

OWASP report

Individual project

Nur Nechushtan | December 2022

Contents

[Versioning Table 3](#_Toc124512573)

[OWAP top 10 4](#_Toc124512574)

[Security risks 5](#_Toc124512575)

[broken access control 5](#_Toc124512576)

[Cryptographic Failures 5](#_Toc124512577)

[Injection 5](#_Toc124512578)

[Insecure Design 6](#_Toc124512579)

[Security Misconfiguration 6](#_Toc124512580)

[Vulnerable and Outdated Components 6](#_Toc124512581)

[Identification and Authentication Failures 7](#_Toc124512582)

[Software and Data Integrity Failures 7](#_Toc124512583)

[Security Logging and Monitoring Failures 7](#_Toc124512584)

[Server-Side Request Forgery 7](#_Toc124512585)

[Conclusion 8](#_Toc124512586)

# Versioning Table

|  |  |
| --- | --- |
| Date | Content |
| 09/12/2022 | * Document initialization * Top 10 OWAP table |
| 17/12/2022 | * Security risks |

# OWAP top 10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Likelihood** | **Impact** | **Risk** | **Actions possible** | **Planned** |
| A1: broken access control | Low | Moderate | Low | Server-side role-based authorization, refresh tokens, JWT tokens stored in HttpOnly cookies. Fixed | Yes |
| A2: Cryptographic  failure | Very low | Moderate | Low | N/A | No |
| A3: Injection | Very low | Moderate | Very low | ORM(JPA). Fixed | Yes |
| A4: Insecure Design | Low | High | Moderate | Testing | Yes |
| A5: Security Misconfiguration | Moderate | Moderate | Moderate | Configure spring security. | Yes |
| A6: Vulnerable and Outdated Components | Moderate | Moderate | Moderate | Remove unused dependencies | No |
| A7: Identification and Authentication Failures | Low | Moderate | Moderate | Not use default credentials, Implement weak password checks | Yes |
| A8: Software and Data Integrity Failures | Low | Low | Low | Integrate OWASP Dependency Check or OWASP CycloneDX | No |
| A9: Security Logging and Monitoring Failures | N/A | N/A | N/A | N/A | No |
| A10: Server-Side Request Forgery | Moderate | Moderate | Moderate | Implement CSRF server in the backend | Yes |

# Security risks

# broken access control

**The risk:**

Access control ensures that users stay within the bounds of their specified permissions. Failures frequently result in the unauthorized disclosure of information, the modification or deletion of all data, or the execution of business operations outside the user's scope, JWT can be tampered and protected actions like deleting a user would be more vulnerable to malicious attacks.

**My actions:**

Implemented server-side role base authorization using spring security, refresh token and server side JWT service using HttpOnly cookies.

# Cryptographic Failures

The risk:

This describes that it is important to protect your data. Especially, it is important to secure information that falls under privacy laws.

My actions:

The only sensitive data in my project is users’ passwords. The password is encrypted before storing it the in the database to secure the accounts even if the database is hacked.

# Injection

The risk:

Injection vulnerabilities allow threat actors to insert malicious data into an application. The injection may include malicious commands that redirect stored data to a malicious website or force changes to the application.

My actions:

Using JPA to preform all commands from and to the database.

# Insecure Design

**The risk:**

Design flaws can introduce critical threats to web applications. They are caused by lack of integration and security controls into the application during the development cycle.

**My actions:**

Create unit, integration and E2E tests and executing them in the pipeline.

# Security Misconfiguration

**The risk:**

Security is important aspect regarding our application. We want to prevent outdated software and unnecessary features that being enabled or installed. Also, it can happen that the values of different frameworks of your application are not set to secure values.

**My actions:**

Configure spring security.

# Vulnerable and Outdated Components

**The risk:**

Using outdated, and vulnerable software components can lead to security risk, when a component is not frequently updated it is more vulnerable to attacks and by using it, we open our application to threats.

**My actions:**

Using dependencies from known sources that are well maintained.

# Identification and Authentication Failures

**The risk:**

These failures are vulnerabilities related to the application authentication. This can occur when passwords are weakly stored. Also, it can happen when the application permits weak passwords, ineffective multi-factor authentication.

**My actions:**

Implement password strength validation, using different environment for testing, development and production to prevent a security risk of having username admin with password admin.

# Software and Data Integrity Failures

**The risk:**

This can occur when code implementation and other infrastructure lack the ability to protect code against all integrity violations. This can happen when you use libraries or repositories from some untrusted source. A hacker can access the system without authorization. The system will become vulnerable to multiple attacks.

# Security Logging and Monitoring Failures

**The risk:**

This vulnerability is related to security monitoring and logging failures. This way we can detect certain suspicious patterns that can lead to attacks. Therefore, by logging and monitoring, we can take the needed action to prevent an attack.

# Server-Side Request Forgery

**The risk:**

An attacker targets applications that uses HTTP requests to a server. From the URLs the attacker abuses the server functions to access or modify data.

**My actions:**

Implement CSRF server in the backend.

# Conclusion

In conclusion, this document highlights several security risks that need to be considered in order to ensure the safe and secure operation of a web application. These risks include broken access control, cryptographic failures, injection vulnerabilities, insecure design, security misconfiguration, vulnerable and outdated components, identification and authentication failures, software and data integrity failures, security logging and monitoring failures, and server-side request forgery. To mitigate these risks, a number of actions were implemented such as server-side role-based authorization, refresh tokens and server-side JWT service, encryption of sensitive data, use of JPA, unit, integration and E2E testing, configuration of spring security, use of dependencies from known sources, password strength validation, and implementation of CSRF server in the backend. These actions provide a robust and secure solution for user authentication and token management in an application.