Security, privacy, and legislation adherence assessment of a whistleblowing web application

University of Turku
Department of Computing,
Faculty of Technology
Master of Science in Technology Thesis
Information and Communication Technology
December 2022
Tino Lehtola

Supervisors:
Ville Leppänen (University of Turku)
Peter Lehto (BeanBakers Ltd)
Ilkka Halonen (BeanBakers Ltd)

UNIVERSITY OF TURKU
Department of Computing, Faculty of Technology

Tino Lehtola: Security, privacy, and legislation adherence assessment of a whistleblowing web application

Master of Science in Technology Thesis, 151 p.
Information and Communication Technology
December 2022

In recent years, web applications have become increasingly more complex as they are required to have more features than ever before. The need for more features comes from both the service providers as well as the end-users, since competition on the Software as a Service (SaaS) market can be fierce. The ever-growing complexity and feature richness of web applications have in turn also increased their attack surface, predisposing them to new threats and vulnerabilities. The evolving web applications have also developed new methods of gathering personal data from its users. User information privacy has become a hot topic of discussion in the past decade, which has led to privacy legislation being enacted in different regions of the world. In 2019, the European Parliament enacted Directive (EU) 2019/1937 into the European law, which is also known as the Whistleblower Directive. The Directive's goal is to establish rules and procedures to protect individuals who report information they have acquired in a work-related context on breaches of EU law in key policy areas. The Directive requires qualifying organizations and municipalities to set up reporting channels that whistleblowers can use to anonymously report these breaches.

The commissioner of this thesis, BeanBakers Ltd, has developed a web application called Vihjaa that is meant to be used by organizations and municipalities as an internal reporting channel that complies with the requirements set for the application by the Directive. The main objectives of this thesis were to identify the requirements set for Vihjaa by EU law and then to conduct security, privacy, and legislation adherence assessments on Vihjaa to gain a deeper understanding of its current status. Furthermore, the procedures and methodology used during the assessments can be used as a framework for future works, which assess the states of other web applications. Our assessment found that Vihjaa's state of security, privacy, and legislation adherence are mostly in a good standing, but there were multiple issues identified that should be addressed. Most of the identified issues were of low severity, for instance, lacking a privacy policy document, missing a incident response plan, and out-dated dependencies. In this thesis, we present the developed framework that can be used to assess web applications of this nature, the results of our assessments, and a ranking of data items collected by a web application based on how critical they are for the process of identifying a specific user.

Keywords: Whistleblower Directive, GDPR, web application security, user privacy, legislation adherence, web application vulnerabilities, privacy concerns

Contents

List of Figures

List of Tables

1 Introduction

Modern web applications have gone through a technical evolution in recent years due to an increased demand of new features and functionality from both service providers and end-users. Those additional features and functionalities in turn increase the attack surface of web applications, as there are more libraries, modules, ports, technologies, and data sources among other targets to attack than ever before. In order to protect this increased attack surface, web applications require security hardening and auditing in order to identify threats and vulnerabilities. Furthermore, there has been an increase in privacy legislature being enacted in different regions of the world to make service providers legally responsible for the data they gather, process, and store from the end-users of their web applications. This factor has also put the pressure on the service providers to create more secure software and implement features and processes to comply with their respective regional legislation.

In this thesis we will be assessing the state of security, privacy and legislation adherence of a whistleblowing web application called Vihjaa. Vihjaa has been developed in-house by the thesis writer's current employer BeanBakers Ltd. This thesis was commissioned by BeanBakers Ltd in order to assess whether or not Vihjaa meets the requirements set for it by legislation, and to gain insight into its current state of security and privacy. The rest of the introductory chapter will provide more background information on the thesis' subject as well as how the the aforementioned assessments are conducted.

1.1 Background and Scope

In 2019, the European Parliament enacted Directive (EU) 2019/1937 into the European law, which is also known as the “Whistleblower" Directive. According to the Directive, its goal is to: “establish rules and procedures to protect “whistleblowers”, individuals who report information they acquired in a work-related context on breaches of EU law in key policy areas.” [1] In practice, the goal of the Directive is to provide individuals effective channels for reporting these breaches, which must also provide a way for anonymous reporting. The target application of this thesis, Vihjaa, aims to provide a web application that functions as an internal channel to customers with a Software as a Service (SaaS)-model. The target application can be used by organizations or municipalities to receive, process, and act on reports, while securing the privacy of whistleblowers and providing useful tools to the organization’s administrators. It is important to note however, that as the target application is a private and commercial product, some technical aspects about the application cannot be discussed in detail.

The scope of this thesis is to provide a general overview of current EU legislation regarding information privacy and the Whistleblower Directive, discuss current trends in web application security including risks, threats, and vulnerabilities as well their mitigation measures and to analyze and improve the privacy and security of the Vihjaa-application. Additionally, we will also assess whether or not Vihjaa complies with the requirements set by EU privacy legislation and the Whistleblower Directive. As a result, the methodology used in this thesis can be used as an approximate framework for future works and research to assess web applications of this nature.

1.2 Problem Description and Goal

As previously mentioned, the target application has to adhere to legislation and allow users to report breaches anonymously. Additionally, the application must also store data securely, ensure that only the appointed users in that organization or municipality have access to information they have the privileges for and secure the privacy its users. In order to find solutions to how these goals can be achieved, the following research questions were formulated:

RQ1: What are the most prominent threats, vulnerabilities & privacy concerns towards web applications at the moment?

RQ2: What user-related information is collected and handled by web applications that can be used to identify a user?

RQ3: What is the current state of security and user privacy of the 'Vihjaa' application and how can it be improved?

RQ4: How can we assess and verify that the application meets the requirements set for it by EU legislation?

The main goal of this thesis is to conduct security, privacy and legislation adherence assessments of the target application, Vihjaa. As a byproduct, we will contribute information that could be useful for future research when answering the research questions. Mainly, for RQ1 we will provide a list of the most prominent threats and vulnerabilities for web applications as well as privacy concerns from the point of view of end-users as well as organizations. For RQ2, we will provide a table of data items collected by web applications as well as a table ranking how critical those different items are in the process of identifying a user. For RQ3 and RQ4 we will provide information on how the most prominent web application vulnerabilities can be evaluated and mitigated and how the legislation requirements can be interpreted and assessed. Furthermore, as alluded to previously, another result of

this thesis is a framework that can be utilized to audit web applications for the currently most prominent vulnerabilities and assess whether or not they comply with the General Data Protection Regulation (GDPR) or the Whistleblower Directive. Overall, the results of this thesis will be of most interest to software developers, software architects, security researchers, and law practitioners with interest in web application privacy and privacy legislation.

1.3 Methodology

In order to answer this thesis' research questions, the main research method for Chapters 2, 3, 4, and 5 is a literature review. The main objective of those chapters is to provide background information that will be used later on the thesis during the assessment chapters. The methodology used for the security and privacy assessments are explained more in-depth in Section 6.1, while the assessed items are listed and discussed in Sections 4.3, 5.3, and 5.4. For the legislation adherence assessments conducted in Section 7, we base our assessments solely on what is written in the General Data Protection Regulation and the Whistleblower Directive. The writer of this thesis is not a student of law and as such, the legal requirements are being assessed strictly from the point of view of a software developer. This means that shall any of the legal requirements be incorrectly interpreted, those sections can be disregarded. The lists of requirements that were interpreted to affect web applications of the same nature as the target application of this thesis from EU legislation are provided in Sections 2.2.2 and 2.3.2.

1.4 Thesis Structure

The rest of the thesis is organized as follows. Chapter 2 introduces the applicable European Union legislation that will be assessed later on in this thesis as well as their requirements for the target application, Vihjaa. Chapter 3 provides an overview of Vihjaa's features as well as discusses some of the main technologies utilized in it to assist later on in the security assessment section. Chapter 4 first discusses the current web application threat and vulnerability landscape in general, then moves on to list off the presently prominent threats and vulnerabilities and their mitigation measures. Chapter 5 focuses on privacy, and reviews the kinds of privacy legislations that are currently enacted in the United States, the European Union and China. Furthermore, we explore web application personal data collection, types of personal information, how critical a piece of personal information collected by a web application is to identify a specific user, end-user and organizational privacy concerns for the application as well as how user privacy can be secured from a technical standpoint. Chapters 6 and 7 detail how the security, privacy, and legislation adherence assessments are conducted, what the results of those assessments are, and what suggestions are given to address the identified issues.

2 Relevant European Union Legislation

As discussed in the introduction chapter, the target application of this thesis is subject to regulation under the European Union (the "EU"). Those regulations being the Whistleblower Directive as well as the General Data Protection Regulation, since the target application processes personal data [2]. As the target application is meant to be used as an effective internal channel for reporting breaches, the importance of protecting the personal information of whistleblowers who "wish to stay anonymous" cannot be understated.

In this chapter, we further examine the current European Union whistleblowing and privacy regulations, what requirements they set for software of this nature, and what type of information they classify as personal data. This information will be useful later on in the thesis as we look at the information types collected by web applications in Section 5.2, as well as in Section 6.3 when analyzing the current state of privacy of the target application. In Section 2.1, we list what applicable European Union legislation exists that should be taken into consideration in the context of this thesis and understand what the differences between EU directives and regulations are. In Sections 2.2 and 2.3, we look at the Whistleblower Directive and the GDPR separately to partly answer RQ4 by listing the requirements set for the target application in Sections 2.2.2 and 2.3.2.

2.1 Applicable European Union Legislation

In the beginning of this section, it is important to understand the differences between legal acts set by the European Council to achieve objectives set out in the treaties of the European Union. In this thesis, we mainly focus on the Whistleblower Directive and the General Data Protection Regulation (the "GDPR"). The main difference between a directive and regulation refers to which EU member countries they apply to, and whether or not they are binding. In terms of regulation, it must be applied in its entirety across the European Union, as was the case with the GDPR. As for directives, individual member countries of the EU devise their own laws to achieve the goal set out by the directive before a given deadline set by the European Parliament, as was the case with the Whistleblower Directive. [3] The GDPR came into effect on the 6th of May in 2016 [2], whereas the Whistleblower Directive had its deadline on the 17th of December, 2021 [1]. In Finland, the enactment of the directive was delayed due to the regulatory extent of the directive and the responses it received from the Finnish Parliament according to the Finnish Ministry of Justice. As it currently stands, the Finnish government's proposal *HE 147/2022 vp* was presented to the parliament in September of 2022. The laws are meant to enter into force as soon as possible. [4]

Another noteworthy piece of EU legislation to mention in the context of this thesis is the 'EU Cybersecurity Act', (EU) 2019/881, which was enacted in 2019. It aims to strengthen the current European Union Agency for Cybersecurity (ENISA) and to create a framework for voluntary cybersecurity certification of IT products, services and processes [5]. The certification scheme's goal is to improve the EU's internal market by increasing the level of overall cybersecurity as well as increase trust towards those goods that have been certified using the framework. However, as this regulation is still new, the framework is still a work in progress and as such cannot be utilized for the purposes of this thesis. It does nevertheless provide a

subject for future research and will be mentioned again at the end of this thesis.

The target application of this thesis, Vihjaa, was designed to be utilized as a reporting channel that fits the internal channel requirements as posed by the Whistleblower Directive Articles 7, 8 & 9 [1]. In Chapter 3, a more detailed description of the application and a short overview of its features will be given. During the reporting process, the application will gather and save necessary personal data about the whistleblower, such as a unique identifier for storing the report in a database connected to the application. As required by the Directive, the whistleblower can decide if they wish to submit a report with their name attached or anonymously [1]. The reason why discussion about the GDPR is included in thesis is due to the fact that Vihjaa processes personal data collected from its users. For instance, the GDPR classifies an individual's name as personal data among other information. Additionally, Vihjaa is operated and used inside the EU and as such, storing or processing any personal information falls under EU legislation [2].

2.2 Whistleblower Directive

The most well known whistleblower of the past decade in the technology field is Edward Snowden, who after years of working under the United States' National Security Agency (NSA), turned over highly classified documents to journalists in 2013 for them to publish. The documents detailed numerous global surveillance programs that were established with the cooperation of telecommunication companies as well as European governments. As a result of his actions, the United States Department of Justice charged him with espionage and theft of government property and revoked his passport. He fled the charges to Russia, where he later gained the right to asylum and where he still lives to this day. [6] The information contained in the documents sparked a global discussion on information privacy, mass surveillance, and government secrecy that is still ongoing today. Nowadays, we can see the effect

that conversation had on the European Union and how the EU legislation mentioned in this thesis came to be. The Whistleblower Directive and the GDPR can be both viewed as responses from the EU to address the issues raised by Snowden and his whistleblowing actions, as well as the discussions that followed.

As mentioned earlier in this thesis, the purpose of the Whistleblower Directive is to safeguard whistleblowers who wish the report on breaches of EU law against retaliation from their employees or colleagues. In the Directive itself, the actions of whistleblowers are described as playing a key role in exposing breaches and safeguarding the welfare of society. The scope of the whistleblowing report does not matter, as the most important thing is to guarantee that whistleblowers feel protected when reporting on a breach. The Directive provides legal protection for the whistleblower, if they qualify for the following criteria: [1]

- they have reasonable grounds to believe that the information they report is covered by the legislation and true at the time of reporting;
- they report the breach to the competent authorities using the provided internal or external channels. Whistleblowers are encouraged to report internally (within the organisation) first, where the breach can be addressed effectively internally and where they consider that there is no risk of retaliation. However, they can choose whether to report first internally or to directly report externally to the competent authorities.

The Directive covers reports on breaches of rules in areas such as product safety and compliance, consumer protection and the protection of privacy and personal data, security & information systems. Additionally, breaches affecting the EU's financial interests, the internal market, national corporate tax rules or EU's competition laws are also included. The areas covered by the Directive are extensive as are the varying regulations that apply to those areas. The range of people this Directive covers is extensive, as it covers workers in both the private and public

sectors, including people that submit a report after their work-based relationship has ended. The job position of the reporting person is irrelevant, as the Directive covers persons from employees, shareholders and management to volunteers, trainees and job applicants. Furthermore, persons that help whistleblowers in a confidential manner or that are connected to them and might suffer retaliation at work because of it are also protected.

2.2.1 Whistleblower Directive's Reporting Channels and Non-Compliance Penalties

The Whistleblower Directive outlines two different types of channels to be used as reporting arrangements: internal and external channels, both ensuring confidentiality. For internal channels, all legal entities in the private sector with 50 or more workers must establish an internal reporting channel that corresponds to the company's size and the level of risk their activities pose to public interests. Moreover, essentially all public entities must also set up an internal reporting channel. Entities that are exempted from setting up internal channels are private and public entities with less than 50 employees and municipalities with less than 10,000 inhabitants. External channels are to be set up by appropriate national authorities or authorised third parties on behalf of legal entities in the private or public sector, provided they can offer appropriate guarantees of respect for confidentiality, data protection, independence and secrecy. [1]

The Directive also mandates EU member countries to take action in order to oversee that legal entities in their jurisdictions implement the reporting arrangements if they are included in the criteria. Those actions include ensuring that the appropriate reporting channels are in place once the Directive has been adapted into local law, taking the necessary measures to prevent retaliation against whistleblowers & respecting the right for a fair trial, presumption of innocence and rights of

defence and effective remedy for persons concerned by the allegations detailed in a whistleblowing report. [1]

Non-compliance towards the Whistleblower Directive by, for instance, not implementing the required reporting channels, retaliating on a whistleblower, attempting to hinder reporting or revealing the identity of a whistleblower will result in penalties. Unlike the GDPR, the Whistleblower Directive does not however set minimum penalties for non-compliance. It is up to each Member State to establish proportionate, effective, and dissuasive penalties to deter entities from hindering the whistleblowing process or from infringing on a whistleblower's rights. Furthermore, the Directive does not determine whether penalties should be based on civil or penal law, so that choice is also up to each Member State. [1]

2.2.2 Whistleblower Directive's Requirements for the Target Application

Given the context of this thesis, the last violation of the Whistleblower Directive that was listed in the previous paragraph is of most interest to us. Since the target application of this thesis handles personal information of whistleblowers, it is essential that the target application does not under any circumstances provide that information to non-authorised persons. This subject matter leads us to the general requirements that the Directive has for internal reporting channels. In this section we will list the requirements that can be seen to affect service providers and the software they provide to entities to be used as internal reporting channels. The requirements themselves are worded in a generic way without any technical details, because an internal reporting channel does not have to be an application. For example, a 'complaint box' operated by a designated party inside an organization or municipality that meets the requirements of the Directive does suffice [1]. However, the additional features and the added layer of anonymity provided by an application

used as a internal reporting channel makes the whole process more efficient. Especially for a large organization or municipality, this approach provides a streamlined process to both the whistleblower as well as the party receiving the report.

In order to efficiently refer to single requirements set in these legislations in upcoming chapters, we will list them in this subsection as WD[X] and in Subsection 2.3.2 as GDPR[X]. Article 9(1) of the Whistleblower Directive lists the requirements for an internal reporting channel for it to qualify to be used in the process of internal reporting. For clarity, the following list only includes requirements that affect the reporting channel itself, not the reporting process. From the requirements, the ones that can be interpreted to affect applications used as internal reporting channels are the following: [1]

WD1: channels are to be designed, established and operated in a secure manner,
WD2: channels ensure the confidentiality of the identity of the reporting person,
WD3: the channel prevents access thereto by non-authorised staff members,
WD4: the channel must provide a method of acknowledge after a report has been received as well as a way to provide feedback on the report,
WD5: reporting via the channel can be done either in writing or orally.

According to WD1, the target application must be designed, established and operated in a secure manner. This means that during the design and development phases of the application, security has been a priority and that when the Vihjaa service is sold to a customer, it can be set up and operated securely. This is further expanded upon by WD2 and WD3 where it is emphasized that the identity of the whistleblower must be protected and that access by unauthorized parties must be disallowed. Requirements WD4 & WD5 introduce specific features that must be present in the application for it to be used as an internal reporting channel. Those requirements will all be assessed in Section 7.1.1, while an overview of the target application's features will be presented in Section 3.1.

2.3 General Data Protection Regulation

In Annex Part I, Section J of the Whistleblower Directive, it is detailed that: "Protection of privacy and personal data, and security of network and information systems" is subject to regulation (EU) 2016/679 "The General Data Protection Regulation". [1] This means that any personal data collected throughout the process of a person using a reporting channel is subject to the GDPR. When it was enacted in 2016 and later enforced in 2018, the GDPR brought major changes to the European Union's existing privacy and security laws which effected companies and organizations worldwide, as long as they targeted or collected data related to EU citizens. The first draft of the GDPR was created in 2012 and after 4 years of debating, it replaced the "Data Protection Directive" 95/46/EC that was enacted in 1995. [7] The regulation itself includes hundreds of pages' worth of requirements for organizations, however most them are not relevant in the context of this thesis. We will nonetheless look at the background and aim of the regulation and its main points from the viewpoints of a EU citizen and a business operating in the EU. Afterwards, we will look at what kind of data the GDPR classifies as private data to aid us answer RQ2 in Chapter 5.

The main motivation behind enacting the GDPR was to overcome the obstacle posed by legal uncertainties stemming from fragmented data protection laws across EU Member States. This led to issues concerning attempts to pursue economic activities at EU level as well as an imbalance of competition on the EU market. On the other hand, increasing the personal data privacy of EU citizens was also noted to be a priority. In the early 2010's, lawsuits were filed against companies that were infringing on a user's data privacy and information was made public by whistleblowers which showed governments as well as companies breaching personal data privacy. With the GDPR, the EU hoped to regain trust from its citizens towards reasonable treatment of their personal data in an attempt to boost the digital economy across

the EU's internal market. With the unification of data privacy regulation amongst Member States, the GDPR will also prove advantageous to businesses looking to operate on the EU market even with its strict requirements. [8], [9]

2.3.1 GDPR for EU Citizens and Business Entities

For EU citizens, the most important changes introduced by the GDPR were the strengthening of existing rights and providing new rights to enable citizens to have more control over their personal data. The main of those changes being providing citizens easier access to their data, the right to data portability, the right of rectification, the right for data erasure or 'the right to be forgotten', the right to know if your personal data has been hacked and the right to lodge a complaint with data protection authorities. [2] Out of those six principles, the right to be forgotten (RtbF) and providing easier control over personal data caused the most heated debate among the legal, academic and business world. The former grants citizens the right of having their data deleted if they no longer want to have it processed and there is no legitimate reason to keep it. The latter meant giving citizens the right to consent and to revoke their consent regarding their information being processed and to also receive information on how and where their data is processed. The heated debate around these principles stemmed from the fact that they both have a pivotal impact on how personal data could be processed in the future in the era of big data and Internet of Things. [7] The impact can be however be mitigated by data controllers, if they implement the technical and organizational measures to record what personal data they keep and who are the recipients of that data [9].

For businesses that are based in the EU or that do business in the EU, the list of requirements is quite extensive. Additionally, the penalties for violating the GDPR are very high as they max out at 20 million euro or 4% of the company's global revenue, depending on whichever is higher. Compared to the GDPR's predecessor

Data Protection Directive or the Whistleblower Directive, these penalties can have a serious impact on a company's financials. The GDPR's requirements towards a company can be split into the following categories: [2]

1. Lawful, fair and transparent processing of data:
    - Have a legal basis and take responsibility for your data processing activities, inform your data subjects about the data processing in your privacy policy and conduct information audits to detail what data you process and who has access to it.
2. Limitation of purpose, transfer and storage:
    - Do not process or collect data you do not have a legitimate purpose for, mandate that none other than necessary personal data is requested from data subjects and delete the personal data once the legitimate purpose has concluded. In the case you utilize third party data processors, sign data processing agreements with them.
3. Data subject rights and consent:
    - Data subjects have the right to file an information request to a data controller to know what data they have collected on them and what the company does with that data. If the data controller has the intent to process personal data for any other than the legitimate purpose it was collected for, clear consent must be asked from the data subject.
4. Data security:
    - Design products with the principle of "data protection by design and by default", encrypt and pseudonymize or anonymize personal data where ever and when ever possible, conduct data protection impact assessments and have a process in place to notify authorities if a data breach takes place.

5. Data Protection Officer, compliance and training:
    - Designate a Data Protection Officer if there is a significant amount of personal data being processed, appoint an employee to ensure GDPR compliance in your company, raise awareness about GDPR compliance and create training for employees. Additionally, if your company is located outside the EU, you must appoint a representative within one of the Member States.

In the GDPR, "personal data" is defined as the following under Article 4(1): "'personal data' means any information relating to an identified or identifiable natural person ('data subject'); an identifiable natural person is one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or to one or more factors specific to the physical, physiological, genetic, mental, economic, cultural or social identity of that natural person". [2] This definition will be used when referring to personal data during this thesis. Other legal terms that were used in the aforementioned list describing the GDPR's requirements towards companies that will also be used in Subsection 2.3.2 while listing the GDPR's requirements, are defined under Article 4 as the following: [2]

- *Data processing* is any action manual or automated action performed on data.
- A *data controller* is the entity that decides why and how personal data is processed.
- A *data processor* is a third party that assists the data controller in processing personal data, such as a cloud server operator.

2.3.2 GDPR's Requirements for the Target Application

In this section we will list the requirements that GDPR places on the target application of this thesis as well as succinctly discuss the impact it has had on software

development collectively. As with the Whistleblower Directive in Subsection 2.2.2, listing the requirements of the GDPR that affect the target software of this thesis is important and will assist us answer RQ4 in Chapter 7. First, we list the requirements that can be interpreted to apply in the context of our target application. Afterwards, we discuss what those requirements mean in practice. From the requirements, the following affect software that are the same in nature as the target application of this thesis: [2]

GDPR1: Data subjects must consent to their data being processed, their consent must be collected into abstract data structures and special considerations must be implemented for children and their guardians (Art. 6-8)

GDPR2: Data subjects can easily request and receive all the information you hold on them (Art. 15),

GDPR3: Data subjects can at any time correct or update incorrect or incomplete information (Art. 16),

GDPR4: Data subjects can easily request to have all their personal data deleted (Art. 17),

GDPR5: Data subjects can easily ask for you to restrict processing their data (Art. 18),

GDPR6: Data subjects can request and receive their personal data in a format that can be transferred to another company (Art. 20),

GDPR7: Data subjects can easily object to their personal data being processed (Art. 21),

GDPR8: Implement appropriate technical and organizational measures ensuring that only personal data which is needed to be processed is being processed, also referred to as data minimization (Art. 24 & 25),

GDPR9: Take data protection into account at all times, from development to production, referred to as "data protection by design and by default" (Art. 25),

GDPR10: Each controller shall maintain a record of processing activities under its responsibility (Art. 30),

GDPR11: The data controller and data processor shall cooperate, on request, with the supervisory authority in the performance of its tasks (Art. 31),

GDPR12: Utilization of pseudonymisation or anonymization and encryption wherever and whenever possible (Art. 32),

GDPR13: Have a process in place that notifies authorities and data subjects in the event of a personal data breach (Art. 33 & 34),

GDPR14: Conduct a data protection impact assessment (DPIA), if the data being processed is "likely to result in a high risk to [a person's] rights and freedoms" (Art. 35).

Requirements GDPR1 - GDPR7 refer to the new rights granted to EU citizens as mentioned in Section 2.3.1 and require the data controller or data processor to implement features into their systems that enables data subjects to give their consent for processing and gives them control over their personal data. GDPR8 - GDPR13 refer more to the requirements for businesses regarding the GDPR as was also discussed in Section 2.3.1. They require the data controllers to put in place the appropriate measures for personal data processing, logging that processing and also reporting possible data breaches to the appropriate authorities. Furthermore, both GDPR9 and GDPR12 specifically require the data controller to implement new features into their software in order to reach compliance. Out of the two requirements, GDPR9 is more extensive as it requires the entire chain from software architects to developers to include data protection measures during every step of the software development process. Out of the requirements, GDPR14 is the one that elicits further conversation. The GDPR gives some examples to what kind of data is likely to result in a high risk to someone's rights and freedoms. Included in this data are, for instance, personal data relating to criminal convictions and offences, data concerning vulnerable data subjects and data being processed on a large scale [2]. However, when taking into account that the target application of this thesis will only collect basic personal information about a whistleblower, and only when they consent to

it, we can conclude that GDPR14 does not affect our target application. We will nonetheless discuss the need of conducting a data protection impact assessment in Chapter 7.

Considering the subject matter of this thesis, it is also fitting to briefly allude to the effects the GDPR has had on software development in general. In a paper released by Jensen et al, it is noted that most proprietary software released nowadays do not provide any insight into the way they processes data, what data is being processed, or what data passes through the system in general. The paper then goes on to discuss how the demand for "GDPR-specific tools for process mining, visualization, and documentation, with a strong focus on processing personal data within these processes" has become evident. The solution the paper suggests for this issue is a data annotation approach that uses data visualization, data management, and standardization. [10] A paper by Hjerppe et al further elaborates on the need for GDPR-specific tools by claiming that while the media has mainly focused on new business constraints introduced by the GDPR, the different software engineering and architectural constraints are far more important in practice. Applying solutions such as implementing new user interfaces, isolating personal data, introducing access control mechanisms and adding logging, pseudonymization and annotations to existing architecture are efficient methods to aid in applying with the GDPR. [11] These solutions can also be applied to new software products, as requirement GDPR9 requires that software has to be developed with "data protection by design and by default". While the requirements that the GDPR has introduced are numerous, it seems that the effect it has had on software development is not that substantial. By incorporating new development principles and using the solutions recommended by both Jensen et al and Hjerppe et al, compliance with the GDPR can be achieved efficiently.

3 Target Application Overview and Technical Background

The target application, Vihjaa, utilizes multiple third party components in its software stack like many other web applications. In short, the term 'software stack' or 'solution stack', refers to the collection of independent components that work together to execute the software. Some of the stack's components run the back-end processes, some perform data transferring and store data, while others run the front-end presentation that is shown to the end-user. The main objective of this chapter is to provide a base level understanding about the components and technologies used to run Vihjaa. This will be useful in Chapters 6 and 7 when answering RQ3, as most of these components and technologies will be referred to and analyzed in those chapters. Since one of the primary goals of this thesis is the analysis of Vihjaa's state of privacy and security, the following subjects have been selected with those in mind. We begin the chapter by looking at the overall software stack of Vihjaa and a brief overview of its functionality. Subsequently, we review some of the key protocols and components utilized by Vihjaa that will provide us with useful information for the upcoming chapters. First, we will examine transport layer security (TLS), hypertext transfer protocol secure (HTTPS) and REST APIs and how they function at the core of Vihjaa by transferring data between the client, the application, and the database. Afterwards, we continue to Amazon Web Services (AWS) and its Key

Management Service (AWS KMS), which provide us with a secure cloud computing platform that can be used to present each customer with their own encrypted MongoDB database. Lastly, we examine how user access control is implemented in Vihjaa with Trivore ID and how the front-end is powered by the Vaadin framework.

3.1 Vihjaa's Software Stack and Overview

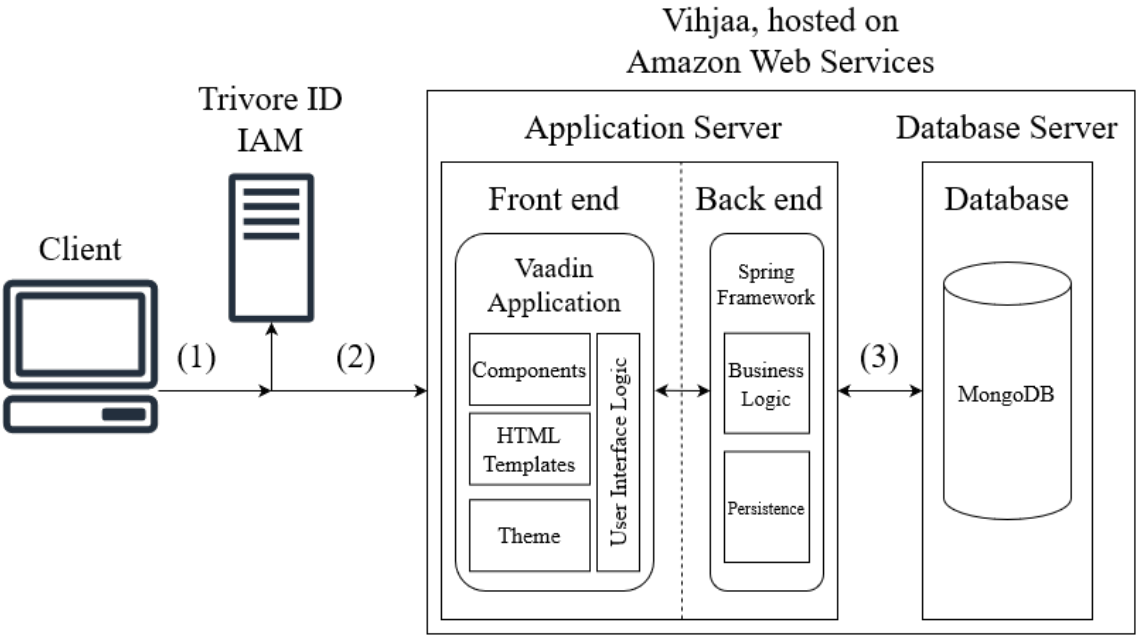

Figure 3.1: The Vihjaa architecture overview

The main components of Vihjaa can be seen in Figure 3.1. A user operates their device and accesses Vihjaa on their browser, represented in the figure as the client. First, the client connects to the application and will be asked to login, which is represented on the figure as (1). User authentication is accomplished using the Trivore ID Identity Access Management platform, that is hosted independently by the Trivore Corporation. Once the user is authenticated, they are allowed to use the application as represented by (2) in the figure. The organization information is fetched from the database, and all subsequent actions such as viewing or submitting

reports on the application will be registered to the database, represented in the figure as (3). Vihjaa is separated into two applications that technically function the same way. The admin panel, which requires users to log in, is mostly used by upper management in an organization to handle things such as incoming reports, user accounts, and the Vihjaa subscription. The second application, which is the reporting application, does not require logging in as it has to provide reporting persons the choice of reporting anonymously. Its main functionality is to act as a form that a reporting user fills in and then submits. In this section we will refer to Vihjaa as a single application for the sake of clarity, and then make a more elaborate distinction between the two applications in Section 6.2. Vihjaa's software stack is as follows:

- The operating system on the Amazon Web Services Elastic Compute Cloud (EC2) instance is a version of Linux.
- The application is running on the Spring framework, which has built-in Apache Tomcat web server support.
- The programming language used for the application is Java.
- The front-end and back-end communicate with each other using the Vaadin framework, which also provides the user interface logic, components, HTML templates, and themes for the front-end.
- Each organization's database is a MongoDB instance running on a separate server, that is accessed by the application's backend over an encrypted connection.

