*Security report*

Fontys University of Applied Sciences

***Logo

Description automatically generated***

|  |
| --- |
| **Date** : 25/05/2023 |
| **Version** : 2.0 |
| **State** : In progress |
| **Author**  : Anna Kadurina |

Table of Contents

[Introduction 2](#_Toc136452557)

[Purpose of this document 2](#_Toc136452558)

[Reasonings 2](#_Toc136452559)

[OWASP Top 10 table 2](#_Toc136452560)

[A01:2021 - Broken Access Control 4](#_Toc136452561)

[A02:2021 – Cryptographic Failures 4](#_Toc136452562)

[A03:2021 – Injection 4](#_Toc136452563)

[A04:2021 – Insecure Design 5](#_Toc136452564)

[A05:2021 – Security Misconfiguration 5](#_Toc136452565)

[A06:2021 – Vulnerable and Outdated Components 5](#_Toc136452566)

[A07:2021 – Identification and Authentication Failures 5](#_Toc136452567)

[A08:2021 – Software and Data Integrity Failures 5](#_Toc136452568)

[A09:2021 – Security Logging and Monitoring Failures 6](#_Toc136452569)

[A10:2021 – Server-Side Request Forgery (SSRF) 6](#_Toc136452570)

[Conclusion 6](#_Toc136452571)

# Introduction

My name is Anna Kadurina, and I am in the 3rd semester of ICT and Software engineering at Fontys University of Applied Sciences. Up till now, I have implemented a functional and effective system for managing bookings, properties, users, reviews and statistics.

# Purpose of this document

The purpose of this document is to explain in detail how my application deals or does not deal with the OWASP top 10 security risks. Along with that, explore what needs to be done so that both backend and frontend can be secure.

# Reasonings

## OWASP Top 10 table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Likelihood | Impact | Risk | Actions possible | Planned |
| A01:2021-Broken Access Control | Low | Medium | Low | * Enable role-based access control with permissions for each role. * Deny by default | Yes |
| A02:2021-Cryptographic Failures | Low | Medium | Medium | * Classify data, save what is needed and use encryption | Yes |
| A03:2021-Injection | Low | Medium | Low | * Avoid special characters, except for the password | Yes |
| A04:2021-Insecure Design | Medium | Medium | Medium | * Write unit and integration tests to validate that all critical flows are resistant to the threat model | Yes |
| A05:2021-Security Misconfiguration | Medium | Medium | Medium | * Sending security directives to clients | Yes |
| A06:2021-Vulnerable and Outdated Components | Low | Medium | Low | * Implement a tool to monitor for libraries that are unmaintained or do not have security patches | Yes |
| A07:2021-Identification and Authentication Failures | Medium | Medium | Medium | * Limit login failure and alert admin * Weak password check | Yes |
| A08:2021-Software and Data Integrity Failures | Medium | Medium | Medium | * Implement tool to sense malicious code or configuration | Yes |
| A09:2021-Security Logging and Monitoring Failures | Low | Medium | Low | * All login, server-side input failures need to be logged with context to identify suspicious behavior | Yes |
| A10:2021-Server-Side Request Forgery | Low | Low | Low | * Sanitize and validate all client-supplied input data | Yes |

## A01:2021 - Broken Access Control

Broken access control happens when users act outside of their intended permissions. This typically leads to the exposing of unauthorized information, modification and destruction of data. To limit the chances of this risk in my application, I deny the access by default for all the information that is not meant for everyоne. I use a spring system with tokens and authorization permissions. Changing the URL in the browser won’t work and any actions without valid request and authentication won’t succeed too. If the action is not in the scope of the permissions of the user role, it will be blocked. During tests, everything succeeds, but there are still some vulnerabilities that can lead to Broken Access Control, such as the usage of CORS configuration.

## A02:2021 – Cryptographic Failures

Cryptographic failures happen when the data is not protected in transit and at rest. The first step for preventing this risk is to determine which data needs to be encrypted and to classify what needs to be saved and what not. My application is not storing any important user information in session, cоokies etc. Only a token is stored to retrieve the needed user data. In my application, the most important personal information is the password. I am using passwordEncoder interface because allows a password to be encoded but there is no way to decode the password to plain text. Possible improvements would be to encrypt every information about the user when the data is fetched via valid JWT token.

## A03:2021 – Injection

An application is vulnerable to attack when the requests are not validated. I use validations of the fields such as minimum, maximum length and JPA framework, which highly reduces the risk of Injection, as it provides an abstraction layer between my application code and the database. Along with that, it offers built-in mechanisms for handling database queries and data access. A possible improvement would be to avoid special characters everywhere except for the password, which has its own validation.

## A04:2021 – Insecure Design

A secure design is a methodology that constantly evaluates threats and ensures that the code is robustly designed and tested to prevent knоwn attack methods. I have implemented unit and integration tests with configured user stories limiting the user resources.

## A05:2021 – Security Misconfiguration

This risk regards data that is improperly set or not secure. My application does not have any default users, it is using a proper port and does not contain anything unnecessary. Everything is unique and proper error messages are only shown when the data is not relevant and SonarQube integration is used to determine any vulnerabilities.

## A06:2021 – Vulnerable and Outdated Components

This risk happens when you do not know the versions of all the components you use and if the software is unsupported or out of date. I am using SonarQube to determine if there are any components that are a vulnerability to my application, also to detect unused imports and components.

## A07:2021 – Identification and Authentication Failures

This risk appears when the application fails to protect functions related to user’s identification, authentication, and session management. My application uses JWT token with expiration time. There are a lot of improvements that can be done such as implementing a weak password check.

## A08:2021 – Software and Data Integrity Failures

This risk is about data integrity failures related to code and infrastructure that does not protect against integrity violations. An example for that is insecure CI/CD pipeline that can introduce the potential for unauthorized access. I am using git.fhict with authorized maintainer (me). The repository cannоt be accessed without me giving permission. Using CI/CD tasks such as build, test and SonarQube review every push to the repository.

## A09:2021 – Security Logging and Monitoring Failures

This risk happens when there are unspecific or incorrect logs that do not provide any insightful information. To address this risk, I have implemented monitoring systems in my application, including the integration of Sentry for both the frontend and backend. Sentry helps capture and track errors, exceptions, and performance issues, providing detailed information for debugging and monitoring purposes.

## A10:2021 – Server-Side Request Forgery (SSRF)

This risk occurs when the application is fetching a remote resource without validating the user-supplied URL. I am using validation for each request and user input data.

# Conclusion

To conclude, my application deals with moderately with the OWASP top 10 security risks. Some of them are with higher risk than others, but this document is beneficial for the development process. Knowing the weak spоts of your application is crucial in order to secure it the best way possible. In conclusion, I rate my project’s security a medium overall.