**Security Report**

***Emergency Web App***

*Netherlands Citizens*

|  |
| --- |
| **Date : 21 December 2023** |
| **Version : 1** |
| **State : Ongoing** |
| **Author : Jennifer Oentoro** |

Contents

[1. The OWASP Top 10 Vulnerabilities 3](#_Toc42673512)

[1.1 SQL Injection Attacks 3](#_Toc42673513)

[1.2 Broken Authentication & Session Management 3](#_Toc42673514)

[1.3 Cross-Site Scripting (XSS) Attacks 5](#_Toc42673515)

[1.4 Insecure Direct Object References 6](#_Toc42673516)

[1.5 Security Misconfiguration 6](#_Toc42673517)

[1.6 Sensitive Data Exposure 7](#_Toc42673518)

[1.7 Missing Function Level Access Control 8](#_Toc42673515)

[1.8 Cross Site Request Forgery Attacks (CSRF) 8](#_Toc42673516)

[1.9 Using Components with Known Vulnerabilities 8](#_Toc42673517)

[1.10 Unvalidated Redirects and Forwards 8](#_Toc42673518)

# The OWASP Top 10 Vulnerabilities

## SQL Injection Attacks

## Backend (Spring Boot with Hibernate and Spring Data JPA):

* Use Parameterized Queries or Prepared Statements: Prefer the use of parameterized queries or prepared statements when interacting with the database. This ensures that user input is treated as data and not executable code.

A screen shot of a computer code

Description automatically generated

* Input Validation and Sanitization: Implement input validation and sanitization to ensure that user input adheres to expected patterns and does not contain malicious code.

A screen shot of a computer

Description automatically generated

1. Frontend (React TypeScript):

* Parameterized Queries in API Calls: When making API calls from the frontend, use parameterized queries to send parameters securely.

A screen shot of a computer code

Description automatically generated

* Avoid Dynamic SQL Queries: Avoid dynamically constructing SQL queries in the frontend based on user input. Instead, define the queries on the server side.

## Broken Authentication & Session Management

1. Backend (Spring Boot with Spring Security):

* Use Spring Security: Leverage Spring Security to handle authentication and authorization. Spring Security provides a comprehensive set of features to secure your application.

A screen shot of a computer

Description automatically generated

* Bcrypt Password Hashing: Continue using Bcrypt for password hashing. Bcrypt is a secure one-way hashing algorithm that is suitable for password storage.

A black background with white text

Description automatically generated

* JWT Authentication: Implement JWT-based authentication for secure and stateless user authentication. Ensure that the JWT token includes necessary user roles and permissions.

A screen shot of a computer program

Description automatically generated

* Token Expiration and Refresh: I have implemented token expiration and refresh functionality in my application, ensuring that JWT tokens have a reasonable expiration time and incorporating mechanisms to refresh tokens, thereby maintaining user sessions securely.

A screen shot of a computer program

Description automatically generated

1. Frontend (React TypeScript):

* Secure Transmission: Ensure that communication between the frontend and backend is encrypted using HTTPS to prevent man-in-the-middle attacks.

## Cross-Site Scripting (XSS) Attacks

1. Backend (Spring boot)

* Output Encoding: Implement proper output encoding for data rendered in HTML responses to ensure that user-supplied content is treated as data and not executable code.

A screen shot of a computer

Description automatically generated

## Insecure Direct Object References

* 1. Backend (Spring boot):
* JWT Claims for Authorization: Include necessary user roles or permissions as claims in the JWT. Verify these claims during authentication and use them for authorization checks.

A computer screen shot of text

Description automatically generated

* 1. Frontend (React Typescript):
* JWT Decoding: Decode the JWT on the frontend and extract necessary claims. Use these claims for client-side authorization checks

A computer screen shot of a code

Description automatically generated

## Security Misconfiguration

1. Backend (Spring boot):

* Secure CORS Configuration: Configure Cross-Origin Resource Sharing (CORS) settings securely to control which domains can access the API.

A computer screen shot of a program code

Description automatically generated

* Error Handling: Customize error handling to avoid exposing sensitive information in error responses. Provide generic error messages to clients and log detailed error information internally.

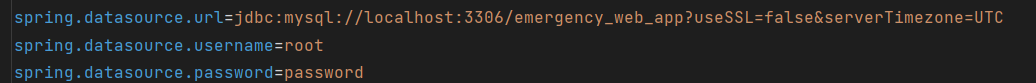
A computer screen with text

Description automatically generated

## Sensitive Data Exposure

1. Backend (Spring boot):

* Secure Password Storage: Safely store sensitive information such as database passwords, API keys, and other credentials using secure methods, such as environment variables or secure vaults.



1. Frontend (React typescript):

* Secure API Requests: Ensure that API requests involving sensitive data are made securely using HTTPS. Avoid sending sensitive information in query parameters and utilize request headers for authentication.

A computer screen shot of a code

Description automatically generated

## Missing Function Level Access Control

* 1. Backend (Spring boot):
* Role-Based Access Control (RBAC): Implement Role-Based Access Control to restrict access to specific functions or features based on the roles assigned to users.

A computer screen shot of text

Description automatically generated

* 1. Frontend (React typescript):
* Client-Side Authorization Checks: Implement client-side authorization checks to conditionally render or enable/disable UI components based on the user's role or permissions.

## Cross Site Request Forgery Attacks (CSRF)

1. Backend (Spring boot):

* CSRF Token Generation: Implement the generation of unique CSRF tokens for each user session. Include these tokens in forms or as headers for state-changing requests

A screen shot of a computer

Description automatically generated

## Using Components with Known Vulnerabilities

* Continuous Integration (CI) Checks: Integrate vulnerability checks into your CI/CD pipeline. This ensures that your project is automatically scanned for vulnerabilities whenever changes are pushed, and builds can be blocked if vulnerabilities are identified.

A screen shot of a computer

Description automatically generated

## Unvalidated Redirects and Forwards

* Controlled Navigation: Use React Router's controlled navigation methods to ensure that redirects and forwards are controlled and follow predefined routes.

A screen shot of a computer program

Description automatically generated