As the aforementioned Spring Framework will not have a subsection dedicated to it in this chapter, it shall be briefly discussed here. The Spring Framework is an open-source application framework that provides comprehensive and modular infrastructure support for developing enterprise-ready Java applications. One feature

that is lacking from the Java platform is the ability to organize different basic building blocks into a coherent whole. The Spring Framework addresses this issue with a component called Inversion of Control (IoC), that allows composing disparate components into a fully working application that is ready for use. In a nutshell, Spring functions as a dependency injection container that manages the different classes and dependencies of an application while allowing the developers to focus on the application's business logic. [12]

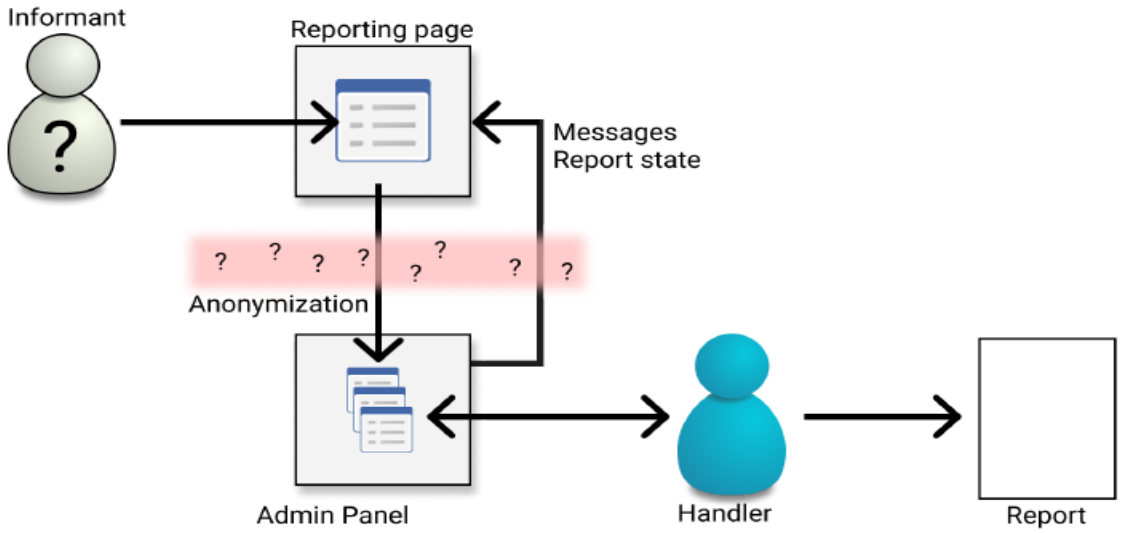

Figure 3.2: Representation of vihjaa's main functionality

The main functionality of Vihjaa is shown in Figure 3.2. In general, the Vihjaa web application is a service that collects reports of misconduct from reporting persons involving an organization or municipality. Vihjaa allows reports to be made anonymously and, if necessary, the option for further reporting to the relevant authorities. As an example, an employee of an organization can file a report by logging into the Vihjaa web application and then accessing the reporting page. The employee, represented in the figure as the "informant", files the report using the reporting page and their information gets anonymized if not otherwise instructed. After the report is submitted, it shows up in the "Reports" tab for the appropriate

handler depending on the report's contents or the user's selection. The employee that filed the report will receive updates on the state of their report to their contact email as well as what action will be taken once the report has been resolved by the handler. In Vihjaa, channels are used as a grouping method to distribute the processing of reports. More specifically, each handler can see reports from one of more channels, and each channel can have one or more handlers. By default, reports arrive in the channel that has been defined as the default channel, however reporting persons can be given the option to choose which specific channel the report is filed to. Channels can be used to divide work by topic or by responsibility, such as "conflict of interest", "occupational health and safety", or "abuse". The processing of reports can be further divided with user groups, which allows reports to be filed to handlers based on, for instance, their organizational branch or geographical region. The admin panel view of the Vihjaa application can be seen in Figure 3.3. Compared to the normal user view, the users that have administrator privileges have access to user administration, channel editing, user groups and organization settings.

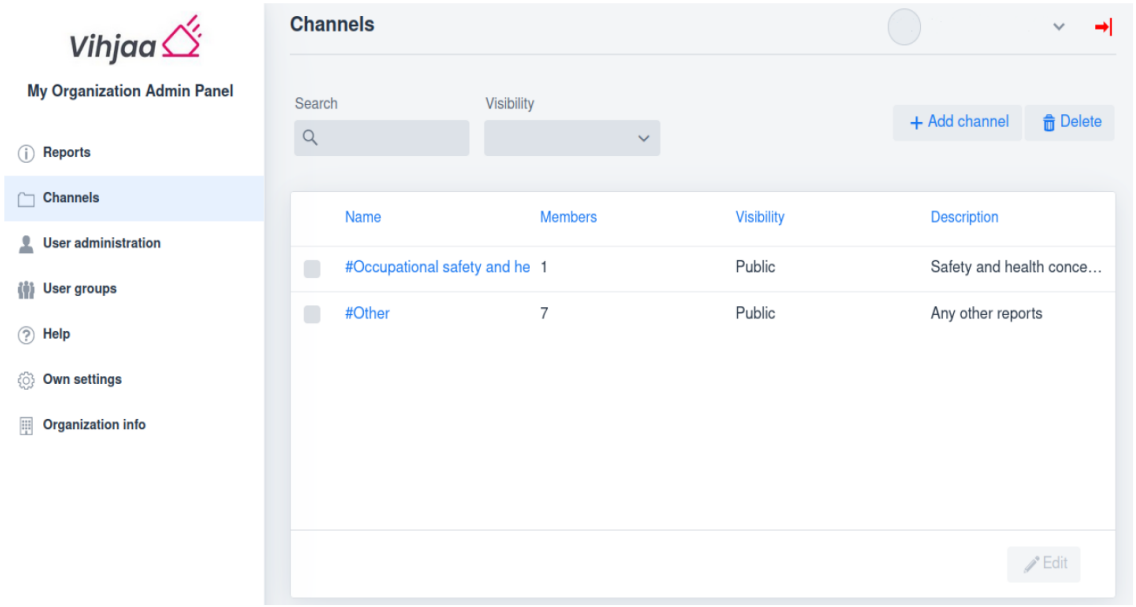

Figure 3.3: Vihjaa's admin panel

That concludes the basic functionality of Vihjaa as well as its software stack. In the following sections, we further examine some of the different components and technologies used by Vihjaa to achieve this functionality.

3.2 Transport Layer Security and Hypertext Transfer Protocol Secure

The Transport Layer Security protocol (TLS) is a *stateful* and *connection-oriented* client-server protocol. Stateful meaning that the client sends the server a request and then sends the request again if it receives no response. Connection-orientated means that before any data is transferred, a communication session or a semi-permanent connection is established. TLS is arguably one of the most widely used security protocols on the Internet as it provides confidentiality, integrity, and authentication to both the client and the server side. TLS is based on the Secure Sockets Layer (SSL) protocol that was developed by Netscape in 1994 and is as such sometimes still referred to as TLS/SSL. The first two versions of SSL had serious security flaws and after releasing SSL 3.0 in 1996, again with security issues, the development moved from Netscape to the IETF TLS working group. Basing the design mostly on SSL, but making key changes to procedures for key generation and authentication, TLS 1.0 was released in 1999. The newest version of TLS was released in 2018 and is called TLS 1.3. [13] Gaining a deep understanding about TLS' or SSL's protocol design isn't part of the scope of this thesis. However, a high-level overview on how a connection functions between a client and a server using TLS is useful in understanding how HTTPS operates which will further be used to analyze connection-based vulnerabilities in chapters 6 and 7 when answering RQ3.

TLS is divided into two parts, the handshake, and the record protocol. The handshake begins with the client sending a server a message including the highest

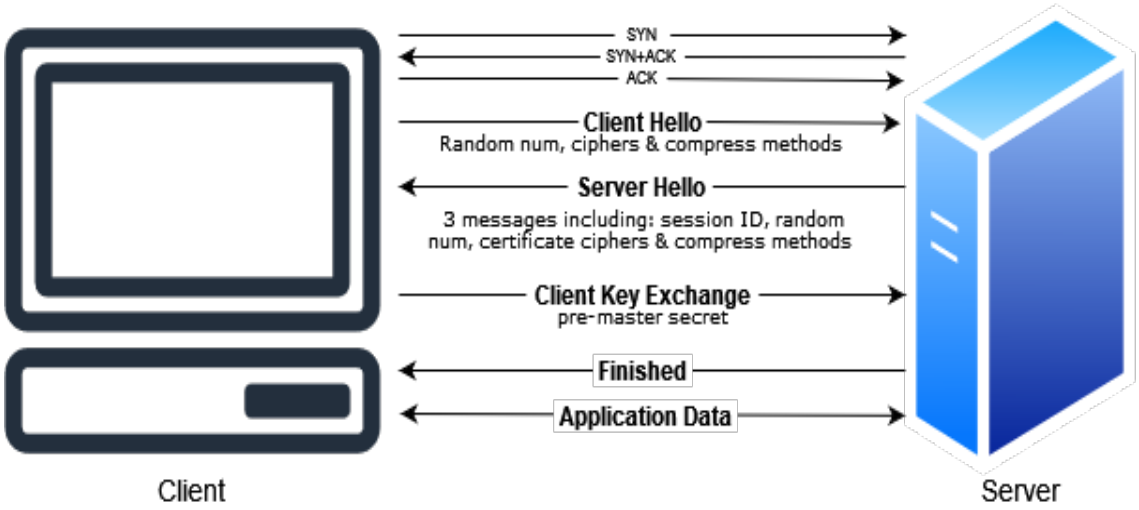

Figure 3.4: The TLS/SSL handshake

supported version of the protocol, client-generated random number and information about the compression method and cipher specification. The server responds with three messages, first of which specifies the used protocol version, cipher suite, compression method, session ID etc. Second message contains a list of certificates which the client will use to authenticate the server and a third message to tell the client to begin the key-exchange progress. The client then proceeds to send three messages to the server, which include a random pre-master secret encrypted with a public key included in the server's certificate and information what cipher specification should be used for hashing and encrypting the messages. The server returns a change cipher spec message and a finished message, which concludes the handshake. The second part of TLS, record protocol, is used to encrypt and authenticate packets when there is a need to send or receive data. [13], [14]

The target application is mainly accessed with a browser, so securing and encrypting the communication between the browser, meaning the client, and the server is important. To accomplish this, a protocol called Hypertext Transfer Protocol Secure (HTTPS) is used. HTTPS wraps the HTTP protocol in an encrypted protocol

layer, which is achieved with the aforementioned TLS. The reason why this is important is that communications in HTTP are conducted using plain text, meaning that if the packets are intercepted by an attacker between the client and server, their contents can be read. HTTPS adds the handshake from TLS before establishing a connection between the two parties and by employing a public key for encryption and a secret private key for decryption, the messages between the client and server are encrypted. As such, even if an attacker were to intercept packages between the client and server, their contents wouldn't be readable without the decryption key. [14], [15]

Considering the scope of this thesis, we will not be analyzing TLS/SSL or HTTPS for potential bugs or vulnerabilities. While there does exist attacks for TLS/SSL, for instance SSL stripping, Man-in-the-Middle attacks, certificate forgery and certificate stealing, they are not linked with the target application or its security. The use of these protocols is nonetheless essential in order to secure connections from our customers to the target application, so understanding their basics is relevant to the subject matter.

3.3 REST and REST APIs

Once a connection has been established between the client and the server, data and information can be exchanged between the two parties. This communication is mostly handled following the REST architecture. The representational state transfer (REST) is a software architectural style introduced and defined by Roy Fielding in his doctoral dissertation in 2000. As with the previous section, gaining a deep understanding of REST or REST APIs is outside the scope of this thesis, but grasping their basic concepts is useful in chapters 6 and 7 when answering RQ3. REST consists of a set of constraints of the Web's architectural style, outlining the requirements a system needs to meet to remain efficient and scalable. Fielding placed the

constraints into six groups, which were client-server, uniform interface, layered system, cache, stateless and code-on-demand. Soon after publishing, developers began to adapt the REST architectural style while developing web services. These web services are web servers that support the needs of a certain site or application and use application programming interfaces (APIs) to communicate between the client and server. [16], [17]

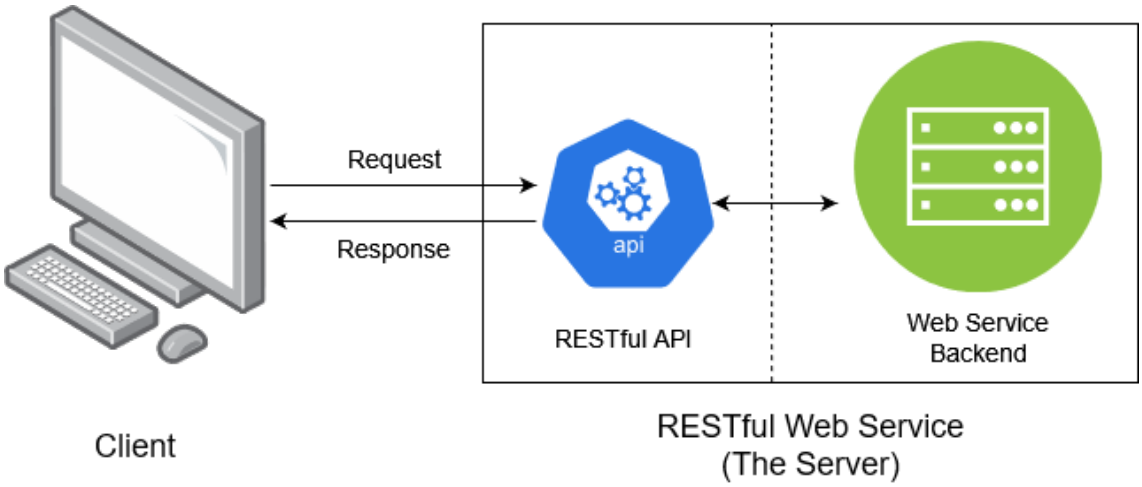

Figure 3.5: Communication between the client and the 'RESTful' web service

In general, the purpose of APIs is to expose a set of data and functionalities to allow and assist software to exchange information between them. When an API is designed using the REST architectural style, it is referred to as a REST API. When a web service uses REST APIs, it makes the web service "RESTful". [16] In Figure 3.1, the relationship of a client and RESTful web service is pictured. For instance, when the client makes changes while using the target application, the client sends a request to the server to the specific API that handles the kind of change the user wanted to make. After making this change in the back-end, the API responds with the updated data. In the case of the target application, the RESTful web service and its MongoDB database are both hosted on Amazon Web Service, which we will discuss shortly.

Unlike TLS/SSL or HTTPS, APIs are directly connected to the target application and are as such dependent on its logic. When adding an API to any software, it is important to define its purpose and set of operations that a caller is permitted to use. Later on in this thesis we will be analyzing the security of the target application's implemented APIs mainly from the perspective of three elements: authentication, available operations and input sanitization. Firstly, some APIs might be only accessible to users with a certain role. In the case of our target application for example, modifying organization information can only be accomplished with the role an administrator. This means that the API will authenticate the user before allowing the use of its set of operations. Secondly, while the individual operations in a certain API may be secured, a combination of them might not be. If a single API allows you to view and remove submitted reports, perhaps these actions be combined or chained to alter records in the database. Lastly, if an application allows its users to input text in any format, it is crucial to sanitize those input fields. These input fields can be exploited by for instance, inputting code that allows operations outside of that API's preset operations or by inputting a very large chunk of text that can consume all available memory. [18] These are the key elements of API security and they will be discussed again in Chapters 6 and 7.

3.4 Amazon Web Services (AWS) and Key Management Service (AWS KMS)

Before delving deeper into AWS itself, it is important to understand the concept of cloud computing as that is what AWS essentially is. Cloud computing refers to the on-demand delivery of IT resources and applications over the Internet using the pay-as-you-go pricing method. This means that your company pays another company to use their infrastructure in order to, for example, run servers, maintain

databases, compute calculations or simply store data. With cloud computing, you can provision the exact type and size of computing resource that you require to provide your products to customers while paying for what you use. It removes the need to setup and maintain the necessary infrastructure yourself, allowing you to deploy applications quickly to production while increasing the agility of the entire software development process. [19]

Amazon Web Services, a subsidiary of Amazon, was established in 2003 and has been offering cloud computing services since 2006. The service is based on a global infrastructure that allows customers to choose which region and 'Availability Zone' their resources are placed in. Each region is designed to function independently of other regions to ensure the highest possible fault tolerance and stability. The regions range from North and South America to Europe, Asia and Oceania and each of them include multiple Availability Zones. The zones are isolated locations inside the regions, connected to each other through low-latency links. [19] Considering the GDPR, keeping the data collected by operating the target application inside the EU is a high priority. This among other factors ultimately led BeanBakers Ltd to choose AWS to provide our customers with a secure and efficient platform to offer our product on.

In order to protect our client data while in the cloud, we utilize the Amazon Web Service Key Management Service (AWS KMS). AWS KMS is a managed service that allows the user to create and control encryption keys used to encrypt their data. To accomplish this, AWS uses Hardware Security Modules (HSMs) validated under the FIPS 140-2 validation program to safeguard and manage cryptographic keys and to perform functions like encrypting and decrypting. [20] Adding a layer of encryption like this is especially important since we have client data going back and forth from our back-end service and database, both operating in the AWS cloud ecosystem.

As with TLS/SSL and HTTPS, the security analysis of AWS or AWS KMS falls outside of the scope of this thesis. Amazon has their own security team as well as third-party researchers and a bug bounty program to find vulnerabilities in their systems. While AWS is an essential part of how we operate and offer our service, Chapters 6 and 7 will strictly focus on the security and privacy of the target application itself.

3.5 MongoDB

To store our clients' data in the cloud, for instance their organization information, whistleblowing reports and other data, we use MongoDB inside the AWS cloud. MongoDB is a database focused on scalability, flexibility, and versatility. Databases are organized collections of structured information stored on computer systems. Usually, databases are relational, meaning that the collection of stored data items in the database have predefined relationships between them. In a relational database, each row in a table is a record with a unique key identifier while the columns of the table carry the data's attributes which makes it easier to establish relationships between data points. MongoDB however is a document-orientated database, which replaces the "rows" of a relational database with a new model called the "document". The document model allows embedding documents and arrays inside documents in order to represent complicated hierarchical relationships within a single record. While relational databases are reliable, robust and scalable, they are lacking in terms of modern applications with huge and generally unstructured data. Additionally, non-relational databases do not use predefined schemas which makes adding and removing fields easier and more efficient. [21], [22]

At the core of MongoDB is the 'document' model, which is its basic unit of data. Essentially, documents are ordered sets of keys with associated values. In simple terms, this allows us to represent complex hierarchical relationships in a

single record, making record keeping easier and more efficient. There are also no predefined schemas for these documents, meaning that adding or removing keys or values becomes way easier. [22] For the purpose of our target application, having a scalable database that takes care of load balancing and loads across a cluster is important to serve our possible client base in the future. MongoDB is able to distribute data to multiple web services at the same time, which will be of use when multiple clients access their organization simultaneously.

Non-relational databases, also referred to as NoSQL databases, also have their own security flaws. Even though trying to find vulnerabilities in MongoDB itself is not in the scope of this thesis, we will regardless briefly look at how customer information and reports are encrypted and stored in the database in Chapters 6 and 7. In those chapters, we will also analyze if its possible to attack the database by exploiting the target application's logic or by using known MongoDB vulnerabilities.

3.6 User Access Control with Trivore ID

In order to authenticate and verify our users and let them access the correct organization and its information, user access control has to be implemented. In the case of our target application, we chose to partner with Trivore Corporation and to utilize their identity and access management platform (IAM). An IAM's main purpose is to ensure that the right people have the right amount of access to appropriate resources at the appropriate time and for the appropriate purposes. The user authentication process consists of three activities and tasks, which include identification, enrolment, and verification. The first two subtasks happen when a user's information is added and registered to the platform. The parameters and configurations of which depend on the IAM or service provider. The final task, verification, is performed whenever a user is trying to access a specific resource or service platform through the IAM. The verification procedure itself is the most crucial stage of any

user authentication system, as it verifies the user's identity and determines whether or not it can be authenticated. The correct implementation of an intelligent identity and access management platform minimizes the number of identities possibly linked to an individual inside an organizational network and simplifies the critical identity activities like audits and password resets. [23]

Trivore Corporation's product called Trivore ID is the IAM platform used for user identification and authentication for the target application. Trivore ID itself has been integrated into our target application meaning that every user gets identified and authenticated using the service before gaining access to the target application. The service is developed and hosted in Finland and can be tailored to utilize a multitude of federated authentication options during the user identification process, such as OpenID Connect, Apple ID, Google account, Microsoft account, or the Suomi.fi e-identification service. [24] For the purposes of providing a basic understanding of how user identification functions, we will briefly examine OpenID Connect (OIDC) and Open Authorization (OAuth) 2.0 protocols. Open Authorization 2.0 is a security standard that permits an application to access a person's data in another application. In other terms, OAuth 2.0 provides authorization for an application to access resources hosted by other applications on behalf of a user. For example, when an online service asks for permission to access a user's social media or phone contacts to find them on that service, OAuth 2.0 allows the user to give that permission to the service in order to accomplish that task on the user's behalf. OpenID Connect builds on OAuth 2.0 by adding a simple identity layer on top of the OAuth 2.0 protocol. This facilitates services to confirm the identity of a user depending on the authentication made by an Authorization Server as well as acquiring profile information about that user. For instance, when a user uses their Google account to access various services on the Internet, circumventing the need to create an account specifically for that service.

The functionality of OpenID Connect protocol in abstract can be seen in Figure 3.6. [25], [26]

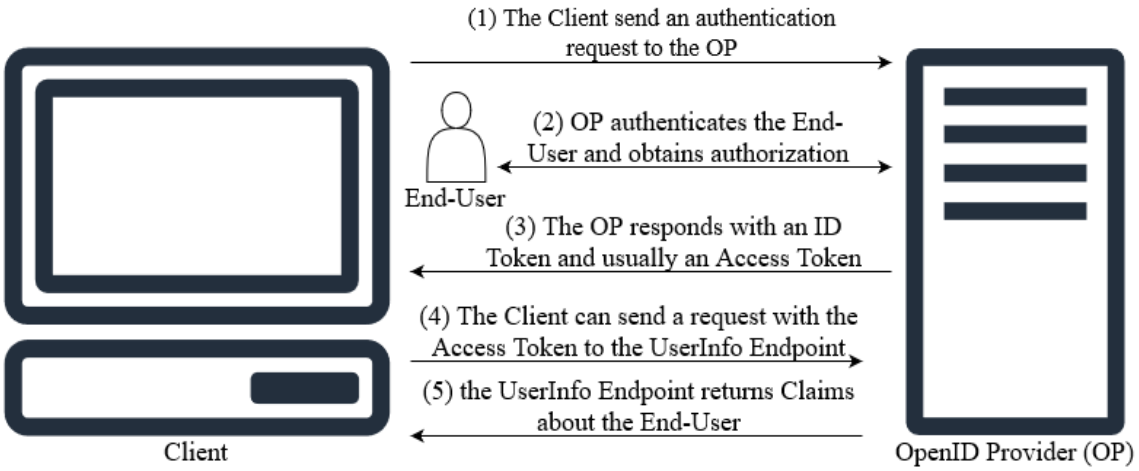

Figure 3.6: OpenID Connect functionality

The figure displays an example of an end-user being authenticated using profile data from an existing service. In the scope of this thesis, the client would be Vihjaa, OpenID Provider would be Trivore Corporation, and the End-User would be the person trying to access Vihjaa. Due to the nature of Trivore ID and it being a stand-alone service disconnected from the main application of Vihjaa, it will not be featured as a part of the security and privacy analyses in Chapters 6 and 7. However, user authentication and authorization will still be evaluated from the point of view of the target application.

3.7 Vaadin

Due to the target application being mostly written in Java, it was decided to utilize a Java framework for streamlining the development process and to increase efficiency by having a library of reusable components to build the application with. Since a part of the development team was already familiar with Vaadin and it fit the team's needs, it was selected. Vaadin is an open-source Java framework that

can be used to create and maintain web-based user interfaces, consisting of a client and a server side. This means that all changes made on the client side will be sent to the back-end, so that the server knows what is happening on the front-end at all times. However, the developer can choose to use either server-side or client-side programming model, or a mixture of both. The server-side model allows developers to program user interfaces entirely in Java or other JVM languages, such as Kotlin, Scala, or Groovy. The client-side uses a HTML Web Component model, which provides the developer with access to Vaadin's UI components library. All of Vaadin's components have both a server and client-side implementation that have a shared state maintained and communicated by the framework. This optimizes transmitting data between the two sides, as the framework only transmits information if a component's state has changed. The different library components can then be used and combined to create modern user interfaces for various web applications and web services. In this section we will briefly go over Vaadin's main functionality to gain a basic level of understanding of the framework that functions at the core of the target application. Firstly, when using the server-side programming model, everything is a UI component. The developer can choose whether to use premade components or layouts from Vaadin's component library, create new ones, or combine them. To illustrate this, below is a code snippet from an example Vaadin project. [27]

```
@Route("")
public class ExampleView extends VerticalLayout {
        public ExampleView() {
          add(new H1("Hello, World!"));
          Button exampleButton = new Button("Example Button");
        }
}
```

In this example the ExampleView application is a UI component that extends VerticalLayout, which is one of Vaadin's premade layouts. Using this layout will order all added components vertically. In the constructor, we add a H1 title component that reads "Hello, World!" as well as a button that reads "Example Button". The empty "@Route" annotation ensures that this view is accessible by the end-user. Vaadin provides many premade components from buttons and accordions to labels, different grids, and table views. In order to add interactivity to the ExampleView user interface, we could add a listener to the example button with "addClickListener()". This would track every time the button is pressed, making it possible to implement a variety of different features. Tracking both the client and server-side component statuses, Vaadin is able to provide a Single-Page Web Application (SPA), which means that the user does not need to reload the page to see their changes. [27] The main shortcomings of Vaadin are its lack of scalability and no way to implement complex UI elements or animations. Scalability here meaning that web applications made with Vaadin are not supposed to support a large number of users, like a social media page. [28] Overall, Vaadin is the correct choice for the target application considering its purpose, UI requirements and user base.

4 Web Application Security & Mitigation Measures

A web application is a program or software that is most often hosted on a server or servers owned by a cloud computing provider that can be accessed by using a web browser. The move towards web applications came with the Software as a Service (SaaS) movement, which shifted software away from programs that are downloaded and installed on a personal computer, onto a server that can be accessed with a web browser from anywhere at any time with an Internet connection. SaaS also has other benefits such as lower costs, integrations, scalability, updates, availability, and additional security. Advancements in networking protocols as mentioned in the previous chapter as well as new security features added to web browsers have made using software over the Internet a viable and efficient option. However, hosting software on a server and making it theoretically accessible by anyone who knows its address, also has it downsides. Malicious actors in the cyberspace have shifted their focus onto the logic and software stack of the web applications themselves. By taking advantage of integrations, bugs, different types of databases, open source dependencies and other vulnerabilities, these malicious actors have the best chance of succeeding in hacking into web applications. The size and complexity of web applications nowadays also plays into the hand of malicious actors as it increases the chance of a vulnerability or bug being found that can be exploited.

One of the focal points of this thesis is the security of web applications and how the threats and vulnerabilities concerning them can be mitigated. The goal of this chapter is to answer the first part of RQ1 as well as to provide information related to the security of web applications that will be useful to us when answering RQ3 in Chapters 6 and 7. In order to answer the security related problems of RQ1, we will discuss the current trends regarding web application security, the Open Web Application Security Project (OWASP) and their Top Ten web application threats and vulnerabilities list, recent threat intelligence reports, and vulnerability and cyberthreat mitigation measures. These discussions will provide us with lists of prominent vulnerabilities and threats as well as mitigation measures that can then be utilized while analyzing the target application in Chapters 6 and 7.

4.1 Current Web Application Threats and Vulnerabilities

According to the European Union Agency for Cybersecurity (ENISA) Threat Landscape report released in 2021, over 83% of the information sector's data breaches were caused by basic web application attacks, errors and system intrusions. Furthermore, 43% of all reported data breaches involved a web application while around 90% of all hacking vectors targeted them. [29] As web applications have become more advanced and able to handle large amounts of data, it comes as no surprise that they are one of the most prominent vectors for data breaches. However, before further discussing the current trends in web application security, it is important to understand the three main terminologies used in this section: [30]

- *Threats* are coercive actions where an attempt is made to make a target act as desired by a malicious actor by influencing the victim or negatively affecting their interests.

- *Attacks* are actions taken against a system in order to damage it or to disrupt its routine operations by exploiting its vulnerabilities whilst using various tools and techniques.
- *Vulnerabilities* are flaws in a system or in its design that allow a malicious actor to access data or information unauthorized, execute commands and/or conduct different kinds of denial-of-service attacks.

Malicious actors combine these concepts to breach web applications and to gain access to data, information or potentially other web applications through integrations. The main security problem with web applications lies at the core of their design and that is the ability of users being able to submit arbitrary inputs. Because the client used by a user to access a web application is not controlled by the application itself, users can input anything into the client and submit it to the server-side application. As such, it is up to the web application to assume anything submitted by the user could be malicious, and has to be checked. This main security problem manifests itself in various scenarios: [31]

- Users can interfere with any piece of data transmitted between the client and the server, which circumvents security controls implemented on the client's side.
- Users can send requests and submit parameters in different stages or sequences than the application expects, or not at all.
- Users can use tools that operate alongside or independently of web browsers to generate requests that could not be made with normal web browsers.

In Chapter 3, HTTPS and TLS/SSL were discussed and how using them improves the security of data transfers between the client and the server-side application. While using HTTPS does stop other users on the network from viewing or modifying data in transit, a user can still send their malicious inputs from the client

to the server-side application through the TLS/SSL tunnel. There are also other key problem factors that contribute to the often lacking security of modern web applications. Web applications are developed using third-party libraries, packages and components that make the development more efficient but also abstract the developer from the underlying technologies. These libraries and components might have vulnerabilities in themselves, or they might be in the logic that combines them into a working web application. New functionalities, underlying legacy technologies and a rapidly evolving threat profile also all add to the security concerns concerning web applications. While the industry has been able to deal with some of the aforementioned concerns, it is still a fact that web applications today still battle with vulnerabilities. Understanding the security threats facing web applications and finding effective and efficient ways to deal with them is a process that is going to last as long as we rely on web applications. [31], [32]

Until now, we have mostly focused on the client-side of a web application. However, an integral part of a web application is the server-side, which includes the web server and the web application's database that processes, stores, and transmits data to the application that is accessed by users. For a usual web server, the application logic runs on top of a software-based web server package, which allows a computer to act as a web server. The web server package, also known as web server software, mainly handles the HTTP requests from the web application, and manages different processes in order to make the application function. There are a couple attacks that target the final application server, most of which leverage the trust between the client-side web application and the server itself. For instance, Denial of Service attacks, Server Side Request Forgery, and injection attacks. Denial of Service attacks, or DoS attacks, come in many forms. Most commonly, the attack is a Distributed Denial of Service (DDoS), where a large network of devices floods a server which requests, making it inaccessible to its intended users. Another slightly less com-

mon DoS attack type is the regular expressions DoS (ReDoS) attack, which takes advantage of the "+" operator in the regular expressions (regex) language. Using the "+" operator in the regex language will test for one or more matches, rather than stopping at the first match found. Exploiting the "+" operator, an attacker can craft an "evil regex" pattern, which gets stuck on crafted input. When that pattern is then inputted into a regex parser on a web server, it can make the web server slow down significantly or even make it crash. Server Side Request Forgery (SSRF) allows an attacker to induce the server-side application to make requests to an unintended location. For example, the attacker might attempt to have the server make a connection to internal-only services within an organization that are not otherwise accessible from the outside. The last example, injection attacks, are mostly known for SQL injections that target a web application's SQL database. Earlier in this thesis we discussed the different database types and their differences, mainly the difference between a Structured Query Language (SQL) and a Not Only SQL (NoSQL) databases. As their name suggests, SQL injections target the former of the two and exploit the way SQL queries are made. Both Server Side Request Forgery and SQL injections are covered further in Section 4.3, so we will not discuss them more here. [31], [32]

Currently, there are three prominent trends concerning web application security. Firstly, old and well-understood vulnerabilities that have been utilized for over a decade still continue to appear, but their prevalence is gradually diminishing. Instances of such vulnerabilities are increasingly hard to find and more difficult to exploit. Secondly, there has a been a steady shift from attacks against the server-side of web applications towards targeting the application's users. While this method also takes advantage of deficits in the application, it usually involves some sort of interaction with another user in order to compromise that user's information regarding the vulnerable application. Thirdly, various changes in web application trends

such as new functionality, integrations, and cloud computing bring with them new attacks as well as variations of old attacks. As we will discuss in Section 4.4, it is clear that despite the changes in web applications and their security, some variations of old and known vulnerabilities are still prevalent today and that will not be changing any time soon. [31]

4.2 The Open Web Application Security Project (OWASP) Foundation

One entity behind enhancing web application security and providing different resources, education and publications for developers, security experts and website owners is the Open Web Application Security Project (OWASP) Foundation. The OWASP Foundation was founded in 2001 as a non-profit organization with the main objectives of improving software security, supporting projects, developing communities, and providing educational publications and resources. Their major publications include the OWASP Top Ten, the OWASP Software Assurance Maturity Model (OWASP SAMM), the OWASP development guide, and the OWASP testing guide. Arguably their most recognized and impactful publication is the OWASP Top Ten, which was first published in 2003, that lists the current trends in web application security to raise awareness about them. The OWASP SAMM project aims to develop a usable framework that helps organizations implement an application security strategy that is tailored to their organization's needs. The OWASP development and testing guides include information about best practices, practical guidance, tools and examples.

In the next section we will answer the former part RQ1 by examining the most prominent threats and vulnerabilities towards web applications at the moment. The latter part of RQ1, which focuses on privacy, will be covered in Sections 5.3 and 5.4.

The next section will concentrate on the most recent "Top 10 Web Application Security Risks" released by OWASP, which lists the most notable threats and vulnerabilities at the moment according to the dataset contributed to and collected by OWASP and their research. The reason why the top ten is featured heavily in this chapter, is due to it offering a fairly comprehensive answer to the former part of RQ1. Furthermore, the OWASP Foundation is considered a trustworthy and reliable source of information by the industry. Their top ten is even used as a standard in some software development companies. As mentioned previously, OWASP are a non-profit organization, meaning that almost everyone associated with the foundation is a volunteer. The people behind the top ten lists are security professionals from around the world and there are countless of articles and research papers evaluating and discussing their findings every time a new list is released. [33] Thusly, it seems reasonable to use their list as a base for answering RQ1 while comparing it to results of other research and white papers. It can of course be argued that the dataset contributed to and collected by OWASP is not representative of all web applications out there, and in order to combat that issue, OWASP selected two of the ten categories based off a community survey to balance out the categories and to cover categories that might not appear yet in the dataset itself. According to OWASP, they received their 2021 dataset from organizations that are testing vendors by trade, bug bounty vendors, or organizations that contributed internal testing data. Furthermore, the dataset covers over 500,000 applications which makes it the largest dataset used for a top ten list so far. [34] All in all, there simply is not another resource available that is as comprehensive and thorough as the OWASP Top Ten list and as such, the answer to RQ1 will be largely based on it.

4.3 The OWASP Top Ten 2021

In this section we will examine the most recent "OWASP Top 10 Web Application Security Risks" list, which was released on the 24th of September 2021 in order to answer the former part of RQ1. The top ten list combines Common Weakness Enumerations (CWEs) under categories created by the OWASP Foundation and ranks them based on findings from contributed data as well as a high-level community survey. The *Common Weakness Enumeration* is a list of software and hardware weaknesses, meaning bugs, flaws, vulnerabilities, or other errors that is managed by the Homeland Security Systems Engineering and Development Institute (HSSEDI). We will analyze the different categories on the lists, what type of CWEs are included in them, and what type of attack vectors are used to exploit them. The ten categories are each given incident rates to showcase how likely CWEs in that category are to occur, exploitability scores for how easy they to discover and exploit, and technical impact scores for how it affects the application, its data and/or its functions. Additionally, the total occurrence of CWEs included in that category is also shown. Below is a table consisting of the OWASP Top Ten 2021 list and related factors to each category:

Category Name	Avg Incidence Rate	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences
A01 - Broken Access Control	3.81 %	6.92	5.93	318,487
A02 - Cryptographic Failures	4.49 %	7.29	6.81	233,788
A03 - Injection	3.37 %	7.25	7.15	274,228
A04 - Insecure Design	3.00 %	6.46	6.78	262,407
A05 - Security Misconfiguration	4.51 %	8.12	6.56	208,387
A06 - Vulnerable and Outdated Components	8.77 %	5.00	5.00	30,457
A07 - Identification and Authentication Failures	2.55 %	7.40	6.50	132,195
A08 - Software and Data Integrity Failures	2.05 %	6.94	7.94	47,972
A09 - Security Logging and Monitoring Failures	6.51 %	6.87	4.99	53,615
A10 - Server-Side Request Forgery	2.72 %	8.28	6.72	9,503

Table 4.1: The OWASP Top Ten Web Application Vulnerabilities list 2021

A01 – Broken Access Control

Beginning with the top rated category, *A01:2021 – Broken Access Control* refers to failures in access control policies that allows unauthorized users to have access to data or information that they are not supposed to. In a nutshell, access control is utilized in applications to limit who can see what in order to minimize risks to a company or organization. Access control is separated into physical and logical types, former referring to limitations in the physical world that restrict access to certain areas, while the latter refers to limitations in computer networks and systems. Besides information disclosure, failures in access control may also lead to the modification or destruction of data and in some cases the execution of a business function or functions that a user does not normally have access to. Broken access control consists of multiple attack vectors, for instance bypassing access control checks, elevation of a user's privileges, metadata manipulation through access control tokens, CORS misconfigurations that allow access to restricted APIs, and impersonating another user by providing their unique identifier among other attack vectors. Out of all of the ten categories, broken access control was clearly the most common with over 300,000 occurrences and an average incident rate of 3.81 %. Some notable CWEs included in this category are CWE-200: Exposure of Sensitive Information to an Unauthorized Actor, CWE-201: Insertion of Sensitive Information into Sent Data, and CWE-352: Cross-Site Request Forgery. [33], [35]

A02 – Cryptographic failures

The second category on the list is *A02:2021 – Cryptographic Failures*, which was previously known as "Sensitive Data Exposure". The reason for the name change is a decision by OWASP to focus more on root causes rather than symptoms, even though the list is still a mix of the two. Focusing on the root cause rather than the symptom whenever possible is more logical in the sense of being able to provide

guidance for identifying and mitigating them. For instance in the case this category, cryptographic failures are a root cause which can lead to the symptom of sensitive data exposure. Cryptographic failures refer to failures related to cryptography or the lack of it. For example, web applications that transmit data in clear text, use old or weak cryptographic algorithms or protocols, use deprecated hash functions like MD5 or SHA1, or use default or weak crypto keys that are re-used or poorly managed. An example attack scenario for this category, for instance, would be an attacker intercepting data from a web application mid-transit, after which attacker would then proceed to read its contents if it is not encrypted. While the total occurrences for this category is the lowest in the top four, it has the highest average incident rate with 4.49 %. Moreover, its average weighted exploit of 7.29 and average weighted impact of 6.81 are both higher than the same factors for broken access control. This means that not only is a cryptographic failure is easier to discover and exploit, exploiting it can lead to a more profound impact on the application. The notable CWEs included for this category are CWE-259: Use of Hard-coded Password, CWE-327: Broken or Risky Crypto Algorithm, and CWE-331 Insufficient Entropy. [33]

A03 – Injection

The third category on the list is *A03:2021 – Injection*, which refers to a broad class of attack vectors. Some more well-known examples of it are SQL -, NoSQL -, OS command -, LDAP -, and code injections. Even though these injections target different components of a web application, the logic behind the attacks is the same. An injection attack occurs when an attacker discovers and exploits a weakness in input validation and then enters an untrusted input that gets processed by the web application. The processed input then changes the intended execution of its backend processing, which leads to the backend completing an operation that the attacker instructed. For instance, let us assume that a web application has a table that

lists products and their quantities. This list also has a search function that filters products based on user input. If this search bar does not filter or sanitize the user's input, the user can input commands into the search bar using a programming language that can lead to the web application displaying information it is not supposed to. Other vulnerabilities that enable injection attacks are the use of dynamic queries or non-parameterized calls without context-aware escaping in the interpreter or data provided by the user is not properly validated and checked. Injections are the second most common overall on the top ten list, with high average weighted exploit and impact. The notable CWEs included in this category are CWE-79: Cross-site scripting, CWE-89: SQL Injection, and CWE-73: External Control of File Name or Path. [33], [35]

A04 – Insecure design

The fourth category on the list is *A04:2021 – Insecure design*, which is the first newly added category for the 2021 top ten compared to previous years. Insecure design refers to "missing or ineffective control design" and as such the category focuses on risks related to design and architectural flaws by placing emphasis on the "Security-By-Design" principle and its importance. The category's description points out that while a secure design can have implementation defects that lead to exploitable vulnerabilities, an insecure design cannot be fixed even with perfect implementation. Examples of identified factors behind insecure design are the lack of threat modeling and business risk profiling to determine the appropriate level of security design for the application, misunderstood project requirements, mishandled resource management, and missing secure development lifecycle. While insecure design is not as exploitable or impactful as the top three on this list, it is still the most prevalent root cause for vulnerabilities to be later found and exploited in an application. For this category, the notable CWEs include CWE-209: Generation of

Error Message Containing Sensitive Information, CWE-256: Unprotected Storage of Credentials, CWE-501: Trust Boundary Violation, and CWE-522: Insufficiently Protected Credentials. [33]

A05 – Security Misconfiguration

The fifth category on the list is *A05:2021 – Security Misconfiguration*, which refers to general security misconfigurations that leaves a web application exposed to security threats. This category covers a variety of topics such as missing security hardening across the software stack, improperly configured permissions on cloud services, enabled default login credentials, error handling that reveals unwanted information to users, implementation of unnecessary or vulnerable features and latest security updates or features not being installed among others. An attack scenario for this category would for example be that an attacker causes an error in the web application which then shows an error message with potentially sensitive information about how the application is configured or its back-end processes. With a high occurrences rate and the highest average weighted exploit value of 8.12, it is not surprising to see this category so high up on the list when considering the complexity and feature richness of today's web applications. Some notable CWEs included are in this category are CWE-16 Configuration and CWE-611 Improper Restriction of XML External Entity Reference. [33], [35]

A06 – Vulnerable and outdated components

The sixth category on the list is *A06:2021 – Vulnerable and outdated components*, which includes components that for one reason or another are now a vulnerability for the security of the web application. More precisely, the category refers to libraries, packages, features, components, frameworks and so on that due for instance being out of date, being malicious to begin with, having received a malicious update or by

causing a version mismatch it creates a vulnerability. As an attack example for this category, in 2021 a remote code execution exploit was discovered in some versions of Apache Log4j, which is a component many computer systems use that records events and then communicates diagnostic messages to users and administrators. The attack could be started by sending a query to a web server running Log4j including a malicious payload that would give the web server commands to execute. The fix for this issue was to remove support for the method used to give the web server commands. [36] This category is hard to test and assess risks for, which led to its average weighted exploit and impact are the default value of 5.0. However, the average incident rate was the highest of all the categories at 8.77 %. The most notable CWE included in this category is CWE-1104: Use of Unmaintained Third-Party Components. [33]

A07 – Identification and Authentication Failures

The seventh category on the list is *A07:2021 – Identification and Authentication Failures*, which refers to vulnerabilities that allow the attacker to exploit and impersonate web application users. This category is different from *A01:2021 – Broken Access Control* in the sense that while that category refers to failures in the access control policies and mechanisms, this category refers to failures in identifying and authenticating a user which can lead to impersonation or session hijacking. There are various attack methods that an attacker can utilize to exploit this category, for instance by guessing or brute forcing weak or guessable login information, stealing exposed session identifiers, taking advantage of missing or ineffective multi-factor authentication, and abusing reused old session IDs among others. One example of an attack scenario of this category would be a credential stuffing attack, where the attacker uses a known list of passwords to attempt to sign into the web application. If the application has not implemented protection for this kind of attack, the attacker

is free to retry as long as they have known passwords. The total occurrences rate for this category is the highest of the final five at over 130,000, but with a relatively low average incident rate of 2.55 %. However, the average weighted exploit rate of 7.40 one of the highest of the listed categories, meaning that this vulnerability can often be exploited. The notable CWEs included for this category are CWE-297: Improper Validation of Certificate with Host Mismatch, CWE-287: Improper Authentication, and CWE-384: Session Fixation. [33], [35]

A08 – Software and Data integrity Failures

The eighth category on the list is *A08:2021 – Software and Data integrity Failures*, which is another newly added category for the 2021 top ten. This category focuses on making assumptions related to software updates, CI/CD pipelines, and critical data without verifying integrity. This category is partly related to *A06:2021 – Vulnerable and outdated components* as it refers to third-party components an attacker gains access to, then releases a malicious update for that component which gets automatically installed for the web application if the integrity of the update is not properly validated. Other examples of this are when a web application relies on a compromised content delivery network (CDN) or uses an insecure CI/CD pipeline that has the potential to result to unauthorized access, system compromise or malicious code being executed. In terms of total occurrences, it is the third lowest on the list but with a high average weighted exploit and impact scores of 6.94 and 7.94 respectively. This category's notable CWEs include CWE-829: Inclusion of Functionality from Untrusted Control Sphere, CWE-494: Download of Code Without Integrity Check, and CWE-502: Deserialization of Untrusted Data. [33]

A09 – Security Logging and Monitoring Failures

The ninth category on the list is *A09:2021 – Security Logging and Monitoring Failures*, which refers to a lack of proper logging and monitoring mechanisms to help detect, escalate, and respond to active breaches. This is the first category to be added to the list from the community surveys, A09 being added from the OWASP industry survey. Simply put, a breach cannot be easily detected without logging and monitoring. Failures in this category allow attackers to conduct their activities undetected, make detecting and investigating breaches more difficult and make dealing with normal errors or anomalies a lot harder. As an example that is relevant to the context of this thesis, the GDPR requires the logging of breaches as any breach involving the personal information of an EU citizen has to be reported to the authorities. The average incident rate for this category is the second highest at 6.51 % with a relatively high average weighted exploit rate of 6.87. Its average weighted impact is however the lowest on the list at 4.99, as logging mechanisms often cannot control the application or its data. Most notably this category's CWE includes CWE-778 Insufficient Logging, but it is also expanded to include CWE-117 Improper Output Neutralization for Logs, CWE-223 Omission of Security-relevant Information, and CWE-532 Insertion of Sensitive Information into Log File. [33], [35]

A10 – Server Side Request Forgery (SSRF)

The final category on the list is *A10:2021 – Server Side Request Forgery (SSRF)*, which is the second category to make the list from the OWASP community surveys. This is also the second category on the list which targets a specific kind of attack type, the other being *A03:2021 – Injections*. It refers to an attack when a web application fetches a remote resource without validating the user-supplied URL. It enables the attacker to send a crafted request to an unexpected destination, even

if the application is protected by a firewall, network access control list or VPN. Because modern web applications provide their users with more and more features, fetching URLs has become a common scenario. This increases both the incident rate and severity of SSRF, meaning that it might be at a higher position in the next released OWASP Top Ten list. Currently, its average incident rate is 2.72 %, with the highest average weighted exploit at 8.28 and a high average weighted impact of 6.72. As a new entry, it only has one notable CWE which is CWE-918: Server-Side Request Forgery (SSRF). [33]

In order to gain more insight on how the current security trends concerning web applications have developed, we can compare the most recently released OWASP top ten list to lists released previously to see how the threats and vulnerabilities of web applications have evolved in the 18 years. In Table A.1, we have combined all of the OWASP Top Ten lists released to date. The list's footnotes can be found in Appendix A, that explain the changes that the list has gone through. One of the main points to take away from the table is how injections, cross-site scripting, broken authentication, security misconfigurations, and broken access control have been continuously featured on the list almost every year since its inception. In Section 4.1, we discussed how well-understood attacks and vulnerabilities still continue to appear, even if they are less prevalent. This can also be seen in the table, how for instance, cross-site scripting and injections that used to be very high on the list are now slowly coming down the rankings. While the renaming and combining of categories is apparent throughout the years, the switch in focus to place more emphasis on root causes rather than symptoms, can be seen in the latter two editions of the list. In summary, while the methods malicious actors use to exploit vulnerabilities and launch attacks may have evolved, the root causes that enable these attacks have stayed mostly the same. Web application vulnerabilities continue to stem mostly from insecure design, security misconfigurations, broken authentication and access

control, and a lack of input sanitization. While these issues will continue to cause security vulnerabilities in web applications for the foreseeable future, entities like OWASP that spread awareness about these problems will also continue to try to make the industry more secure.

4.4 Presently Prominent Web Application Threats and Vulnerabilities

To start off this section, we will look at other research regarding current web application threats and vulnerabilities to establish whether or not the OWASP Top Ten list is comprehensive enough to be used as a base to answer the first part of RQ1. Accomplishing that will also in turn present an overview of the currently prominent web application threats and vulnerabilities. In the beginning of Section 4.1, the ENISA Threat Landscape report from 2021 was mentioned. While that report focuses on the overall threat cybersecurity threat landscape, it also includes information about web applications. For the year 2021, ENISA listed eight prime threat groups that were as follows: Ransomware, malware, cryptojacking, e-mail related threats, threats against data, threats against availability and integrity, disinformation & misinformation, and non-malicious threats. Out of these eight categories, threats against data and threats against availability and integrity can be seen to affect web applications. The numbers discussed in the beginning of Section 4.1 touched upon the former category, where web application attacks, errors and system intrusions were the primary patterns of the information sector's data breaches. The report does not list any specific web application attack types that lead to these data breaches, so we will discuss other sources shortly. The other threat group, threats against availability and integrity, lists Distributed Denial of Service attacks as well as web-based attacks as their most prominent attack types. It is mentioned

that for instance, web applications can be used to distribute a web-based attack that allows a malicious actor to make a network of compromised devices that they then use to carry out a denial of service attack. Furthermore, it is discussed that web applications continue to be vulnerable to web-related threats such as injections and cross-site scripting. This information is corroborated upon by the 2021 OWASP Top Ten list. In a section discussing the main trends of web-based attacks, the ENISA report lists following: A noticeable rise in detected security misconfigurations, automated brute force attacks being increasingly adopted, cyber-attackers turning security defenses into weapons, untrusted composite services and the OWASP top ten list. Not only do ENISA's findings reflect those of OWASP, but they also use the OWASP Top Ten list as a source for their report. [29]

Radware, a cybersecurity research and solutions provider, also discusses the current trends in web application security in their annual "Global Threat Analysis Report". According to that report, in the year 2021 web application attack activity was dispersed across an array of industries, none of which particularly stand out from the rest. The most attacked industries included banking, finance, SaaS providers, retail, and high-tech industries, with most of the attacks originating in the United States, Russia, India, the United Kingdom, and Germany. The report also refers to the 2017 OWASP Top Ten list and its categories, concluding that according to Radware's data, over three-quarters of all web application attacks were either broken access control or injection attacks. As can be seen in Appendix A.1, OWASP ranked injections as the top application security risk and broken access control as the fifth in their 2017 Top Ten list. In addition, the number of blocked malicious web application requests grew by 88 % from 2020 to 2021. This comes as no surprise, as the amount of malicious activity detected by Radware was higher in every quarter in 2021 when contrasted to 2020. [37]

In Microsoft's annual "Microsoft Digital Defense Report" it is stated that modern applications and services require user authentication prior to granting access as that is seen as one of the most prominent attack vectors. Moreover, web applications were listed as one of the most common network attack types while HTTP is the most common network protocol to be exploited. The report also mentions Web Application Firewalls (WAFs) that have in recent years evolved from focusing on injection-type attacks to include attacks from malicious bots and API abusers. This is due to the increasing number of botnets and vulnerabilities found in APIs. [38] More on WAFs in Section 4.5.

Based on information gathered from the OWASP Top Ten 2021 list and the reports mentioned in this section, we can ascertain the currently most prominent web application threats and vulnerabilities in order to answer the first part of RQ1. We can use the two categories linked with web applications used in the ENISA report to group these threats and vulnerabilities as either concerning a web application's availability and integrity or concerning the data that is handled within one. Some of these threats or vulnerabilities might affect a web application on both of these two fronts, however for the sake of clarity, they will be grouped according to which category they would fall under in most circumstances.

Beginning with threats and vulnerabilities concerning a web application's availability and integrity, the most prominent threats are Denial of Service attacks. Especially the amount of DDoS attacks has been on the rise in recent years, with attackers utilizing multiple attack vectors, new and technically advanced strategies, and botnets. [29] Last of which is a network of vulnerable or infected devices controlled by an attacker without the owner's knowledge in order to carry out various cyberattacks or scams. In the case of DDoS attacks, an attacker can use a botnet with hundreds or thousands of devices in different geographical locations to overload a service or web application with requests. The most prominent vulnerability of a

web application's integrity are software integrity failures. As discussed in Section 5.2.2 as list number A08 of the OWASP Top Ten 2021, these refer to vulnerabilities caused by code and infrastructure that does not protect against integrity violations. These violations include, for instance, relying on third-party libraries, plugins or modules that are not being constantly monitored for malicious activity, and using insecure production pipelines that allow a malicious actor to tamper with the integrity of a web application. Before moving onto threats and vulnerabilities against data, it is good to mention prominent vulnerabilities that can be seen to affect a web application's availability and integrity as well as their data. Security misconfigurations and insecure design can leave a web application vulnerable against threats focusing on its data, availability, or integrity. For example, a web application that does not have any DoS attack protection, security measures that are improperly configured on cloud services or other external services, security settings that are not set to correct values, or the application simply being designed to be insecure from the beginning that allows it or its data to be tampered with later on.

Finally, we will focus on the most prominent threats and vulnerabilities against the data of a web application. Based on the reports mentioned in this section as well as the OWASP Top Ten list, the most prominent threat against web applications currently are injection attacks. These attacks were heavily featured in every report and according to the OWASP Top Ten list, 94 % of web applications are vulnerable to some form of injection. [33] These attacks can be detrimental to a web application's security, as injection attacks may in the worse case expose its whole database to the attacker. The most prominent vulnerability is broken user authentication as it was also featured in every report and had the number one spot on the OWASP 2021 Top Ten list. According to that list, 94 % of web applications were tested to have some sort of access control vulnerabilities. [33] Broken access control is a vulnerability that can lead to an unauthorized user accessing data or

information they should not be able access. As previously mentioned in this section, Radware found that nearly 80 % of cyber incidents regarding web applications in the year 2021 involved either injection attacks or exploited vulnerabilities due to broken access control. [37] The second most prominent threat is the predictable resource location attack that most of all utilizes broken access control. According to Radware, the attack counted for over 40 % of their customer's security violations in the year 2021. [37] The predictable resource location attack is an attack method utilizing brute force that tries to uncover hidden website content and functionality by guessing file and directory names. By making educated guesses for common names for directories, files, backups or similar resources, the attacker can access these resources even though they were not meant to be accessed or viewed by users. The second most prominent vulnerability is unencrypted data, mostly referring to cryptographic failures as covered in Section 5.2.2. Transmitting data unencrypted or using old or weak cryptographic algorithms exposes data to multiple different threats on different attack vectors. According to the OWASP Top Ten 2021 list, nearly 70 % of web applications had vulnerabilities related to cryptographic failures. [33] Lastly, the third most prominent vulnerability is broken user authentication or identification. As discussed in Section 5.2.2, this vulnerability can be linked to automated and brute force attacks such as the predictable resource attack. Without properly authenticating users and their application sessions, attackers can impersonate users or hijack their sessions to carry out attacks. According to the OWASP Top Ten 2021 list, almost 40 % of tested web applications had vulnerabilities related to identification or authentication failures. [33]

In this section we compared cyber threat, vulnerability, and digital defense reports from three known industry entities and their findings to the OWASP top ten list. We did this to both distinguish whether or not the list could be used to answer the first part of RQ1 and to gain an overview of the current web application

threat and vulnerability landscape. While highlighting different aspects, the reports corroborated on the OWASP Top Ten list and its ranking with their data. As mentioned before, the "Threat Landscape" report from ENISA even directly refers to the OWASP list in their section on web-based attack trends [29]. After assuring that OWASP Top Ten list can be used to partly answer RQ1, we combined information from the reports and the Top Ten list and listed the most prominent threats and vulnerabilities towards web applications at the moment. We split the answer into two categories borrowed from the ENISA report. These categories consisted of threats and vulnerabilities against a web application's availability and integrity, and against its data. For the first category, we found that different kinds of DoS attacks, software integrity failures, and security misconfigurations. For the latter category, injection attacks, broken access control and user authentication, predictable resource location attacks, and cryptographic failures were ranked the highest among the reports and the top ten list. This answers the security focused part of RQ1, whereas the web application privacy concerns will be addressed in the following chapter along with RQ2.

4.5 Mitigation Measures

In this section we will discuss and list the mitigation measures and strategies that can be used to secure web applications and prevent malicious actors from finding vulnerabilities to exploit. As the list of different methods, measures, and strategies is quite extensive, this section will focus on listing the ones that can be seen as general good practice as well as ones targeting specific threats or vulnerabilities based on the OWASP Top Ten list as discussed in Section 4.3. We will not be delving into any technical details or specifications on how these measures or strategies are implemented or operated in this section, but instead we will compile a comprehensive list of them to aid us answer RQ3 in Chapters 6 and 7. Furthermore, should the

measures or strategies listed here be brought up or utilized during those chapters, we will go into further detail as needed. First off, we will begin by discussing the general good practices and strategies in software development. Afterwards, we will briefly examine measures and strategies in order to mitigate and prevent the threats and vulnerabilities as listed in Section 4.3 from being exploited.

The first step of mitigating threats and vulnerabilities takes place before any of the code related to the web application has been written. During the planning and architectural phase, there are already important steps to take. Firstly, the architecture of the application should be designed with the current security recommendations in mind, while implementing a security framework if the software stack supports one. The objective of most web applications is to efficiently transmit data from point A to B, so it is important to focus on how the web application transmits, operates on and stores data, and how those actions and the data itself can be secured. Secondly, setting up a formal process that emphasizes security during the development phase not only helps streamline development but also makes addressing possible security issues a part of the process. For instance, the security oriented development strategy called "Secure Development Life Cycle" helped Microsoft achieve great results which focuses on different aspects from developer training, code review and verification as well as how updates are released and how incidents are responded to after the application has been put into production [39]. During the development phase, conducting comprehensive code reviews, making use of vulnerability analysis and management, and writing different sorts of tests, is paramount. Code reviewing each commit or multiple commits in a separate meeting is a crucial step to ensure that the application will meet the set security standards. Discovering, analyzing, and managing vulnerabilities either internally or through bug bounty programs and third party penetration testers during and post development, minimize the possibility of a vulnerability getting exploited once the application has reached production.

Tests like the regression test that assert whether or not a vulnerability has been actually fixed by a patch are also a useful measure in making sure that bugs or vulnerabilities that have been previously fixed are not exploitable sometime in the future. Overall, during the architectural and development phases of a web application's lifespan the most important mitigation measures are designing a secure application architecture, setting up a secure software development life cycle, using secure coding best practices, vulnerability management, writing tests and educating developers to have a secure-by-default mindset. [39]–[41]

Another mitigation measure is the implementation of a Web Application Firewall (WAF). The purpose of a WAF is to offer in depth security in the means of deep packet inspection of network traffic between the client and the server side. Generally the WAF is set up in one of two ways: Either it only allows traffic that matches preset positive policies, or it allows all traffic and attempts to block specific traffic that is represented by negative policies. In order to create these policies, most of the WAFs on the market offer either machine learning so that the WAF itself knows what kind of traffic is normal for the application or manual configuration. There are problems with both of these approaches as the automatically trained firewall only knows what is normal based on the learning material, whereas manual configuration requires a security professional that knows all possible valid and invalid inputs and outputs that the application processes. Most importantly, while implementing a WAF adds an extra layer of security to an application, it in no way can replace the mitigation measures and strategies listed in the previous paragraph. Implementing a default or misconfigured WAF can be in fact lead to more vulnerabilities or problems in the application itself. [42]

The target application of this thesis has however already reached production, so for the rest of this section, we will focus on what measures and strategies could be implemented once an application has reached this phase. It is also noteworthy that

in general it is difficult if not impossible to write completely vulnerability-free code, even if an application is developed with the security-by-design approach. That is why having the proper methods, measures, and strategies to update and improve an application after it has been put into production is crucial. [39] As mentioned at the beginning of this section, we will compile a list of mitigation measures based on the most recent OWASP Top Ten list. Some of the vulnerabilities on that list include very specific mitigation measures and technical details that concentrate on preventing a certain vulnerability. Hence, the following list is a compilation of general mitigation measures that can be implemented at any point of an application's life cycle and that raise its overall level of security. If during Chapters 6 or 7 a certain vulnerability on that list is discovered, we will further examine the prevention measures linked with that vulnerability. List of mitigation measures: [33], [39], [40]

- Access control should be properly configured. Block all access to other than public resources by default and implement access control mechanisms that exist throughout the application.
- Implement multi-factor authentication, check for weak passwords and limit failed login attempts.
- Disable server directory listing and ensure that no metadata or backfiles are present within web roots.
- Ensure that you have proper logging, log access control failures while notifying the admins when appropriate and encode the log files.
- Secure and limit API and controller access.
- Do not unnecessarily store sensitive data and classify what data is being processed and stored.
- All data should be encrypted at rest and in transit.
- Implement input sanitization.
- Apply strong, non-deprecated cryptographic functions, algorithms and padding schemes.

