Kaloyan Kulov

Fit sphere

Security Report

**Table of Contents**

1. Introduction
2. Risk Overview
3. Risk Explanation

**Introduction**

This report provides an overview of the top 10 OWASP security risks and the measures taken to mitigate them in the "FitSphere" project. For each risk, we outline the likelihood, impact, overall risk level, preventative measures implemented, potential additional measures, and explanations for accepted risks (highlighted for clarity).

**Risk overview**

A chart with different colored squares

Description automatically generated

**Risk Explanation**

|  |
| --- |
| A01:2021-Broken Access Control: |
| Users should not be able to do modifications to data they do not own. Accessing protected resources without being logged in. |

Currently, "FitSphere" uses JWT tokens that are valid for a short period to prevent data theft in case of token leakage. These tokens are generated using a secret key. On the backend, service methods validate if a user has the authority to modify data or if the user is an admin. Additionally, the server only accepts requests from localhost:5173.

I assess this risk as HIGH because the access token is currently stored in a session, making it vulnerable to Cross-Site Scripting (XSS) attacks. Another contributing factor is the absence of a refresh token mechanism, as its implementation is beyond the scope of this semester.

To mitigate this risk, I plan to introduce refresh tokens and implement OAuth through Clerk, a service designed for secure user authentication. This will provide enhanced security by leveraging professionally maintained systems. Additionally, incorporating a monitoring system to track user activity would be beneficial. However, due to time constraints and technical limitations, this feature will not be implemented at this time.

|  |
| --- |
| A02:2021-Cryptographic Failures: |
| Sensitive data should be protected (passwords). The website should make use of a security certificate and avoid deprecated features in both the FE and BE. |

I have implemented a password hasher to ensure passwords are securely stored, and sensitive data, such as passwords, is excluded from user data responses. I have marked this risk as MODERATE due to the use of HTTP instead of HTTPS. I'm accepting this risk because implementing HTTPS is beyond the scope of this semester.

If I had the necessary technical expertise, I would have implemented HTTPS to encrypt all server responses, making it more difficult for hackers to intercept and access data.

|  |
| --- |
| A03:2021-Injection: |
| User-supplied data should be always validated. The SQL can be modified by a hacker to extract data or modify it. Users should be able to make request only from authorized  origins. |

The system uses authorization and authentication for different data. The idea is that sensitive data should be kept in safe. I have also implemented ORM in the server and all the input data is validated by the service classes and by Spring boot beans.

|  |
| --- |
| A04:2021-Insecure Design: |
| Data should be validated. The logic flow should be tested. There should be an additional security measures if something continuously fails. |

Currently my system is covered by SonarQube and it is passing the check for code validation. I’m testing the classes where there is some logic inside, so mainly the business layer.

For the future I am planning to implement more E2E testing. This risked is marked as Low because I do not expect problems due to my validation.

|  |
| --- |
| A05:2021-Security Misconfiguration: |
| Passwords should be encrypted, there should be a security handling, the software is up to date and does not use deprecated features. Users do not have administrative rights in the database. |

I cannot mitigate this risk significantly at the moment, which is why I am accepting it. Ideally, to reduce this risk, I would create a dedicated user account in the database for the application's users, restricting permissions to prevent table/database creation or deletion. Additionally, I would transition from a development environment to a production environment.

This risk is HIGH because my system currently lacks adequate protection. Some of the required features for semester 3 use deprecated technologies that I must implement, making it unavoidable. Furthermore, creating a restricted database user account would necessitate modifications to application properties, potentially introducing unexpected bugs that could negatively impact my final grade.

|  |
| --- |
| A06:2021-Vulnerable and Outdated Components: |
| The system should be monitored for deprecated features and components/frameworks. |

The risk is LOW, as I use CI/CD pipeline to monitor this.

As a prevention measure, I can only think about regularly checking the dependencies I use for their lates versions.

|  |
| --- |
| A07:2021-Identification and Authentication Failures: |
| This risk is related to the authentication part of the application and what the user can do. |

Currently, I have implemented password hashing and enforced password complexity on the front end. My JWT token is stored in the browser session. I have marked this risk as MODERATE due to the ease with which a token can be stolen from the session and because I do not enforce password complexity on the backend.

I accept this risk concerning the token due to my limited technical knowledge. Regarding password complexity and login attempt restrictions, I am constrained by time limitations and do not want to introduce potential bugs in the final sprint.

In the future, I plan to enforce password complexity on the backend and implement login attempt restrictions for users.

|  |
| --- |
| A08:2021-Software and Data Integrity Failures: |
| The application should use a secure libraries and dependencies (such as Gradle or Maven in my case). The CI/CD pipeline should not provide sensitive data. |

I consider this risk to be LOW because I utilize Gradle for dependency management. Additionally, in my pipeline, all sensitive data is safeguarded using GitLab variables.

As an additional measure, I would like to implement a monitoring system to validate data integrity.

|  |
| --- |
| A09:2021-Security Logging and Monitoring Failures: |
| The application should have a Monitoring system that checks for suspicious activities. A good add would be to establish or adopt an incident response and recovery plan in case of being hacked or the server going down. |

I consider this risk to be HIGH because I currently lack the technical knowledge to address it effectively. If I had the capability, I would have implemented a system and plan to mitigate this risk.

|  |
| --- |
| A10:2021-Server-Side Request Forgery: |
| Web application should not fetch a remote resource without validating the user-supplied URL because of crafted request to an unexpected destination. |

This risk is LOW for "FitSphere" as I do not use any external APIs.

**Conclusion:**

The current state of the "FitSphere" project meets the needs for semester 3 solidly. I have implemented additional measures such as integration tests and achieved nearly complete unit test coverage, with plans for end-to-end (E2E) testing. My pipeline is secure.

However, if I had more time and technical knowledge, I would implement a refresh token, OAuth, and use HTTPS to make the application production-ready. Overall, I believe I have done a decent job meeting the semester requirements.