When defining the scope of the project, we need to be sure that we're actually able to do the things in time.

**Type:** Project
**Risk:** Report and project is not completed within the deadline
**Consequence:** High
**probability:** Low
**Action:** The project plan will need to be followed, and tracking of issues will be used to follow the progress. Adjustments to the plan will be made if deemed necessary.

**Type:** Development
**Risk:** Integration between questionnaire and security scanner does not work as expected
**Consequence:** Medium
**probability:** Low
**Action:** Additional work will be needed to

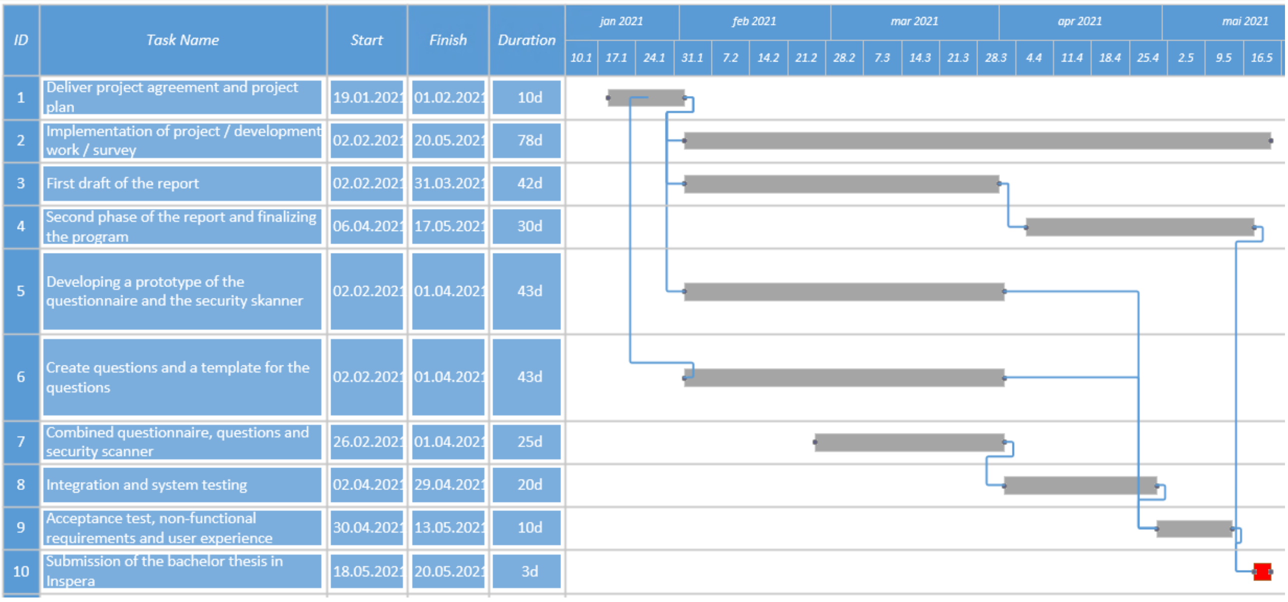
**Type:** Development
**Risk:** Major merge issues
**Consequence:** Medium
**probability:** Medium
**Action:** If multiple people are working on the same files merging can be difficult. We will therefore try not to work on the same files, and absolutely not work on the same pieces of code.

**Type:** Development
**Risk:** Loss of local changes
**Consequence:** Low
**probability:** Low
**Action:** By pushing often and keeping separate branches for features we will ensure that a loss of local changes due to hardware failure, or user error will not have a high impact.

**Type:** Development
**Risk:** Lack of knowledge to achieve the features we want for the project
**Consequence:** Medium
**probability:** Low
**Action:** Re-planning, new agreement with client and adjust the feature requirement, so that the project can be fulfilled.

# 7 Project plan

## 7.1 Gantt



| ID | Task Name | Start | Finish | Duration |
|---|---|---|---|---|
| 1 | Deliver project agreement and project plan | 19.01.2021 | 01.02.2021 | 10d |
| 2 | Implementation of project / development work / survey | 02.02.2021 | 20.05.2021 | 78d |
| 3 | First draft of the report | 02.02.2021 | 31.03.2021 | 42d |
| 4 | Second phase of the report and finalizing the program | 06.04.2021 | 17.05.2021 | 30d |
| 5 | Developing a prototype of the questionnaire and the security skanner | 02.02.2021 | 01.04.2021 | 43d |
| 6 | Create questions and a template for the questions | 02.02.2021 | 01.04.2021 | 43d |
| 7 | Combined questionnaire, questions and security scanner | 26.02.2021 | 01.04.2021 | 25d |
| 8 | Integration and system testing | 02.04.2021 | 29.04.2021 | 20d |
| 9 | Acceptance test, non-functional requirements and user experience | 30.04.2021 | 13.05.2021 | 10d |
| 10 | Submission of the bachelor thesis in Inspera | 18.05.2021 | 20.05.2021 | 3d |

Figure 6: Gantt diagram of the project plan

## 7.2 Milestones

- 19 January - 1 February
  Deliver project agreement and project plan

- 2 February - 20 May
  Implementation of project / development work / survey

- 2 February - 31 March
  First draft of the report

- 6 April - 17 May
  Second phase of the report and finalizing the program

- 2 February - 1 April
  - Developing a prototype of the questionnaire and the security scanner
  - Create some questions and a template for the questions

- 26 February - 1 April
  Combined questionnaire, questions and scanner

- 2 April - 29 April
  Integration and system testing

- 3 May - 14 May
  Acceptance test, non-functional requirements and user experience

- 18 May - 20 May
  Submission of the bachelor thesis in Inspera

  * The spaces between some days are because of the national holidays like 1 April until 6 April is the Easter holiday or 17 May which is the Norwegian Constitution day which collides a Monday.

## 7.3 Time and resource scheduling

Due to the situation as the Covid-19 requires and also different job times each student might have we work mostly online together. However we are a high ambitious group of students with high competence, and have high expectations of ourselves. Therefor the amount of work expected of each of us in this project is set to 30 hours per week. We have some freedom of choosing when to work after we divide the tasks and we have also good control systems such as Trello and Clockify to keep track of the progress of each task delegated to each of us. The group rules will help us to to achieve this goal and fulfil the roles delegated to each of us. For group rules and agreement please see appendix A.
More precise time management during the project is illustrated in the Gantt diagram.

### 7.4 Work breakdown structure

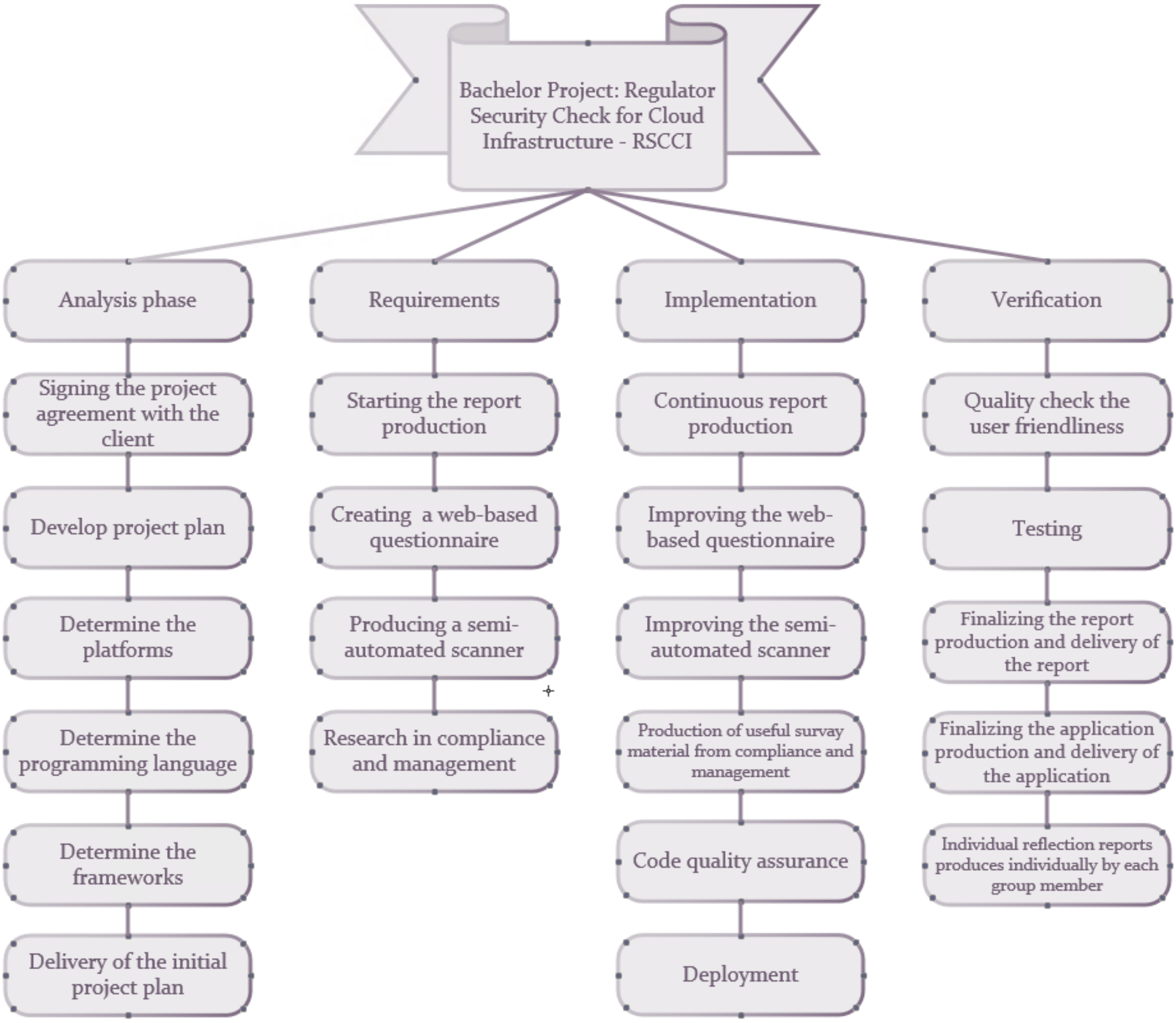

Bachelor Project: Regulator Security Check for Cloud Infrastructure - RSCCI
Analysis phase
Signing the project agreement with the client
Develop project plan
Determine the platforms
Determine the programming language
Determine the frameworks
Delivery of the initial project plan
Requirements
Starting the report production
Creating a web-based questionnaire
Producing a semi-automated scanner
Research in compliance and management
Implementation
Continuous report production
Improving the web-based questionnaire
Improving the semi-automated scanner
Production of useful survay material from compliance and management
Code quality assurance
Deployment
Verification
Quality check the user friendliness
Testing
Finalizing the report production and delivery of the report
Finalizing the application production and delivery of the application
Individual reflection reports produces individually by each group member

Figure 7: Work breakdown structure

# References

[1] Axeptia, *Axeptia homepage*, `https://www.axeptia.com/`.

[2] Microsoft, *Penetration testing rules of engagement*, `https://www.microsoft.com/en-us/msrc/pentest-rules-of-engagement?rtc=1`, (Downloaded 16/01/2021).

[3] Atlassian, *What is kanban*, `https://www.atlassian.com/agile/kanban`, (Downloaded 16/01/2021).

# D Group Rules

# Axeptia bacheloroppgave grupperegler:

**Formålet** med dette dokumentet er å skape klare retningslinjer og regler for arbeidet fremover og for å unngå usikkerhet ved eventuelle intern- eller eksterne konflikter.

**Verdigrunnlaget** for gruppen er at alle studenter som bidrar til gruppearbeidet får utbytte av studiene og bacheloroppgaven som bidrar til enhver sitt fremtidige gruppearbeid både i studieprosessen og i arbeidslivet.

## Gruppemedlemmer består av:

- Øyvor Vejlgaard Sørensen
- Harald Aarseth
- Kristian Fjelstad
- Gloria Treider

## (Hoved)Ansvarsdelingen består av:

**Teknisk:** Øyvor, Kristian, Harald

**kontaktpunkt mot Axeptia**: Øyvor

**Compliance/management/jus**: Gloria

**kontaktperson mot NTNU**: Gloria

**Sekretær**: Gloria

**Vara sekretær:** Øyvor

**Kontrollør og ansvarlig for Trello som er relatert til workflow:** Harald

**Loggbok ansvarlig:** Harald

**Overleaf spesialist**: Kristian

**Confluence ansvarlig:** Kristian

**Bitbucket repository og jira ansvarlig:** Øyvor

## Definisjon av rollene:

**Kontaktpunkt mot Axeptia:** Har ansvaret for å ta kontakt med de som er behov for på Axeptia. Denne kommunikasjonen blir gjennomført via denne kontaktpersonen.

**Kontaktperson mot NTNU:** Har ansvaret for å ta kontakt med de som er behov for på NTNU inkludert veilederen (Kelly) på NTNU, og denne kommunikasjonen blir gjennomført frem og tilbake via kontaktpersonen til gruppe. Skriver også ukentlig rapport til Kelly.

**Teknisk:** De som blir tildelt det tekniske ansvaret skal delta i utvikling av programmet, oppsett av infrastruktur og databasen. Velge teknologi og programmeringsspråket som kan bidra til oppgaven.

*Forslag: I denne delen av gruppen beslutninger om de tekniske delene av oppgaven blir tatt. Denne delen er 3 studenter som bidrar. For å ta beslutninger enten alle er enige, eller to mot en, eller hver av de tre gruppemedlemmene har sin mening. I dette tilfelle kan forslag fra andre del gruppen i Compliance/management/jus høres og bør høres, og hvis det ikke løser seg kan konflikten drøftes i møte med veileder.

Generelt forslag fra andre gruppedelen som jobber med Compliance/management/jus kan og bør høres etter at den studenten også har samme utdanning og forståelsen, men beslutning tas av selve medlemmer i den tekniske delen som jobber mest med de tekniske temaene.

* Forslag: Oppretter issues i Jira for bugs/problemer som oppstår underveis, og lukker issues dersom de er løst.

**Compliance/management/jus:** Dette går ut på spørreundersøkelsen som skal lages. Spørsmålene som kommer og blir brukt i sikkerhetssjekken. I tillegg til hva som skal sjekkes ift. ISO 27xxx standarder.

*Forslag: I denne delen av gruppen blir oppgavene som er delegert gjennomført av en student som har ansvaret. Forslag fra andre studenter i tekniske delen blir hørt og tatt hensyn til. Hvis en mener en endring i arbeidet som er gjort er absolutt nødvendig så kan det drøftes med ansvarlig for denne delen. Hvis det ikke oppnås enighet, kan det tas stemmer. Dersom stemmene er to mot to, så kan konflikten tas opp i møte med veileder for å ta en siste beslutning om problemstillingen.

Alle i gruppen kan komme og bør komme med forslag til Compliance/management/jus delen etter at alle har sikkerhetsutdanning, men må gi vei for at den studenten som har ansvaret for denne delen skal kunne fullføre jobben.

**Sekretær:** Skriver ukentlig møtereferat hver gang vi har møte, setter opp timer til neste møte etter avtalt tid i gruppen, dokumenterer alt som blir sagt i gruppemøte. Dette inkluderer oppgavene som er gjort som blir levert i gruppemøte, og oppgavene som skal gjøres til neste uke av gruppemedlemmer.

**Kontrollør** - Passer på at alle oppgavene som er diskutert er lagt inn i Trello under "To Do" listen etter at forslag har kommet ut og blitt diskutert i gruppen. Kontrollerer at alle gruppemedlemmer blir satt til oppgaver som er skrevet i Trello, at alle har fått oppgave, og at de krysser av når oppgaven er ferdiggjort. Denne personen har også ansvaret for Trello.

Kontrollør har også ansvaret for loggbok som har Clockify som plattform og han sørger for at den er oppdatert.

**Overleaf spesialist –** I tillegg til at alle i gruppen bidrar til rapporten og dennes struktur, er en person ansvarlig for å sørge for at rapporten i Overleaf ryddes og fikses der det er behov for det ift. at Overleaf trenger litt mer oppmerksomhet av koding og struktur.

**Confluence ansvarlig:** Passer på at dokumentasjon i Confluence er oppdatert og organisert. Oppdaterer eller spør personer som har gjort endringer om å oppdatere dersom noe ikke stemmer lenger. Ansvarlig for å dokumentere veivalg i utviklingen.

**Bitbucket repository og Jira ansvarlig:** Følger med på issues og deres progresjon, passer på at påkrevd funksjonalitet og større problemer/bugs har en issue. Vedlikeholder README, og gjør nødvendige oppryddinger i Git og hjelper til med å løse problemer med brancher.

*Disse rollene er definert for å ha en klar definisjon av forventninger av hver rolle, men hvis noen ønsker å bidra med mer til oppgaven og gruppearbeidet så er veien åpen. Oppgaven hører til alle medlemmer i gruppen og alle skal føle eierskap til oppgaven i helhet.

## Definisjon av arbeid-, dokumentasjon- og kommunikasjonsplattform:

**Kommunikasjonsplattform:** Gruppen kommuniserer på Discord for å diskutere, ha voice-møter, avtale nye møtetider eller melde fra om fravær og lignende

**Arbeidsplattform:** Overleaf, OneDrive på Office365, Trello, Bitbucket, Confluence, Jira og Clockify

> Vi gjør arbeidsoppgaver og endelige innleveringen i **Overleaf**. Oppgaven blir dokumentert i Overleaf fra starten og til siste innlevering.
>
> **OneDrive** på Office365 blir brukt for å samle referat fra møter og diverse dokumenter som skal deles gjennom gruppen.
>
> **Trello** blir brukt for å ha kontroll på arbeidsoppgave delinger generelt for bachelor oppgaven, og for compliance
>
> **Jira** brukes som issue tracker og for å ha kontroll på arbeidsoppgaver knyttet til programmering og utviklingen av software.
>
> **Confluence** brukes som dokumentasjonsverktøy for programmering og software utviklingen. Dokumentasjon av veivalg for utviklingen.
>
> **Clockify** brukes for timeføring.

**Teams og NTNU-epostplattform:** Kommunikasjon med NTNU, blant annet med veilederen Kelly pågår via Teams og epost på NTNU plattform.

**Kommunikasjonsplattform med Axeptia:** Teams blir brukt for møter, epost blir brukt av kontakt person for enkelte kommunikasjon.

## Møteorganisasjon:

Møtene holdes for å dele progresjon av ukeoppgaver og diskutere rundt dette hver tirsdag 10:00 –11:00. Dersom det er nødvendig, har vi et ekstra møte etter avtalt tid. Hvis et medlem mener det, trengs et ekstra møte må det tas opp med gruppen senest 24t før møtetid.

Tiden vi bruker på dette møtet skal være fleksibel og gruppemedlemmer tar det som kommer. Sekretær(ene) skriver referat fra hvert møte.

## Forventet arbeidsmengde per uke:

Denne bachelor oppgaven har totalt **22,5** studiepoeng og veier som to fag. Hvert fag på **10** studiepoeng tar **15** timer inkludert forelesning, undervisningene, og relaterte møter og arbeid per uke. Dette betyr at arbeidet med bachelor oppgaven skal bestå av **30** timer per uke per student.

**I denne sammenheng antall timer forventet av hvert gruppemedlem per uke:**

**30** timer inkludert alle møter, forberedelser, arbeid og tid vi legger inn for oppgave fremgangen.

Av denne kan det nevnes:

**Foreslåtte møtetider med veileder (Kelly) per uke:**

Torsdager 1200-1230 med Kelly (30 minutter etter gruppens behov) på Teams

Gruppemøte etter møte med Kelly max **2** timer (kan holdes innen en time) på Discord

- avtalt timers ekstra møte*
- 30 timer individuelt arbeid med ukeoppgave basert på de delte oppgavene og mengder arbeid som er behov for å gjøres. 30 timer er inkludert alt som blir gjort ift. Bachelor oppgaven som møter, lesing, arbeid, skriving osv.