- Write and run unit, integration and regression tests.
- Remove and do not install any unused features or frameworks.
- Continuously inventory client and server-side components and their dependencies and keep them up-to-date. Ensure that software like npm or Maven consume from trusted repositories.
- Secure your Continuous Integration / Continuous Delivery (CI/CD) pipeline and your version control repositories.

5 User Information Privacy & Target Application Overview

When people visit websites or use online services and web applications, they are constantly being tracked. The simplest and most common way of tracking are cookies, which are small text files websites installed on our devices that get processed and stored in our web browsers. While cookies themselves are harmless, they can be used to track a user browsing the web in order to build a unique profile of them. This profile can then be sold to advertising companies for a personalized advertisement experience. There does exist various browser plug-ins and settings to disable cookies and tracking, but websites can use other information like your IP address, user agent, hardware, location, or browser plug-ins to identify you. In the European Union, both the GDPR and a Directive called "the ePrivacy Directive" regulate how websites can install cookies or track users from the EU. In the case of the GDPR, Recital 30 says that cookies that are used to identify users qualify as personal data and are as thus subject to the GDPR. The ePrivacy Directive (EPD) passed in 2002 and later amended in 2009 states that website visitors must consent to cookies, which is why every website has cookie-related pop ups the first time you visit them [43].

The goal of this chapter is to answer RQ2 by identifying what user-related information is collected and handled by web applications, how that information can

then be used to identify a user, and how relevant that piece of information is in that process. First, we look at privacy as a concept, what personal data is, and how it is linked with user information. Then, we identify the types of user information that are collected and processed by web applications and rank how critical they are in order to identify a user. Afterwards, we discuss privacy concerns related to web applications as well as briefly examine how user information can be secured in web applications from a technical standpoint.

5.1 Privacy, Personal Data & User Information

Privacy as a concept and related privacy issues are as old as mankind. What once began as a right to protect one's body and home first evolved towards our current understanding of privacy in 1891 with an article written by American lawyers Samuel Warren and Louis Brandeis describing the right to privacy as "the right to be left alone". During the 1970's and 1980's global attention shifted towards information privacy, as progress in technology gave us computer systems and the Internet in 1983. In 1986, privacy was identified as one of the four ethical issues of the information age with two forces recognized as the biggest threats to our privacy: the growth of information technology and the increasing value of information in decision making [44]. These two forces fairly accurately predicted what kind of privacy related issues people were going to face in the future. Firstly, advances in technology have enabled governments, organizations, and companies to gather information using video surveillance, biometric and genetic data, smart cards, GPS system, networks, the Internet, and applications among others. Secondly, the collected data can then be used to identify individuals, create profiles, be data mined for analytics, get sold to third-party entities or get used by policy makers even if they violate a person's privacy. [45] Regulations and directives passed in the EU in the past 20 years like the aforementioned GDPR and EPD have both improved

the protection of data and privacy for EU citizens and forced companies to abide by them if they wish to operate inside the EU. However, it is still important to note that even with these regulations in place, every single piece of private information we give out about ourselves is seen as another breadcrumb ready to be processed, used, and sold off by profit-oriented organizations.

The terms *privacy protection* and *data protection* might seem similar in nature as personal information can be turned into points of data, but it is however important to make a distinction between the two. The first one refers to the protection of reliable and accurate information related to all aspects of an individual's everyday life, such as their home, family, or correspondence. This means that it concerns the protection of an individual's personal identity. The latter emphasizes the informational dimension, referring to the process of safeguarding data from corruption, compromise, and loss, while maintaining the capability to restore it should something render the data unusable or inaccessible. [45], [46] In essence, data protection is used to safeguard data that may include personal data that in turn effects the privacy of a user. Next, we will see how personal data is defined by different governing bodies and how it is related to user information. For instance, the European Commission defines personal data as the following: *"Personal data is any information that relates to an identified or identifiable living individual. Different pieces of information, which collected together can lead to the identification of a particular person, also constitute personal data. Personal data that has been de-identified, encrypted or pseudonymised but can be used to re-identify a person remains personal data and falls within the scope of the GDPR. Personal data that has been rendered anonymous in such a way that the individual is not or no longer identifiable is no longer considered personal data. For data to be truly anonymised, the anonymisation must be irreversible."* [47] The National Institute of Standard and Technology (NIST) that functions as a part of the U.S Department of Commerce define personally identifi-

able information (PII) as: *"information that can be used to distinguish or trace an individual's identity, either alone or when combined with other information that is linked or linkable to a specific individual."* [48] China's equivalent to the European Union's GDPR came into effect in November 2021. Called Personal Information Protection Law (PIPL), it is China's first comprehensive legislation regarding data and privacy protection. In it, personal information is defined as the following as per Article 4: *"Personal information is all kinds of information, recorded by electronic or other means, related to identified or identifiable natural persons, not including information after anonymization handling."* [49]

Overall, the definition of personal data is very similar regardless of a person's geographical location. Personal data can as such be any information that can be linked to an individual and their identity alone or combined with other information. The definition of user information is not as clear however, as different companies and agencies categorize different types of personal data as user information in their terms of service. In order to give a general definition for user information, we need to combine some of these categorizations. Essentially, user information means all information, data or other content that is entered by or collected from a user whilst using a service, software or application that can be used to either identify a user or that can be somehow linked to them. This means that in the framework of a web application or web service, user information, personal information, and personal data can be seen to mean the same thing. The only difference between the personal information and user information is that singular data points of personal data can always be linked to an individual, while singular data points of user information might not be. This is true in the cases of screen size, operating system, or browser information for example, as they can rarely alone be used to identify an individual but can prove very helpful when used along with other data points. However, the GDPR definition of personal data as mentioned previously does state that: "Dif-

ferent pieces of information, which collected together can lead to the identification of a particular person, also constitute personal data." This means that even those singular pieces of user information may be classified as personal data if they can be used to identify a person together with other data.

5.2 Personal Data Collection and Types of Personal Information

We have now established what personal data is, what user information is, and what the relation between the terms is. Next, we will discuss what personal data is collected from users by applications, how it is collected and how critical different pieces of information are in order to identify a specific user. Types of user information can be separated into three categories by how they are collected:

I. User input the information themselves,
II. The application indirectly gathers the information from the user,
III. The application appends known user information from other sources to information collected by method I or II.

The first method involves the user themselves entering their information to the service or application in order to have a unique account that they can be identified by. The second method involves the application tracking the user while using the application, collecting identifiable information like IP addresses, device information and cookies, if available. The third method involves the administrator of the application or service either buying or gaining access to user information collected by other applications or analytics services that can be linked to specific users in their application and then combining that information. The first method usually includes personal data, as services and applications often ask the full name, address,

phone number, banking details etc. of their users depending on the type of service or application. As discussed in the previous section, all of this information is classified as personal data as it can be used to identify an individual. Considering the scope of this thesis, we will focus on methods II and III of information collection as for a mostly anonymous application as the target application of this thesis, we are most of all interested in what information is collected from a user that they are potentially unaware of.

In general, websites, web services, web applications, mobile applications and so on gather data on the user using methods II and III. This is done with online trackers that can be divided into two categories: same-site and cross-site trackers. In simple terms, same-site trackers track your activity on a certain website or application while cross-site trackers follow you from website to website. Online trackers, like cookies, fingerprinters, and tracking pixels all collect your personal and user information but in slightly different ways. As mentioned at the beginning of this chapter, cookies are small text files that a website or application sends to your browser that gets installed on your device. Fingerprinters are installed on websites and applications that collect information such as a user's language, keyboard layout, time zone, browser version and much more. Tracking pixels are hidden or camouflaged 1x1 pixel sized graphics that include a link to a server which the user's browser follows and accesses that server. Cookies and fingerprinters are the most common types of online trackers that enable website and web service providers to gather, process, and store your personal data and user information. [50]

In order to gain a basic understanding of what types of information is actually collected by web services and mobile applications, we will look at two studies conducted by authors at the University of Turku. In the studies, the network traffic of 34 web services and 32 mobile applications provided and maintained by Finnish public sector bodies was analyzed to find out what kind of personal data is collected

from users and potentially sent to third-party analytics services. In the first study, web services made available by the Finnish government, cities, or municipalities were tested by logging into the service, invoking their most important functionality, navigating through the service, and accepting all cookies. While using the web service, the browser network traffic was captured using Google Chrome's Developer Tools and downloaded into a log file. Almost every single tested web service collected some piece of personal data using different kinds of trackers, some more than others. The different personal data items that were collected by the web services can be seen in Table 5.1. Furthermore, the study found that the privacy policies of the web services included in the study often did not adequately cover what personal data was collected from the user. The second study which focused on mobile applications provided by Finnish public sector bodies came to same conclusions. In that study, a smartphone connected to the Internet through a Linux computer acting as a Wi-Fi access point that recorded the phone's network traffic. Through these traffic logs the authors could identify what personal data items were sent to the mobile applications and their findings can be seen in Table 5.1. One limitation that both of these studies had was the fact that some of the application level data could be encrypted or encoded so it cannot be seen on the network traffic logs. Nonetheless, the findings of these studies show that web services and mobile applications collect a lot of personal data and their privacy policies do not necessarily always accurately reflect that. [51], [52]

There are many different kinds of personal data, from a person's name and social security number to their IP address and purchase history. All of these can be collected from a user using methods I – III depending on what web service or application the user is using. Privacy regulations like the GDPR have in recent years improved the privacy of users by requiring websites, services, and web applications to disclose how they track their users and providing their users an option to opt out

Public sector web services		Public sector mobile applications	
Sent data item	Number of services (Percentage)	Sent data item	Number of services (Percentage)
IP Address	32 (94.1%)	IP Address	29 (85.3%)
Browser	32 (94.1%)	Phone Brand and Model	21 (61.8%)
Operating System	32 (94.1%)	Phone OS	21 (61.8%)
Screen Size	32 (94.1%)	Phone OS Version	20 (58.8%)
Window Size (viewport)	15 (44.1%)	Window Size (viewport)	6 (17.6%)
Color Depth	25 (73.5%)	Screen Size	21 (61.8%)
User Identifier	2 (5.9%)	Processor	2 (5.9%)
Device/Browser Identifier	15 (44.1%)	Is the Phone Rooted?	2 (5.9%)
Other Unknown Identifier	31 (91.2%)	User Identifier	3 (8.8%)
Timestamp	26 (76.5%)	Device Identifier	3 (8.8%)
Language	25 (73.5%)	Other Unknown Identifier	19 (55.9%)
List of Purchased Products / Services	1 (2.9%)	List of Purchased Products / Services	1 (2.9%)
Viewed Contents / Performed Actions	32 (94.1%)	Performed Actions	12 (35.3%)
Connection Type	6 (17.6%)	Location (Country)	11 (32.4%)
		Location (City, Area)	1 (2.9%)
		Timezone	7 (20.6%)
		Timestamp	20 (58.8%)
		Language	12 (35.3%)
		Internet Provider	14 (41.2%)

Table 5.1: Personal data items sent to the analyzed web services and mobile applications [51], [52]

of getting tracked. The decision to track users and collect their data is almost always in the hands of the service provider. That is without counting the rare occasions where a service or website has been hacked in order to hide malicious trackers or other data collectors by malicious actors. In order to discern between the different high-level characteristics of personal data, it can be classified into three dimensions: [53]

- Static vs. Dynamic personal data:

  Static: Does not typically change over time, such as the user's name, gender or email address.

  Dynamic: Changes in short or medium time intervals, such as user's session ID, user's IP address or connection type.

- Unique vs. Non-unique personal data:

  Unique: Distinctly identifies a person from others, such as email address, phone number or user identifier.

  Non-unique: May be shared among other persons, such as date of birth, gender or name.

- Shared vs. Distinct personal data:

  Shared: Likely to be shared across other services or applications, such as email or mailing address.

  Distinct: Potentially different for each service or application, such as user identifier or last login information.

Another way of personal data collection that web applications can utilize apart from trackers and information that the user input themselves, is metadata collected from the content that a user might upload to that application. Content such as images, videos or audio files can include embedded information that can be used for cataloging and contextualization but also identification. In the case of a photo, information such like where the photo was taken, what it was taken with, when it was taken, the device's serial number, the name of the image creator or the device's owner among other data items. Image metadata is usually stored in a standardized format such as Exif, IPTC or XMP, which makes removing metadata from images uploaded to different services or applications more efficient with automated tasks. Content metadata is nonetheless an important piece of personal data that gets often overlooked by users that they unknowingly share with services and applications. [54]

In Table 5.1 different personal data items collected and processed by web applications are listed and ranked depending on how critical that item is in identifying an individual. The list of data items will be narrowed down by only listing personal data items most likely to be gathered by web application of the same type as this thesis' target application, Vihjaa. In other words, commercial web applications that gather basic information about the user using methods I-III either for

analytics or to improve the user experience. It is important to note however, that the target application does not collect any analytical data from its users to protect their anonymity. The personal data items in this table are partly based on the aforementioned research conducted by Heino et al and Carlsson et al as seen in Table 5.1 combined with other sources [51], [52]. These other sources include the NIST guide to protecting the confidentiality of PII [55], the GDPR examples [2], and personal experience from working as a web application developer. The reason for this stems from the fact, that after extensively searching for credible articles, publications, white papers, and industry standards for research on this particular issue, it seems there has so far been none conducted. More specifically, research on how web applications gather and process personal data. Additionally, a ranking of personal data items based on how critical they are to identify an individual also seems to not have been completed. This lack of research will be further discussed in the conclusion to further elaborate on this point as well as to provide subjects for future research. Thus, the data items have been gathered from these different sources and in some cases combined to create a more coherent ranking. The ranking system functions as follows: Data items will be ranked as 'low' for items that are unlikely to lead to the identification of a user, 'medium' for items that when used in combination with other items can identify a user, and 'high' for items that alone can lead to the identification of a user.

Ultimately, we can now use the information gathered during this section to answer RQ2. Firstly, we must keep in mind that according to the GDPR, all collected user-related information that can be used to identify a user by itself or along with others is classified as personal data. In Table 5.2 we listed and ranked the separate user information items based on how critical they are in identifying a specific user. We found that almost all of the personal data items handled by web applications that are similar in nature as the target application can lead to the identification of

Personal data item	Static?	Unique?	Shared?	Rating
Name	x		x	High
Gender	x		x	Medium
Social Security Number	x	x	x	High
Mailing Address	x		x	High
Email Address	x	x	x	High
Phone Number	x	x	x	High
Data of Birth	x		x	High
IP Address	x		x	High
MAC Address	x	x	x	High
Browser			x	Low
Operating System	x		x	Low
Screen Size	x		x	Low
Color Depth	x		x	Low
User Identifier	x	x		High
Username	x	x	x	High
Language	x		x	Medium
Performed Actions / Viewed Contents		x		Medium
Personal Characteristics (Including Photographic Image)	x	x	x	High
Date and Time (Image Metadata)	x		x	Medium
Geo Location (Image Metadata)	x		x	Medium
Device (Image Metadata)	x		x	Medium

Table 5.2: Ranking of web application data items based on how critical they are in identifying a user

a user either together with other information or by itself. The table additionally includes a classification based on the three dimensions that personal data can be classified into. It is worth noting that these classifications are based on a default presumption about that certain item of personal data. For example, while a person's name is in most cases unique, there do exist cases where individuals share the exact same name especially in cultures where middle names are not commonplace. Another example is all of the technical personal data items gathered from a user.

For the ranking, we ranked "browser", "operating system", "screen size" and "color depth" with a rating of "low". While these items can change if the user decides to use the web application using another device, the default presumption is that a user mainly accesses the application from the same device. However, the aforementioned technical data items are usually shared among many users, making it difficult to identify a specific user using those items. The broad definitions given to personal data, or personally identifiable information, by different bodies of authority as discussed in Section 5.1 encompasses every item collected by web applications as personal data. More specifically, the definition that every piece of data that can be used by its own or together with other pieces of information to identify a person leads to a situation where every piece of data collected from a user by web applications does classify as personal data. In conclusion, we can use this information to answer RQ2 with some caveats. The main one being the unfortunate lack of research conducted into what type of user-related information is collected and processed by web applications. As such, Table 5.2 should be treated as a compilation of research findings gained from studying similar types of software as well as insights based on personal software developer experience. While lacking an exhaustive answer of every single user-related data item collected by web applications, Table 5.2 can still be considered a baseline for the types of information end-users can expect web applications to collect from them. The answer to RQ2 can then be formulated by taking the information compiled in Table 5.2 and combining it with the fact that each of those data items can be used to identify a user, be it by itself or together with other data items.

5.3 Web Application End-User Privacy Concerns

Privacy concerns as a concept refers to a person's beliefs about the potential negative consequences and risks associated with sharing personal information. [56] Web applications are information collectors and disseminators that do not necessarily always disclose how they gather data on their users as concluded in the previous section. In Chapter 4, we discussed and partly answered RQ1, while leaving the part concerning privacy to be answered in this section. In order to answer this question, we need to understand the interdependent and complementary relationship between privacy and security. Designing a secure software that safeguards privacy begins with a basic concept made up of three pillars called the "CIA Triad": *Confidentiality, Integrity* and *Accessibility*. These three pillars act as the fundamental tenets of information security and by extension information privacy. In general, data cannot be confidential if it is not private and secured. Additionally, data integrity cannot be guaranteed if the data it is not private and secured. Furthermore, data cannot be accessed if it is not stored properly, or the security or privacy settings are misconfigured. This means that, as aforementioned, security and privacy are complementary and interdependent concepts. Without data security it is impossible to safeguard data privacy and without data privacy it is impossible to ensure data security. [39] As such, it could then be argued that privacy concerns and security concerns are also interdependent. However, the objective of this part of RQ1 was to establish what kind of privacy concerns the end-users of the target application might have towards it. Thus, we will list some of the prominent privacy related concerns recently conducted studies and surveys have found end-users to have and then analyze whether or not those concerns have been addressed in the target application in Chapters 6 and 7.

A meta-analysis of online privacy concerns and privacy management from 2017 found that an individual's privacy concerns affect the extent that individuals use online services and share their information online. Furthermore, individuals who are more literate and familiar with privacy literature, tend to be more concerned about their privacy. Additionally, individuals with higher privacy concerns tend to share less personal information while utilizing privacy protective measures more. The paper also notes that some individuals "endure" those higher privacy concerns to receive the key benefits of social networking services, such as relational development and self-clarification that gets derived from self-disclosure. From the paper, we can derive some privacy concerns that can be also applied to web applications. [56] Furthermore, we can also use the results of an empirical study focused on online privacy concerns to add to our list of prominent privacy concerns: [56], [57]

UPC1: What personal data does the application collect about the user?
UPC2: How is that personal data processed and stored?
UPC3: Is that personal data shared with other parties or entities and if so, how?
UPC4: What measures designed to protect user privacy does the application provide?
UPC5: Does the application's privacy policy accurately reflect the application?
UPC6: How can the application guarantee the confidentiality of a user's messages or emails?
UPC7: Is a user's activities monitored while using the application?
UPC8: Can the user's personal data saved on their device only be accessed with their permission?

A study on the general perception of social media privacy concerns by users conducted in 2018 concludes that social privacy as a whole has become one of the major concerns for users in today's environment. Additionally, even though social networking services provide users with various security measures, there is still a lack of user information security. [58] This sentiment can in most cases also be extrapolated to point out the same flaw in other online services or software, such as

web applications, web services or mobile applications. Furthermore, while legislation such as the GDPR aids in addressing most of these concerns, the different web applications, web services, and social media websites should in general be more proactive and do more to improve the privacy of their users.

5.4 Web Application Technical and Organizational Privacy Concerns

In this section, we shift our focus from end-users to technical and organizational privacy concerns regarding web applications. The objective is to gather a similar list as in the previous section to complete our answer concerning the most prominent privacy concerns towards web applications at the moment as outlined by RQ1. This section will consist of discussing the Open Web Application Security Project (OWASP) Top 10 Privacy Risks list from 2021, which provides us with the last remaining information to answer RQ1. The list was compiled by surveying 60 privacy and security experts that evaluated the most prominent technical and organizational privacy concerns for web applications. The list is as follows: [59]

TOPC1: Web Application Vulnerabilities
TOPC2: Operator-sided Data Leakage
TOPC3: Insufficient Data Breach Response
TOPC4: Consent on Everything
TOPC5: Non-transparent Policies, Terms and Conditions
TOPC6: Insufficient Deletion of User Data
TOPC7: Insufficient Data Quality
TOPC8: Missing or Insufficient Session Expiration
TOPC9: Inability of Users to Access and Modify Data
TOPC10: Collection of Data Not Required for the User-Consented Purpose

The first privacy concern on the list is *TOPC1: Web Application Vulnerabilities*, which refers to all web application vulnerabilities that can lead to data exposure or leakage. The most prominent web application vulnerabilities were listed in Chapter 4 and from that list, broken access control, injection, security misconfiguration and identification and authentication failures can all result to the exposure of sensitive user data just to name a few. Failure to design a secure application, correctly address web application vulnerabilities, or detect a problem are all likely to result in a privacy breach.

The second privacy concern on the list is *TOPC2: Operator-sided Data Leakage*, which refers to failure to preventing the leakage of data that can be classified as personal data, to any unauthorized party that results in a loss of data confidentiality. Most often caused by an intentional malicious breach or an unintentional mistake caused by, for instance, lack of awareness, issues with storing data, or insufficient access management controls.

The third privacy concern on the list is *TOPC3: Insufficient Data Breach Response*, which refers to failures when dealing with a data breach. This includes informing the affected persons about a possible data leak or breach, determining the cause of the breach, failure to remedy the situation and not attempting to limit the spread of the breach or leak.

The fourth privacy concern on the list is *TOPC4: Consent on Everything*, which refers to the inappropriate use or aggregating end-user consent to all data processing. End-user consent should be collected separately for each purpose and the user should be informed of how their data is collected and processed.

The fifth privacy concern on the list is *TOPC5: Non-transparent Policies, Terms and Conditions*, which refers to the terms and conditions of web applications that do not accurately represent how the application actually collects, processes, stores, and shares end-user data. This phenomenon was corroborated on in Section 5.4 as

per [51], [52] as well as UPC5 in the previous section.

The sixth privacy concern on the list is *TOPC6: Insufficient Deletion of User Data*, which refers to failures in effectively or timely deleting personal data upon request or after termination on a specific purpose. For instance, the GDPR allows EU citizens to submit data deletion requests to companies that they must adhere to.

The seventh privacy concern on the list is *TOPC7: Insufficient Data Quality*, which refers to the use of incorrect, outdated, or fraudulent user data as well as failures to correct or update it.

The eighth privacy concern on the list is *TOPC8: Missing or Insufficient Session Expiration*, which refers to failures to effectively enforce session termination. This may result in user data being collected from the user without their awareness or consent.

The ninth privacy concern on the list is *TOPC9: Inability of Users to Access and Modify Data*, which refers to failures in providing users the ability to access, modify, or delete data related to them.

The tenth privacy concern on the list is *TOPC10: Collection of Data Not Required for the User-Consented Purpose*, which refers to collecting demographic, descriptive or any other end-user related data that is not needed for the purposes of the application. Additionally concerns data that has been collected without the user's consent. The aforementioned items on the OWASP Top Ten Privacy Risks list stem from technical or organizational issues, in some cases both. The concerns in most cases are due to flaws in the web application itself or faulty practices put in place by organization running and maintaining the application. Combining the information learned in Sections 5.3 and 5.4, we can answer the privacy concern part of RQ1. The part focused on the prominent threats and vulnerabilities towards web applications of RQ1 was answered in Section 4.4. The most prominent privacy

concerns towards web applications at the moment are separated into two different categories in this thesis, those being the privacy concerns from the point of view of end-users as well as organizations. The end-user privacy concerns mostly consist of how the data controllers collect, control, process and share their personal data and if they are accurately informed on those matters. The organizational privacy concerns mostly consist of putting the correct processes and practices in place to be able to adhere to the applicable privacy legislation depending on what region they operate in. In addition, possible technical issues and vulnerabilities affecting the web application can lead to data breaches or data leakage whether intentionally or not. While technical and organizational privacy concerns could be separated into their own categories, the technical concerns mostly affect the organization or company that either owns or operates the web application. As such, they can be viewed as being an integral part of an organization's privacy concerns. The list of organizational privacy concerns, including technical concerns, was listed previously in this section. These two lists can be used to list the most prominent web application privacy concerns at the moment from the point of view of the user using the application as well as the organization operating it.

5.5 Securing User Information Privacy from a Technical Standpoint

The task of providing and securing the privacy of a user ultimately comes down to the development team of that application as well as the organization that operates it. In Section 4.5, we discussed how security has to be a part of the development cycle from the architectural stage for the application to be even considered secure. The same is true for user privacy as well, since in order to adhere to privacy related legislation and respect the user's privacy, web applications have to be designed with

those things in mind. In this section we will briefly discuss a couple of the main concepts of securing user information privacy from a technical standpoint. This refers to the actual features and design choices software architects and developers can make in order to make their application able to secure a user's privacy. The concepts covered include implementing and respecting the user's privacy settings, encrypting connections, securing user data, and preserving user data. As mentioned, this section will only provide a brief overview of those concepts to aid us in analyzing the current implementation of the target application in the upcoming chapters. An important phase of a web application's development is the architectural phase. We established and discussed security-by-design earlier in this thesis and how crucial the concept is for the security of any application. A similar concept to security-by-design is privacy-by-design, which was developed by the former Information and Privacy Commissioner for the Canadian province of Ontario, Ann Cavoukian. She has stated to have developed the concept to promote the view that privacy cannot be only achieved by complying with regulation, rather by becoming an organization's default mode of operation. Regarding technical solutions, privacy-by-design's foundational principles include implementing solutions such as having privacy as the default setting, end-to-end security, and transparency for end-users. [60] Another important solution for user information privacy is respecting a person's choice whether or not they want cookies or trackers installed from a web application. In some regions like the EU, it is now mandated by the GDPR to ask permission from a user if they consent to their information being collected and processed. Apart from these mandatory checks, some browsers allow their users to enable a Do Not Track setting. This setting does not automatically disable tracking in their browser but instead tells web applications that a user does not want to be tracked. Detecting this setting has to be implemented in web applications, for instance in the following manner: [61]

```
var dntSetting = navigator.doNotTrack;
    if (dntSetting !== 1) {
        // enable cookies only if DoNotTrack is not enabled
        document.cookie = 'example';
    }
```

In Chapter 3 we discussed the different technologies utilized in the target application, one of them being HTTPS. Encrypting the connection between the client and the server is an essential part of securing the privacy of users using a web application. Enabling HTTPS on a server is quite a technical process, but it can be simplified to enabling HTTPS on the web application's server with configuration changes, then applying for, receiving, and installing a certificate from a certificate authority on that server. Implementing end-to-end encryption is a likewise a very technical process, but it mostly boils down to implementing basic cryptographic primitives, such as signing, key exchanging, and symmetric encryption. Last major part to account for is the processing and storing of personal data and how the privacy of users can be protected in those states. Once the web application receives user data over an encrypted transmission from an authenticated and authorized user, the data itself should be encrypted and possibly anonymized depending on the purpose the data was collected for. Different libraries exist that can be utilized to encrypt data before it is stored, such as the *crypto* library that is built-in to Node.js. Node.js is an asynchronous event-driven JavaScript runtime, that powers many of today's most prominent web applications. Once the data has been encrypted and stored in an appropriate manner, the data controller must be able to comply with different requests, such as the deletion of a person's data or a copy of a person's collected data. In order to comply with these requests, features must be implemented that for example, allow a user to request the deletion of all their personal data collected by the web application. [59], [61]

The technical solutions presented in this section were discussed on a high-level in order to provide an overview of how user information privacy is secured from a technical standpoint. Additionally, the objective was to further elaborate on the interdependent and complimentary relationship between security and privacy. Almost all of the privacy enhancing technical solutions mentioned in this section also enhance the overall security of the web application. The same can be said for some of the measures and strategies mentioned in Section 4.5, where enhancing the security of the web application also helps preserve the privacy of the end-users. With this information in mind, the rest of this thesis will focus on the target application and answering RQ3 and RQ4.

6 Security and Privacy Assessments of the Target Application

In this chapter we focus on conducting a security and privacy assessment on the target application, Vihjaa. The assessments are based on the lists compiled in Chapters 4 and 5, beginning with the OWASP Top Ten Web Application Security Risks list and moving on to the privacy concern lists. Before the assessments, the general setup used for the assessments as well as the tools and methodology are explained and discussed. Afterwards, the results will be analyzed, and possible suggestions will be given in order to improve the security or user information privacy in the application. At the end of the chapter, the results of the assessments as well as the given suggestions will be combined to answer RQ3.

6.1 Setup Used For Analysis

This section will cover the setup used for assessing the current state of security and privacy of the target application as well as its adherence to the requirements set by legislation. It is divided into brief subsections that cover the different parts of the setup, beginning with the operating system, then the tools used, and lastly methodology. As an overview, the operating system used for the assessment is Kali Linux, which is a Debian-based Linux distribution focused on advanced penetration testing and security auditing. The main tools used for the assessment, some

of which come pre-packaged with the Kali Linux distribution, include Burp Suite, Nikto, Wapiti, Dependency-Check and npm-audit. What these tools are used for and how they work are explained in the sections that they are used in. The methodology will follow the guidance and information provided by a couple different books on web application security auditing as well as guidelines written by the OWASP Foundation, which will be further elaborated on in the methodology section.

The assessment will be conducted on the newest released production version of Vihjaa as of writing this thesis. Vihjaa will be run in a development environment during the assessment, which simulates the Amazon Web Services cloud environment of the actual Vihjaa web application that is in production. The development environment is running on the aforementioned Kali Linux, where Vihjaa is run using the IntelliJ IDEA Integrated Development Environment (IDE). In order to simulate Vihjaa's database, a MongoDB instance is run inside a Docker container. Docker is a software platform, which allows developers to package software into standardized units called containers, which include everything that software needs to run. This means that for the assessment, the database is seemingly separated from the environment used to run Vihjaa while populated with example data of a fictional customer.

