**OWASP SECURITY REPORT**

**Knockout Ticket**

Valentin Gabriel Morisca

04/06/2023

Owasp Top 10 risk for 2021

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures

A10:2021-Server-Side Request Forgery

1.Broken Access Control

This refers to the improper management of access controls, allowing an unauthorized user to perform authorized actions they shouldn’t be able to do within a system.

I am dealing with this problem in this manner:

I am using @PreAuthorize to perform a check if the id from the request matches with the id from the Access Token.

A screen shot of a computer code

Description automatically generated

2. Cryptographic Failures

Cryptographic Failures involves weaknesses and flaws in the implementation or use of cryptographic algorithms, which can lead to the compromise of security.

In my case, I am hashing the passwords of the user.

A computer screen shot of a program

Description automatically generated

3. Injection

Injection vulnerabilities occur when untrusted data is sent to an interpreter as part of a command or query. This can lead to the execution of unintended commands, such as SQL Injection in Databases or Code Injection in Web Applications.

In my own code, I am dealing with it in this manner:

Parameterizing variables in the query enhances security and prevents SQL Injection.

A computer screen shot of a program

Description automatically generated

4. Insecure Design

Insecure Design reflect flaws in the overall chosen architecture of design from a system. This can create vulnerabilities, poor choices in data flow, communication protocols, or overall system structure and design.

In my application, I am dealing with this by using the principles of architectures like S.O.L.I.D.

More regarding this manner is present in the Design Document.

5. Security Misconfiguration

Security Misconfigurations happen when security settings are not properly configured or maintained. This can include default configurations, unnecessary services running, poor choice of dependencies, overall, providing opportunities for attackers.

In my own application, there is a Web Security Configuration present, and also, I am using Roles that are handled by Spring Security.

A computer screen shot of a computer

Description automatically generated

A screen shot of a computer code

Description automatically generated

6. Vulnerable and Outdated Components

Using outdated or vulnerable software components, libraries, or frameworks can expose weak points in the system or can expose a system to known security flaws or vulnerabilities that have been patched in future releases.

In my own application, I have been using the latest versions of wide known frameworks and libraries, which have a rich ecosystem around them which ensures a smooth and secure software application.

7. Identification and Authentication Failures

This involves weaknesses in the processes of identifying and authenticating users. It includes issues like weak passwords, insecure authentication methods, thus making it easier for unauthorized users to gain access.

In my own application I use Spring Security for authentication and authorization, which is known for being a good security framework. Also, I encrypt personal details (like passwords) using BCrypt.

A computer screen shot of a computer screen

Description automatically generated

8. Software and Data Integrity Failures

Software and Data Integrity Failures occur when unauthorized changes or alterations to software or data are possible, compromising the integrity and reliability of the system.

In my own application I ensure that this problem is dealt with in the following manner:

I am using JPA (Java Persistence API), which ensures data consistency and integrity.

A screen shot of a computer program

Description automatically generated

9. Security Logging and Monitoring Failures

This refers to the inadequacy of logging mechanisms, making it difficult to detect and respond to suspicious activity in a timely manner.

In my application, this could be improved, but there are a few points of logging present, specifically in the Frontend.

10. Server-Side Request Forgery

Server-Side Request Forgery, or widely known as SSRF, occurs when an attacker can make a server perform requests on behalf of them, potentially leading to unauthorized access to internal resources and services.

In my implementation I ensured that I do not fetch or use any external data, which largely reduces the probability of SSRF.

Besides the 10 OWASP Points, I have also maintained a standard in security in the following manner:  
  
I ensure that important keys and tokens are hidden and are not shown in the code, and also, are not pushed to the repository.

For example, in this case, I am storing the JWT-SECRET in a separate file.  
A screenshot of a computer

Description automatically generated

Also, the environment variables used for docker-compose and in the pipeline, are stored in Git’s CI/CD settings, which adds another layer of security.

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated