**OWASP Top 10 Risks**

****2021**

***Vibe Check***

|  |
| --- |
| **Date : 17/01/2025** |
| **Version : 1.0** |
| **State : Finished** |
| **Author : Nuno Dias** |

#### Version history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Changes** | **State** |
| 0.1 | 05/01/2025 | Nuno Dias | First Draft | Finished |
| 1.0 | 17/01/2025 | Nuno Dias | Expanded on some risks | Finished |

Contents

[1. Top 10 Risks - 2021 4](#_Toc187017818)

[1.1 Overview 4](#_Toc187017819)

[1.2 Reasonings 4](#_Toc187017820)

[1.2.1 A1: broken access control 4](#_Toc187017821)

[1.2.2 A2: Cryptographic failure 4](#_Toc187017822)

[1.2.3 A03: Injection 5](#_Toc187017823)

[1.2.4 A04: Insecure Design 5](#_Toc187017824)

[1.2.5 A05: Security Misconfiguration 5](#_Toc187017825)

[1.2.6 A06: Vulnerable and Outdated Components 5](#_Toc187017826)

[1.2.7 A07: Identification and Authentication Failures 5](#_Toc187017827)

[1.2.8 A08: Software and Data Integrity Failures 5](#_Toc187017828)

[1.2.9 A09: Security Logging and Monitoring Failures 5](#_Toc187017829)

[1.2.10 A10: Server-side request forgery 5](#_Toc187017830)

[2. Conclusion 6](#_Toc187017831)

# Top 10 Risks - 2021

## Overview

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Likelihood** | **Impact** | **Risk** | **Actions possible** | **Planned** |
| A1: Broken access control | High | Severe | Medium | Self and Outside Review | Yes |
| A2: Cryptographic failure | Likely | Severe | Low | Updated B-Crypt package, HTTPS encryption | Risk accepted |
| A03 Injection | Very unlikely | Severe | Low | Verifying string conversion in request | No |
| A04: Insecure Design | Likely | Severe | High | Request reviews by Experienced Developers | Yes |
| A05: Security Misconfiguration | Unlikely | Severe | Moderate | Request reviews by Experienced Developers | Yes |
| A06: Vulnerable and Outdated Components | Unlikely | Severe | Moderate | Replace 1 year+ old components | Risk accepted |
| A07: Identification and Authentication Failures | Likely | Moderate | Moderate | Connection limits and other defences | Yes |
| A08: Software and Data Integrity Failures | Likely | Moderate | Moderate | More regular updates to tests and pipeline | Yes, but time constraint |
| A09: Security Logging and Monitoring Failures | High | Low | Low | Thorough logging and monitoring frequently saved | Risk accepted |
| A10: Server side request forgery | High | Moderate | Moderate | Improve framework implementation | Risk accepted |

## Reasonings

### A1: broken access control

Broken access coontrol refers to an attacker having access to resources that they should not be able to with their permission.

To solve this, all requests only apply to current user ID which is read from the JWT token. Also, in case of failure there is no sensitive information to be leaked since all profiles are public and passwords are not shared. As for updating and deleting accounts, the user Id is read from the JWT token.

### A2: Cryptographic failure

Being reliant on an external package there is not much to do about a possible Cryptographic failure. As for https depends on a proper deployment solution which is currently not a priority.

### A03: Injection

All requests are received by a Springboot REST Api that automatically converts them to objects. These are then processed by the JPA hibernate Java frameworks that automatically escape characters.

### A04: Insecure Design

This is a big risk but as inexperienced developer there is not much to do besides researching patterns before coding and asking for reviews from experienced developers which is planned.

### A05: Security Misconfiguration

This issue is a possibility but by adhering to the minimal permissions philosophy it should not be a problem (closing all entries and only opening the necessary ones).

### A06: Vulnerable and Outdated Components

Components save time and with the scale and timeframe of this project it is impossible to implement everything from scratch, and it would probably have more security flaws. This is the least of two evils and an acceptable risk.

### A07: Identification and Authentication Failures

Due to time constraints and lack of testing basic protection will be added but wont be tested thoroughly.

### A08: Software and Data Integrity Failures

Testing takes a lot of time that is not sustainable for small projects with constant change. A fix of all tests weekly is about the best compromise possible that is still viable.

### A09: Security Logging and Monitoring Failures

For a small-scale project that will not be serviced real-time, custom monitoring is slightly overkill. A framework as well as basic statistics and some monitoring, however, are part of the plan.

### A10: Server-side request forgery

This risk is accepted since there are few attack vectors (embedded links only).

# Conclusion

The biggest and most severe risks come lack of experience with good/proven patterns, configuration and design. These will mostly be dealt with by research and reviews by people with more knowledge.

The second biggest risk are dependencies on frameworks, components and outside code. This is an accepted risk that cannot be avoided given the expertise, scale and timeframe of this project. Self-maintained code has a higher chance of creating vulnerabilities.

Finally, there is a risk with lack of monitoring, logging and other forms of awareness regarding the code (tests, pipelines, etc.) regarding the application. Again, due to the scale and timeframe of the project while the basics are implemented there is no time or need for the same kind of tools and overview in a million-user application.