6.1.1 Operating System

As mentioned in the previous paragraph, the operating system used for the security, privacy, and legislation adherence assessments in this thesis is the Debian-based Linux distribution called Kali Linux. Kali Linux, previously known as BackTrack Linux, was originally released on the 26th of May in 2006. However, it was rebranded and rebuilt around the Debian distribution in March 2013 by an American information security and penetration testing company called Offensive Security. Kali Linux was chosen as the operating system for the assessment part of this thesis for three

main reasons. First, it comes pre-packaged with a large variety of tools aimed at various information security tasks, configurations, and automations to get the user set up quickly. Second, there is a lot of information available on how to conduct a security audit using the platform. Third, it is free to install and use, making it the ideal platform to use for non-profit projects. The only major downside is a fairly steep learning curve, as the operating system is targeted cybersecurity professionals and hobbyists that know what they are doing. However, by sacrificing some of the user experience, Kali Linux manages to offer an efficient way to conduct security auditing and penetration testing. [62]

6.1.2 Methodology

The methodology used for the security, privacy and legislation adherence assessments can be split into two categories. The security assessment follows the instructions provided by different sources listed shortly, while the privacy and legislation adherence assessments are performed based on how each concern or requirement was interpreted. The security assessment will mostly follow the instructions on the OWASP Foundation's Top Ten Vulnerabilities list related to each vulnerability [33] and Stuttard et al [31], Hoffman [40], Singh [63] and Najera-Gutierrez [64]. During each vulnerability assessment section, the methodology of how each vulnerability was assessed is discussed in addition to what instructions were followed. For the privacy and legislation adherence, each privacy concern and legislation requirement is interpreted and discussed and then assessed based off of knowledge and information available to the thesis writer at the time of writing. This is due to the fact that while most of the legislation requirements are quite straightforward to assess after interpretation, the privacy concerns are very case specific and as such, there does not exist any frameworks or instructions that could be followed during that section. Thus, both the privacy and legislation assessment sections aim to explain how each

concern and requirement was interpreted and what the most appropriate method of assessing them was.

6.2 Security Analysis

In this section we will conduct the security assessment of the Vihjaa web application using the OWASP Top Ten Web Application Vulnerabilities 2021 list that was covered in Section 4.3. The vulnerabilities themselves and how they can be exploited were mostly explained during that section, so this assessment will strictly focus on testing if those vulnerabilities are found in Vihjaa. Each vulnerability will be tested for using the instructions and information provided by the references covered in Section 6.1 where applicable. Additionally, if any tools are used, they will be mentioned. The security assessment will mostly focus on Vihjaa's admin panel as the reporting side is open for anyone without registering. In the case that a vulnerability is found to be applicable also to the reporting side, it will be mentioned separately in that section.

A01 – Broken Access Control

In order to test Vihjaa for the broken access control vulnerability, we created two user accounts for the admin panel as shown in Figure 6.1. One account had the access rights of a "super user", meaning that they have access to every section of the admin panel and also modifying organization settings, user information, channels and so on. The other account had the access rights of "maintainer", meaning that this is a role suitable for IT support personnel as they only have access to organization information, user groups and user administration. As such, the "maintainer" account should not be able to access reports, channels, or subscription information.

The vulnerability was then tested for using information regarding access control from Singh [63] and Stuttard et al. [31] along with the Burp Suite application

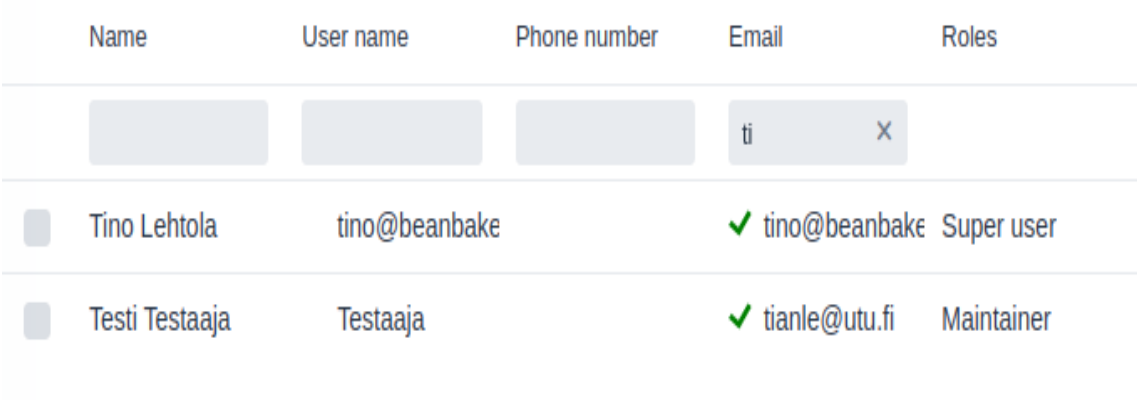

Figure 6.1: Users accounts used for the security assessment

security testing software. First, Burp Suite was used to build a site map of Vihjaa's admin panel from the point of view of both the "super user" and the "maintainer" accounts to see what sections of the application they had access to and what different requests are sent when navigating the application. Additionally, POST requests sent from the "super user" were saved within Burp Suite in order to resend those requests from the "maintainer" account to see if, for instance, requests to modify channel names would be accepted. Furthermore, the Vihjaa APIs were also tested without logging in, which in every tested scenario resulted in a "403 Forbidden" response. That was also the case when trying to resend "super user" modification requests from the "maintainer" account as mentioned previously. However, a vulnerability was detected when attempting to visit the different tabs of the admin panel. In Figures 6.2 and 6.3, the subscription tab from the point of view of the "super user" and the "maintainer" roles are shown. In both, the GET request sent to the "/subscription" URL gets accepted with the HTTP status 200 that is shown with Burp Suite.

In the case of the "super user", the subscription tab is shown normally on the left in the navigation bar, although it is not shown for the "maintainer" role. That is because a user with the "maintainer" role is not supposed to be able to access the subscription tab or change information within it. The subscription tab includes information about the customer and the payment method used to pay for the Vihjaa

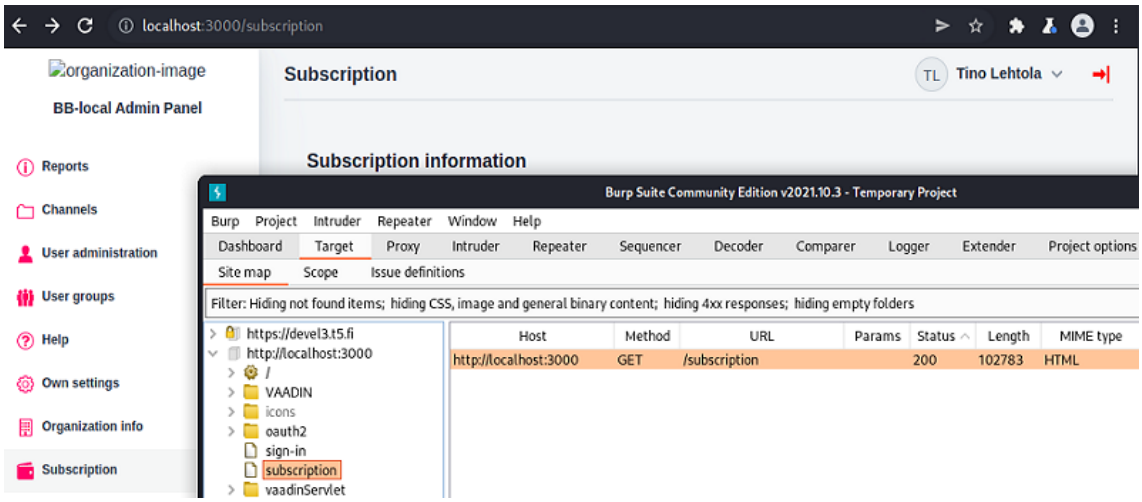

Figure 6.2: Vihjaa admin panel subscription view as a "super user"

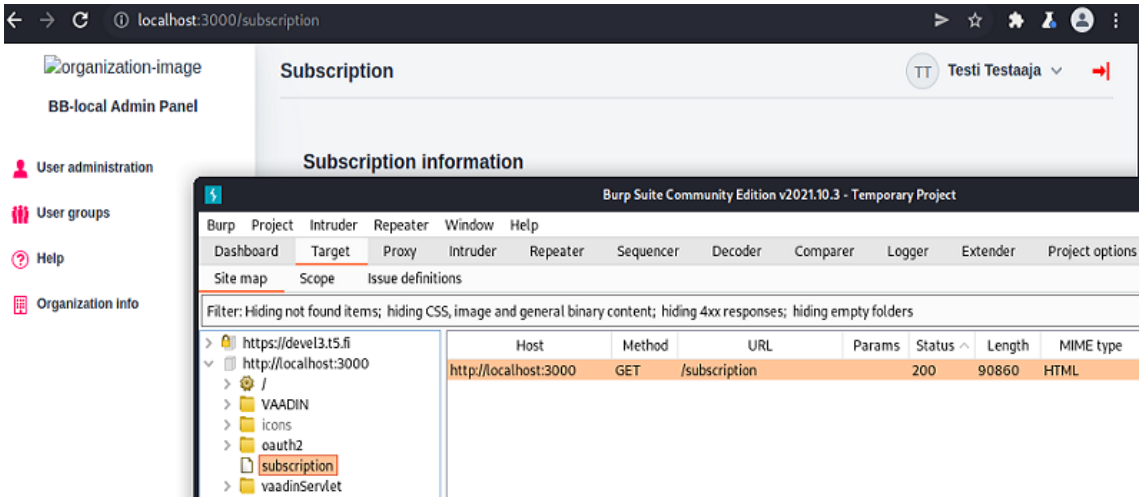

Figure 6.3: Vihjaa admin panel subscription view as a "maintainer"

service, which are all information that should not be accessible by a “maintainer”. The access control check was bypassed by simply adding the “/subscription” parameter to the base URL, which does not work for any of the other tabs that should be hidden from the “maintainer” role. Those tabs correctly display a message telling the user that they are not authorized to access that information and to choose a valid tab from the navigation bar. This vulnerability could easily just be an oversight, as the subscription page and its related functionality was added just before

this security assessment was made. It should nevertheless be patched out as soon as possible, so that personal data from the person that is listed as the contact person on the subscription does not leak. Other than that, no other issues or vulnerabilities were found with the access control of Vihjaa's admin panel.

A02 – Cryptographic failures

In Section 4.3 we discussed what different technical shortcomings in a web application can lead to a web application having vulnerabilities related to cryptographic failures. In order to assess this vulnerability, we will utilize the description given by the OWASP Foundation in their top ten list. From that description we will choose the weaknesses that can be applied to Vihjaa and tested in an environment specified earlier in this section. Firstly, we will assess the HTTPS and TLS protocols that were introduced in Section 3.2. As was discussed in that section, HTTPS requires a TLS certificate that ensures the secure connection between the client and the server. Unfortunately we cannot install such a certificate using the current assessment setup, meaning that we will instead test it on the BeanBakers Vihjaa reporting application that is currently in production. As seen in Figure 6.4, the ilmoita.vihjaa.fi website has a TLS certificate provided by Amazon which enables the HTTPS protocol. This is accomplished by using the AWS Certificate Manager (ACM), which a service provided by Amazon that lets developers easily provision, manage and deploy private and public SSL/TLS certificates for their applications in the AWS ecosystem. As a part of assessing the A05 security misconfiguration vulnerability in this section, we also scanned Vihjaa for open ports. The only open ports found were ports 443 and 80, that are used for the HTTPS protocol communication. Furthermore, as will be shown in the beginning of the privacy assessment section later on, Vaadin's in-built security features encrypt every request made to the application which means that there is no data transmitted in clear text.

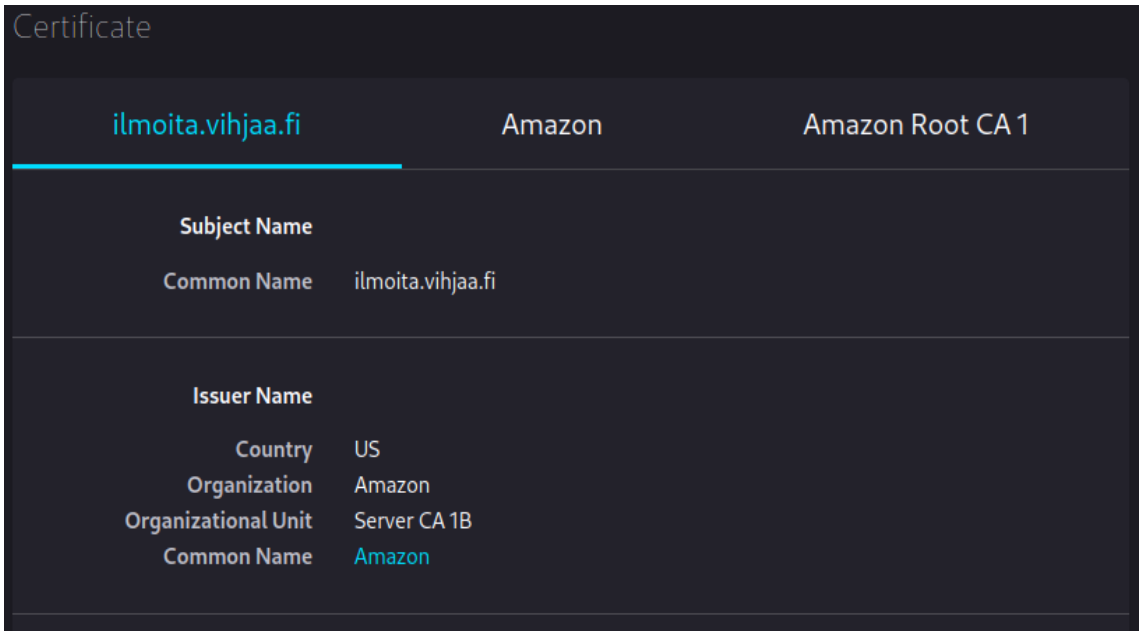

Figure 6.4: The TLS certificate for ilmoita.vihjaa.fi provided by Amazon

When the reporting application receives data that needs to be stored in the database, the data is encrypted before transmitting. For encryption, Vihjaa utilizes the industry standard open-source encryption library called Tink that was created by cryptographers and security engineers at Google and released in 2018 [65]. Tink functions on top of the Java Cryptographic Architecture and adds additional security features such as support for external Key Management System, which in Vihjaa's case is the AWS KMS that was discussed in Section 3.4. From Tink, Vihjaa relies on the Authenticated Encryption with Associated Data (AEAD) primitive which offers secrecy and authenticity using the AES128_GCM key type with the SHA256 hash function. The AEAD ensures that every encrypted message has different encrypted outputs, even if the plaintext input is the same. This increases the strength of the encryption as possible attacks on the encrypted outputs cannot identify patterns for the same inputs. As was discussed during the AWS KMS section mentioned previously, each organization has their own encryption keys that are saved in the AWS KMS service that also performs additional cryptographic operations on the

data before it is stored in the database. Using AWS KMS removes the weaknesses related to weak crypto keys, crypto keys re-usage, proper key management and encryption keys being checked into source repositories as all of those are handled inside Amazon's secure environment. Overall, we can conclude that based off of the assessments we can perform using our testing environment, Vihjaa does not have any vulnerabilities related to cryptographic failures.

A03 – Injection

The basics of how an injection attack works was discussed during Section 4.3, but as a short recap, injection attacks take advantage of how user inputs and SQL queries are handled by backend processes. For instance, when a user logs into a web application using their login information, that information is verified against a database and their information is fetched if the login information matches a database entry. A login injection attack would capture a legitimate login attempt request, then modify that request by changing to parameters to escape a SQL query into their login attempt and then resend that request. If the web application is vulnerable to an injection attack, the response might include information about valid user accounts saved in the database. The attacker can then use that information to login on someone else's account. As explained in Section 3.5, Vihjaa uses MongoDB as their database which is a NoSQL database, and all personal data is encrypted before saving. As explained during that aforementioned section, the language and queries used for relational and non-relational databases are different, meaning that we must use information that explicitly targets a non-relational database. Furthermore, as was mentioned in Section 3.6, users logging into the admin panel log in through an identity access management platform that is hosted by the Trivore Corp. This means that user login information is saved on Trivore's servers and performing a security assessment on their IAM is not part of the scope of this thesis. Thus, we must focus

on the data saved in each organization's MongoDB database hosted in the AWS cloud. That data namely being the reports, personal data of the reporting persons, and other organizational information like subscription information. Utilizing the information presented by Hoffman [40] about injecting into MongoDB databases, we will test that method on the admin panel. There are not many fields in the Vihjaa admin panel that support user input and those that do, for instance the search bar in the "channels" tab, has implemented user input sanitization on the backend side. Additionally, trying to modify the requests where data was either being added to the database or already existing data was being modified did not work either, as those requests received either the "401 Unauthorized" or the "403 Forbidden" responses. The request responses were the same also from the reporting side of the application. Thus, we will conclude that no injection vulnerabilities were found in Vihjaa.

A04 – Insecure design

In Section 4.3 it was discussed that insecure design is not considered a vulnerability itself, but it is the most prevalent root cause for vulnerabilities to get discovered later on and possibly exploited. The most important takeaway from that section was the idea that if a web application has insecure design, it cannot be fixed even with perfect implementation. As we only have the finished Vihjaa product to assess, it is difficult the determine whether or not Vihjaa has followed the "Security-By-Design" principle or what choices were made during the early stages of development that have played a part in Vihjaa's general design. Due to this fact, we will examine the prevention methods the OWASP Foundation has listed for this vulnerability and assess whether or not those measures have been considered. We will focus our assessment on the prevention methods that could still be implemented or utilized at this stage of development now that Vihjaa has reached production. The first

prevention method from the list is to establish and use a library of secure design patterns and ready to use components. This method has already been implemented in Vihjaa's design, because BeanBakers Ltd has a library of frameworks and common components that are periodically updated and used in multiple projects including Vihjaa. The second measure is the use of threat modeling overall in the application, focusing especially on authentication, access control, business logic, and key flows. Based off of discussions with Vihjaa's development team, threat modeling has been utilized during the planning and development of the application. However, it was pointed out that it could be used to a greater extent and that the results of this thesis will be adopted into the threat modeling process in the future. The third measure is to write unit and integration tests in order to validate that all critical flows are resistant to the threat model. This is one way how threat modeling could be used in a greater extent to automatically test Vihjaa for vulnerabilities or general bugs that would improve Vihjaa's overall state of security in the long run. The fourth measure is hiring the assistance of application security professionals to help further evaluate and assist in improving Vihjaa's state of security as well as privacy-related concerns. The final prevention method from the list that to some extent incorporates most of the prior measures, is the utilization of the OWASP Foundation's Software Assurance Maturity Model (SAMM), which is an open framework aimed to aid organizations analyze their current software security practices and improve upon them [66]. The framework incorporates measures like threat modeling, security testing, incident management and operational management and provides examples on how they can be used to improve software security posture. In conclusion, insecure design is difficult to assess once the target application has reached production without having been part of the development phase of an application. This resulted in us having to assess whether or not Vihjaa has made use of the prevention methods presented by the OWASP Foundation that would help identify and address possible

issues in Vihjaa's design. We found that most of the measures mentioned have been used, but they could be utilized to greater extent to further improve Vihjaa's overall state of security.

A05 – Security Misconfiguration

Like insecure design, security misconfiguration is another difficult vulnerability to assess as it can be seen as an umbrella term for multiple different security issues that lead to vulnerabilities. Additionally, the definition of what security issues can be classified as security misconfigurations might change depending on who provides the definition. This is mostly due to how broad the term "security misconfiguration" is, and how many things in a web application the term can refer to. As such, we will assess this vulnerability by determining if Vihjaa has any of the security issues that the OWASP Foundation has listed under the security misconfiguration vulnerability. We will adapt some of the listed security issues to make them applicable to Vihjaa and the scope of this security assessment. Furthermore, we will address each issue inside the list provided below and then sum up our findings at the end of this section. With that in mind, Vihjaa has security misconfiguration related vulnerabilities if:

- *Unnecessary features are enabled or installed, for instance ports, pages, or privileges.* Port scans of the Vihjaa admin panel and the reporting application show open ports 80/tcp and 443/tcp. Port 80 is most commonly used for the Hypertext Transfer Protocol (HTTP) discussed in Section 3.2, while port 443 is most commonly used for HTTPS which was discussed in the same section. This means that there are no unnecessary ports open for either application. Unnecessary pages were tested for during A01, where we found a security vulnerability while accessing the admin panel's subscription page with privileges that should not be able to access it. In terms of privileges, Vihjaa has four user roles: user, admin, maintainer, and super user. These roles were also

tested in A01 and it was found that the privileges function as defined in the documentation. In general, there were no unnecessary libraries or features implemented into Vihjaa that would predispose the web application to outside threats.

- *Default accounts and their passwords are still enabled and unchanged.* Due to the fact that all identification and authentication for the Vihjaa admin panel goes through the Trivore IAM, there are no default accounts or default passwords enabled for any organization. The first admin level account for the service is always created per customer organization by the customer themselves. Instructions on creating an account are sent from the Trivore IAM by someone from BeanBakers Ltd and the customer can then use that account to create any further needed user accounts for the service.
- *Error handling reveals stack traces or other overly informative error messages to users.* After seeing many error messages during the assessment of vulnerabilities A01 and A03, not once were there overly informative error messages that would have exposed sensitive information to the user. Displaying error messages in the front-end is handled by the Vaadin framework, which has built-in security features that prevent the application from revealing stack traces or other over informative error messages. Due to this, the user is only displayed a general error message while the actual stack trace is logged in the back-end for further analysis.
- *Security settings in the application frameworks are not set to secure values.* The two main application frameworks used in Vihjaa that have been discussed previously in Section 3.1 in this thesis are Vaadin and Spring Boot. Vaadin is fully compatible with Spring Boot's security framework called Spring Security and it also has built-in security helpers that streamline the process of implementing security. Spring Security is implemented into a project by importing the security framework dependency in the project's pom.xml and then

configuring it in a security configuration file. The security configuration file, for example called "SecurityConfiguration.java", should then either extend the web security superclass from Vaadin's side called "VaadinWebSecurity" or from the Spring Security framework called "WebSecurityConfigurer". After assessing Vihjaa's security configuration file for default and insecure values, we can conclude that it follows industry guidelines and does not have any settings set to insecure values.

- *Software is out of date or vulnerable.* This issue is addressed as a part of the A06 – Vulnerable and outdated components section.

Overall, we can conclude that Vihjaa does not have any security misconfigurations as defined by the OWASP Foundation. The only issue we found was with account privileges, but that was already identified during A01 where the vulnerability was also discussed.

A06 – Vulnerable and outdated components

Manually checking through all the possible vulnerable or outdated components in a project that is of the scale of Vihjaa is both ineffective and inefficient. That is why we will use a tool made by the OWASP Foundation called "OWASP Dependency-Check", which attempts to automatically detect publicly disclosed vulnerabilities contained within a project's dependencies as well as check for outdated components. It does this by analyzing the dependencies listed in the project's pom-file and then comparing that information to the National Vulnerability Database (NVD) that is maintained by the National Institute of Standard and Technology (NIST). Using the tool itself is simple enough, as the tool can be downloaded from the OWASP Foundation's website and then executed in the command line [67].

For example, one of the scans conducted was started with the following command: *$ ./dependency-check.sh –project Vihjaa-backend –scan /home/tino/Documents/BeanBakers/vihjaa-backend –out /home/tino/Documents/results/vihjaa-backend.* This automatic analysis was conducted on all repositories containing Vihjaa-related files. In addition, because the reporting side of the application uses the Node Package Manager (NPM) for managing its dependencies, we also ran a security audit scan using NPM's built-in feature called "npm-audit". The information was combined into a neat HTML report using another node package called "npm-audit-html", which can be installed into the project using *$ npm I -g npm-audit-html.* The audit was then performed with *$ npm audit -json | npm-audit-html*, which combines the information into HTML format by providing the "npm-audit-html" pipe in the command. While the results of these audits cannot be disclosed in this thesis, there were multiple possibly vulnerable or outdated components found by the scans. The severity of the vulnerabilities ranged from low to critical, meaning that some of the issues must be addressed immediately. As such, the reports showing the results of the scans were given over to the team responsible for the development of Vihjaa. In the future, it would be good practice to automate a scan of the Vihjaa repositories periodically so that possible vulnerabilities could be addressed as soon as possible.

```
[INFO] Analysis Started
[INFO] Finished Archive Analyzer (2 seconds)
[INFO] Finished File Name Analyzer (0 seconds)
[INFO] Finished Jar Analyzer (0 seconds)
[INFO] Finished Central Analyzer (0 seconds)
[INFO] Finished Dependency Merging Analyzer (0 seconds)
[INFO] Finished Version Filter Analyzer (0 seconds)
[INFO] Finished Hint Analyzer (0 seconds)
[INFO] Created CPE Index (2 seconds)
[INFO] Finished CPE Analyzer (6 seconds)
[INFO] Finished False Positive Analyzer (0 seconds)
[INFO] Finished NVD CVE Analyzer (0 seconds)
00:00  INFO: Vulnerability found:
00:00  INFO: Vulnerability found:
00:00  INFO: Vulnerability found:
00:00  INFO: Vulnerability found:
00:00  INFO: Vulnerability found:
00:00  INFO: Vulnerability found:
00:00  INFO: Vulnerability found:
[INFO] Finished RetireJS Analyzer (1 seconds)
[INFO] Finished Sonatype OSS Index Analyzer (0 seconds)
[INFO] Finished Vulnerability Suppression Analyzer (0 seconds)
[INFO] Finished Dependency Bundling Analyzer (0 seconds)
[INFO] Finished Unused Suppression Rule Analyzer (0 seconds)
[INFO] Analysis Complete (12 seconds)
[INFO] Writing report to: /home/tino/Documents/Tulokset/Backend/dependency-check-report.html
```

Figure 6.5: Dependency Checker analysis example

A07 – Identification and Authentication Failures

As was explained in Section 3.6 and discussed during the injection vulnerability assessment of this section, users accessing the Vihjaa admin panel are identified and authenticated using the Trivore IAM platform. Since the goal of this thesis is not to perform a security assessment on Trivore's platform, we are quite limited in terms of assessing Vihjaa's identification and authentication failures. We can nonetheless try to identify weaknesses linked to this vulnerability in both the Trivore IAM on a surface level and in Vihjaa and assess those instead. Here is a list of weaknesses that will be assessed in this section:

Trivore IAM: Are weak passwords permitted, are there weak or ineffective password recovery processes and is there multi-factor authentication.

Vihjaa admin panel: Are session URLs exposed in the URL or requests, are session identifiers reused and are application session timeouts set correctly (single sign-on (SSO) tokens).

Beginning with the Trivore IAM, the default password security settings can be seen in Figure 6.6. As can be seen from that figure, the password settings range from number of required character classes to number of accepted failed sign-in attempts. In the former, character classes refer to lower case letters, upper case letters, numbers, and special characters. Additionally, these exists a setting to block dictionary words and usernames from being used in passwords. Thus, we can conclude that weak passwords are not permitted. Password recovery is managed by sending a password recovery link to an email address, if that email address is linked to an account, and multi-factor authentication can be setup for an account but at this time it is not mandatory. As a suggestion, persons that have super user or admin level privileges to an organization's Vihjaa admin panel should be instructed to setup two-factor authentication in the Vihjaa user manual or in a separate document. Overall, we can conclude that the Trivore IAM does not have any of the weaknesses mentioned previously.

Moving on to the Vihjaa admin panel, we will first test if session URLs are exposed either in the URL or requests. Checking the URL itself is simple since all it takes is to look at the address bar if it includes any session URLs or identifiers, which it does not at any point when using the Vihjaa admin panel. Next, we will login Vihjaa using our super user account and test whether or not session URLs are exposed in the requests. Following the testing instructions of Najera-Gutierrez [64] regarding session URLs, we use the Burp Suite intercepter tool to view every request sent during the login process to see if session URLs are exposed. Neither during the Trivore IAM authentication process nor the redirection to the Vihjaa admin panel can session URLs be viewed in the requests. Session URLs not being exposed also mitigates the chance of a session identifier reuse attack since the attacker cannot figure out a user's session identifier. However, because every user accessing the Vihjaa admin panel has to authenticate themselves through the Trivore IAM, they

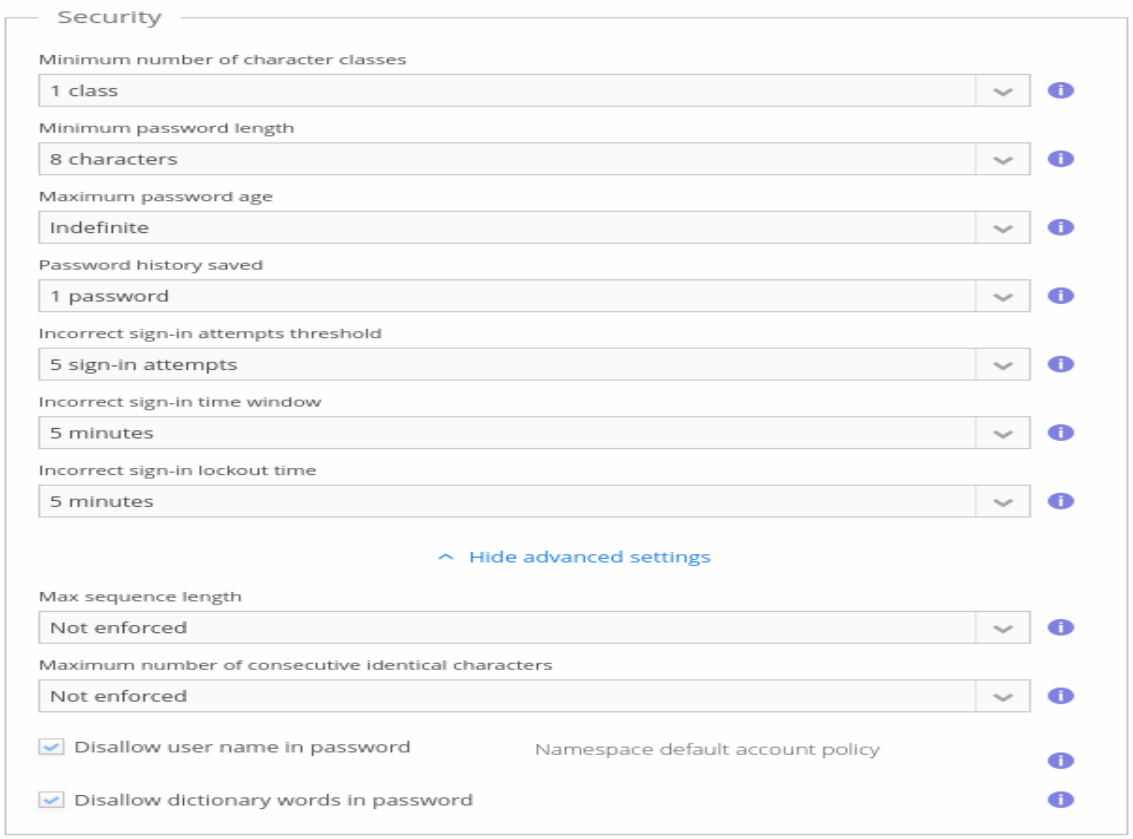

Figure 6.6: Trivore IAM password security settings

receive a new session URL each they login, also mitigating this type of attack. Finally, the application session timeouts will be discussed further in Section 6.3.2, but we found that sessions are not correctly timed out if a user stays inactive on any page of the Vihjaa admin panel. This is an important security feature that should be added, as users sometimes leave applications open on their devices when they leave their workstations which could lead to an unauthorized person accessing the Vihjaa admin panel. Overall, we can conclude that none of the listed vulnerabilities were discovered in the Trivore IAM and out of the three weaknesses mentioned for the Vihjaa admin panel, issues were only found with session expiration.

A08 – Software and Data integrity Failures

As was explained in Section 4.3, software and data integrity failures mainly refer to code and infrastructure that does not protect against integrity violations. These include a web application relying on libraries or modules from untrusted sources or repositories, insecure continuous integration / continuous delivery (CI/CD) pipeline that introduces potential for unauthorized access to the source code, and automatic updates that get downloaded for the application without sufficient integrity verification. The first of these weaknesses was covered with the A06 vulnerability, which found that there are third party libraries and modules implemented into Vihjaa at this moment with varying levels of vulnerability severity ratings. The best solution, although one that takes quite a lot of time to implement, would be to setup a repository manager for Vihjaa. This repository manager would include all the dependencies installed through Maven and Node Package Manager for the Vihjaa project in its own repository, which would make verifying dependency integrity a lot more efficient. The basic idea of a repository manager would be to create a repository with the latest verified versions of all of Vihjaa's dependencies and then always have Vihjaa only update them through that repository. The repository manager would periodically and automatically search trusted sources for updates for those repositories, as well as alert the developer if there is a new vulnerability detected in any of the dependencies. The manager would never automatically update the dependencies, instead always asking the repository admin for permission to update dependencies. This would create a sort of safe zone of trusted and verified dependencies that would negate the weaknesses of Vihjaa downloading dependencies from untrusted sources or automatically downloading a malicious update for its libraries or modules. Additionally, if the web application has a higher risk profile, OWASP Foundation suggests hosting an internal known-good repository that is vetted manually. Moving on to discussing the CI/CD pipeline, Vihjaa uses GitLab as its DevOps

platform. On GitLab, Vihjaa is only shared to members of the development team and the writer of this thesis for the duration of the writing and assessment process. Additionally, in order to access the Vihjaa repositories, the GitLab account must have two-factor authentication enabled. Continuing on to code integrity checking during the CI/CD pipeline, some shortcomings were identified during the assessment process. Most importantly, the fact that there are no automatic tests or security tools that would test either every commit or every merge. Tools such as the OWASP Dependency Check that was used in A06, general code integrity checkers that would automatically scan every commit for bugs or security vulnerabilities like SonarQube, Codacy, Embold, or DeepSource, and custom unit or integration tests that would also test Vihjaa for bugs are all missing. Not only would these tools and tests aid the developers to discover bugs or vulnerabilities in the code, but they would also improve Vihjaa's overall state of security and make sure that no broken code is committed to the repositories. In conclusion, there were a lot of suggestions given during this section on how to mitigate software and data integrity failures in Vihjaa. These ranged from using a repository manager that would automatically maintain dependencies to tools and tests that should be implemented to run during the CI/CD pipeline. Implementing some or all of these suggestions would do a lot to improve not only Vihjaa's state of security, but also its code quality.

A09 – Security Logging and Monitoring Failures

Failures with security logging and monitoring is another vulnerability that is hard to test. The OWASP Foundation suggests that logging should be tested by penetration testing the application or interviewing developers attached to the project if they have noticed anything in the logs. Since we only have access to our local testing environment version of Vihjaa, we cannot test how the security logging or monitoring would work in the AWS environment. That is why we will mostly rely on interviews

with Vihjaa's development team and what measures they have currently put in place in that environment. We discovered during A07 that the Trivore IAM has security settings for incorrect sign-in attempts, meaning that they have the capability to log those events. Based off on interviews, currently that information is only logged onto Trivore's servers, meaning that BeanBakers Ltd or its developers do not have access to those logs. Suggestion for this would be a way for who ever is on the security response duty for Vihjaa to either have access to those logs, or at least receive alerts if it seems that an organization's Vihjaa admin panel is being attacked. For the admin panel itself, at the moment only mundane server activity is being logged on the server itself. It is worth mentioning that the AWS control panel does have logging tools and analytics of the activity on all servers running Vihjaa, but it would be a good idea to setup automatic server alerts when there is an unexpected spike in activity, or if a server begins showing a lot of errors or warnings in its log. This way the person on server duty would receive a notification as suspicious activity is taking place and provide them a chance to investigate before any damage is dealt. As such, suggestion for this vulnerability is to establish a way of sharing logs with the Trivore IAM and implement effective monitoring and alerting on the AWS dashboard that lowers the chance of an attack going unnoticed.

A10 – Server Side Request Forgery (SSRF)

Server Side Request Forgery vulnerabilities were shortly explained in Section 4.3, but we will go slightly into more detail here. In a typical SSRF attack scenario, an external attacker is trying to access a server or resource inside a private network that is blocked by a firewall or other network security measures. Since the attacker identifies that they cannot directly attack the victim server, they must try to identify a vulnerable web server that is accessible from the outside that has access to the victim server inside the private network. Once an attacker has gained access to the

vulnerable web server, they can use tools like Burp Suite to modify requests made to the server and instead direct them towards the victim server. A high-level overview of how the attack proceeds is shown in Figure 6.7.

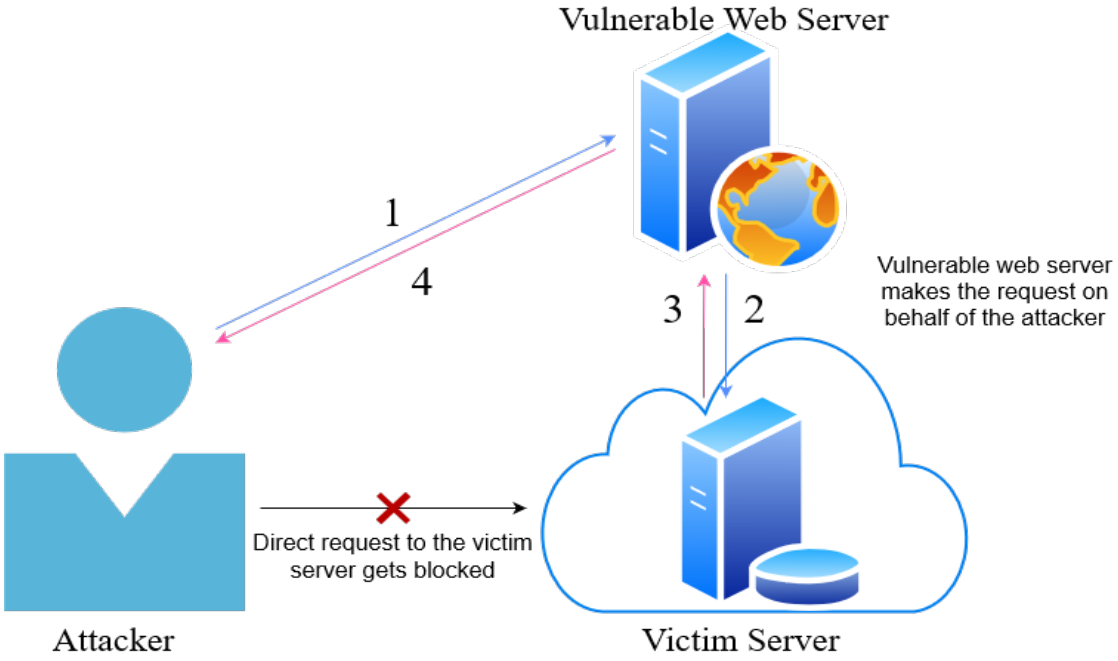

Figure 6.7: High-level overview of a SSRF attack

Due to how the attack functions and what it tries to accomplish, it is not applicable to Vihjaa. The main reason being that SSRF as a vulnerability is mitigated with the technology and environment used to run Vihjaa. Firstly, the Vaadin framework has-in built protections against both cross-site (CSRF) and server side request forgery attacks, meaning that requests on neither the admin panel nor the reporting side can be exploited to take advantage of this vulnerability [68]. Secondly, because Vihjaa is hosted on the Amazon Web Services cloud service, assessing this vulnerability falls out of the scope of this thesis. This is due to the fact that the party affected by the SSRF vulnerability in that case would be AWS, not Vihjaa itself. It is worth nothing however, that in recent years SSRF vulnerabilities have been identified in AWS that allowed the attacker to exfiltrate server side metadata [69].

Even so, we can conclude that the Server Side Request Forgery vulnerability was not detected within Vihjaa, and no actions are needed.

In Table 6.1, we have combined the results of the Vihjaa security assessment conducted in this section.

Vulnerability Code	Were Issues Found Related to this Vulnerability? (No / Yes)	Suggestion / Notes
A01	Yes	The "/subscriptions" URL was found to be accessible on accounts that should not ahve to privileges to access that page. Suggestion to double check the permissions for every page of the web application.
A02	No	Vihjaa was not found to have any weaknesses or cryptographic failures related to the A02 vulnerability.
A03	No	Vihjaa was not found to have any vulnerabilities that would predispose it to an injection attack.
A04	Yes	We identified that while most of the measures mentioned during the section were implemented, they could be utilized to a greater extent. Most importantly the use of threat modeling and writing automated tests that ensure code integrity would improve Vihjaa's overall security.
A05	No	Vihjaa was not found to have any security issues listed by the OWASP Foundation under A05. The only identified issues was with account privileges, but that was already identified in A01.
A06	Yes	The dependency scans found outdated and/or vulnerable dependencies being used in Vihjaa. The scan results were given to the Vihjaa development team and suggestion to implement dependency checks to the CI/CD pipelines.
A07	Yes	For the Trivore IAM, customers should be instructed to setup multi-factor authentication for the admin and super user accounts. For the Vihjaa admin panel, session expiration should be implemented so that Vihjaa cannot be left open in the browser idling.
A08	Yes	Suggestions given for this vulnerability included setting up a repository manager for Vihjaa and adding automated tools for dependency checking, checking code integrity and unit tests to the CI/CD pipeline.
A09	Yes	Better logging should be implemented for the Vihjaa server-side that would detect suspicious activity and alert the Vihjaa development team if necessary. This would decrease the chance of an attack on the application being successful, since the attack could be responded to while it is happening.
A10	No	The SSRF vulnerability was not was not found to be applicable to Vihjaa, so no actions are needed.

Table 6.1: Vihjaa security assessment results

In the "were issues found related to this vulnerability?" column, each security vulnerability is rated with a "no" or "yes" rating. "No" rating means that no issues were found related to that vulnerability. "Yes" rating means that issues were found and suggestions for remedying the issue are given in the next column. In the "suggestion / notes" column, either suggestions or notes are given on what actions should be taken for that vulnerability to be addressed, if any.

6.3 Privacy Analysis

In this section we will conduct the privacy analysis of the Vihjaa web application using the privacy concern lists gathered in Sections 5.3 and 5.4. The assessments are divided into their own subsections, former of which being the end-user privacy concerns and latter being the technical and organizational privacy concerns.

Before assessing the privacy concerns however, we will analyze whether or not Vihjaa actually tracks or monitors the users while using the web application. Since the web application is split into the admin panel and the reporting side, we will look at both individually. The methodology for this analysis will be mostly the same that was used in Heino et al. and Carlsson et al. during Section 5.2 [51], [52]. We will use Google Chrome's Developer Tools (DevTools), which is a set of tools that allow inspecting the rendered HTML and network activity of web pages. This allows us to see the network activity of a web page and assess whether or not it gathers personal data about the user or monitors their activities. Additionally, we used a browser extension called Ghostery which displays if a website has installed any trackers [70]. First, we will analyze Vihjaa's admin panel. For this analysis, Vihjaa's network traffic was gathered by visiting the different tabs of the web application, which can be seen in Figure 6.8. Each time a new tab was opened, Vaadin sent a new request that can be seen from the aforementioned figure as the requests with the name "?v-r=uidl&v-uiId=7". Additionally, as can be seen in Figure 6.9, there

are no trackers on the site and the application cannot track the Vaadin requests, meaning that a user's actions cannot be monitored.

Name	Status	Type	Initiator	Size	Time
channels	200	docum...	Other	160 kB	1.15 s
webcomponents-loader.js	200	script	channels	6.9 kB	45 ms
vaadin-bundle-1f485ac0af4e...	200	script	channels	16.0 MB	177 ms
vaadin-gizmo-f520d0b6951b...	200	script	channels	475 kB	66 ms
vaadinPush.js?v=2.4.0	304	script	channels	262 B	25 ms
client-C4B53966888EAB6C...	304	script	channels	262 B	23 ms
data:application/fo...	200	font	channels	(memo...	0 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	437 B	5 ms
orgLogo.jpg	200	jpeg	client-C4B539...	18.6 kB	76 ms
?v-r=push&refresh_connection	101	webso...	VaadinDevmo...	0 B	Pending
?v-r=push&v-uiId=7&v-pushI...	101	webso...	vaadinPush.js...	0 B	Pending
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	337 B	6 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	56.4 kB	1.11 s
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	438 B	5 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	337 B	12 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	337 B	8 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	36.3 kB	698 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	438 B	5 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	338 B	5 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	339 B	7 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	13.5 kB	303 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	341 B	9 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	339 B	8 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	339 B	7 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	10.4 kB	438 ms
Logo	200	octet-s...	client-C4B539...	18.6 kB	13 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	339 B	7 ms
?v-r=uidl&v-uiId=7	200	xhr	client-C4B539...	339 B	7 ms

Figure 6.8: Network traffic of Vihjaa's admin panel

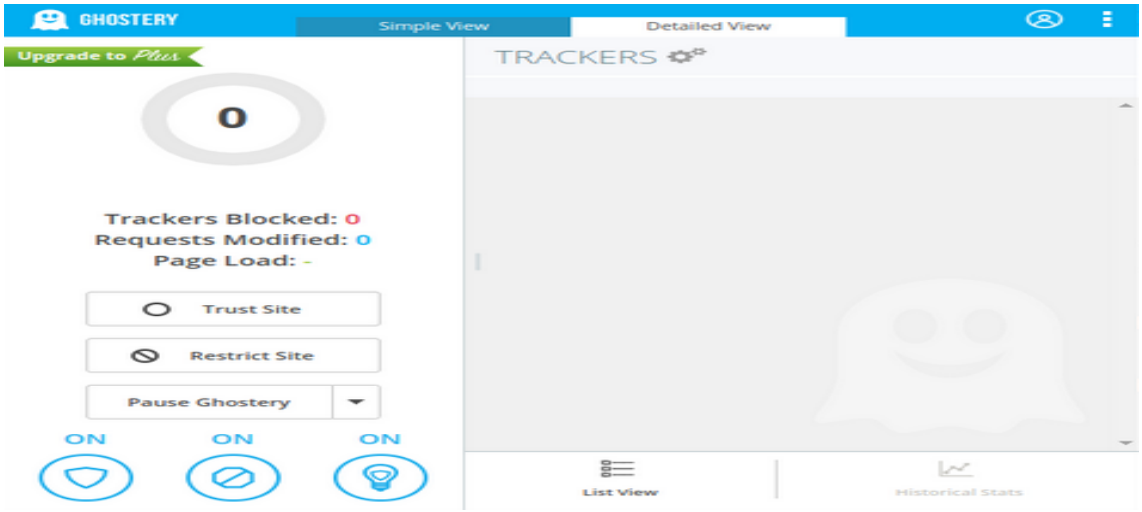

Figure 6.9: Ghostery report of Vihjaa's admin panel

Name	Status	Type	Initiator	Size	Time
beanbakers-oy	304	docum...	Other	201 B	22 ms
app.js	304	script	beanbakers-oy	203 B	19 ms
chunk-vendors.js	304	script	beanbakers-oy	204 B	204 ms
css2?family=Lora:ital@0;1&...	200	stylesh...	beanbakers-oy	(disk c...	17 ms
css2?family=Lora:ital@0;1&f...	200	stylesh...	beanbakers-oy	(disk c...	17 ms
materialdesignicons.min.css	200	stylesh...	beanbakers-oy	(disk c...	28 ms
0.js	304	javasc...	beanbakers-oy	203 B	25 ms
1.js	304	javasc...	beanbakers-oy	203 B	30 ms
10.js	304	javasc...	beanbakers-oy	203 B	29 ms
11.js	304	javasc...	beanbakers-oy	202 B	20 ms
12.js	304	javasc...	beanbakers-oy	202 B	20 ms
13.js	304	javasc...	beanbakers-oy	202 B	22 ms
14.js	304	javasc...	beanbakers-oy	202 B	17 ms
15.js	304	javasc...	beanbakers-oy	202 B	17 ms
16.js	304	javasc...	beanbakers-oy	202 B	26 ms
2.js	304	javasc...	beanbakers-oy	203 B	29 ms
3.js	304	javasc...	beanbakers-oy	202 B	20 ms
4.js	304	javasc...	beanbakers-oy	202 B	19 ms
5.js	304	javasc...	beanbakers-oy	203 B	22 ms
6.js	304	javasc...	beanbakers-oy	203 B	17 ms
7.js	304	javasc...	beanbakers-oy	203 B	16 ms
8.js	304	javasc...	beanbakers-oy	203 B	9 ms
9.js	304	javasc...	beanbakers-oy	202 B	9 ms
2.js	200	script	app.js:919	(prefet...	3 ms
11.js	200	script	app.js:919	(prefet...	4 ms
info?t=1666651619577	200	xhr	sockjs.js?9be...	451 B	4 ms
beanbakers-oy	200	xhr	xhr.js?b50d:187	2.8 kB	300 ms
logo.png	200	png	Other	1.4 kB	11 ms
websocket	101	webso...	sockjs.js?9be...	0 B	Pending
materialdesignicons-webfont...	200	font	materialdesig...	(disk c...	4 ms
Logo	200	octet-s...	VImg.ts?4649:...	(disk c...	6 ms

Figure 6.10: Network traffic of Vihjaa's reporting application

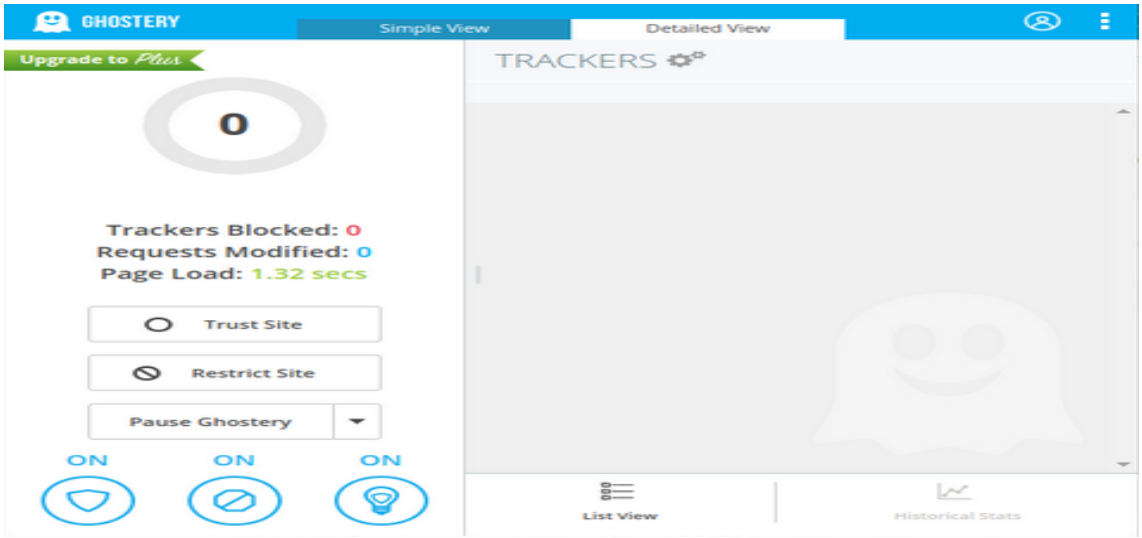

Figure 6.11: Ghostery report of Vihjaa's reporting application

For the reporting side of the application, the results stayed the same. For the analysis, we used BeanBakers Ltd's Vihjaa reporting channel and gathered its network traffic as we did with the admin panel. In Figure 6.10 the reporting application's traffic can be seen, which shows there are no trackers active on the site. The highlighted XMLHttpRequest seen in the figure also includes the line "cookies_needed: false", which communicates to the browser that the web application does not have cookies enabled. The full JSON response of that request can be seen in Appendix B. Additionally, the Ghostery report that can be seen in Figure 6.11 also shows that no trackers are active on the site meaning that the user cannot be tracked. As such, we can conclude that neither Vihjaa's admin panel or reporting application collects personal data about the user or monitors a user's activities while using the web application.

6.3.1 End-User Privacy Concerns

This section covers the end-user privacy concerns that were gathered from the two research papers discussed in Section 5.3. Each end-user privacy concern will be interpreted and how they might apply to Vihjaa will be explained, if not self-explanatory. Afterwards, the end-user privacy concern will be analyzed from Vihjaa's point of view, and possible suggestions will be given on how those concerns can be addressed.

UPC1 – What personal data does the application collect about the user?

This end-user privacy concern focuses on what kind of personal data is collected by the web application about the user with or without their knowledge. In Vihjaa's case, users can be separated into two categories. Those with user accounts that can access the admin panel, and the reporting persons using the application that do not need a user account. For the those in the organization with a user account that can be used to access that organization's Vihjaa admin panel, only their basic contact

information is collected and that is stored with the Trivore Corporation as Vihjaa utilizes their IAM platform as discussed in Section 3.6. For the reporting persons, only their email address, first name, last name, and phone number are stored, if the person gives permission for it. In that case, the reporting person must input their information on the report themselves. Other than that, no personal data is collected about the end-user, neither are their actions tracked while using the web application in any way as was analyzed and concluded at the beginning of the privacy assessment section.

UPC2 – How is that personal data processed and stored?

For Vihjaa in general, personal data is processed and stored on the Amazon Web Services cloud service in Europe as explained in Chapter 3. Users that have access to the organization specific Vihjaa admin panel, personal data is processed and stored by the Trivore Corporation on the servers of their IAM platform in Finland. On Trivore's platform, collected personal data is only used to authenticate the user before being able to access Vihjaa's organization specific admin panel. Furthermore, for both admin users and reporting persons, processed personal data only includes basic contact information.

UPC3 – Is that personal data shared with other parties or entities and if so, how?

Personal data that is collected from the reporting persons using Vihjaa is not shared with other parties or entities as of now. Users that are granted access to Vihjaa's organization specific admin panel must go through Trivore's registering process, which handles the personal data of those users for authentication purposes. In the future, it is possible that a feature is added to send submitted reports to supervisory authorities if necessary. However, that feature has not yet even reached the design

phase, so the answer to this concern at this time is no.

UPC4 – What measures designed to protect user privacy does the application provide?

In order to protect user privacy, several methods have been incorporated into Vihjaa's core design. These methods were discussed and explored in Sections 3.1, 5.5, and 6.2. In order to avoid unnecessary repetition, for Vihjaa to be able to pass the security assessment as well as the upcoming legislation compliance assessment, multiple privacy protection methods have been implemented into both the web application and the database.

UPC5 – Does the application's privacy policy accurately reflect the application?

As of the writing of this thesis, Vihjaa does not have a privacy policy document. Additionally, Vihjaa's Terms of Service document can currently only be accessed from the organization specific admin panel. As such, suggestions to address this concern is to create a privacy policy document and make Vihjaa's Terms of Service either widely available on its website or when a reporting person accessed the application.

UPC6 – How can the application guarantee the confidentiality of a user's messages or emails?

Vihjaa is not designed for the express purpose of sending user messages or emails. However, we can apply this concern to Vihjaa by discussing the functionality required by WD4 as discussed in Section 2.2.2 of being able to provide feedback on a submitted report as the reporting person. The dialogue between the report handler and reporting person is encrypted and at no point is the identity of the reporting person disclosed to the handler, unless the reporting person has provided that in-

formation on the report. Additionally, Vihjaa's encryption methods were analyzed in the previous section and those were found to be functioning as intended and up to the industry standard.

UPC7 – Is a user's activities monitored while using the application?

The actions of any user, be it someone using the admin panel or someone submitting a report on the reporting side, are not and cannot be monitored or tracked in any way. This functionality has not been built into Vihjaa, as was verified during the privacy assessment at the beginning of the privacy assessment section.

UPC8 – Can the user's personal data saved on their device only be accessed with their permission?

In general, Vihjaa cannot access any personal data or files saved on a user's device without their knowledge or consent. The only time that this is possible, is when a reporting person decides to add a file attachment to their report. However, Vihjaa cannot do this without permission or without the user choosing to do so themselves.

In Table 6.2, we have combined the results of the Vihjaa end-user privacy concern assessment conducted in this section. In the "has the Concern Been Addressed?" column, each end-user concern is rated with a "no", "somewhat", or "yes" rating. "No" rating means that the concern is valid and that Vihjaa is lacking a feature or documentation. "Somewhat" rating means that Vihjaa has addressed the concern, but there is still missing documentation, or it is explained unclearly. "Yes" rating means that Vihjaa has addressed the concern and that addressing it does not require any additional work. In the "suggestion / notes" column, either suggestions or notes are given on what actions should be taken for that concern to be addressed, if any.

UPC Code	Has the Concern Been Addressed? (No / Somewhat / Yes)	Suggestion / Notes
UPC1	Yes	Data collection and processing is kept to a minimum, and data is only collected from reporting persons that want to give out their information.
UPC2	Yes	Personal data processing and storing was explained as a part of this section.
UPC3	Somewhat	Currently no personal data is shared with other parties or entities, however in the future it is possible that such a feature might be implemented to share information directly with the supervisory authorities.
UPC4	Yes	Multiple measures have been implemented in Vihjaa's design as required by both the Whistleblower Directive and the GDPR.
UPC5	No	Vihjaa does not have a privacy policy document as of writing this thesis. Thus, such a document should be written.
UPC6	Yes	Due to the requirements posed by the Whistleblower Directive for the web application, encryption is used for all data processing and storing.
UPC7	Yes	The user's activities are not monitored while using the application.
UPC8	Somewhat	Vihjaa can only access the reporting person's personal data and files on their device if they decide to upload an attachment to their report. This should be declared in the privacy policy document.

Table 6.2: Vihjaa end-user privacy concern assessment results

6.3.2 Technical and Organizational Privacy Concerns

This section covers the OWASP Foundation's Top Ten Technical and Organizational privacy risks that was discussed in Section 5.4. Each privacy concern will be analyzed from either a technical or organizational point of view, and possible suggestions will be given on how those concerns can be addressed.

TOPC1 – Web Application Vulnerabilities

The OWASP Top Ten Web Application vulnerabilities were analyzed and addressed earlier in this chapter in Section 6.2.

TOPC2 – Operator-sided Data Leakage

In order to assess whether or not Vihjaa has addressed this concern, we must first interpret what is meant by "operator". From the PowerPoint presentation available on the OWASP Top Ten Privacy Risks website, we can construe that what the OWASP Foundation is referring to as "operator", the GDPR for instance refers to as "data processor". [59] The definition of which was given in earlier in this thesis in Section 2.3.1. In Vihjaa's case, operator and data processor both refer to BeanBakers Ltd, as they are the party processing the personal data behalf of the data controllers, which are the customer organizations of the Vihjaa service. Focusing on the concern itself, one of the objectives of the security and privacy assessments in this thesis is to identify the possible security or privacy vulnerabilities that can lead to operator-sided data leakage. As such, this concern can be partly addressed with the findings of those assessments conducted for this thesis. The results of those assessments show that the likelihood of an operator-sided data leak is low, but that does not mean that it cannot happen. As a suggestion to address this concern, the databases of organizations should be monitored for any suspicious activity and Vihjaa's state of security should be constantly maintained.

TOPC3 – Insufficient Data Breach Response

Vihjaa does not currently have an incident response or disaster recovery plans in case of a data breach. This issue will also be discussed again in the next chapter as a part of the GDPR legislation adherence. As such, we will shortly discuss some concrete steps on how a data breach response plan is built in this section. First of all, a great framework for building an incident response plan is the computer security incident handling guide released by the National Institute of Standard and Technology in 2012 [71]. The key steps of data breach response are having an incident response team with clear roles, that has prepared for an incident beforehand and has access

to tools and resources needed to detect and locate data breaches. In the case of a breach, the spread and severity of the data breach has to be identified, the leakage of data has to be stopped and involved parties need to be notified. After the situation has been taken into control, the method of entry has to be identified and fixed and other post-incident activities should be performed. [71] As such, the suggestion to address this concern is to design and implement a data breach response plan.

TOPC4 – Consent on Everything

Vihjaa does not currently collect consent from reporting persons submitting reports in the web application. This issue is further elaborated upon in the GDPR section of the next chapter, mainly when discussing GDPR1 in Section 7.1.2. The suggestion given in that section is to implement some sort of method to collect consent from reporting persons that add their personal data into their report. In order to address this concern, the way consent is collected should make sure to the reporting person what personal data is collected, how it is processed, and how it is stored. Additionally, since the collection and processing of personal data is kept to a minimum, there is no threat of inappropriate use of consent.

TOPC5 – Non-transparent Policies, Terms and Conditions

As identified as a part of UPC5 in Section 6.3.1, Vihjaa does not currently have a privacy policy document. However, Vihjaa's Terms of Service document does accurately represent how Vihjaa collects, processes, and stores personal data that can be easily understood by a layman. The access to the Terms of Service document should be nonetheless made easier, as was already suggested during Section 6.3.1. Additionally, a privacy policy document should also be created.

TOPC6 – Insufficient Deletion of User Data & TOPC7 – Insufficient Data Quality

These two privacy concerns have been grouped together, as they both directly refer to requirements made by the GDPR that are addressed in Section 7.1.2 of the following chapter. TOPC6 is discussed and addressed as a part of GDPR4 and TOPC7 is discussed and addressed as a part of GDPR3. Suggestions on how to comply with those requirements are also made in their respective sections.

TOPC8 – Missing or Insufficient Session Expiration

Session expiration is not a big problem on the reporting side of the application, as there are no user sessions, and a reporting person simply uses the web application to write and submit their report. However, on the Vihjaa's admin panel organization admins and handlers use their user accounts to log into the web application and perform their duties. Testing the admin panel's session expiration resulted in two discoveries. First, leaving the web application to go to another website in the same tab and coming back to it does log the user out, which means that it is working as intended. However, the second discovery was that a user will stay logged in for extended periods of time even if they are not interacting with Vihjaa. This can lead to a situation, where someone with access to the Vihjaa admin panel can leave it open in a browser tab which leaves it open for misuse. This can happen either by someone physically using the computer while that person is away, or by someone using the computer remotely. As such, the suggestion to address this concern is to implement effective session termination if the user has not interacted with the web application for a certain period of time. For instance, display a warning after 15 minutes of inactivity that tells the user that they will be disconnected in 5 minutes if they do not interact with the web application. If they do not do that in the following 5 minutes, terminate the session and redirect the user to the sign-in screen.

TOPC9 – Inability of Users to Access and Modify Data & TOPC10 – Collection of Data Not Required for the User-Consented Purpose

These last two privacy concerns have also been grouped together as was done with TOPC6 and TOPC7, as they both directly refer to requirements made by the GDPR. In this case, TOPC9 referring to GDPR3 and TOPC10 referring to GDPR8. Additionally, TOPC10 was also analyzed as a byproduct of the end-user privacy assessment conducted in the previous section. In that section we found that Vihjaa does not collect any data that it does not require, or process the data in question in a way that the user has not consented to.