* Kan avvike

## Møtestruktur:

- **Hvert medlem presenterer det de har jobbet med** siden sist.
- Diskusjon rundt det som blir presentert.
- Sjekke at Trello er oppdatert
- Idemyldring
- **Arbeidsoppgaver til kommende uke** blir fordelt.
- Møte tidsrom er absolutt i henhold til avslutning.

## Grupperegler:

1. Alle skal følge med på meldinger og bekrefte eller si om de har problem og komme til ny enighet etter avtale, men det er viktig at gruppemedlemmer bekrefter at de har mottatt meldingen og svarer på det.
2. Hvis og når det blir satt fysiske gruppemøter så skal det avtales bestemt start og sluttid, og gruppen skal holde seg til den avtalte tiden. Fysiske gruppemøter skal ha frivillig fysisk oppmøte, og det skal legges til rette for at gruppemedlemmer kan være med via teams/discord.
3. Det skal holdes respekt til arbeid hver gruppe medlem gjør, slik at endring på jobben som er gjort skal tas opp ved å sette kommentar og diskutere den, eller diskutere den direkte med personen til de kommer til en enighet, eller det tas opp i gruppemøte.
4. Gruppen ble enige om å ha møte med veileder **torsdager 1200-1230**
5. Gruppemedlemmer skal kommunisere med hverandre i gruppe om nye oppgaver og oppgaver som har behov for endring. God kommunikasjon er en viktig faktor for at alle i gruppen får deltatt aktiv i gjennomføring av oppgaven og eventuelle misforståelser blir klargjort.

6. Gruppereglene som ble det enighet om skal signeres eller skrives en bekreftelse på av gruppemedlemmer at hver person har lest og godtar reglene.
7. Vi er en gruppe av 4 studenter. Alle eier og deltar i oppgaven likt. Ingen individ eier ene og alene oppgaven og oppgaver eller konflikter som oppstår i oppgaveløsningen skal løses på en demokratisk måte i gruppen.
8. Hvert medlem er ansvarlig for å oppdatere Trello på det de jobber med
9. Alle skriver sin del av oppgaven i Overleaf.
10. Dersom et medlem av spesielle årsaker ikke klarer å gjøre arbeidsoppgaven sin ferdig før møte skal personen selv sørge for å gi beskjed til gruppen i god tid.
11. Alle må møte opp tidsnok til møtene. En må så tidlig som mulig gi beskjed til gruppen dersom en ikke klarer å møte opp eller kommer for sent.
12. Hvis det er uenigheter i gruppen, bestemmer flertallet.
13. Det er hvert medlems ansvar å holde seg "oppdatert" på gruppens framgang i oppgaven.
14. Hvis et gruppemedlem ikke kan delta på møte (veiledningsmøte eller gruppemøte) som er satt opp og personen har gitt beskjed til gruppen, skal vedkommende gi melding/spørsmål/oppgaven sin til et annet gruppemedlem slik at dette kan tas opp i møtet.
15. Gruppen skal være informert om ting som skal sendes til emne ansvarlig, Axeptia, eller veileder på vegne av gruppen før den blir sendt.
16. Når gruppen møtes skal det jobbes med oppgaven, og det skal unngås bortkast tid i gruppemøter som for eksempel lang prating om andre temaer enn det gruppen har samlet seg for.
17. Ettersom at Øyvor både jobber for Axeptia og har satt opp oppgaven i utgangspunktet så veier hennes stemme som 2. (mer enn en)

## Brudd på gruppereglerr:

- Ved brudd på regler angående ukentlig oppgaver gis det påminnelse før en advarsel blir sendt ut.
- Ved mindre alvorlige brudd på reglene tas det opp med gjeldende person snarest.
- Ved alvorlige eller gjentatte brudd på reglene skal alle medlemmene på gruppen møtes, og det blir snakket om problemstillingen.
- Det skal gis skriftlig advarsel til medlemmer som ikke opprettholder reglene. Advarslene skal dokumenteres.
- Ved tre skriftlige advarsler skal emneansvarlig (Erik) involveres.

## Kontaktpersoner på Axeptia:

**Per Nestor Warp**, per.nestor.warp@axeptia.com

**Håkon Viervoll,** hakon.viervoll@axeptia.com

## Veileder på NTNU:

**Jia-Chun Lin (Kelly)** jia-chun.lin@ntnu.no

**Signatur/bekreftelse** av gruppemedlemmer om at gruppereglene er lest og bekreftes/godtas:

**Øyvor Vejlgaard Sørensen:** Øyvor V. Sørensen

**Harald Aarseth:** Harald Aarseth

**Kristian Fjelstad:** Kristian Fjelstad

**Gloria Treider**: Gloria Treider

# E Prod Setup and Prod Update

## Prod setup

1. Do updates and upgrades, reboot and setup a basic firewall (step 4 only): https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04
2. Install a LEMP stack (1-3 only): https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-on-ubuntu-20-04
3. Create non-root user for the DB: https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql
4. Create a DB

   ```
   mysql -u $DBUSER -p
   CREATE DATABASE bachelor;
   exit;
   ```

5. Edit `/etc/nginx/sites-available/bachelor` to contain:

```
server {
    listen 80;
    server_name example.com;
    root /var/www/bachelor/bachelor-cloud-2021/public;

    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Content-Type-Options "nosniff";

    index index.php;

    charset utf-8;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location = /favicon.ico { access_log off; log_not_found off; }
    location = /robots.txt  { access_log off; log_not_found off; }

    error_page 404 /index.php;

    location ~ \.php$ {
        fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
        fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.(?!well-known).* {
        deny all;
    }
}
```

6. Symlink new sites file, and remove old

```
sudo rm /etc/nginx/sites-available/default
sudo rm /etc/nginx/sites-enabled/default
sudo ln -s /etc/nginx/sites-available/bachelor /etc/nginx/sites-enabled/bachelor
sudo service nginx restart
```

7. Install requirements:
   `sudo apt-get install php7.4-mbstring php7.4-xml php7.4-zip php7.4-curl`
8. Install Nodejs and NPM:
   Easy mode (older version): `sudo apt install npm`
   Medium mode, follow option 2 (recent version): https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-20-04

9. Install and configure redis (step 1 and step 4):
   https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-redis-on-ubuntu-20-04
10. If there is no regular user, create one and log in as it:

    ```
    adduser bachelor
    usermod -aG sudo bachelor
    login bachelor
    ```

11. Create directory and cd:

    ```
    sudo mkdir /var/www/bachelor
    sudo chown -R bachelor:www-data /var/www/bachelor
    cd /var/www/bachelor
    curl -sS  https://getcomposer.org/installer  | sudo php -- --
    install-dir=/usr/local/bin --filename=composer
    ```

