OWASP Report

Name: Tony Jiang

Semester:6

# Introduction

This document aims to determine if the application addresses the OWASP Top 10 security risks. It will provide insights on how these risks will be covered in the application and assess whether this coverage is necessary in application.

# Top 10 security risks

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Likelihood | Impact | Risk | Action possible | Planned |
| A01:2021-Broken Access Control | High | Severe | High | Add authentication | Yes |
| A02:2021-Cryptographic Failures | Unlikely | Severe | Low | Password hashing | Yes |
| A03:2021-Injection | Unlikely | Moderate | Low | Implement Repository Methods in Spring Data JPA | Yes |
| A04:2021-Insecure Design | Likely | Moderate | Moderate | Create a technical design document | Yes |
| A05:2021-Security Misconfiguration | High | Severe | High | Create a testing strategy and test report | N/A |
| A06:2021-Vulnerable and Outdated Components | Likely | Severe | Moderate | OWASP Dependency-Check | N/A |
| A07:2021-Identification and Authentication Failures | Likely | Severe | High | Implement authentication testing | N/A |
| A08:2021-Software and Data Integrity Failures | Unlikely | Moderate | Low | Implement Repository Methods in Spring Data JPA | Yes |
| A09:2021-Security Logging and Monitoring Failures | High | High | High | Implement logging across all critical components and monitoring the logs | N/A |
| A10:2021-Server-Side Request Forgery | High | High | High | Validate input parameters that control server-side requests | N/A |

## A01:2021-Broken Access Control

**Reasoning:** It will lead to other people finding sensitive information of a user and give access to a user that has no permission to do some function on the website for example create song on the website.

**Implementation:** Here is an example of a "get all users" function, which can only be accessed with the admin role. The "IsAuthenticated" annotation is custom-made with a dependency on Spring Security.

A black screen with yellow and orange text

Description automatically generated

A screenshot of a computer program

Description automatically generated

With Spring Security, it is necessary to configure the permissions and assign user roles to access these components.

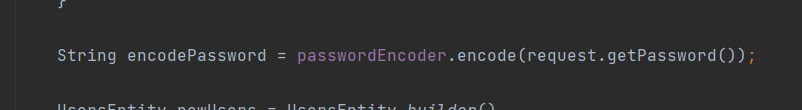
A screen shot of a computer program

Description automatically generated

## A02:2021-Cryptographic Failures

**Reasoning:**  Passwords can be exposed to unauthorized parties. This risk is severe if users reuse the same password across multiple accounts. If an attacker obtains a user's password from one account, they could potentially gain access to all of the user's other accounts.

**Implementation:** Here is how I implemented password hashing using Bcrypt. Bcrypt is a widely used hashing algorithm for securing passwords.



The hashing is achieved by using Spring Security and configuring it to use Bcrypt hashing.A screen shot of a computer

Description automatically generated

## A03:2021-Injection

**Reasoning:** Injection vulnerabilities, such as SQL injection, can harm the database. For example, an attacker might inject an SQL statement to delete database tables or gain unauthorized access to the website by bypassing the login mechanism.

**Implementation:** I use Spring JPA, which provides methods that help prevent SQL injection. Here is an example of its usage. The methods are provided by Spring JPA and can be used when creating the class.

A screen shot of a computer program

Description automatically generated

## A04:2021-Insecure Design

**Reasoning:** Insecure design refers to the lack of security considerations during the design phase of software development. This can result in fundamental security weaknesses that are difficult or impossible to mitigate later.

**Implementation:** I created a technical design document that identifies any weaknesses in the application's design. This document provides an overview of how to address and improve the design later if any issues are discovered.

## A05:2021-Security Misconfiguration

**Reasoning:** Security misconfigurations can often be prevented by thoroughly checking all configurations, though this process can be time-consuming. Implementing a testing strategy and creating test reports can streamline this process and reduce the time required to identify and correct misconfigurations.

**Implementation:** The testing strategy is a work in progress, and the test report has not been created yet. These tasks are planned to be completed.

## A06:2021-Vulnerable and Outdated Components

**Reasoning:** There might be outdated dependencies in the project that need to be checked.

**Implementation:** The OWASP Dependency-Check has not been implemented yet but will be implemented in due time.

## A07:2021-Identification and Authentication Failures

**Reasoning:** There is currently no authentication testing set up in the project, which is a critical gap that needs to be addressed.

**Implementation:** This is not yet implemented and will come in due time.

## A08:2021-Software and Data Integrity Failures

**Reasoning:** By utilizing Spring JPA methods, which prevent data integrity failures, the data remains consistent, reliable, and accurate.

**Implementation:** Here is an example of the song repository implemented using Spring JPA.

A screenshot of a computer program

Description automatically generated

## A09:2021-Security Logging and Monitoring Failures

**Reasoning:** There is no critical logging, or any logging set up in the project, and there is also no monitoring system in place.

**Implementation:** Not yet implemented, it will come in due time.

## A10:2021-Server-Side Request Forgery

**Reasoning:** There are no mechanisms currently in place to validate input parameters that control server-side requests.

**Implementation:** Not yet implemented, will look into it.

# Conclusion

The report indicates that the application is 50% compliant with the OWASP Top 10. The remaining 50% still needs to be addressed and evaluated. Additionally the application needs to cover all the OWASP top 10.