In Table 6.3, we have combined the results of the Vihjaa technical and organizational privacy concern assessment conducted in this section. In the "has the concern been addressed?" column, each end-user concern is rated with a "no", "somewhat", or "yes" rating. "No" rating means that the concern is valid and that Vihjaa is lacking a feature, documentation, or a planned course of action. "Somewhat" rating means that Vihjaa has mostly addressed the concern, but it is still lacking in functionality or documentation. "Yes" rating means that Vihjaa has addressed the concern and that addressing it does not require any additional work. In the "suggestion / notes" column either suggestions or notes are given on what actions should be taken for that concern to be addressed, if any. In the case of concerns TOPC6, TOPC7, TOPC9 and TOPC10 suggestions are given in Section 7.1.2, as those concerns were interpreted to directly refer to the requirements identified from the GDPR and are as such discussed during that section.

TOPC Code	Has the Concern Been Addressed? (No / Somewhat / Yes)	Suggestion / Notes
TOPC1	Yes	Vihjaa's web application vulnerabilities were assessed in the security section earlier in this chapter.
TOPC2	Yes	The databases of customer organizations should be monitored for suspicious activity and Vihjaa's state of security should be constantly
TOPC3	No	Vihjaa does not currently have a set incident response or disaster recovery plan. These plans and processes should be defined in order to efficiently deal with possible incidents.
TOPC4	Somewhat	Vihjaa should implement a way to ask reporting persons for consent for data processing as detailed in GDPR1. However, data collection and processing is kept to a minimum, so a reporting person unknowingly consenting to their data being processed is unlikely.
TOPC5	Somewhat	As mentioned in UPC5, Vihjaa does not have a privacy policy document as of writing this thesis. Vihjaa's Terms of Service do however accurately represent the web application.
TOPC6	No	Discussed as a part of GDPR3 in Section 7.1.2.
TOPC7	Somewhat	Discussed as a part of GDPR4 in Section 7.1.2.
TOPC8	Somewhat	A feature should be implemented that terminates a registered user's Vihjaa admin panel session if it is left idle for a certain period of time.
TOPC9	No	Discussed as a part of GDPR3 in Section 7.1.2.
TOPC10	Yes	Discussed as a part of GDPR8 in Section 7.1.2.

Table 6.3: Vihjaa technical and organizational privacy concern assessment results

6.4 Current State of Security and Privacy of Vihjaa

In this section we will conclude and summarize the security and privacy assessments conducted in this section in order to answer RQ3. First in Section 6.2, we assessed Vihjaa's state of security by conducting a security audit on Vihjaa that focused on identifying whether or not the web application has any of issues related to the vulnerabilities listed in Section 4.3. Out of the ten vulnerabilities, Vihjaa did not have issues related to four vulnerabilities, while having issues related to six of them. The four vulnerabilities that did not have any issues related to the vulnerabilities were *A02 - cryptographic failures*, *A03 - injection*, *A05 - security misconfiguration*, and *A10 - SSRF*.

The vulnerabilities that issues were identified in relation to were *A01 - broken access control*, *A04 - insecure design*, *A06 - vulnerable and outdated components*, *A07 - identification and authentication failures*, *A08 - software and data integrity failures*, and *A09 - security logging and monitoring failures*. The issues related to those vulnerabilities and suggestions on how to fix them were summed up in Table 6.1. Overall, most of the issues found were lacking proactive measures and features that should be implemented in order to improve Vihjaa's overall state of security. Only vulnerabilities A01 and A06 were identified to have technical issues within Vihjaa that should be addressed. For A01, broken access control were detected for the "/subscription" page within the Vihjaa admin panel and for A06, outdated and/or vulnerable dependencies were detected. Both of these vulnerabilities should be addressed as soon as possible using the given suggestions in their respective sections.

After conducting the security assessment, we moved on to assess how Vihjaa has addressed the privacy concerns listed in Sections 5.3 and 5.4. The first of the two, the end-user privacy concerns, were assessed in Section 6.3.1 and the results were then combined in Table 6.2. Out of the eight end-user privacy concerns, five had been fully addressed, two had been somewhat addressed and one had not been addressed. The concern that had not been addressed was UPC5, which questioned whether or not Vihjaa's privacy policy accurately reflects the application. This could not be assessed however, since Vihjaa does not have a privacy policy document as of writing this thesis, so the suggestion of writing such a document was given. The concerns that were somewhat addressed, UPC3 and UPC8, were mostly due to lacking or unclear documentation. For UPC3, the Whistleblower Directive requires that reports can be sent to supervisory authorities, if necessary. This means that a feature enabling this functionality should be implemented into Vihjaa. However, since as of the writing of this thesis, the Whistleblower Directive has not been made into law in Finland, so the supervisory authority is unknown at this time. This

feature might be implemented into the web application in the future, which means that both the Terms of Service as well as the privacy policy document should be updated to reflect this change. The second concern that also received the "somewhat" rating was UPC8, for which the suggestion was to include in Vihjaa's privacy policy that the web application can only access a reporting person's personal data or files saved on their device when they are uploading an attachment to their report.

The latter of the two privacy related assessments, which was the technical and organizational privacy concerns list from the OWASP Foundation, was assessed in Section 6.3.2 and the results were combined in Table 6.3. Out of the ten concerns, Vihjaa fully addressed three concerns, somewhat addressed four concerns, and failed to address three concerns. Most of the issues found with addressing the privacy concerns stemmed from organizational concerns, not technical ones. The only missing technical functionality was identified with TOPC8, where it was suggested that Vihjaa should implement effective session termination in order to prevent misuse. The organizational privacy concerns that were not linked to GDPR requirements were TOPC3 and TOPC5. Those concerns consisted of not having a set incident response plan or process in place, and Vihjaa not having a privacy policy document. Privacy concerns TOPC4, TOPC6, TOPC7, TOPC9, and TOPC10 were all identified to effectively require features and documentation that will elaborated upon in Section 7.1.2 and will as such be addressed during that section. Overall to answer RQ3, we can conclude that Vihjaa's current state of security is mostly in a good standing with some minor issues, while its state of privacy still has work to be done for it to address all of the privacy concerns assessed in this thesis.

7 Analysis on the Target Application's Legislation Compliance

In this chapter we conduct a legislation adherence assessment on the target application, Vihjaa. The assessment will be based on the lists compiled in Chapter 2, beginning with the requirements set for the application by the Whistleblower Directive and moving on to the requirements set for it by the General Data Projection Regulation (GDPR). After the assessments, the results will be analyzed, and possible suggestions will be given in order to guarantee that Vihjaa meets the requirements set for it by legislation. At the end of the chapter, the results of the assessment as well as the given suggestions will be combined to answer the latter part of RQ4.

7.1 Legislation Adherence Analysis

This section is split into two subsections, first of which analyzes whether or not Vihjaa meets the requirements set for a reporting channel by the Whistleblower Directive. The latter subsection covers the requirements set by the GDPR, which can be seen to affect a web application of Vihjaa's nature as discussed in Section 2.3.2.

7.1.1 Whistleblower Directive

In Section 2.2.2 it was discussed that the requirements set for reporting channels to be used in whistleblowing that falls under the Whistleblower Directive are quite broad, as channels can vary from web applications such as Vihjaa to locked post-boxes. At the time of writing this thesis, the Finnish government is yet to adopt the Whistleblower Directive into law, which might introduce additional requirements for reporting channels such as Vihjaa. However, we will assess Vihjaa's legislation adherence based on the requirements listed in the Directive as the possible additional requirements are unknown at this time.

WD1 – channels are to be designed, established and operated in a secure manner

WD1 was the first requirement identified from the Whistleblower Directive that can be seen to affect Vihjaa. While the wording of the requirement is quite broad, we can analyze how it can be applied to web applications. Firstly, channels are designed in a secure manner can be seen to refer to the architectural and planning phase of Vihjaa. The security of Vihjaa's design as well as the use of the Security-By-Design principle were discussed and analyzed in Chapter 6 as a part of the security analysis. Secondly, channels are established in a secure manner can be seen to refer to the overall level of security and privacy of the users of Vihjaa. These factors were also analyzed in the previous chapter. Thirdly, the channels are operated in a secure manner can be seen to refer to how Vihjaa is operated in the cloud and how the application is designed to function in general. As discussed in Section 3.1, Vihjaa is hosted in Europe on the industry standard Amazon Web Services cloud service, utilizing Amazon's cloud security tools which include the AWS KMS. Vihjaa's main functionality was also discussed in Section 3.1, which displayed how reports and user data are processed and how the contents of the reports can only be seen by the

designated report handlers. Considering the information presented and analyzed in chapters 3.1 and 5.5, we can conclude that Vihjaa meets the requirements of WD1.

WD2 – channels ensure the confidentiality of the identity of the reporting person

WD2 is the second requirement identified from the Whistleblower Directive. This requirement requires the reporting channel to ensure the confidentiality of the identity of the reporting person, meaning that if the reporting person decides to submit their report anonymously, their identity is kept confidential. In Vihjaa, if the reporting person decides to submit an anonymous report, their information is anonymized before the report can be viewed by a handler as showcased in Section 3.1. The anonymization procedure itself is analyzed as a part of the end-user privacy concerns in Section 6.3.1, which was found to functioning as designed. As such, we can conclude that Vihjaa also meets the requirements of WD2.

WD3 – the channel prevents access thereto by non-authorized staff members

WD3 is the third requirement identified from the Whistleblower Directive. This requirement is fairly self-explanatory, as it simply requires that the reporting channel does not grant access to submitted reports by non-authorized members of staff. As showcased and discussed in Section 3.1, submitted reports are categorized into channels depending on the selection made by the reporting person. The report can then only be seen by handlers assigned to that channel and the Vihjaa administrators as chosen by the organization or municipality. The Vihjaa administrator role is separated into three different roles, those being: admin, maintainer, and super user. Out of the three, only admins and super users are allowed to see and read submitted reports. Non-authorized staff members do not have access to these channels, and

they cannot view submitted reports. This was also corroborated on by the analysis findings of the previous chapters, as there was no way to view submitted reports without having been given the necessary role to view them. Thus, we can conclude that Vihjaa also meets the requirements of WD3.

WD4 – the channel must provide a method of acknowledge after a report has been received as well as a way to provide feedback on the report

WD4 is the fourth requirement identified from the Whistleblower Directive. This requirement is a part of Vihjaa's main functionality as discussed in Section 3.1, where it was explained that the report's state changes automatically based on how the report handling process is progressing. The reporting person receives notifications every time the report changes state to inform them that the report is being processed. The report state screen as viewed by a handler is displayed in Figure 7.1. Vihjaa also has functionality that allows the handler and the reporting person to discuss about additional details among other things which is displayed in Figure 7.2. As such, we can conclude that these two functionalities meet the requirements of WD4.

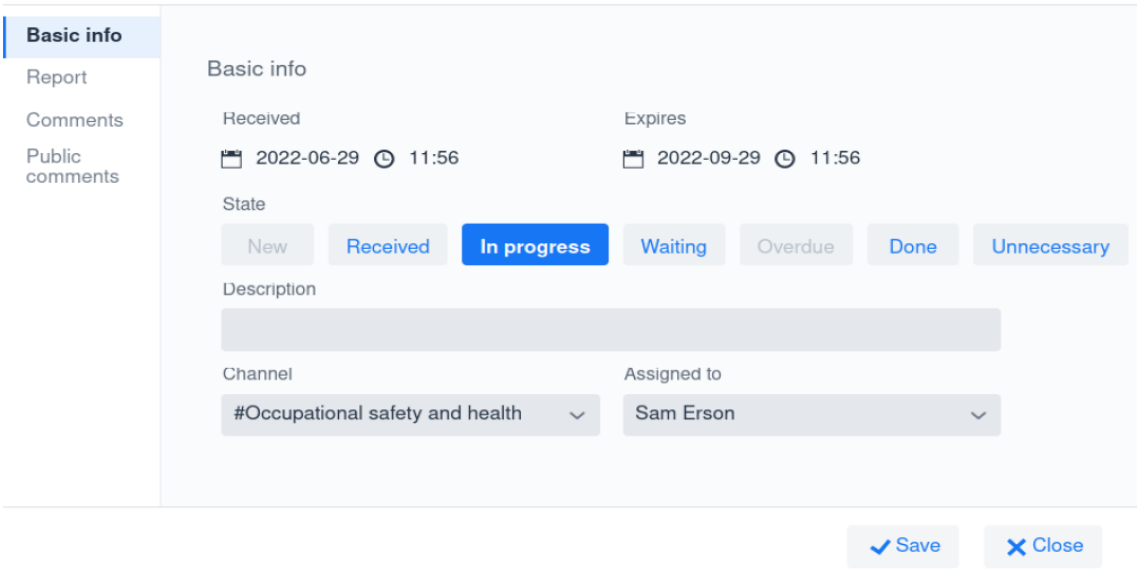

Figure 7.1: Vihjaa report state screen

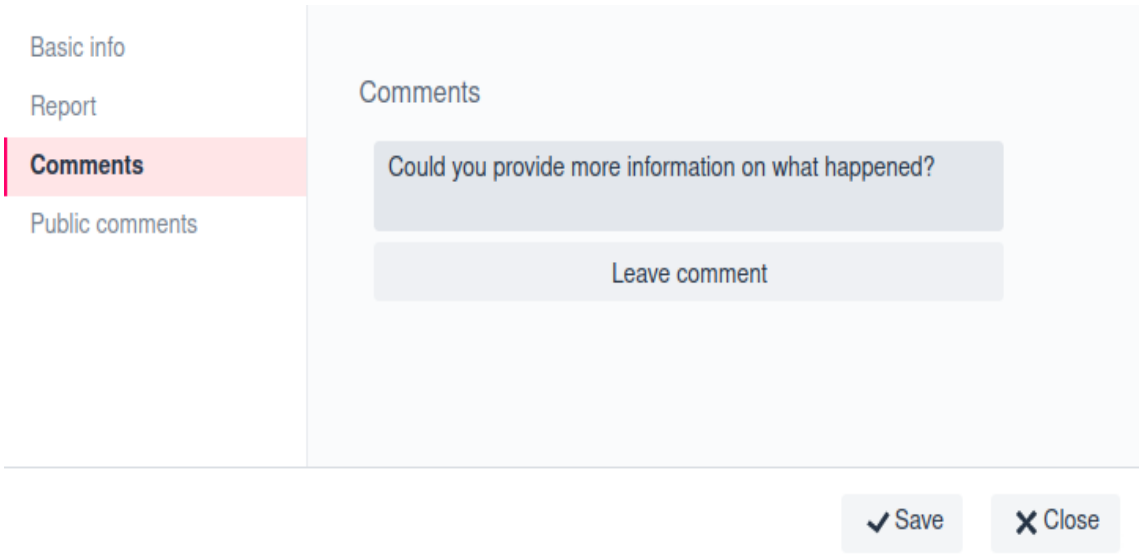

Figure 7.2: Vihjaa report comments screen

WD5 – reporting via the channel can be done either in writing or orally

WD5 is the fifth and final requirement of the Whistleblower Directive. Due to the nature of the Vihjaa application, it currently only has the functionality of submitting reports in writing. Since the directive requires providing the reporting person the ability to also report orally, the functionality to upload a sound file should also be added. As it stands, Vihjaa accepts the uploading of attachments as long as they are either pictures or Portable Document Format (PDF) files. This should be either extended to also cover sound files or a separate upload area should be provided earlier in the report submitting process. Perhaps this upload area could be displayed after the reporting person has chosen whether they want to submit their report in writing or orally. As such, we can conclude that Vihjaa does not currently meet the requirements of WD5.

In Table 7.1, we have combined the results of the Whistleblower Directive legislation compliance adherence assessment conducted in this section. The "Level of Compliance" column, which ranges from "low" to "high", shows the Vihjaa's level of compliance with that row's Whistleblower Directive requirement. Since Vihjaa

did not score “low” on any of the requirements, that will be skipped. For the two remaining levels, “medium” means that an effort has been made to comply with the requirement while there is still a missing feature or something to improve and “high” meaning that Vihjaa complies fully with the requirement.

WD Code	Level of Compliance (Low / Medium / High)	Suggestion / Notes
WD1	High	The security assessment concluded that the channels are designed, established, and operated in a secure manner. Attention should be paid that the channels keep operating in a secure manner in the future.
WD2	High	The confidentiality of the identity of the reporting person is ensured using anonymization and encryption. These were assessed in Section 6.3.1 and found to be functioning as expected.
WD3	High	This requirement is met with the basic functionality of Vihjaa as explained during the WD3 section.
WD4	High	This requirement is also met with the basic functionality of Vihjaa as explained during the WD4 section.
WD5	Medium	Writing cannot currently be done orally, a feature should be implemented that allows the uploading of audio files.

Table 7.1: Vihjaa Whistleblower Directive compliance assessment results

7.1.2 General Data Protection Regulation

In Section 2.3, we explored EU’s General Data Protection Regulation and how it affects Vihjaa. Since Vihjaa does process user information for everyone that is a registered user of the application as well as the reporting persons that choose to give include their personal information in their reports, it is affected by the GDPR. Additionally as a clarification, the Whistleblower Directive required Vihjaa to provide the possibility for reporting persons to either report anonymously or using their name, which means that only reports that the reporting person has submitted with their contact information are subject to GDPR. Out of the GDPR, we identified

14 requirements that can be seen to affect Vihjaa and as such, Vihjaa's adherence to those requirements will be analyzed in this section. The requirements themselves were expanded upon a little bit in Section 2.3.2, but they will be described in more detail in this section. As a disclaimer, the writer of this thesis is not a student of law and is strictly interpreting these requirements from a software developer's point of view. As such, if the interpretations of the GDPR articles below are in any way inaccurate or if some the suggested solutions do not guarantee compliance, those sections can be disregarded.

GDPR1 – Data subjects must consent to their data being processed, their consent must be collected into abstract data structures and special considerations must be implemented for children and their guardians

GDPR1 refers to Articles 6 - 8, which requires data controllers to ask for consent from data subjects for processing their data and that information has to be saved into a database that is updated regularly. Additionally, if data is collected from children, special considerations must be implemented for them and their guardians. However, considering that Vihjaa's customers are mostly companies and organizations, we do not need to consider that option at this time. When a reporting person accesses their organization's reporting page, Vihjaa does mention how personal data provided by the reporting person is used, but it does not outright ask for consent for data processing. As such, in order for Vihjaa to comply with GDPR1, it needs to implement a method to ask both reporting persons as well as the administrator users for consent and then save that consent in an organization specific database. For instance, consent could be asked from a reporting person before they can submit a report in the case that they have either given their email address or other personal data with the report itself. In Figure 7.3, this could be implemented so that the "SEND" button is grayed out and a dialogue option on top of it asks for the

reporting person's consent if they have input any personal data in the report. In the case of a registered admin user of Vihjaa, consent could be asked as they access the organization specific Vihjaa admin panel for the first time. With these suggestions, we can conclude that Vihjaa does not currently meet the requirements of GDPR1.

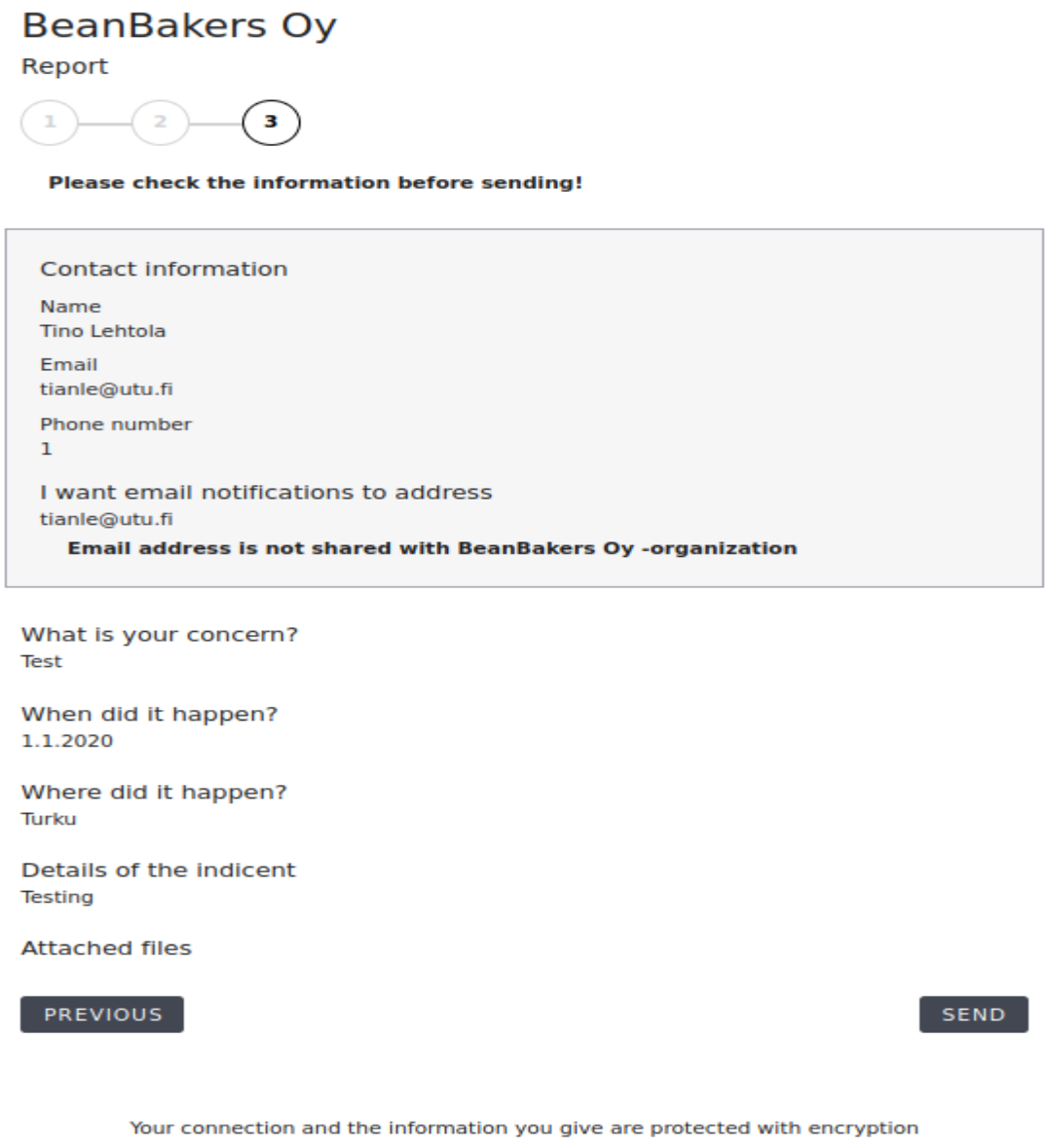

Figure 7.3: Vihjaa report submit screen

GDPR2 – Data subjects can easily request and receive all the information you hold on them

GDPR2 refers to Article 15, which is known as the 'right of access' by the data subject. This means that Vihjaa should have a feature or documentation that lets users know how and when their information is being processed and also a tool that exports all of the collected information about them. In Vihjaa's Terms of Service document, the following is said about this requirement: "The organization that has purchased the service [Vihjaa] will as the data controller take responsibility to inform the reporting persons of the processing of their personal data." This means that legally, the responsibility of informing the reporting persons about how and when their personal data is processed is with the organization that purchased the service. However, these organizations do not have direct access to the databases that store the personal information of reporting persons, meaning that BeanBakers Ltd should provide a method that enables the exporting of all personal data collected from a reporting person. The details of how such a method could be implemented are currently unknown as it requires the decrypting of personal data from an organization's database, but in order to fully meet the requirements of GDPR2, one should be put in place.

GDPR3 – Data subjects can at any time correct or update incorrect or incomplete information

GDPR3 refers to Article 16, which is known as the 'right to rectification'. This requirement requires that data subjects can rectify inaccurate personal data concerning them without undue delay. Vihjaa makes this possible by allowing registered users of Vihjaa's admin panel to update their personal information at any time. In the case of unregistered reporting persons however, rectifying incorrect personal information submitted with a report is a lot more complex. That process would

require someone with the access to that organization's database to go into the entry of a submitted report and manually change the inaccurate information. As a suggestion, if the reporting person submits inaccurate information initially with their report, perhaps they can be sent a link to view and possibly modify their report to the email address that they provided. Through this link, they could also partake in a dialogue with the person handling the report. However, simply sending a link to a page like that is not the most secure method so perhaps it can be reiterated upon to reach an efficient solution to this issue. At this time, we can conclude that Vihjaa does not completely comply with GDPR3.

GDPR4 – Data subjects can easily request to have all their personal data deleted

GDPR4 refers to Article 17, which is known as the 'right to be forgotten'. This requires the data controller to erase any personal data concerning a data subject shall they choose so. As mentioned in GDPR2, Vihjaa's Terms of Service name the organization purchasing the service as the data controller. This means that it is up to the organization to comply with a data erasure request from a data subject. However, as discussed in GDPR3, the organization does not have direct access to the database. This means that in order to comply with this requirement, Vihjaa should either have some sort of a channel that reporting persons can submit data erasure requests to, or a separate page in the web application for this specific purpose. As such, we can conclude that Vihjaa does not comply with GDPR4.

GDPR5 – Data subjects can easily ask for you to restrict the processing their data

GDPR5 refers to Article 18, which is known as the 'right to restriction of processing'. It requires data controllers to be able to comply with requests from data subjects to

restrict the processing of their personal data. As discussed in GDPR2 and GDPR4, the data controller, meaning the purchaser of the Vihjaa service, can not comply with such request as they do not have direct access to the database. This means that the suggestions given in GDPR4 could be expanded upon by giving data subjects the option to also ask for the processing of their personal data to be restricted. Thus, we can conclude that Vihjaa does not comply with GDPR5.

GDPR6 – Data subjects can request and receive their personal data in a format that can be transferred to another company

GDPR6 refers to Article 20, which is known as the 'right to data portability'. It requires the data controller to be able to export all personal data collected from a user in a format that is machine-readable. With this requirement, the same problem of the data controller does not have direct access to the database as also mentioned in GDPR2, GDPR4, and GDPR5 remains. The necessity of implementing a feature that complies with this requirement can be debated, as Vihjaa does not collect much personal data from the user, only their email, first and last names, phone number and a possible other contact email. However, if such a feature is necessary to implement in order to comply with GDPR6, the data subjects could be given an option to either to submit a data request either by email or by a separate page in the web application. The MongoDB database is capable of exporting information in either a JSON or CSV format, which are both machine-readable. In conclusion, Vihjaa does not currently comply with GDPR6.

GDPR7 – Data subjects can easily object to their personal data being processed

GDPR7 refers to Article 21, which is known as the 'right to object'. The requirement is in its essence the same as GDPR5, except that it gives the data subject the

option to ask for the data controller to stop processing their personal data entirely. Nevertheless, the same issue as was identified with GDPR5 still remains, which restricts the data controller from complying with these kinds of requests from data subjects. As a suggestion, the feature that was explored in GDPR4 and further elaborated upon in GDPR5 could also be used for this purpose. Thus, Vihjaa does not currently comply with GDPR7.

GDPR8 – Implement appropriate technical and organizational measures ensuring that only personal data which is needed to be processed is being processed, also referred to as data minimization

GDPR8 refers to Articles 24 and 25, which require the data controller to be responsible with data collecting and processing as well as only processing necessary personal data. Vihjaa only collects a limited amount of personal data as explained in GDPR6 and only if the reporting person so chooses. That personal information is then processed and encrypted in the backend and sent to the database, where it is stored encrypted and not processed unless the personal information is needed to proceed with handling the report. As such, we can conclude that Vihjaa does utilize data minimization and does comply with GDPR8.

GDPR9 – Take data protection into account at all times, from development to production, referred to as "data protection by design and by default

GDPR9 refers to Article 25, which is called ‘Data protection by design and by default’. It requires data protection to be taken into account at every point of the web application’s lifecycle, aspects which are by proxy also being evaluated during Chapters 5.5 and 7. In Article 25 itself, it requires the data controller to implement the appropriate technical and organizational measures for ensuring that only per-

sonal data that is needed for each specific purpose is being processed by default. Due to being required to comply with the requirements from the Whistleblower Directive, Vihjaa is complying with this requirement not only by choice, but also as a byproduct of adhering to other requirements. That is why we can conclude that Vihjaa also complies with GDPR9.

GDPR10 – Each controller shall maintain a record of processing activities under its responsibility

GDPR10 refers to Article 30, which requires data controllers to maintain a record of processing activities under its responsibility. This means that every organization that has purchased the Vihjaa service, should keep a record containing all the information required by Article 30. For example, the name and contact details of the data controller, the purposes of data processing and a description of the categories of data subjects, as well as the categories of collected personal data. Due to Vihjaa's Terms of Service dedicating the role of the data controller to the customer, this document should be maintained by them. In order to assist in the process, a template document could be drafted by BeanBakers Ltd for their customers for them to be able to comply with this requirement. Due to the nature of this requirement, we will note that Vihjaa does comply with GDPR10 as it currently stands.

GDPR11 – The data controller and data processor shall cooperate, on request, with the supervisory authority in the performance of its tasks

GDPR11 refers to Article 31, which requires data controllers and data processors to cooperate with supervisory authorities if necessary. As at the time of writing this thesis, the Whistleblower Directive has not been made into law in Finland yet, so we are currently uncertain of who that supervisory authority is. For example, if it happens to be the regional state administrative agency, BeanBakers Ltd should

draft a document which outlines the steps of how them and a customer organization can collaborate in order to cooperate with the agency. We can conclude, that while BeanBakers Ltd is ready to cooperate with any supervisory authority as it stands, they can take proactive steps to help support the authorities when needed. As such, Vihjaa does comply with GDPR11.

GDPR12 – Utilization of pseudonymisation or anonymization and encryption wherever and whenever possible

GDPR12 refers Article 32, which is called 'Security of processing'. It requires personal data to be processed using pseudonymization, anonymization, and encryption whenever and wherever possible. As required by WD2, Vihjaa already utilizes anonymization and encryption when processing and storing personal data. The cryptographic methods used for those were assessed as a part of the security analysis in Section 6.2 and were found to abide by the industry standards and best practices. Using this knowledge, we can conclude that Vihjaa does comply with GDPR12.

GDPR13 – Have a process in place that notifies authorities and data subjects in the event of a personal data breach

GDPR13 refers to Articles 33 and 34, which require the data controller and processor to have a process in place that notifies the correct authorities and the affected data subjects in the event of a personal data breach. Article 33 also specifies that the data processor shall notify the controller without undue delay after becoming aware of a personal data breach and notify the authorities within 72 hours. Regarding data subjects, in Article 34 it is set that if the data controller has implemented appropriate technical and organization protection measures, the personal data breach does not need to be communicated to the data subject without undue delay. In Vihjaa's case, there does not currently exist an incident response plan or process for if a data

breach should occur. As such, it is suggested that such a plan or process is put in place as soon as possible. In conclusion, Vihjaa does not currently completely comply with GDPR13.

GDPR14 – Conduct a data protection impact assessment (DPIA), if the data being processed is "likely to result in a high risk to [a person's] rights and freedoms"

GDPR14 refers to Article 35, which requires data controllers to conduct a data protection impact assessment (DPIA) if the processed personal data is "likely to result in a high risk to [a person's] rights and freedoms". Vihjaa only collects and processes a reporting person's personal data if given permission to as explained in GDPR6, where the processed personal data items were also listed. The thesis writer's interpretation on the matter is that those personal data items are not likely to result in a high risk to a person's rights and freedoms as they can be categorized as basic contact information. As such, we will conclude that Vihjaa does not need to conduct a DPIA and does comply with GDPR14.