12. Clone repo and set up
    Add a read-only ssh key or use password to get git repo

    ```
    git clone git@bitbucket.org:oyvorvs/bachelor-cloud-2021.git
    cd bachelor-cloud-2021
    composer install --no-dev
    ```

13. Edit the `.env` file to contain local secrets, such as the mysql db/user/password. Also make sure environment is set to production, and that debuging is disabled
14. Run migration and generate key:
    `php artisan migrate:fresh`
    `php artisan key:generate`
15. Install supervisor for horizon:

    ```
    sudo apt-get install supervisor
    sudo useradd --no-create-home --shell /usr/sbin/nologin --system
    horizon
    ```

    /etc/supervisor/conf.d/horizon.conf

```
[program:horizon]
process_name=%(program_name)s
command=php /var/www/bachelor/bachelor-cloud-2021/artisan horizon
autostart=true
autorestart=true
user=horizon
redirect_stderr=true
stdout_logfile=/var/log/horizon.log
stopwaitsecs=3600
```

16. Run optimizations and cache commands:

```
composer install --optimize-autoloader --no-dev
php artisan config:cache
php artisan route:cache
php artisan view:cache
```

17. Set up CSS/JS

```
npm i
npm update
npm audit fix
npm install
npm run production
```

18. Fix permissions:

```
sudo usermod -aG www-data bachelor
sudo chown -R www-data:www-data storage/
sudo chown -R www-data:www-data bootstrap/cache
```

More permission changes are probably needed, make sure to not give things 777 permissions, but add www-data as owner or group

## Prod update

```
git pull
sudo chown -R www-data:www-data storage/
php artisan config:cache
php artisan route:cache
php artisan view:cache
```

# F Bitbucket README.md

### README.md - Grip

This is the BitBucket repo for the source code produced for bachelor thesis "Regulatory Security Check For Cloud Infrastructure" 2021.

# Proposed rules

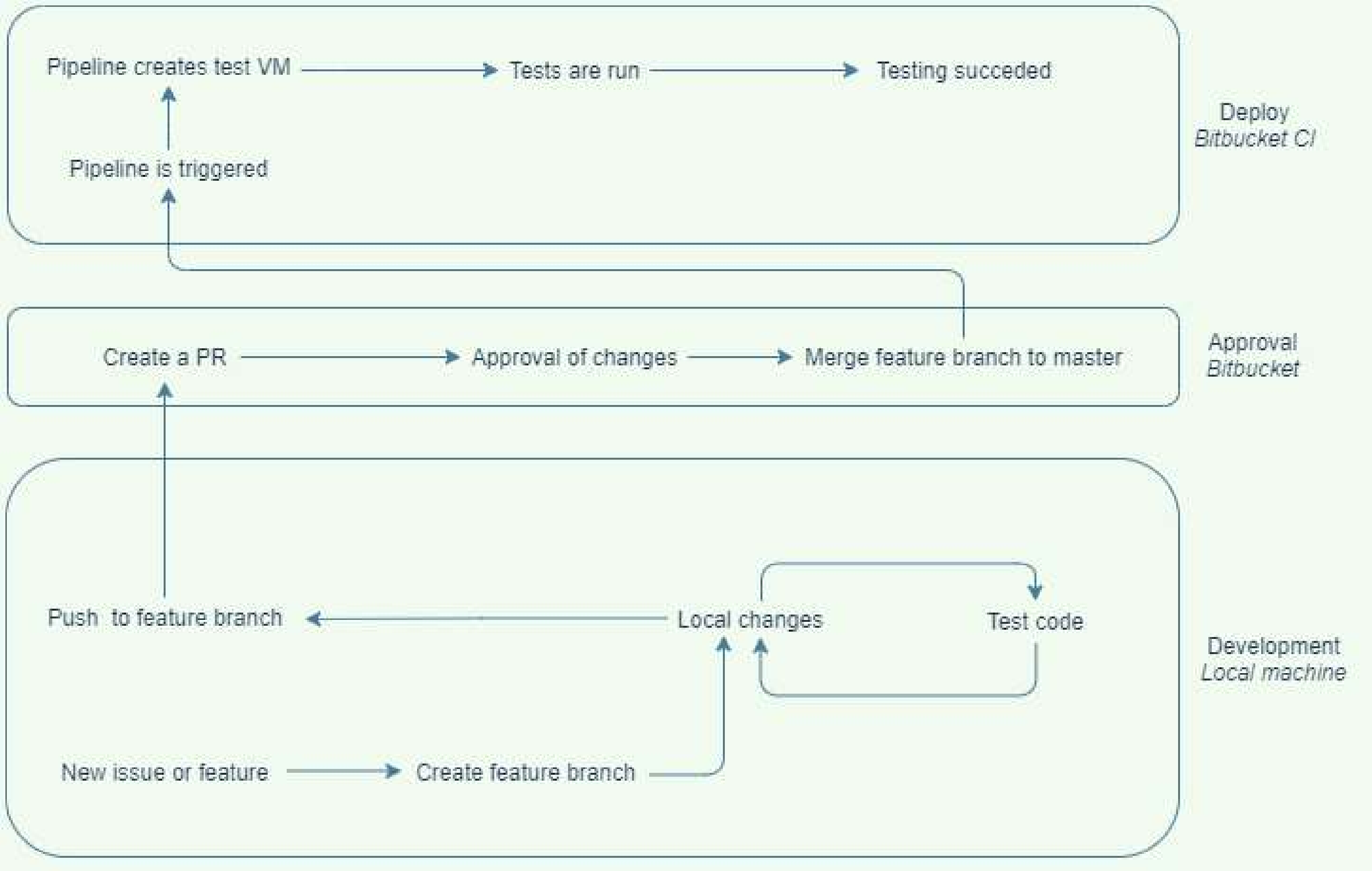
- All commits shall have a professional message in English that briefly explains what has been done with the modified file(s). Example: "Added logging of database queries".
- A contributer is not allowed to merge his or her changes to master without another reviewers approval.
- If two or more contributers has worked together on a change (pair programming), another contributer does not need to review the changes.
- All changes has to be relevant to the project goal, and should have a task/issue/trello-card connected to it.
- When possible, one or more unit-tests should be created when adding new functionality to the solution.

# Documentation of code

Code documentation will be done in Confluence. It is recommended that the documentation happens on an ongoing basis rather than afterwards to ease the documentation process and not fall behind. If necessary, a user guide for the final product should be conducted.