In Table 7.2, we have combined the results of the GDPR legislation compliance adherence assessment conducted in this section. In the "Level of Compliance" column, we have assessed on what level Vihjaa complies with that row's GDPR requirement and then given suggestions on how it could be improved, if applicable. The "low" level means that a feature required for complying with this requirement is either fully missing or that the requirement has not been taken into consideration when designing the web application. The "medium" level means that some steps were taken in order to comply with the requirement, but there are still some features missing or something could be improved. The "high" level means that Vihjaa complies with that requirement and that no actions are needed at this time.

GDPR Code	Level of Compliance (Low / Medium / High)	Suggestion / Notes
GDPR1	Low	Consent for data processing could be gathered from reporting persons before they can press "SEND" on the reporting form.
GDPR2	Low	A method should be implemented that enables the exporting of all personal data collected from a reporting person.
GDPR3	Medium	Reporting persons should be provided a method with which they can correct or update incorrect or incomplete information. For registered users this is possible through the admin panel.
GDPR4	Low	Either a channel or a separate page in the Vihjaa application should be provided for reporting persons to submit data erasure requests.
GDPR5	Medium	Either a channel or a separate page in the Vihjaa application should be provided for reporting persons to request restrictions on the processing of their personal data. However, considering the nature of the Vihjaa application, personal data is only used to save the contact information of the reporting persons, so their data is only processed when their report is being handled.
GDPR6	Low	A method for reporting persons to request their personal data from that organization's database should be provided.
GDPR7	Medium	A channel or a separate page in the Vihjaa application should be provided for reporting persons to request that their personal data is no longer processed. However, the same point as made with GDPR5 remains that their personal data is only processed when their report is handled.
GDPR8	High	The security, privacy and Whistleblower Directive adherence assessments found that Vihjaa complies with this requirement.
GDPR9	High	As with GDPR8, the findings of the assessments showed that Vihjaa does comply with this requirement.
GDPR10	Medium	BeanBakers Ltd could provide the customers of the Vihjaa service templates to help them as the data controller to comply with this requirement.
GDPR11	High	BeanBakers Ltd is prepared to help supervisory authorities with their tasks, although processes for this could be planned beforehand.
GDPR12	High	Vihjaa utilizes anonymization and encryption as required by WD1 and WD2.
GDPR13	Medium	While BeanBakers Ltd and the data controllers are prepared to comply with this requirement, the process should be planned and defined beforehand in the case of an incident occurring.
GDPR14	Not Applicable	We interpreted that Vihjaa does not need to comply with this requirement.

Table 7.2: Vihjaa GDPR compliance assessment results

7.2 Current State of Vihjaa's Legislation Adherence

In this section we will sum up the results of Vihjaa's legislation adherence assessment and answer the latter part of RQ4. First, we assessed Vihjaa's compliance with the requirements interpreted from the Whistleblower Directive that would affect a web application like Vihjaa that has been developed to fit the role of an internal reporting channel as was discussed in Section 2.2.1. Our assessment found that Vihjaa is fully compliant with requirements WD1 – WD4, but was lacking a feature required by WD5 as can be seen from Table 7.1. The missing feature is a method that provides reporting persons the possibility of reporting orally. The suggestion for this issue was that Vihjaa should implement a way for a reporting person to upload an audio file so that they can attach a recording of themselves speaking to make Vihjaa compliant with WD5.

While Vihjaa was almost fully compliant with the Whistleblower Directive, the same was not the case with the interpreted requirements of the General Data Protection Regulation. As can be seen from Table 7.2, Vihjaa scored a "low" rating on requirements GDPR1, GDPR2, GDPR4, and GDPR6 as well as a "medium" rating on requirements GDPR3, GDPR5, GDPR7, GDPR10 and GDPR13. Out of the fourteen identified requirements, we found that GDPR14 was not applicable to Vihjaa. Vihjaa's non-compliance with some of the GDPR requirements mostly came down to how Vihjaa collects and stores personal data. This is the case, because Vihjaa collects, encrypts, and stores personal data in a way that neither the data controller nor the data processor can easily provide a reporting person a way to request for their data to be corrected, modified, or deleted. Additionally, consent is not collected from a reporting person that chooses to give out their contact information, which leads to a situation where the reporting person does not know how their data is processed nor can they withdraw their consent. The suggestions for these issues were quite vague, since it will take time to plan how Vihjaa can comply with

these requirements, considering that a new interface has to be implemented for the database for reporting persons to interact with their personal data. Thus to answer RQ4, we can conclude that Vihjaa almost fully complies with the requirements of the Whistleblower Directive, while there were a number of issues identified regarding the General Data Protection Regulation. The suggestions to improve Vihjaa's level of legislation adherence can be found in Tables 7.1 and 7.2.

8 Conclusion

After reviewing literature based on applicable EU legislation, currently prominent web application threats and vulnerabilities, and privacy, we created criteria that the target web application of this thesis had to meet. In the assessment chapters, we then assessed the target application based on the created criteria. The target application of this thesis, Vihjaa, is a whistleblowing web application designed to be used as an internal reporting channel that complies with the requirements set for it by the Whistleblower Directive. The main objective of this thesis was to assess the target application in order to gain insight into the application's current state of security, privacy, and legislation adherence. Additionally, we formulated four research questions related to the subject matter that were answered during this thesis. In the next section, we will summarize our answers to the research questions and review the main contributions of this thesis. Afterwards, we will discuss how well the thesis succeeded in meeting its goal, the thesis' limitations, and topics for future research.

8.1 Contributions

RQ1: What are the most prominent threats, vulnerabilities & privacy concerns towards web applications at the moment?

In Section 4.4, we reviewed recent threat intelligence reports as well as information provided by the OWASP Foundation on web application security in order to

answer the former part of RQ1 regarding threats and vulnerabilities. We split the prominent threats and vulnerabilities into two categories: those that target a web application's availability and integrity, and those that target their data. We found that the most prominent threats and vulnerabilities that target a web application's availability and integrity are DDoS attacks, general software integrity vulnerabilities, security misconfigurations, and insecure design. The most prominent threats and vulnerabilities that target the data of web applications are injection attacks, predictable resource location attacks, broken user authentication, broken access control, unencrypted data, and weak cryptographic algorithms. The most prominent privacy concerns towards web applications at the moment were listed in Sections 5.3 and 5.4. We split the privacy concerns into two categories as well, those from the viewpoint of end-users and those from the viewpoint of organizations. The end-user privacy concerns were formulated based on research papers that had surveyed the end-users of web applications for their biggest privacy concerns. For the organizational privacy concerns, we utilized the OWASP Foundation's Top Ten Privacy risks list. In summary, most of the privacy concerns were related to the processing and storing of data, how it is used, and who it is shared with.

RQ2: What user-related information is collected and handled by web applications that can be used to identify a user?

In Section 5.2, we discussed what personal information is collected from the end-users of web applications. The discussion was based on two research papers, Heino et al and Carlsson et al, which investigated what types of personal data is collected by 34 web services and 32 mobile applications, all maintained and provided by Finnish public sector bodies respectively [51], [52]. In Table 5.1, we listed the different data items collected by web applications that can be used to identify a user. We then ranked those data items in Table 5.2, based on how critical a data item is for the identifying process.

RQ3: What is the current state of security and user privacy of the 'Vihjaa' application and how can it be improved?

Vihjaa's current state of security was assessed using the OWASP Foundation's Top Ten Web Application vulnerabilities list from 2021. In Section 4.4, we concluded that the OWASP list can be used to assess the current state of security of the Vihjaa application as it corroborates on the findings of other respected entities in the industry. Overall, we found out that Vihjaa had security weaknesses related to three of the ten vulnerabilities listed by OWASP while general suggestions for improvement were given for another three vulnerabilities. The summary of Vihjaa's security assessment, its findings, and provided suggestions can be seen in Table 6.1.

Vihjaa's current state of privacy was assessed using a combination of the end-user concern list devised in Section 5.3 as well as the OWASP Top 10 Privacy Risks list discussed in Section 5.4. The results of the assessments were summarized in Tables 6.2 and 6.3. The end-user privacy concern assessment found that out of the eight concerns, one had not been addressed, and two concerns were somewhat addressed. For the organizational and privacy concerns list, four out of the ten concerns were linked to GDPR requirements and thus assessed in Chapter 7. From the remaining concerns, one had not been addressed, while three had somewhat been addressed. Overall, we found that actions should be taken in order to address the privacy concerns that suggestions were provided for in Tables 6.2 and 6.3.

RQ4: How can we assess and verify that the application meets the requirements set for it by EU legislation?

In Sections 2.2.2 and 2.3.2, we identified and listed the requirements set for the target application by EU legislation. We concluded that the application is mostly affected by the Whistleblower Directive as well as the General Data Protection Regulation, and as such, those were used for the legislation assessment. The results of the assessments can be seen in Tables 7.1 and 7.2. Vihjaa's compliance with the

Whistleblower Directive was found to be high, only missing a feature that allows the reporting person to report orally. Assessing Vihjaa's compliance with the GDPR was found to be a more difficult task, as some of the requirements and how they affect the application were dependent on interpretation. As was noted in the beginning of Section 7.1.2, the writer of this thesis is not a student of law, which might have led to misinterpretations of the GDPR. Should any of the requirements be found not to apply to Vihjaa or that they have been misinterpreted, they can be disregarded. In summary, the results of our assessment showed that there many aspects in which Vihjaa should improve in order to reach full compliance with the GDPR. Most of the provided suggestions revolved around features that should be implemented to give reporting persons more access to their personal data, which is saved in Vihjaa's database.

Developed framework to assess whistleblowing web applications

Over the course of this thesis, we have developed a framework that others can follow to assess whistleblowing web applications and their states of security, privacy, and legislation adherence. While interpreting EU legislation and identifying the requirements from them was mostly a straightforward process, the reader can still find value in how those requirements were assessed. The main contributions of this thesis in terms of the framework, are however the security and privacy assessment procedures and how those assessments are conducted in practice. For the security assessment, we argued that the OWASP Top Ten Web Application Vulnerabilities list is enough to provide us with a general overview of Vihjaa's state of security, and then we utilized information from various sources in order to assess those vulnerabilities in practice using the setup explained in Section 6.1. Furthermore, we listed tools that can be used to conduct a security assessment to test web applications for vulnerabilities and provided examples for how those tools should be used.

For the privacy assessment, we first combined the results of multiple research papers to formulate the end-user privacy concerns list. Additionally, we decided to assess Vihjaa using the Top Ten Privacy Risks list provided by the OWASP Foundation in order to understand privacy related issues from the organizational point of view as well. These two lists were then used as the basis for Vihjaa's privacy assessment, which focused on how to assess the application for privacy concerns and how well the listed privacy concerns have been addressed in the application.

Using this developed framework, we managed to gain a broad understanding of Vihjaa's state of security, privacy, and legislation adherence. For each of the assessments, we found issues of varying severities that we provided suggestions for on how they could be addressed. In conclusion, most of the issues that were identified during the assessments were of low severity, meaning that the overall state of Vihjaa and its different aspects that were assessed during this thesis are in a good standing.

8.2 Discussion

In this section we will briefly discuss the target audience of this thesis, the thesis' usability, general suggestions, limitations, and directions for future research. Since this thesis can be seen to be comprised of a security, privacy, and legislation section, the thesis' target audience can be quite broad. However, considering the thesis' main objective and its results, the thesis will be the most interesting to software developers, software architects, security researchers, and law practitioners with interest in web application privacy and privacy legislation. The usability of this thesis can be argued to be quite broad as well, since the security and privacy assessments can be applied to any web application. The same can be said for the GDPR assessment as well, as it is applicable to any web application that collects, processes, and/or stores personal data gathered from end-users. This broad usability is also partly due to the framework that was developed during the thesis. There are also some suggestions

to the reader that is looking to either follow or take influence from the framework developed in this thesis. First of all, there is a steep learning curve related to the security assessment conducted in this thesis, if the reader has no prior knowledge of it. The tools and the environment take their own time to learn, but utilizing all possible resources in terms of instructional guides, videos, or books, will help tremendously. Furthermore, since every web application is different, it is important to take the time to get familiar with the web application and its source code before starting with the assessments, as this will make the process a lot more easier.

In retrospect, if the writer of this thesis could change something, it would be the scope of the thesis. Depending on the web application, it could be argued that a thesis could be written that focuses only on conducting one of the three assessments that were conducted in this thesis. However, one of the main motivations of this thesis was that the shareholders wanted an insightful overview of the state of the application from those three different aspects. Related to the assessments themselves, it might have been worthwhile to consult an expert in the GDPR before conducting the assessment to discuss which requirements actually affect the target application. Nonetheless, we can conclude that the thesis did succeed in its primary goals of presenting a detailed overview of Vihjaa's state of security, privacy, and legislation adherence in addition to creating a framework that can be used to assess other web applications of this nature.

Limitations of this thesis

The two limitations of this thesis concern the conducted security assessment and the lacking research into how web applications collect data from users. First of all, because we had to restrict the security assessment to only analyze Vihjaa for the vulnerabilities listed on the OWASP Top Ten Web Application Vulnerabilities 2021 list, it does not provide a comprehensive overview of Vihjaa's state of security.

However, since some of the vulnerabilities listed by OWASP on the aforementioned list can be interpreted as umbrella terms that consist of multiple vulnerabilities each, we can conclude that the security assessment conducted in this thesis provides the Vihjaa shareholders with an accurate representation of its current state of security. Additionally, the information provided in this thesis can be used by third parties that further assess Vihjaa for other vulnerabilities. The second limitation of the thesis was the missing research into web applications and how they collect data from their users that was identified in Section 5.2. It could be argued, that the research papers used in that section do not provide an all-encompassing overview of the data items collected by web applications from end-users. However, that section does cover data items that are most commonly collected by web applications. This lack of research is nonetheless mentioned in the following section, as research into this subject would be useful information for many future works.

Suggestions for future research

Continuing from the last section, further security testing for Vihjaa is recommended as there are multiple attack techniques and methods that were not used in our assessment. Furthermore, the writer of this thesis did not have any prior knowledge or experience in conducting security audits for web applications before this thesis, so a more experienced security professional might come up with different results. The other subject for future research that was mentioned in the previous section was the lack of research related to the personal data that is collected and processed by web applications. This would be a worthwhile direction for future research, since it would provide end-users with more transparency on how web applications gather data from their users, and how that data is processed. In more general terms, once Finland does enact the Whistleblower Directive into law, it would be interesting to see whether or not it has additional requirements for whistleblowing web applications that should

be considered in future legislation adherence assessments. Finally, in Section 2.1 the EU Cybersecurity Act was mentioned, which mostly concerns developing a voluntary IT cybersecurity certification framework. Once the development of that framework is finished, the impact it has on the EU's internal market as well as the level of cybersecurity of products sold on it would be another fascinating direction for future research.

Appendix A OWASP Top Ten Web Application Vulnerabilities List Changes

In Table A.1, the evolution of the OWASP Top Ten Web Application Vulnerabilities list is shown between each release. The first top ten list was released in 2003 and in 2017, two "Release Candidates" were released, which are referred to as RC1 and RC2. The references in the table correspond to the list of changes below: [33]

1 Renamed “Broken Access Control” from T10 2003
2 Split “Broken Access Control” from T10 2003
3 Renamed “Command Injection Flaws” from T10 2003
4 Renamed “Error Handling Problems” from T10 2003
5 Renamed “Insecure Use of Cryptography” from T10 2003
6 Renamed “Web and Application Server” from T10 2003
7 Split “Insecure Configuration Management” from T10 2004
8 Reconsidered during T10 2010 Release Candidate (RC)
9 Renamed “Unvalidated Parameters” from T10 2003
10 Renamed “Injection Flaws” from T10 2007
11 Split “Broken Access Control” from T10 2004
12 Renamed “Insecure Configuration Management” from T10 2004
13 Split “Broken Access Control” from T10 2004

14 Renamed “Improper Error Handling” from T10 2004

15 Renamed “Insecure Storage” from T10 2004

16 Renamed “Failure to Restrict URL Access” from T10 2010

17 Renamed “Insecure Cryptographic Storage” from T10 2010

18 Split “Insecure Cryptographic Storage” from T10 2010

19 Split “Security Misconfiguration” from T10 2010

20 Split “Broken Access Control” from T10 2013

21 Merged into “Security Misconfiguration” from T10 2021

22 Merged into “Injection” from T10 2021

23 Merged into new category “Software and Data Integrity Failures” from T10 2021

24 Renamed to “Identification and Authentication Failures” from T10 2021

25 Renamed to “Security Logging and Monitoring Failures” from T10 2021

OWASP Top Ten	2003	2004	2007	2010	2013	2017 RC1	2017 RC2	2021
Unvalidated Input	A1	A1 [9]	x	x	x	x	x	x
Buffer Overflow	A5	A5	x	x	x	x	x	x
Denial of Service	x	A9 [2]	x	x	x	x	x	x
Injection	A6	A6 [2]	A2	A1 [10]	A1	A1	A1	A3
Cross Site Scripting (XSS)	A4	A4	A1	A2	A3	A3	A7	A3 [22]
Broken Authentication and Session Management	A3	A3	A7	A3	A2	A2	A2	A7 [24]
Insecure Direct Object Reference	x	A2	A4 [11]	A4	A4	A4 [20]	A5 [20]	x
Cross Site Request Forgery (CSRF)	x	x	A5	A5	A8	A8	x	x
Security Misconfuguration	A10	A10 [3], [5]	x	A6	A5	A5	A6	A5
Broken Access Control	A2	A2 [1]	A10 [13]	A8	A7 [16]	A4 [20]	A5	A1
Insufficient Attack Protection	x	x	x	x	x	A7	x	x
Unvalidated Redirects and Forwards	x	x	x	A10	A10	x	x	x
Information Leakage and Improper Error Handling	A7	A7 [14],[4]	A6	A6 [8]	x	x	x	x
Malicious File Execution	x	x	A3	A6 [8]	x	x	x	x
Sensitive Data Exposure	A8	A8 [6], [5]	A8	A7	A6 [17]	A6	A3	x
Insecure Communications	x	A10	A9 [7]	A9	x	x	x	x
Remote Administration Flaws	A9	x	x	x	x	x	x	x
Using Known Vulnerable Components	x	x	x	x	A9 [18], [19]	A9	A9	A9
Unprotected APIs	x	x	x	x	x	A10	x	x
Insecure Deserialization	x	x	x	x	x	x	A8	A8 [23]
XML External Entity (XXE)	x	x	x	x	x	x	A4	A5 [21]
Insufficient Logging Monitoring	x	x	x	x	x	x	A9	A9 [25]
Cryptographic Failures	x	x	x	x	x	x	x	A2
Insecure Design	x	x	x	x	x	x	x	A4
Identification and Authentication Failures	x	x	x	x	x	x	x	A7
Software and Data Integrity Failures	x	x	x	x	x	x	x	A8
Security Logging and Monitoring Failures	x	x	x	x	x	x	x	A9
Server Side Request Forgery (SSRF)	x	x	x	x	x	x	x	A10

Table A.1: OWASP Top Ten Web Application Vulnerabilities list changes

Appendix B XMLHttpRequest JSON response

Listing B.1: XMLHttpRequest response from Vihjaa reporting application

```
{
   "websocket":true,
   "origins":[
      "*:*"
   ],
   "cookie_needed":false,
   "entropy":1836239907
}
```

References

[1] European Council, "Directive (eu) 2019/1937 of the european parliament and of the council of 23 october 2019 on the protection of persons who report breaches of union law", 2019. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/LSU/?uri=CELEX:02019L1937-20211110.

[2] European Council, "General data protection regulation (eu) 2016/679", 2016. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679.

[3] European Council, "Types of legislation", 2022. [Online]. Available: https://european-union.europa.eu/institutions-law-budget/law/types-legislation_en.

[4] The Finnish Ministry of Justice, "The government's proposal on the protection of whistleblowers is delayed", 2022, In Finnish. [Online]. Available: https://oikeusministerio.fi/-/hallituksen-esitys-ilmoittajansuojasta-myohastyy.

[5] European Council, "The eu cybersecurity act", 2019. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/LSU/?uri=CELEX:32019R0881.

[6] Edward Snowden, *Permanent Record*. Pan Macmillan, 2019.

[7] E. Politou, E. Alepis, and C. Patsakis, "Forgetting personal data and revoking consent under the gdpr: Challenges and proposed solutions", *Journal of cybersecurity*, vol. 4, no. 1, tyy001, 2018.

[8] B. Wolford, "What is gdpr, the eu's new data protection law?", 2020. [Online]. Available: `https://gdpr.eu/what-is-gdpr/`.

[9] P. Voigt and A. Von dem Bussche, "The eu general data protection regulation (gdpr)", *A Practical Guide, 1st Ed., Cham: Springer International Publishing*, vol. 10, no. 3152676, pp. 10–5555, 2017.

[10] M. Jensen, S. Kapila, and N. Gruschka, "Towards aligning gdpr compliance with software development: A research agenda.", in *ICISSP*, 2019, pp. 389–396.

[11] K. Hjerppe, J. Ruohonen, and V. Leppänen, "The general data protection regulation: Requirements, architectures, and constraints", in *2019 IEEE 27th International Requirements Engineering Conference (RE)*, IEEE, 2019, pp. 265–275.

[12] R. Johnson, J. Hoeller, K. Donald, *et al.*, "The spring framework documentation", version 5.3.23, 2022. [Online]. Available: `https://docs.spring.io/spring-framework/docs/5.3.23/reference/pdf/index.pdf`.

[13] S. Turner, "Transport layer security", *IEEE Internet Computing*, vol. 18, no. 6, pp. 60–63, 2014.

[14] D. Naylor, A. Finamore, I. Leontiadis, *et al.*, "The cost of the" s" in https", in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, 2014, pp. 133–140.

[15] E. Rescorla *et al.*, "RFC 2818: Http Over TLS", 2000.

[16] M. Masse, *REST API design rulebook: designing consistent RESTful web service interfaces*. "O'Reilly Media, Inc.", 2011.

[17] R. Richards, "Representational state transfer (rest)", in *Pro PHP XML and web services*, Springer, 2006, pp. 633–672.

[18] N. Madden, *API security in action*. Manning Publications, 2020.

[19] J. Varia, S. Mathew, *et al.*, "Overview of amazon web services", *Amazon Web Services*, vol. 105, 2014.

[20] Amazon Web Services, *Aws key management service*, 2022. [Online]. Available: `https://docs.aws.amazon.com/kms/latest/developerguide/overview.html`.

[21] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain, "A survey and comparison of relational and non-relational database", *International Journal of Engineering Research & Technology*, vol. 1, no. 6, pp. 1–5, 2012.

[22] S. Bradshaw, E. Brazil, and K. Chodorow, *MongoDB: the definitive guide: powerful and scalable data storage*. O'Reilly Media, 2019.

[23] I. Hadi, "Intelligent authentication for identity and access management: A review paper", *Iraqi Journal for Computers and Informatics*, vol. 45, no. 1, pp. 6–10, 2019.

[24] Trivore Corporation, "Trivore id", 2022. [Online]. Available: `https://trivore.com/trivore-id`.

[25] N. Naik and P. Jenkins, "Securing digital identities in the cloud by selecting an apposite federated identity management from saml, oauth and openid connect", in *2017 11th International Conference on Research Challenges in Information Science (RCIS)*, IEEE, 2017, pp. 163–174.

[26] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, "Openid connect core 1.0 incorporating errata set 1", 2014.

[27] Vaadin Team, *Book of Vaadin, Vaadin 14 Edition*. Vaadin, 2019. [Online]. Available: `https://vaadin.com/book`.

[28] P. Hauer, "Evaluating vaadin: Strengths and weaknesses", 2015. [Online]. Available: https://phauer.com/2015/evaluating-vaadin-strengths-weaknesses/.

[29] European Union Agency for Cybersecurity, *Enisa threat landscape 2021*, 2021. [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021.

[30] M. Humayun, M. Niazi, N. Jhanjhi, M. Alshayeb, and S. Mahmood, "Cyber security threats and vulnerabilities: A systematic mapping study", *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 3171–3189, 2020.

[31] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*. Wiley, 2011.

[32] A. Mohammed, J. Alkhathami, H. Alsuwat, and E. Alsuwat, "Security of web applications: Threats, vulnerabilities, and protection methods", *International Journal of Computer Science & Network Security*, vol. 21, no. 8, pp. 167–176, 2021.

[33] OWASP Foundation, *Owasp top ten web application security risks*, 2021. [Online]. Available: https://owasp.org/Top10/.

[34] OWASP Foundation, *Owasp top ten web application security risks - introduction*, 2021. [Online]. Available: https://owasp.org/Top10/A00_2021_Introduction/.

[35] M. Bach-Nutman, "Understanding the top 10 owasp vulnerabilities", *arXiv preprint arXiv:2012.09960*, 2020.

[36] R. Hiesgen, M. Nawrocki, T. C. Schmidt, and M. Wählisch, "The race to the vulnerable: Measuring the log4j shell incident", *arXiv preprint arXiv:2205.02544*, 2022.

[37] Radware, *Global threat analysis report*, 2021. [Online]. Available: `https://www.radware.com/getattachment/3d26f50b-f2a7-4ffa-9a84-1b5a598a0b27/2021-2022-Global-Threat-Analysis-Report_2022-FINAL-V2.pdf.aspx`.

[38] Microsoft, *Microsoft digital defense report*, 2021. [Online]. Available: `https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RWMFIi`.

[39] V. M. Deshpande, D. M. K. Nair, and D. Shah, "Major web application threats for data privacy & security – detection, analysis and mitigation strategies", *International Journal of Scientific Research in Science and Technology*, vol. 3, no. 7, pp. 182–198, 2017.

[40] A. Hoffman, *Web Application Security: Exploitation and Countermeasures for Modern Web Applications*. O'Reilly Media, 2020.

[41] K. Nirmal, B. Janet, and R. Kumar, "Web application vulnerabilities – the hacker's treasure", in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, IEEE, 2018, pp. 58–62.

[42] V. Clincy and H. Shahriar, "Web application firewall: Network security models and configuration", in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, vol. 1, 2018, pp. 835–836.

[43] European Parliament, "Directive 2002/58/ec of the european parliament and of the council of 12 july 2002 concerning the processing of personal data and the protection of privacy in the electronic communications sector", *JL 201, 31.7. 2002, at 37.(Directive on Privacy and Electronic Communications)*, 2002.

[44] R. O. Mason, "Four ethical issues of the information age", in *MIS Quarterly, Vol. 10*, Management Information Systems Quarterly, 1986, pp. 5–12.

[45] J. Holvast, "History of privacy", in *The history of information security*, Elsevier, 2007, pp. 737–769.

[46] A. Neves, "Protection of personal data regulation and public liberties: A polyhedron with unexpected effects", in *Personal Data Protection and Legal Developments in the European Union*, IGI Global, 2020, pp. 1–18.

[47] European Parliament, "What is personal data?", 2002. [Online]. Available: `https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en`.

[48] P. Grassi, M. Garcia, and J. Fenton, "Digital identity guidelines", National Institute of Standards and Technology, Tech. Rep., 2020.

[49] R. Creemers and G. Webster, "Translation: Personal information protection law of the people's republic of china—effective nov. 1, 2021", *DigiChina Project, August*, vol. 20, 2021.

[50] M. Burgess, "The quiet way advertisers are tracking your browsing", *WIRED*, 2022. [Online]. Available: `https://www.wired.com/story/browser-fingerprinting-tracking-explained/`.

[51] T. Heino, R. Carlsson, S. Rauti, and V. Leppänen, "Assessing discrepancies between network traffic and privacy policies of public sector web services", in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–6.

[52] R. Carlsson, T. Heino, L. Koivunen, S. Rauti, and V. Leppänen, "Where does your data go? comparing network traffic and privacy policies of public sector mobile applications", in *World Conference on Information Systems and Technologies*, Springer, 2022, pp. 214–225.

[53] Y. Liu, H. H. Song, I. Bermudez, A. Mislove, M. Baldi, and A. Tongaonkar, "Identifying personal information in internet traffic", in *Proceedings of the 2015 ACM on Conference on Online Social Networks*, 2015, pp. 59–70.

[54] B. Henne, M. Koch, and M. Smith, "On the awareness, control and privacy of shared photo metadata", in *International Conference on Financial Cryptography and Data Security*, Springer, 2014, pp. 77–88.

[55] E. McCallister, *Guide to protecting the confidentiality of personally identifiable information*. Diane Publishing, 2010, vol. 800.

[56] L. Baruh, E. Secinti, and Z. Cemalcilar, "Online privacy concerns and privacy management: A meta-analytical review", *Journal of Communication*, vol. 67, no. 1, pp. 26–53, 2017.

[57] C. Prince, N. Omrani, A. Maalaoui, M. Dabic, and S. Kraus, "Are we living in surveillance societies and is privacy an illusion? an empirical study on privacy literacy and privacy concerns", *IEEE Transactions on Engineering Management*, 2021.

[58] K. Sharma, S. Gupta, P. Gupta, and P. Arora, "User's perception on social media privacy concern", in *2018 4th International Conference on Computing Sciences (ICCS)*, IEEE, 2018, pp. 80–83.

[59] OWASP Foundation, *Owasp top 10 privacy risks*, 2021. [Online]. Available: `https://owasp.org/www-project-top-10-privacy-risks/`.

[60] A. Cavoukian *et al.*, "Privacy by design: The 7 foundational principles", *Information and privacy commissioner of Ontario, Canada*, vol. 5, p. 12, 2009.

[61] A. D. Scott, *Building Web Apps that Respect a User's Privacy and Security*, First Edition. O'Reilly, 2016.

[62] Offensive Security, *The kali linux documentation*, 2022. [Online]. Available: `https://www.kali.org/docs/`.

[63] G. D. Singh, *The Ultimate Kali Linux Book: Perform advanced penetration testing using Nmap, Metasploit, Aircrack-ng, and Empire, 2nd Edition*. Packt Publishing Ltd, 2022.

[64] G. Najera-Gutierrez, *Kali Linux Web Penetration Testing Cookbook*. Packt Publishing Ltd, 2016.

[65] Google, *Tink cryptographic library*, 2022. [Online]. Available: `https://developers.google.com/tink`.

[66] The OWASP Foundation, *The owasp software assurance maturity model*, 2022. [Online]. Available: `https://owaspsamm.org/`.

[67] OWASP Foundation and J. Long, *Owasp dependency-check*, 2022. [Online]. Available: `https://owasp.org/www-project-dependency-check/`.

[68] Vaadin, *Security in vaadin applications*, 2022. [Online]. Available: `https://vaadin.com/docs/v14/flow/advanced/framework-security#cross-site-request-forgery-csrf-xsrf`.

[69] S. McElroy, "Detecting server-side request forgery attacks on amazon web services", *ISSA Journal February 2020*, vol. 18, 2 2020.

[70] Ghostery, *Ghostery ad & tracker blocker for any device*, 2022. [Online]. Available: `https://www.ghostery.com/ghostery-ad-blocker`.

[71] P. Cichonski, T. Millar, T. Grance, K. Scarfone, *et al.*, "Computer security incident handling guide", *NIST Special Publication*, vol. 800, no. 61, pp. 1–147, 2012.