# Proposed workflow

- Each contributer clones the project down to their local computer.
- Each contributer will work on their own branch, no code is ever to be pushed directly to Master. Exception: README.md.
- There should be created a new branch for each task/feature with a short, descriptive name. When the changes are merged to Master, the branch used should be closed/deleted.
- When a contributer is done with their task, a pull request (PR for short) is to be made with at least one other contributer as reviewer.
- When one reviewer has looked at the changes related to the task, the contributer is allowed to merge their changes into the Master.
- In case of pair programming, the PR does not need to be reviewed and approved by another reviewer.
- Each time there is a merge with the master branch, the other contributers has to pull down the changes and merge them into their own branch to prevent merge hell later.



## Workflow to work on a local branch and then merge it to master (from scratch)

User-guide for proposed workflow:
1: Clone down the repo from you own computer, and make sure it is up-to-date with the master branch:

```
git clone git@bitbucket.org:oyvorvs/bachelor-cloud-2021.git
cd bachelor-cloud-2021
git checkout <name of your branch>
git merge master
```

If you don't have a branch, use he following command to create one:

```
git checkout -b <name of your branch>
```

2: Make desired changes, test them locally, then push to your branch:

```
git status
git add <file> # The files from the status command you want to commit
git commit -m "<message>" # An informative commit message
git checkout master
git pull
git checkout <name of your branch>
git merge master
git status # Check that all is okay before push
git push
```

3: Make a PR. When one of the reviewers is happy with your code, merge it!

- Go to Pull requests.
- Create new.
- Add a title, e.g. the name of the new feature.
- Write a short description of the new feature for the reviewers to read.
- Add one or more reviewers.

- If you want to automaticly close/delete your branch after merge, choose "Close branch after merge" (or something like that).

4: Wait for pipeline to finnish

- For every merge to Master, a pipeline is triggered. This pipeline runs all unit tests in the project to make sure that code that makes tests fail is not deployed to the working product.
- If the pipeline failes: Check the output, find the broken test, and fix whatever making it fail (is it the code, or is the test testing something wrong after new functionality was added?)
- If the pipeline finnishes without failure: DISCLAIMER - Not decided yet, research has to be conducted. Alternatives: Manual deploy to Azure environment or automatic deploy to Azure environment

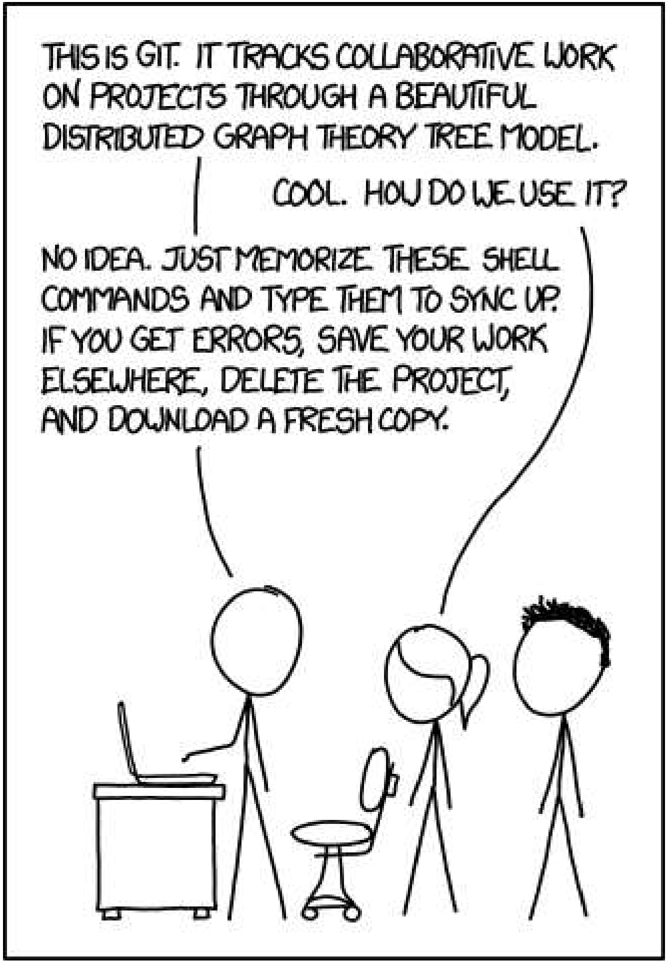
5: Congrats, your code is in production!

6: Sync up your branches!

```
git checkout master
git pull # Pull down the new changes
git fetch --prune # This command syncs up wich branches is active reomte (origin)
git branch -a # Shows both local and remote branches
git branch -d <name of deleted branch> # Delete a local branch
git branch -a # Verify the branch is deleted
```

Remember to merge Master into other branches if you have multiple :)

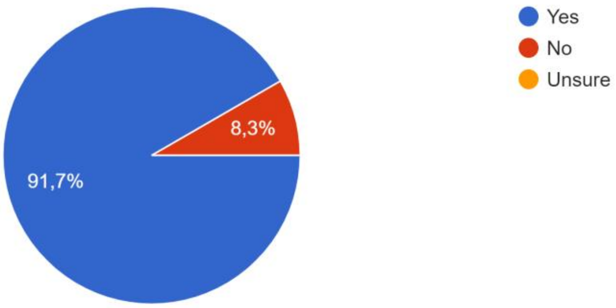
**TL;DR**


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

## G Axeptia's Answers to the User Survey

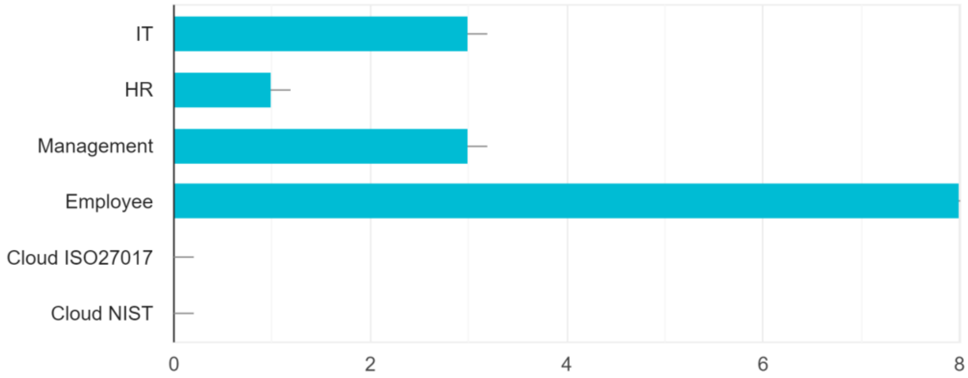
Did you like the design of the website?

12 svar



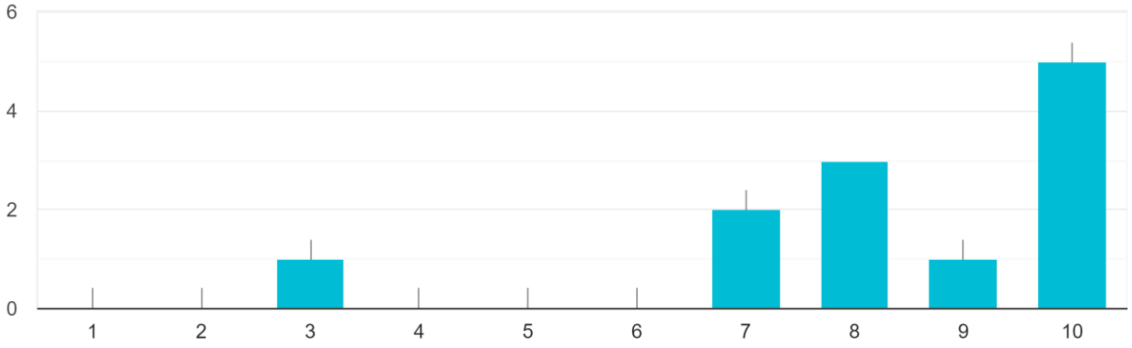
Which questionnaire(s) did you take?

12 svar



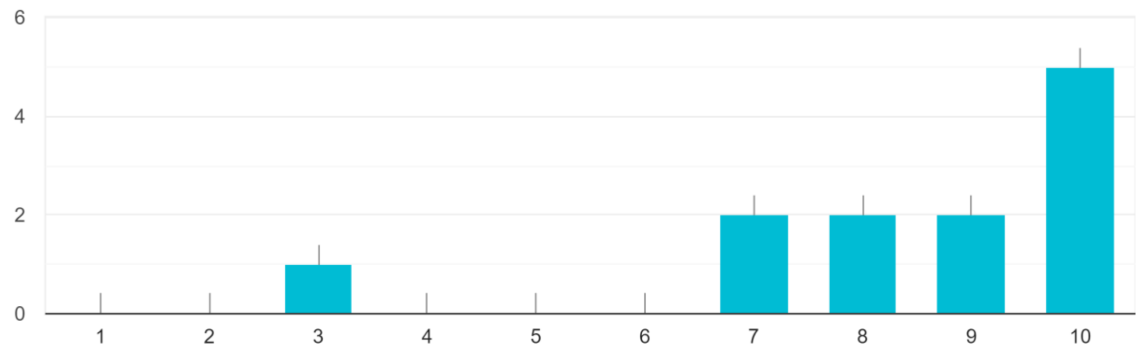
On a scale from 1 to 10, how understandable where the questionnaire results?

12 svar



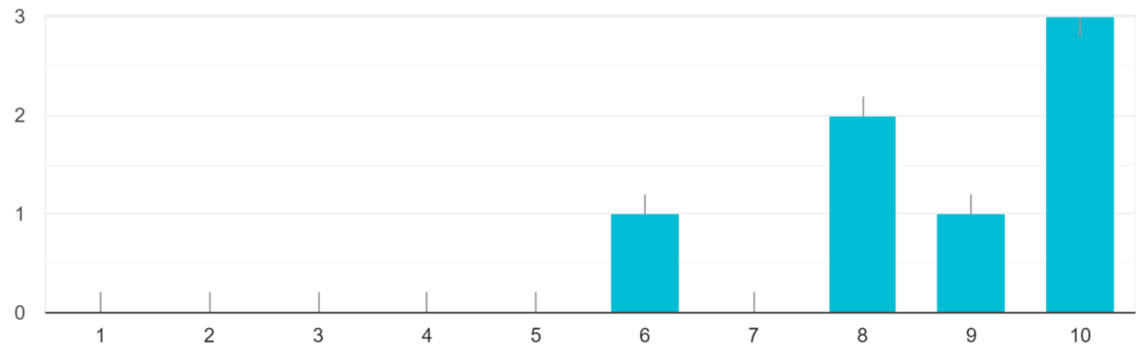
On a scale from 1 to 10, how useful were the questionnaire results?

12 svar



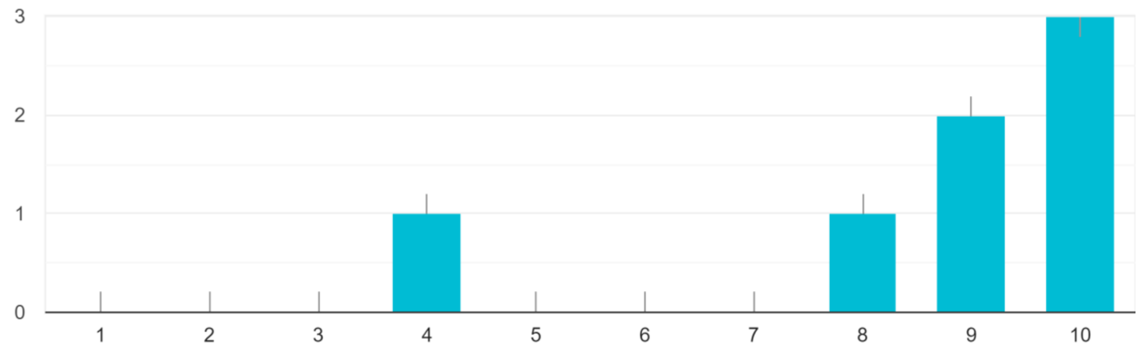
On a scale from 1 to 10, how understandable were the scan results?

7 svar



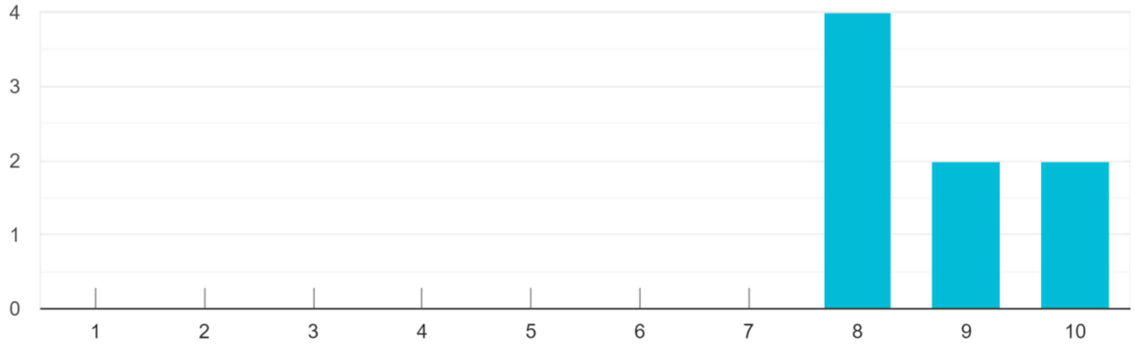
On a scale from 1 to 10, how useful were the scan results?

7 svar



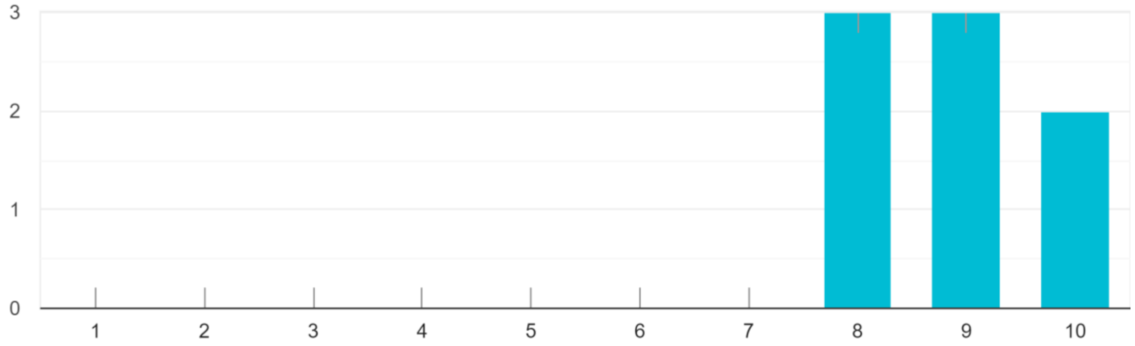
On a scale from 1 to 10, how understandable were the summary results?

8 svar



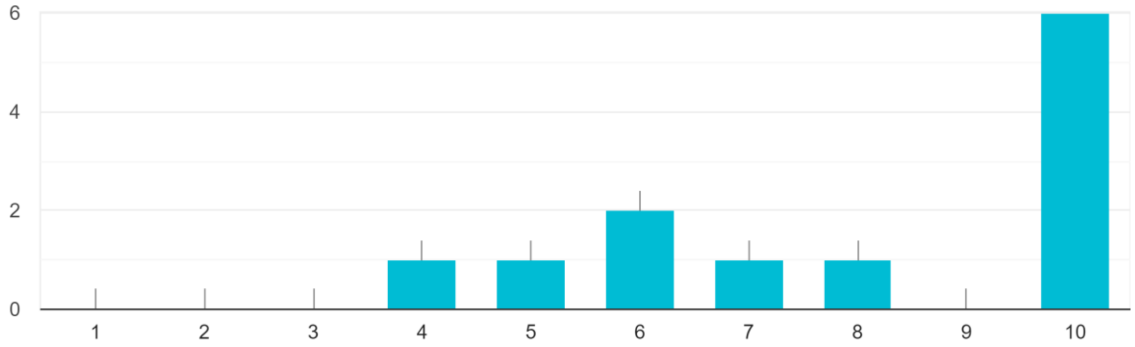
On a scale from 1 to 10, how useful were the summary results?

8 svar



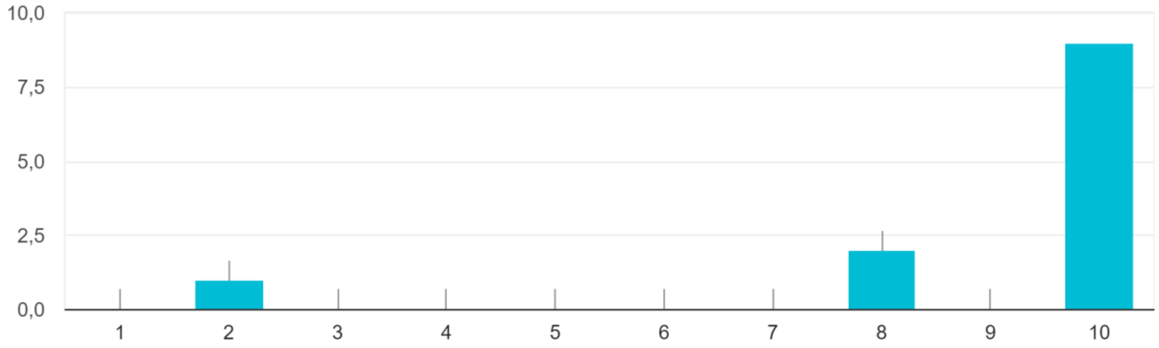
On a scale from 1 to 10, how much more insight did you get of your cloud security after using RSCCI?

12 svar



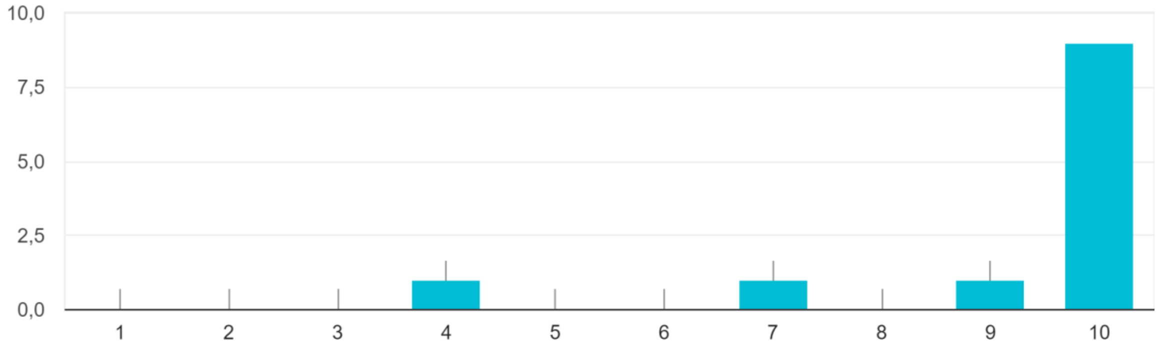
On a scale from 1 to 10, if this was released as a service, how likely is it that you would use it in the future for evaluating your company's cloud security?

12 svar



On a scale from 1 to 10, how likely is it that you would recommend the service as a security evaluation tool to other companies?

12 svar



Any other feedback you would like to give us? :)

2 svar

Lav score på scan er grunnet for lite bruk av funksjonen, og dermed er jeg ikke blitt trygg på resultatene. Håper å få prøvd den mer fremover, det virker som et knallbra verktøy som jeg er overbevist over har veldig stor verdi!

Når det gjelder spørreskjemaene har de enda større nytteverdi enn jeg så for meg. Enormt bra! Et utrolig bra verktøy, på tross av (eller som følge av, egentlig) at IT-skjemaet er litt tungt å komme gjennom...

Systemet er uten tvil noe vi kommer til å bruke som et verktøy for sikkerhetsevaluering!

Fantastisk bra jobba!

Skulle ønske jeg kunne se hva jeg svarte feil på, slik at jeg kunne lært av det.

Sørensen, Aarseth, Fjelstad, Treider

Regulatory Security Check for Cloud Infrastructure

NTNU
Norwegian University of
Science and Technology